Mach Learn (2011) 82: 281–314
DOI 10.1007/s10994-010-5192-9

Learning to compete, coordinate, and cooperate in repeated games using reinforcement learning

Jacob W. Crandall · Michael A. Goodrich

Received: 13 February 2008 / Revised: 10 March 2010 / Accepted: 27 April 2010 /
Published online: 20 May 2010
© The Author(s) 2010

Abstract We consider the problem of learning in repeated general-sum matrix games when a learning algorithm can observe the actions but not the payoffs of its associates. Due to the non-stationarity of the environment caused by learning associates in these games, most state-of-the-art algorithms perform poorly in some important repeated games due to an inability to make profitable compromises. To make these compromises, an agent must effectively balance competing objectives, including bounding losses, playing optimally with respect to current beliefs, and taking calculated, but profitable, risks. In this paper, we present, discuss, and analyze M-Qubed, a reinforcement learning algorithm designed to overcome these deficiencies by encoding and balancing best-response, cautious, and optimistic learning biases. We show that M-Qubed learns to make profitable compromises across a wide-range of repeated matrix games played with many kinds of learners. Specifically, we prove that M-Qubed's average payoffs meet or exceed its maximin value in the limit. Additionally, we show that, in two-player games, M-Qubed's average payoffs approach the value of the Nash bargaining solution in self play. Furthermore, it performs very well when associating with other learners, as evidenced by its robust behavior in round-robin and evolutionary tournaments of two-player games. These results demonstrate that an agent can learn to make good compromises, and hence receive high payoffs, in repeated games by effectively encoding and balancing best-response, cautious, and optimistic learning biases.

Keywords Multi-agent learning · Reinforcement learning · Game theory

Editor: Avrim Blum.

J.W. Crandall (✉)
Computing and Information Science Program, Masdar Institute of Science and Technology, Abu Dhabi, UAE
e-mail: jcrandall@masdar.ac.ae

M.A. Goodrich
Computer Science Department, Brigham Young University, Provo, UT, USA
e-mail: mike@cs.byu.edu

1 Introduction

Many real-world environments require an agent to interact repeatedly with other learning agents. Learning successfully in such circumstances is challenging since the changing strategies of learning associates creates a non-stationary environment that is problematic for traditional artificial learning algorithms. Furthermore, the way that an agent's associates change their behavior is based, in part, on the how the agent behaves. Thus, a self-interested agent seeking to maximize its payoffs over time must balance three, often conflicting, objectives when selecting a strategy: (1) bound individual losses, (2) play "optimally" with respect to its beliefs about associates' current strategies and (3) influence associates' future behavior.

The importance of these three objectives and the way that an agent must reason and act to achieve them depends on the payoff structure of the environment (or game). When payoffs constitute a fully competitive, constant-sum, game, an agent is primarily concerned with playing "optimally" and bounding its losses, though it may possibly use deceit to influence opponents' behavior. On the other hand, when agents share common interests, an agent should focus on influencing its associates to maximize its payoffs. Specifically, an agent should make its future actions and intentions as obvious and consistent as possible so that associates know what to expect. In games of conflicting interest, influence and optimality may be in competition, as seeking to maximize payoffs according to current beliefs or utilities could cause associates to learn to play undesirable strategies. Thus, to maximize payoffs, an agent might need to refrain from seeking to play "optimally" with respect to its current beliefs (Frank 1988).

Learning effectively over repeated interactions becomes particularly challenging when the agent cannot observe associates' payoffs and, hence, does not know the competitive nature of the game. In such situations, an agent must be open to compromise. Compromise can produce high payoffs, but being open to compromise incurs risk. The ability to bound losses (i.e., *security*) mitigates this risk and is an important element of compromise. However, in many situations, an agent can do much better. Good compromises require agents to *cooperate* in order to achieve payoffs that approach or exceed the value of the game's Nash bargaining solution (Nash 1950), or NBS. The ability to cooperate entails the ability to coordinate behavior in shared-interest games.

Thus, we advocate that, in repeated general-sum games, successful algorithms should have the following three properties when associating with a wide-range of agents:

1. Security—Guarantee that long-term average payoffs meet a minimum threshold.
2. Coordination—Ability to coordinate behavior when associates share *common interests*.
3. Cooperation—Ability to make compromises that approach or exceed the value of the NBS in games of *conflicting interest*.

These properties, which are derived from a single metric presented in Sect. 2, are difficult to achieve simultaneously since security mechanisms (property 1) must not interfere with an algorithms' ability to coordinate and cooperate (properties 2 and 3) and vice versa. An algorithm simultaneously possessing these properties in repeated general-sum games has not been identified in the literature when associates' payoffs are unobservable.

In this paper, we present, analyze, and discuss a reinforcement learning algorithm called M-Qubed.[1] M-Qubed implements and balances best-response, optimistic, and cautious learning biases to anchor the learning process in order to achieve effective compromises

[1] M-Qubed was first presented at ICML 2005 (Crandall and Goodrich 2005). In this paper, we present a slightly modified version of M-Qubed, and analyze and explain its dynamics and properties more thoroughly.

across a large number of two-player repeated general-sum matrix games. Specifically, we prove that M-Qubed is secure in all general-sum matrix games. We also show via empirical studies that M-Qubed learns to coordinate and cooperate in self play in many two-player repeated games (Sect. 4). Furthermore, we show that these properties help M-Qubed to be highly effective in evolutionary and round-robin tournaments involving 17 different learning algorithms from the literature in a variety of two-player matrix games (Sect. 5). These results (1) demonstrate the robustness of M-Qubed in repeated general-sum games, (2) establish the need to encode and balance competing learning biases in multi-agent learning, and (3) demonstrate how this balance can be achieved.

2 Background

Before presenting and analyzing M-Qubed in Sect. 3, we first define assumptions and terminology, and then discuss related work.

2.1 Terms and notation

A repeated game consists of a sequence of *stages* played by n players. In each stage t, each player i selects an action a_i^t from a finite (and constant) set of actions A_i. The resulting *joint action*, $\mathbf{a}^t = (a_1^t, \dots, a_n^t) \in \mathbf{A} = A_1 \times \cdots \times A_n$, results in the payoff vector $\mathbf{r}(\mathbf{a}^t) = (r_1(\mathbf{a}), \dots, r_n(\mathbf{a}))$, where $r_i(\mathbf{a}^t)$, also denoted r_i^t, is the payoff to player i when the joint action $\mathbf{a}^t$ is played.[2] For simplicity, we assume that each player's payoffs are such that its highest possible payoff is one, and its lowest possible payoff is zero.

A *strategy* (or *policy*) for player i is a probability distribution π_i over its action set A_i. A strategy may be a *pure strategy* (the player selects an action deterministically) or a *mixed strategy* (otherwise). A *joint strategy* played by the n players is denoted by $\boldsymbol{\pi} = (\pi_1, \dots, \pi_n)$. Also, let $\mathbf{a}_{-i}$ and $\boldsymbol{\pi}_{-i}$ refer to the joint action and strategy, respectively, of all players except player i. Let $r_i(\boldsymbol{\pi})$ be the expected payoff to player i when the joint strategy $\boldsymbol{\pi}$ is played, and let $r_i(\pi_i, \boldsymbol{\pi}_{-i})$ be the expected payoff to player i when it plays π_i and its associates play $\boldsymbol{\pi}_{-i}$.

In this paper, we assume that each player i observes the full joint action $\mathbf{a}^t$, but only observes its own payoff r_i^t; player i does not observe r_j^t for $j \neq i$. Thus, a player can construct its own payoff matrix, but not that of its associates. This restriction implies that each player does not know the competitive nature of the game. For example, Table 1(a) shows the payoffs for a row player in a two-player, two-action game. This payoff matrix could be part of a competitive (constant-sum) game (Table 1(b)), a common-interest game (Table 1(c)), or a game of conflicting interest (Table 1(d)).

For simplicity, we assume a player knows its own payoff matrix prior to the beginning of the game. We also assume that the same n players play each stage of the repeated game, and that the game repeats a large (possibly infinite) and unknown number of times.

2.2 Learning metrics

We adopt the prescriptive agenda for multi-agent learning (Shoham et al. 2007). Thus, we seek an agent that maximizes its accumulated payoff over the course of the repeated game,

[2] When applicable, we use subscripts to specify the player and superscripts to specify time (stage number).

Table 1 (a) Row player's payoff matrix. (b)–(d) Three different kinds of games that the row player could be playing. In each cell, the payoff to the row player is listed first, followed by the payoff to the column player

(a)

	c	**d**
a	0.3	1.0
b	0.0	0.7

(b) Competitive game

	c	**d**
a	0.3, 0.7	1.0, 0.0
b	0.0, 1.0	0.7, 0.3

(c) Common interest

	c	**d**
a	0.3, 0.3	1.0, 1.0
b	0.0, 0.0	0.7, 0.7

(d) Conflicting interest

	c	**d**
a	0.3, 0.4	1.0, 0.0
b	0.0, 1.0	0.7, 0.6

or equivalently, maximizes its average payoff per stage over the course of the game. However, the varying competitive nature of general-sum games and the wide range of possible associates make it difficult to quantify what it means for an agent to "maximize payoffs." A number of metrics have been defined, of which we review the two most commonly discussed in the literature: no regret and best response.

2.2.1 No regret

One metric of success in repeated games is *no-regret* (Foster and Vohra 1999), also known as universal consistency (Fudenberg and Levine 1998). An algorithm is said to have no-regret if it performs at least as well in the limit as its best strategy would have performed against its associates' observed actions. However, it is difficult for this metric to capture how one's actions affect its associates' future behavior (de Farias and Megiddo 2004). As such, most no-regret learning algorithms learn myopic, unsuccessful, strategies in many important repeated games, though they do provide a bound on the minimum average payoff a player receives. Recently, less-myopic no-regret algorithms have been proposed (Chang and Kaelbling 2005; Chang 2007). However, under the knowledge restrictions we consider in this paper, it remains unclear how these algorithms can be designed to effectively influence associates' behaviors across the space of repeated general-sum games.

2.2.2 Best response

Another potential objective of multi-agent learning is to learn to play a *best response* to associates' strategies. That is, a player's best response is the strategy that maximizes its expected payoffs with respect to its associates' current strategies. Formally, player i's best response is given by: $\pi_i^* = \arg\max_{\pi_i} r_i(\pi_i, \boldsymbol{\pi}_{-i})$. When all players play a best response, the result is a *Nash equilibrium* (NE) in which no player has incentive to unilaterally change its strategy. Many algorithms have been proposed that play a best response with respect to some form of current beliefs or utility estimates, including Fictitious Play (Fudenberg and Levine 1998) and Q-learning (Watkins and Dayan 1992). Bowling and Veloso (2002) took this concept one step further by arguing that a successful multi-agent learning algorithm should both (a) learn to play a best response and (b) converge when associates' strategies converge. These two properties are achieved by AWESOME (Conitzer and Sandholm 2003).

While the best response can be a useful learning objective, it does not, by itself, ensure successful learning. One reason for this is the presence of multiple NE in many games, which entails that a successful learner cannot focus solely on *unilateral* changes in strategy. For example, the folk theorem states that many repeated games have an infinite number of *NE of the repeated game* (rNE) (Gintis 2000). In such situations, a successful learning algorithm should *influence* associates to play a desirable equilibrium. Littman and Stone (2005) give an algorithm to compute the rNE corresponding to the NBS (Nash 1950). While the NBS, which we review in Sect. 2.2.3, produces payoffs that correspond to a desirable rNE (it is pareto efficient and matches Nash's definition of *fairness*), a self-interested agent will sometimes find it possible to strike a compromise that is more beneficial to it individually (e.g., the *Bully* agent proposed by Littman and Stone 2001). Thus, in general, no single rNE should be the universal objective of a self-interested agent in repeated general-sum games.

2.2.3 Desirable properties: security, coordination, cooperation

While no single equilibrium concept effectively defines "optimal" behavior in repeated general-sum games, a self-interested learning algorithm should satisfy a set of properties to be successful. In this paper, we define a set of properties to measure an algorithm's ability to make and accept profitable compromises. These properties are based on a metric called *level of compromise*, which is computed using two points of comparison.

The first point of comparison is the *maximin value* v_i^{MM}, which is the highest average payoff that an agent can guarantee itself without some form of cooperation from associates. v_i^{MM} and the associated *maximin strategy* π_i^{MM} are given by:

$$v_i^{\mathrm{MM}} = \max_{\pi_i} \min_{\mathbf{a}_{-i}} r_i(\pi_i, \mathbf{a}_{-i}), \tag{1}$$

$$\pi_i^{\mathrm{MM}} = \arg\max_{\pi_i} \min_{\mathbf{a}_{-i}} r_i(\pi_i, \mathbf{a}_{-i}). \tag{2}$$

Since we assume that player i knows its own payoff matrix, the maximin value and strategy can be computed in polynomial time using linear programming (Shapley 1953; Chvatal 1983).[3] Let $\mathbf{v}^{\mathrm{MM}} = (v_1^{\mathrm{MM}}, \ldots, v_n^{\mathrm{MM}})$.

The second point of comparison is the *NBS value* v_i^{NBS}, or player i's expected payoff when the NBS is played. The NBS, originally defined by Nash (1950) for two-player games, can be defined as follows. Let $P = \{\forall \mathbf{a} \in A, \mathbf{r}(\mathbf{a}) - \mathbf{v}^{\mathrm{MM}}\}$ and let $\mathcal{C}$ be the set of points in the first quadrant (including the origin) of the convex hull formed by the points in the set P. Then, $\mathbf{v}^{\mathrm{NBS}} = (v_1^{\mathrm{NBS}}, \ldots, v_n^{\mathrm{NBS}})$ is given by

$$\mathbf{v}^{\mathrm{NBS}} = \arg\max_{\mathbf{c} \in \mathcal{C}} \prod_{j=1}^{n} (c_j - v_j^{\mathrm{MM}}). \tag{3}$$

Since the NBS is pareto efficient and satisfies a notion of fairness (Nash 1950), it offers a desirable level of proficiency that a player would potentially like to achieve. While a successful player's average payoff should never be below v_i^{MM}, it should, ideally, approach or

[3] We assume that player i's associates can potentially view player i's payoffs and can communicate without its knowledge. Hence, all associates can act as a single agent, and v_i^{MM} can be computed using linear programming. Borgs et al. (2008) show that, for ($n > 2$)-player games, it is NP-hard for player i's associates to compute the minimax strategy when they cannot communicate without player i's knowledge.

exceed v_i^{NBS}. To capture these goals, we define a player's *level of compromise* ℓ_i as follows:

$$\ell_i = \frac{v_i^{\text{avg}} - v_i^{\text{MM}} + 1}{v_i^{\text{NBS}} - v_i^{\text{MM}} + 1}, \tag{4}$$

where v_i^{avg} is player i's average payoff per stage of the repeated game as $t \to \infty$. When payoffs are bound to the range $[0, 1]$, then $0 \le \ell_i \le 2$. A level of compromise of $\ell_i = 1$ means that player i's average payoff matches the NBS value, $\ell_i > 1$ means that its average payoff exceeds the NBS value, and $\ell_i < 1$ means that its average payoff is less than the NBS value. Furthermore, when $\ell_i < \frac{1}{v_i^{\text{NBS}} - v_i^{\text{MM}} + 1}$, its average payoff is less than its maximin value.

The effectiveness of a learning algorithm is determined by the level of compromise ℓ_i that it is able to achieve when associating with a target class of associates in a target set of games. Furthermore, a robust learning algorithm should achieve high levels of compromise when the target class of associates and games is large. With this in mind, we define the term *(G, Ω, C)-efficient.*

Definition 1 *(G, Ω, C)-Efficient.* Let Ω be a target class of algorithms, let G be a set of matrix games, and let $0 \le C \le 2$ be a target level of compromise. Player i is (G, Ω, C)-efficient if and only if $\ell_i \ge C$ in all games $g \in G$ played with any of the associates in Ω.

We use (G, Ω, d_i)-efficiency to define three properties of compromise that a successful learning algorithm should satisfy in general-sum games. The first property of compromise, called security, is the ability to bound losses regardless of the game being played or how associates behave.[4]

Property 1 (Security) *Let G_{all} be the set of all general-sum matrix games, and let Ω_{all} be the set of all possible algorithms. Player i is secure if and only if it is $(\{g\}, \Omega_{\text{all}}, \frac{1}{v_i^{\text{NBS}}(g) - v_i^{\text{MM}}(g) + 1})$-efficient for all $g \in G_{\text{all}}$, where $v_i^{\text{NBS}}(g)$ and $v_i^{\text{MM}}(g)$ are the NBS and maximin values for player i in the game g. Equivalently, player i is secure if* $\lim_{t\to\infty} \frac{1}{t}\sum_t r_i^t \ge v_i^{\text{MM}}$ *regardless of the game or associates.*

An agent can easily satisfy the security property by playing its maximin strategy π_i^{MM}. However, this static strategy is typically undesirable as a much higher level of compromise is typically achievable. Ultimately, a successful learning algorithm should achieve an $\ell_i \ge 1$ level of compromise against associates $\Omega' \subset \Omega_{\text{all}}$ that can be induced to cooperate. This behavior is captured by the properties of coordination and cooperation.

Property 2 (Ω-Coordination) *Let G_1 be the set of* common-interest *games. Player i satisfies this property if it is $(G_1, \Omega, 1 - \epsilon)$-efficient, where $\epsilon > 0$ is small.*

Property 3 (Ω-Cooperation) *Let G_2 be the set of* conflicting-interest *games. Player i satisfies this property if it is $(G_2, \Omega, 1 - \epsilon)$-efficient, where $\epsilon > 0$ is small.*

[4] We assume that associates can possibly view player i's policy π_i, but cannot determine how the agent randomizes when π_i is a mixed strategy.

As noted by the prefix Ω in the two previous definitions, discussions of the coordination and cooperation properties must be cast in context of the set Ω. For simplicity throughout this paper, we refer to the Ω-coordination and Ω-cooperation properties as simply *coordination* and *cooperation*, and quantify the use of these terms by specifying the set Ω against whom a player satisfies these properties.

To summarize, we desire learning algorithms that are secure, and that satisfy the coordination and cooperation properties against a wide-range of learners. M-Qubed, the algorithm presented and analyzed in this paper, takes a step in that direction. First, it is provably secure. Second, it approaches $\ell_i = 1$ levels of compromise in *self play* in many two-player repeated matrix games. Third, it achieves high levels of compromise over a large number of matrix games when associating with many learning algorithms from the literature, as demonstrated by its high performance in a series of round-robin and evolutionary tournaments.

M-Qubed achieves these characteristics by encoding and balancing competing learning biases. While interesting to study in the context of M-Qubed, these results also suggest that encoding and balancing these learning biases are general principles for successful multi-agent learning in repeated games.

2.3 Assessing the state-of-the-art

While security, coordination, and cooperation are central to successful multi-agent learning, they are difficult to achieve simultaneously when the payoffs of associates are unknown. As illustrated by Table 2, this is true even when Ω is constrained to copies of one's self for the coordination and cooperation properties.

Most of the algorithms listed in Table 2 satisfy no more than a single property. The algorithms are typically designed either for constant-sum or shared-interest games. Only Ma-

Table 2 Characteristics of typical learners in repeated matrix general-sum games. * indicates that the algorithm requires knowledge of associate's payoffs or the kind of game being played, which we disallow in this paper

Learning algorithm	Security property	Cooperation (self play)	Compromise (self play)
Q-learning (Watkins and Dayan 1992)	No	No	No
Fictitious play (Fudenberg and Levine 1998)	No	No	No
MinimaxQ (Littman 1994)	Yes	No	No
WoLF-PHC (Bowling and Veloso 2002)	No	No	No
GIGA-WoLF (Bowling 2005)	Yes	No	No
Hyper-Q (Tesauro 2004)	No	No	No

Satisficing learning
(Karandikar et al. 1998;
Stimpson and Goodrich 2003;
Crandall 2005)
	No	Yes	Many games

OAL (Wang and Sandholm 2003)	No	Yes	No
CoLF (de Cote et al. 2006)	No	Yes	Many games
WPL (Abdallah and Lesser 2008)	Yes	No	No
* NashQ (Hu and Wellman 1998)	No	No	No
* FFQ (Littman 2001)	Yes	Yes	No
* Manipulator (Powers and Shoham 2005)	Yes	Yes	Yes

nipulator, which requires knowledge of the payoff matrices of all players, satisfies all three properties. Manipulator computes the NBS and seeks to enforce this strategy for a certain number of stages. When its associates learn to play their portions of the NBS, Manipulator continues to play its portion of the NBS. If associates fail to respond, Manipulator switches to a fictitious play-like learning rule, and ultimately falls back to the maximin strategy if fictitious play fails to generate secure payoffs.

Manipulator could be altered to satisfy the knowledge restrictions we consider (i.e., no knowledge of the payoffs of associates) by spending the initial rounds of the game communicating its payoff matrix through its actions. If associates did the same, Manipulator could construct the complete payoff matrix and then proceed as before. This would allow it to achieve an $\ell_i = 1$ level of compromise in self play while remaining secure. However, outside of self play, this algorithm would likely become a secure version of fictitious play, meaning that it would likely have difficulties making successful compromises in some important games.

While self play is important, we seek to develop algorithms that can learn to compromise more robustly; namely algorithms that are secure and that learn to coordinate and cooperate in both self play and when associating with other algorithms. With this goal in mind, we present the M-Qubed algorithm.

3 M-Qubed

To make and accept profitable compromises in repeated general-sum games, M-Qubed (*M*ax or *M*ini*m*ax Q-learning = M^3 − Q = M-Qubed) encodes and balances three learning biases, each of which has been encoded in one form or another in learning algorithms found in the literature. The first learning bias, called the *best-response learning bias*, encourages M-Qubed to learn to play a best response with respect to its current beliefs about the environment (i.e., its Q-estimates). Specifically, M-Qubed uses Sarsa reinforcement learning (Rummery and Niranjan 1994) and recurrent state (Moody et al. 2004; Sandholm and Crites 1996) to estimate its future discounted reward for taking an action from a given state. The use of recurrent state allows M-Qubed to reason about how its actions affect the actions of associates, and how the actions of associates affect its future payoffs.

Second, M-Qubed encodes a *cautious learning bias* in order to bound its losses when its beliefs are in error. Specifically, M-Qubed's policy moves toward its maximin value when other compromises fail (Littman 1994). In so doing, M-Qubed is secure in all general-sum matrix games without interfering with its ability to learn to make other, more profitable, compromises. An alternate, but somewhat related, security mechanisms is used by (Powers and Shoham 2005).

Third, M-Qubed encodes an *optimistic learning bias* by initializing its Q-values to optimistic values. This learning bias, called the *optimism-in-uncertainty* principle, is known to provide a natural method for determining how to balance exploration and exploitation (Brafman and Tennenholtz 2003). In so doing, M-Qubed's behavior is biased toward strategies with potentially high future rewards, even if such strategies expose the player to high risks. This allows the agent to avoid *action shadowing* (Fulda and Ventura 2007; Stimpson et al. 2001), in which the estimated utility for taking an action corresponding to a good outcome is low because the other possible outcomes for taking that action are low.

In the remainder of this section, we describe the details of how M-Qubed encodes and balances these three learning biases. Specifically, we describe how M-Qubed represents and updates its utilities, and how it selects actions based on these utilities.

3.1 Representation

While repeated matrix games are often considered to be stateless, M-Qubed uses the previous ω joint actions to define its state. Let $H(\omega)$ denote the set of all joint-action histories of length ω. Then, M-Qubed's current state is given by $s^t = (\mathbf{a}^{t-\omega}, \ldots, \mathbf{a}^{t-1}) \in H(\omega)$. This representation of state, which has been called *recurrent state* (Moody et al. 2004), has been shown to increase cooperation in the prisoners' dilemma (Sandholm and Crites 1996), as it allows agents to learn a signaling system (Skyrms 2004).

For each $a \in A_i$ and state $s \in H(\omega)$, M-Qubed learns its expected future discounted reward $Q(s,a)$. $Q(s,a)$ is given by

$$Q(s,a) = r(s,a) + \gamma V(s,a), \tag{5}$$

where $r(s,a)$ is the expected reward M-Qubed receives immediately after taking action a in state s, $0 < \gamma < 1$ is a discount factor, and $V(s,a)$ is the future rewards that M-Qubed receives in subsequent states after taking action a in state s.

While discounting future rewards (5) is somewhat inconsistent with the goal of maximizing payoffs over time, M-Qubed uses reward discounting for two reasons. First, reward discounting offers a mathematically convenient method for estimating $Q(s,a)$ when using standard reinforcement learning in an infinitely repeated game. High values of the discount factor γ give a reasonable, and finite, approximation of the utility of taking action a from state s. Second, given the uncertainty of future payoffs caused by learning associates, M-Qubed uses reward discounting to mitigate the effects of inaccurate estimates of future rewards while maintaining a long-term perspective. In single-agent reinforcement learning, this is unnecessary since future rewards will necessarily converge given appropriate learning and exploration rates. However, in the multi-agent case, the strategies of learning associates can and do change over time, meaning that future rewards remain uncertain. Future work should explore other mechanisms for dealing with this uncertainty.

3.2 Q-updates

As in standard reinforcement learning (Sutton and Barto 1998), M-Qubed estimates its future discounted reward by iteratively updating Q-estimates after each stage of the game:

$$Q^{t+1}(s^t, a_i^t) = Q^t(s^t, a_i^t) + \alpha\left[r_i^t + \gamma V^t(s^{t+1}) - Q^t(s^t, a_i^t)\right], \tag{6}$$

where α is the agent's learning rate and $V^t(s^{t+1})$ is the estimated value of being in state s^{t+1}. Watkins and Dayan (1992) showed that, when

$$V^t(s) = \max_a Q(s,a), \tag{7}$$

$Q^t(s,a)$ is guaranteed to approach its true Q-values as $t \to \infty$, provided that each state-action pair is visited infinitely often, the learning rate α is decreased appropriately, and the environment is stationary. However, since matrix games between learning agents effectually produce non-stationary environments, these results do not hold in repeated games.

Several alternatives for estimating $V^t(s)$ have been proposed to extend Q-learning to repeated games. In these algorithms, $V^t(s)$ is estimated using game theoretic concepts, including minimax (Littman 1994), NE (Hu and Wellman 1998; Bowling 2000), correlated equilibria (Aumann 1974; Greenwald and Hall 2003), and the NBS (Qiao et al. 2006). However, each of these methods requires knowledge of associates' payoffs and assumes that associates will play as expected. Furthermore, while sometimes successful, none of these approaches have demonstrated successful behavior across the range of general-sum games.

As in Sarsa (Rummery and Niranjan 1994), M-Qubed uses an on-policy method for estimating $V^t(s)$:

$$V^t(s) = \sum_{a \in A_i} \pi_i^t(s,a) Q^t(s,a), \tag{8}$$

where $\pi_i^t(s,a)$ is the probability at time t that player i takes action a from state s. We chose this formulation to better deal with the need to play mixed strategies in repeated games.

3.3 Strategy selection

M-Qubed uses its Q-estimates to determine how to bias its behavior toward actions that produce the highest payoffs given its experience (best-response learning bias). However, since its Q-estimates are not guaranteed to reflect actual payoffs, M-Qubed also encodes a cautious learning bias which encourages it to play its maximin strategy. To balance these two competing objectives, M-Qubed probabilistically chooses between these two learning biases in an attempt to learn good compromises. Lastly, to help avoid convergence to myopic solutions, M-Qubed encodes an optimistic learning bias that encourages it to play solutions that yield profitable, but potentially risky, solutions. We describe each of these mechanisms in detail.

3.3.1 Encoding a best-response learning bias

In standard reinforcement learning, an agent plays the pure strategy corresponding to its highest Q-value in the current state (with some exploration):

$$a_i^t = \pi_i^* = \arg\max_{a \in A_i} Q^t(s^t, a). \tag{9}$$

However, in repeated games, an agent's best strategy is sometimes a mixed strategy. For example, in many games, the maximin strategy π_i^{MM} is a mixed strategy. As such, M-Qubed must have the ability to play mixed strategies. Unfortunately, while sometimes necessary, using mixed strategies can make it difficult to strike profitable compromises in some important repeated games, since stochastic behavior can portray unclear signals (Axelrod 1984; Skyrms 2004).

Thus, when the competitive nature of the game is unknown a priori, an effective learning algorithm for repeated general-sum games must learn whether to play pure or mixed strategies. Importantly, an agent should learn to act predictably when associates are apt to coordinate and cooperate, and unpredictably in competitive situations. While many algorithms choose to do this by learning along the continuum of mixed strategies (Zinkevich 2003; Bowling and Veloso 2002; Abdallah and Lesser 2008), M-Qubed only considers playing its maximin strategy π_i^{MM} and each of its pure strategies $a \in A_i$. Specifically, M-Qubed encodes the following *best-response strategy rule*:

$$\overline{\pi}_i^*(s) \leftarrow \begin{cases} \arg\max_{a \in A_i} Q^t(s,a) & \text{if } \max_{a \in A_i} Q^t(s,a_i) > \frac{v_i^{\mathrm{MM}}}{1-\gamma}, \\ \pi_i^{\mathrm{MM}} & \text{otherwise.} \end{cases} \tag{10}$$

In words, this strategy rule selects a pure strategy if its highest Q-value in the current state meets or exceeds the discounted sum of v_i^{MM} (given by $\sum_{t=0}^{\infty} \gamma^t v_i^{\mathrm{MM}} = \frac{v_i^{\mathrm{MM}}}{1-\gamma}$). Otherwise, M-Qubed reasons that its maximin strategy will produce higher payoffs, so it selects π_i^{MM}.

3.3.2 Encoding a cautious learning bias

One fault of the best-response strategy rule is that, when associates learn, Q-estimates are not guaranteed to converge to actual payoffs. This means that an agent using the best-response strategy rule can potentially be exploited and is not secure. To ensure security, M-Qubed employs an alternate strategy rule that does not depend on its utility estimates.

Let $L_i^{\text{accum}}(t)$ be player i's *accumulated loss* up to time t, where loss is defined as how much its payoffs fall short of the maximin value v_i^{MM}. Formally, let

$$L_i^{\text{accum}}(t) = \max\left(0, t[v_i^{\text{MM}} - v_i^{\text{avg}}(t)]\right) \tag{11}$$

be player i's accumulated loss up to time t. Also, let L_i^{tol} be player i's *risk tolerance*, or the amount of accumulated loss that player i is willing to risk or tolerate. Then, M-Qubed encodes the following *cautious strategy rule*:

$$\underline{\pi}_i^*(s) \leftarrow \begin{cases} \arg\max_{a \in A_i} Q^t(s,a) & \text{if } \forall \tau \leq t,\, L_i^{\text{accum}}(\tau) < L_i^{\text{tol}}, \\ \pi_i^{\text{MM}} & \text{otherwise.} \end{cases} \tag{12}$$

In words, this strategy rule selects a best response from among its pure strategies until M-Qubed's accumulated loss L_i^{accum} reaches its risk tolerance L_i^{tol}. If M-Qubed's risk tolerance is ever exceeded, this strategy rule specifies that M-Qubed play its maximin strategy π_i^{MM} in all stages thereafter.

While guaranteeing security, the cautious strategy rule can reduce an agent's opportunities to pursue more profitable compromises. For example, when average payoffs in early stages of the game are below v_i^{MM}, the cautious strategy rule causes an agent to play π_i^{MM}, even when other strategies would eventually produce higher payoffs as associates learned. Thus, by encoding this cautious learning bias, M-Qubed meets the security requirement, but reduces its capabilities with respect to the coordination and cooperation properties.

3.3.3 Balancing the best-response and cautious learning biases

In repeated games, the best-response strategy rule encourages the player to maximize its payoffs, while the cautious strategy rule allows a player to be secure. M-Qubed achieves a balance between these learning biases by probabilistically selecting between the best response strategy rule $\overline{\pi}_i^*(s)$ and the cautious strategy rule $\underline{\pi}_i^*(s)$, and updating this probability after each stage of the game based on which strategy rule is more successful. Let β_i^t be the probability that player i uses the cautious strategy rule $\underline{\pi}_i^*(s)$ in stage t, and let $1 - \beta_i^t$ be the probability that it uses the best-response strategy rule $\overline{\pi}_i^*(s)$ in stage t. Then, M-Qubed's strategy in state s is (without exploration):

$$\pi_i^*(s) = \beta_i^t \underline{\pi}_i^*(s) + (1 - \beta_i^t)\overline{\pi}_i^*(s). \tag{13}$$

The probability β_i^t is based on the ratio of player i's *accumulated loss* L_i^{accum} to its risk tolerance L_i^{tol}, and is given by

$$\beta_i^t \leftarrow \begin{cases} 1 & \text{if } \exists \tau \leq t \text{ such that } L_i^{\text{accum}}(\tau) \geq L_i^{\text{tol}}, \\ (\frac{L_i^{\text{accum}}(t)}{L_i^{\text{tol}}})^{10} & \text{otherwise.} \end{cases} \tag{14}$$

To encourage coordination and cooperation in early stages of the game, the ratio $\frac{L_i^{\text{accum}}(t)}{L_i^{\text{tol}}}$ is (heuristically) raised to the power of ten to bias the agent toward the best-response strategy rule $\overline{\pi}_i(s)$. Raising this ratio to the power of ten works sufficiently well for our purposes, but other values could be explored in future work.

Thus, the probability β_i^t is used to balance the cautious and best-response learning biases based on a tolerance to risk. For M-Qubed, risk is measured as the difference between the maximin value and the average payoff. This encodes how much loss the agent will tolerate while seeking to learn a profitable compromise. The variable L_i^{tol} allows the designer to specify how much risk M-Qubed should take. Lower values of L_i^{tol} specify less risk, but provide less time for M-Qubed to learn a more profitable compromise.

Balancing the best-response and cautious learning biases in this way can potentially limit M-Qubed's ability to learn to play a best response. Since the cautious learning biases kicks in as $L_i^{\mathrm{accum}}(t)$ approaches L_i^{tol} and the player's average payoff is below the maximin value, a finite sequence of exploratory actions could cause M-Qubed to converge to its maximin strategy even when other strategies would be more profitable. Thus, M-Qubed is not guaranteed to play a best-response against a stationary opponent. However, the probability that the cautious learning bias will prohibit M-Qubed from learning a best response against a stationary associate becomes vanishingly small as L_i^{tol} is increased. Additionally, as long as L_i^{tol} is finite, M-Qubed is secure.

We provide this latter result as a theorem in the next section. Before doing so, however, we discuss how M-Qubed encodes its optimistic learning bias. This learning bias encourages the agent to seek for outcomes with high potential future discounted rewards, even when these outcomes carry considerable risk.

3.3.4 Encoding an optimistic learning bias

Since M-Qubed uses the previous ω joint-actions as its state, its best-response learning bias causes it to learn to play strategies that lead to future joint-action histories (i.e., future states) that produce high Q-values. However, since the strategies of learning associates continue to change, the value of future states are likely to be inaccurate until all of the player's strategies converge. The result is that the best-response mechanism tends to produce myopic strategies, strategies that produce high instantaneous rewards but not necessarily high future rewards. Convergence to optimal strategies with respect to the long-term behavior of learning associates is not guaranteed, nor is it necessarily even likely.

M-Qubed implements a mechanism to encode an optimistic learning bias to help overcome this problem. Like (Brafman and Tennenholtz 2003), M-Qubed sets its initial Q-values to their highest possible discounted reward ($\frac{1}{1-\gamma}$). Coupled with traditional Q-updates ((6) and (8)) and recurrent state, this creates a bias for actions that lead to states with high Q-values. Thus, M-Qubed is biased to play actions that lead to states (i.e., joint-action histories) that either have produced high payoffs in the past (best-response learning bias) or have not been visited often in the past (optimistic learning bias). Effectually, this produces a relaxation search, similar to the relaxation searches encoded by satisficing learning (Stimpson et al. 2001), in which M-Qubed's Q-values descend over time until it finds a strategy that yields an expected discounted reward at least as high as its highest Q-value over all states. In subsequent sections, we show that this learning bias allows M-Qubed to learn to play highly successful, non-myopic solutions that algorithms with straight-forward best-response learning biases do not.

The optimistic learning bias produced by setting initial Q-values high affects M-Qubed's strategy selection until Q-estimates relax to realistic levels. However, if the agent's losses during this relaxation search exceed the agent's risk tolerance L_i^{tol}, then the optimistic and best-response learning biases are overridden by the cautious learning bias (so the agent begins to play its maximin strategy). Thus, to adequately balance the optimistic and cautious learning biases, the learning rate α and the loss tolerance L_i^{tol} must be set so that M-Qubed can adequately explore various compromises while minimizing its losses. A heuristic for setting the learning rate α to achieve this balance and other parameter heuristics are discussed in the rest of this section.

3.4 Algorithm summary and parameter heuristics

The complete M-Qubed algorithm is summarized in Table 3. The remainder of this section discusses heuristics for setting the learning rate α and for determining how M-Qubed explores. These treatments are sufficient given the scope of this paper; future work should consider whether alternatives to these heuristics would improve behavior.

3.4.1 Learning rate

The success of M-Qubed depends in large part on its learning rate α. To effectively balance its optimistic and cautious learning biases, the agent must learn slowly enough to thoroughly explore potential compromises, but fast enough to keep potential losses low. M-Qubed attempts to set its learning rate so that, under exploitation, its Q-estimates fall to the discounted sum of its maximin value $\frac{v_i^{\text{MM}}}{1-\gamma}$ as its accumulated loss L_i^{accum} approaches its risk tolerance L_i^{tol}. Using the approximations discussed in Appendix A, this is achieved with

$$\alpha = \frac{1 - \left(\frac{\zeta}{1 - v_i^{\text{MM}} + \zeta}\right)^{\frac{\zeta |A_i| |H(\omega)|}{L_i^{\text{tol}}}}}{1 - \gamma} \tag{15}$$

where $\zeta > 0$ is the small positive constant defined in Appendix A. In this way, given the discount factor γ and the history length ω, the balance among the cautious, best-response, and optimistic learning biases is controlled by the variable L_i^{tol}.

Finally, we note that the convergence of Q-values in stationary environments requires that the learning rate α decays slowly, but not too slowly (Watkins and Dayan 1992). However, in multi-agent environments, where convergence of Q-values is not guaranteed anyway, we do not find it useful to decrease the learning rate over time. Thus, M-Qubed maintains a constant (but small) learning rate α so that it can continue to learn as associates learn.

Table 3 The M-Qubed algorithm

```
(1) Select $\alpha$ (according to (15)), $\gamma$, and $\eta$
(2) $\forall s \in H(\omega)$ and $a \in A_i$, set $\pi_i(s, a) \leftarrow \frac{1}{|A_i|}$, $Q(s, a) \leftarrow \frac{1}{1-\gamma}$
(3) Set $t \leftarrow 1$
(4) Repeat,
    (a) Take action $a$ according to $\pi_i(s^t)$
    (b) Observe reward $r_i^t$ and joint action $\mathbf{a}^t$:
          (i) Determine $s^{t+1}$
         (ii) Update $Q(s^t, a)$ using (6) and (8)
        (iii) Update $\pi_i(s^t)$ using (16)
    (c) $t \leftarrow t + 1$
```

3.4.2 Exploration

Another design consideration in reinforcement learning algorithms is the trade-off between exploration and exploitation (Kaelbling et al. 1996). In multi-agent environments, this trade-off can be particularly important (de Farias and Megiddo 2005). In addition to the exploration caused by setting initial Q-values high, M-Qubed uses a form of ε-greedy exploration (Sutton and Barto 1998) when it has not recently visited the state with the highest Q-value (or close thereto) and when its risk tolerance has not been exceeded (i.e., $\beta_i^t < 1$). This helps M-Qubed to avoid convergence to myopic, locally optimal, strategies.

Let $S^* \subseteq H(\omega)$ denote the set of states that have a Q-value within a small margin of the highest Q-value over all states $s \in H(\omega)$, and let $S^{\text{prev}} \subseteq H(\omega)$ be the set of states that M-Qubed has visited in the last $|H(\omega)|$ stages of the game. Then, $S^* \cap S^{\text{prev}}$ is the set of states that M-Qubed has recently visited that have high Q-values. When this set is empty, M-Qubed reasons that its current strategies, along with the strategies of associates, are keeping it in a local maxima. In such situations, it explores with a small probability in attempt to improve its future payoffs.

Thus, with exploration, M-Qubed's strategy in state s is given by:

$$\pi_i^t(s) \leftarrow \begin{cases} \pi_i^*(s), & \text{if } \beta_i^t = 1 \text{ or } S^* \cap S^{\text{prev}} \neq \emptyset, \\ [1-\eta(t)]\pi_i^*(s) + \eta(t)\chi_i, & \text{otherwise}, \end{cases} \tag{16}$$

where χ_i denotes the uniform probability distribution over the action set A_i, and $\eta(t) \in [0, 1)$ is an exploration rate set so that $\eta(t) \to 0$ as $t \to \infty$. In words, M-Qubed randomly selects actions with probability $\eta(t)$ if it has cause to believe that its current strategies are causing it to converge to local maxima, and uses (13) otherwise.

4 Properties of M-Qubed

To demonstrate the usefulness of encoding and balancing cautious, best-response, and optimistic learning biases in repeated general-sum games, we begin to analyze M-Qubed with respect to the security, coordination, and cooperation properties in this section. We first prove that M-Qubed is secure in all general-sum matrix games. We then demonstrate empirically that, in self play, M-Qubed reaches $\ell_i = 1$ levels of compromise in many conflicting- and common-interest games. In the next section, we analyze M-Qubed's ability to compete, coordinate, and cooperate when associating with 16 representative algorithms from the literature.

4.1 Security

Recall that M-Qubed encodes a cautious learning bias that "kicks in" as its losses $L_i^{\text{accum}}(t)$ approach its risk tolerance L_i^{tol}. This learning bias makes it secure in all general-sum matrix games, regardless of how its associates behave. We establish this as a theorem.

Theorem 1 *M-Qubed is secure in all repeated general-sum matrix games.*

Proof We have two cases: either $L_i^{\text{accum}}(t) < L_i^{\text{tol}}$ for all t, or $\exists t$ such that $L_i^{\text{accum}}(t) \geq L_i^{\text{tol}}$. In the first case, M-Qubed's losses are bounded by L_i^{tol}. In the second case, since M-Qubed plays its maximin strategy π_i^{MM} with probability one once $L_i^{\text{accum}}(t) \geq L_i^{\text{tol}}$ (16), its expected

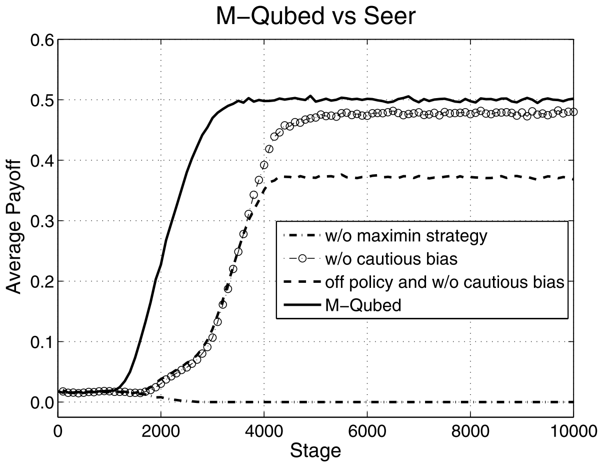


Fig. 1 Payoffs, averaged over 200 trials, of M-Qubed and variants against a seer in *Rock, Paper, Scissors* (Table 4(l))

accumulated loss thereafter is less than or equal to zero. Thus, M-Qubed's expected losses in this case are bounded by $L_i^{\text{tol}} + v_i^{\text{MM}}$. In either case, $E[v_i^{\text{avg}}(t)] \geq v_i^{\text{MM}} - \frac{L_i^{\text{tol}}+v_i^{\text{MM}}}{t}$. Since $\frac{L_i^{\text{tol}}+v_i^{\text{MM}}}{t} \to 0$ as $t \to \infty$, M-Qubed is secure. □

As an illustration of Theorem 1, Fig. 1 shows M-Qubed's payoffs over time in *Rock, Paper, Scissors* (Table 4(a)) against a *seer* agent, which has full knowledge of M-Qubed's strategy $\pi_i^t(s)$. The figure shows that M-Qubed's average payoffs are initially close to zero (the lowest payoff) in each stage. However, once M-Qubed's cautious strategy rule begins to take effect, its payoffs improve until its average payoff reaches $v_i^{\text{MM}} = 0.5$ after approximately 3,500 stages. A lower loss tolerance L_i^{tol} (parameters were set according to Table 7 in Appendix B) would decrease the amount of time it takes M-Qubed to converge to its maximin strategy.

Figure 1 also shows the performance of three variants of the M-Qubed algorithm against the seer agent in this game. The first variant, labeled *w/o maximin strategy*, does not encode its maximin solution; it plays pure strategies as given by (9) rather than select its strategy using (13). As such, the seer exploits it for the duration of the repeated game. The second variant, labeled *w/o cautious bias*, does not encode the cautious learning bias. Rather, it uses the best-response strategy rule defined in (10) in place of (13) to determine $\pi_i^*(s)$. While this agent learns to play its maximin strategy most of the time, its Q-estimates are sometimes inflated, causing it to play a pure strategy. In such cases, the seer exploits it, so its average payoffs fall short of the maximin value throughout the duration of the repeated game. The third variant, labeled *off-policy and w/o cautious bias*, is identical to the second variant, except that it estimates the value of its next state as in Q-learning (7) rather than Sarsa (8). This results in less accurate Q-estimates, which lead to the agent being exploited more frequently.

These results demonstrate that a reinforcement learning algorithm should encode a cautious learning bias in repeated games. Furthermore, when considering playing mixed strategies in repeated games, an on-policy rather than off-policy method should be used by reinforcement learners.

Table 4 Payoff matrices for selected matrix games. Payoffs are rounded to two decimal points

(a) Common interest game

	c	**d**
a	1.0, 1.0	0.0, 0.0
b	0.0, 0.0	0.5, 0.5

(b) Coordination game

	c	**d**
a	1.0, 0.5	0.0, 0.0
b	0.0, 0.0	0.5, 1.0

(c) Stag hunt

	c	**d**
a	1.0, 1.0	0.0, 0.75
b	0.75, 0.0	0.5, 0.5

(d) Tricky game

	c	**d**
a	0.0, 1.0	1.0, 0.67
b	0.33, 0.0	0.67, 0.33

(e) Prisoners' dilemma

	c	**d**
a	0.6, 0.6	0.0, 1.0
b	1.0, 0.0	0.2, 0.2

(f) Battle of the sexes

	c	**d**
a	0.0, 0.0	0.67, 1.0
b	1.0, 0.67	0.33, 0.33

(g) Chicken

	c	**d**
a	0.84, 0.84	0.33, 1.0
b	1.0, 0.33	0.0, 0.0

(h) Shapley's game

	d	**e**	**f**
a	0.0, 0.0	1.0, 0.0	0.0, 1.0
b	0.0, 1.0	0.0, 0.0	1.0, 0.0
c	1.0, 0.0	0.0, 1.0	0.0, 0.0

(i) Security game

	c	**d**
a	0.84, 0.33	0.84, 0.0
b	0.0, 1.0	1.0, 0.67

(j) Offset game

	c	**d**
a	0.0, 0.0	0.0, 1.0
b	1.0, 0.0	0.0, 0.0

(k) Matching pennies

	c	**d**
a	1.0, 0.0	0.0, 1.0
b	0.0, 1.0	1.0, 0.0

(l) Rock, paper, scissors

	d	**e**	**f**
a	0.5, 0.5	1.0, 0.0	0.0, 1.0
b	0.0, 1.0	0.5, 0.5	1.0, 0.0
c	1.0, 0.0	0.0, 1.0	0.5, 0.5

4.2 Cooperation and compromise in self play

Recall that M-Qubed attempts to establish profitable compromises by blending cautious, best-response, and optimistic learning biases. We now analyze how these learning biases promote coordination and cooperation. To do so, we begin to analyze M-Qubed in self play in the suite of two-player matrix games shown in Table 4(a)–4(j). These ten matrix games test an agent's ability to learn to coordinate actions, deal with risk, and share profits. We also evaluate M-Qubed in random two-player matrix games with between two and five actions. As a point of comparison in each game, we also analyze the performance of four other representative algorithms from the literature, namely Sarsa (with random initial Q-values), CoLF, GIGA-WoLF, and stochastic fictitious play (sFP), in self play. The parameter settings used by each of the algorithms are given in Appendix B.

Figure 2 shows the average level of compromise achieved by M-Qubed and the four other algorithms after 300,000 stages in the selected games. In nine out of ten of these games, M-Qubed's average level of compromise in self play is close to $\ell_i = 1$. Each of the other algorithms' levels of compromise over these ten games are statistically lower than the levels of compromise achieved by M-Qubed ($p < 0.001$). In particular, M-Qubed outperforms each of the other algorithms by a wide margin in the *Stag hunt* and the *Security game*. In

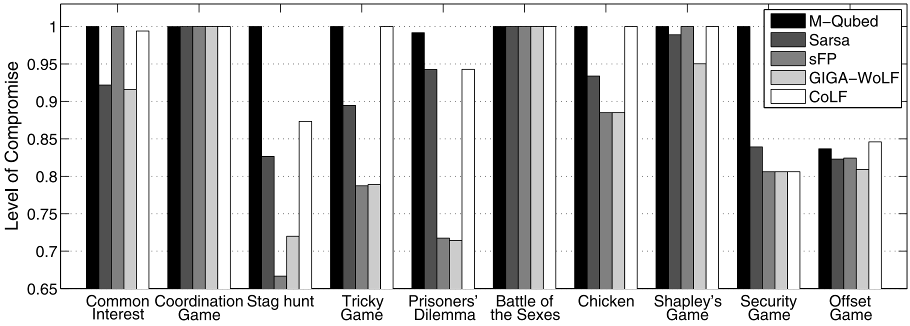


Fig. 2 Average asymptotic levels of compromise achieved in self play. Levels of compromise are measured as the average level of compromise achieved by the agents in the last 10,000 stage games

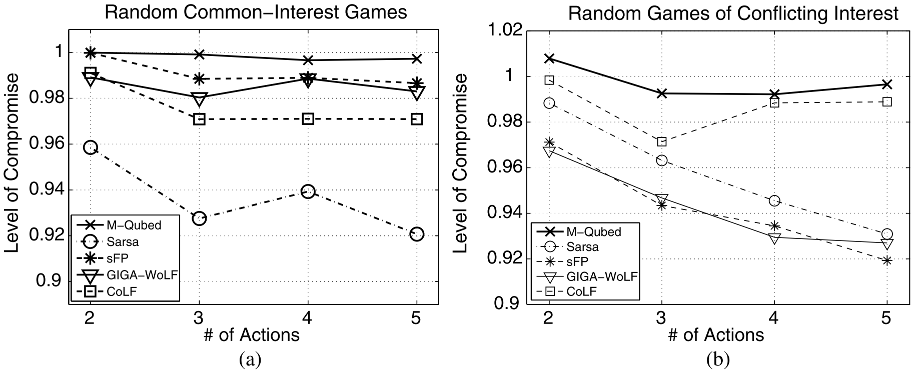


Fig. 3 Average levels of compromise achieved in self play in (**a**) randomly generated common-interest games and (**b**) randomly generated games of conflicting interest after 300,000 stages. Results are an average of 50 random games for each action size

each of these games, action shadowing causes the other algorithms to learn to play less risky, but less profitable, solutions.

M-Qubed's ability to coordinate and cooperate is further verified by its behavior in random games, as shown in Fig. 3. In both random common- and conflicting-interest games, M-Qubed's average level of compromise is near $\ell_i = 1$, and its average payoffs are higher than the other four algorithms in self play. Pairwise statistical comparisons show statistical differences between M-Qubed's performance and that of Sarsa, GIGA-WoLF, and CoLF in the random common-interest games ($p < 0.001$), but show no statistical difference between M-Qubed and sFP ($p = 0.357$). For the random conflicting-interest games, pairwise comparisons show statistical differences between M-Qubed and Sarsa, GIGA-WoLF, and sFP ($p < 0.001$), but not CoLF ($p = 0.592$). These results exemplify that, at least in self play, M-Qubed satisfies the coordination and cooperation properties in most two-player repeated general-sum games.

While M-Qubed is able to reach $\ell_i = 1$ levels of compromise in most two-player matrix games when its state is defined by a single joint action (i.e., $\omega = 1$), it does not do so in the *Offset game* (Fig. 2). In this game, both players depend on each other to receive any positive payoff. A reasonable compromise, which corresponds to the NBS of the game, is for the

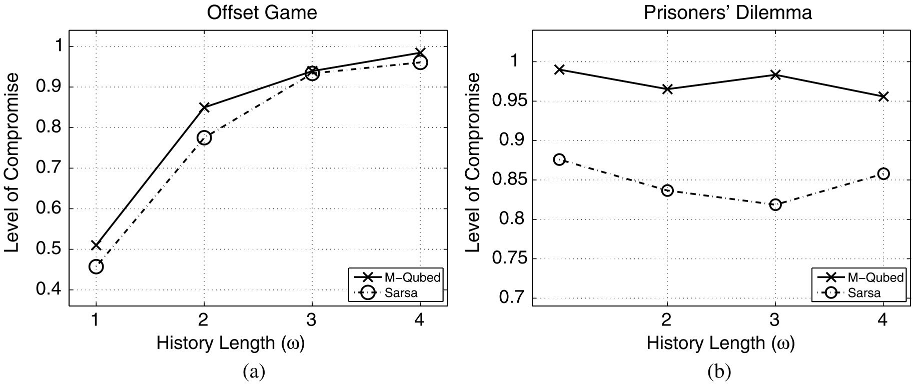


Fig. 4 Average asymptotic levels of compromise in self play given the history length ω in (**a**) the offset game and (**b**) the prisoners' dilemma

agents to take turns receiving the positive payoff. Unfortunately, dependencies in Q-values make it difficult for either algorithm to consistently learn this solution when $\omega = 1$. However, both M-Qubed and Sarsa learn to play the NBS in self play when ω is increased (Fig. 4a). Furthermore, increasing ω does not appear to substantially decrease M-Qubed's asymptotic payoffs in self play in other games (e.g., Fig. 4b), though increasing ω does increase the size of M-Qubed's state space exponentially, thus decreasing how fast M-Qubed learns.

4.3 What makes M-Qubed successful in self play?

The previous results demonstrate that M-Qubed learns to coordinate and cooperate in self play across a wide-range of repeated general-sum games. To better understand what makes M-Qubed successful in self play, we compare M-Qubed with five other algorithms that differ from M-Qubed in at least one way:

- *M-Qubed without an optimistic learning bias*—This variant does not encode the optimistic learning bias. Rather, it sets its initial Q-values randomly in the range $[0, \frac{1}{1-\gamma}]$.
- *M-Qubed without maximin strategy*—This variant does not encode the cautious learning bias or its maximin strategy. In this variant, π_i^* is given by (9) rather than (13).
- *M-Qubed without an exploration stopping rule*—This variant replaces (16) with:

  $$\pi_i^t(s) = \left[1 - \eta(t)\right]\pi_i^*(s) + \eta(t)\chi_i.$$

  Thus, rather than always playing a best response when current strategies appear to be effective, this variant continues to explore randomly with probability $\eta(t)$ in each stage of the game.
- *M-Qubed with decreasing α*—Recall that the decision to maintain a constant learning rate α was made to allow M-Qubed's beliefs (or Q-estimates) to continue to evolve in order to adapt to associates' changing behavior. This variant decreases its learning rate α over time in order to help Q-estimates converge.
- *Sarsa with optimistic learning bias*—This M-Qubed variant combines the differences of the previous three variants. In other words, it does not encode the maximin strategy, it does not encode M-Qubed's exploration stopping rule, and it decreases its learning rate α over time. It does, however, initialize its Q-values optimistically.

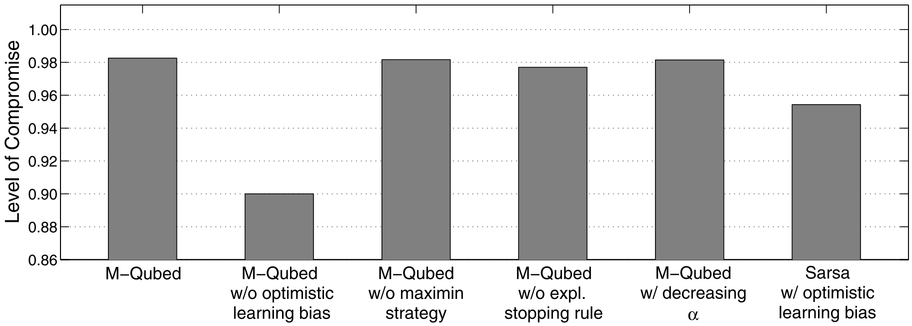


Fig. 5 Average levels of compromise over selected games for M-Qubed and five variants

The average levels of compromise achieved by M-Qubed and these five variants over the ten selected matrix games are shown in Fig. 5. The figure shows that, in self play, the variant that achieved the lowest average level of compromise was the one that did not encode the optimistic learning bias. Its level of compromise was statistically lower than M-Qubed's ($p < 0.001$). This shows that M-Qubed's ability to coordinate and cooperate in self play is significantly increased by its initial optimism coupled with its use of the previous joint actions to define state.

Figure 5 also indicates that, individually, M-Qubed's stopping rule, its constant learning rate, and its encoding of the maximin value did not positively or negatively impact the level of compromised it achieved in self play in common- and conflicting-interest games. However, *Sarsa with optimistic learning bias*, which is equivalent to M-Qubed except that it does not encode any of these three characteristics, achieved a slightly lower, but statistically significant ($p < 0.001$), level of compromise in self play in the selected games. This latter set of observations should not be given too much weight since these results are only for self play in which agents have similar parameter settings. We further evaluate the importance of a constant learning rate and stopping exploration when playing against other learning algorithms in Sect. 5.3, and find that they have little, if any, effect on M-Qubed's performance.

In summary, the results of this section suggest that the optimistic learning bias is necessary to learn to coordinate and cooperate in self play. This is similar to how the cautious learning bias appears to be necessary to ensure security. In the next section, we analyze M-Qubed's ability to learn good compromises when paired with other learning algorithms.

5 Associating with other learners

The goal of Sect. 4.2 was to provide preliminary evidence that M-Qubed could learn to make profitable compromises by balancing cautious, best-response, and optimistic learning biases. However, most of the results of that section are limited to self-play. We now analyze M-Qubed's behavior in a series of round-robin and evolutionary tournaments played with 16 other learning algorithms from the literature. In this section, we describe the design of these tournaments, and present and analyze their results.

5.1 Tournament descriptions

Following the lead of (Axelrod 1984), we evaluated M-Qubed's ability to make good compromises in two kinds of tournaments. First, we conducted a set of *round-robin tournaments* in which each of the 17 algorithms was paired with every other algorithm and itself in each game. Each pairing consisted of a repeated game lasting 300,000 stages. An algorithm's performance was measured as its average payoff per stage when paired with each of the 17 algorithms. To mitigate stochastic effects, each tournament was repeated 50 times.

The second set of tournaments we conducted were *evolutionary tournaments*. In these tournaments, a large population of agents, each employing one of the 17 algorithms, was evolved over a series of generations according to the fitness of the algorithms. In a generation, each agent in the population was randomly paired with another agent in a repeated game lasting 300,000 stages. Initially, each algorithm was equally represented in the population. The proportion of the population employing each algorithm was then altered in each subsequent generation using the replicator dynamic (Taylor and Jonker 1978). An algorithm's performance in these tournaments was measured as the percentage of the population employing that algorithm after 1000 generations. Appendix C describes these evolutionary tournaments in more detail.

The round-robin tournaments place equal weight on an algorithm's performance against all associates. Thus, an algorithm can be successful if it can find a way to exploit a few less successful algorithms. On the other hand, a successful algorithm in the evolutionary tournaments must perform well when associating with other successful algorithms, since successful algorithms will dominate the population in later generations.

We conducted both round-robin and evolutionary tournaments in each of the twelve games shown in Table 4. We also conducted both kinds of tournaments in random constant-sum games, random common-interest games, and random games of conflicting interests. Each set of random games consisted of 200 games with between two and five actions.

5.1.1 Algorithms and parameters

The 17 algorithms entered into the tournaments are listed in Table 7 in Appendix B. These algorithms were chosen based on the following criteria. First, each algorithm entered into the tournament appeared in a top conference or journal over the past couple of decades. Many of these algorithms have been repeatedly cited in the literature. Second, the algorithms were chosen to represent different schools of thought within the community, including belief-based algorithms, reinforcement learning algorithms, gradient-ascent algorithms, no-regret algorithms, and a satisficing learning algorithm. Third, none of the algorithms require knowledge of associates' payoffs.

The parameter values used by the various algorithms were selected to preserve the various schools of thought advocated by the authors of the algorithms and to ensure that each of the algorithms converged long before the completion of 300,000 stages. Where possible, we used the parameter settings indicated by the authors of the algorithms. Details concerning the parameter values used by each algorithm are given in Appendix B.

5.2 Results

5.2.1 Selected games

The results of both the round-robin and evolutionary tournaments in the twelve selected games are summarized in Table 5. While no algorithm was dominant in all games, M-Qubed

Table 5 Summary of tournament results for the twelve selected games. *Average round-robin payoffs* refer to the average payoff per stage against all associates. *Population after 1000 generations* refers to the average composition of the population after 1000 generations in the evolutionary tournament. Only the top three scorers are listed, except when M-Qubed did not place in the top three

Game	Average round-robin payoffs		Population after 1000 generations	
Common interest game	1. WMA	0.988	1. FP	19.3%
	2. Exp3	0.986	2. **M-Qubed**	**18.3%**
	3. **M-Qubed**	**0.984**	3. AP	16.6%
Coordination game	1. FP	0.938	1. FP	100%
	2. Hyper-Q	0.877		
	3. GIGA-WoLF	0.857		
	11. **M-Qubed**	**0.666**		
Stag hunt	1. S-alg	0.679	1. S-alg	66.3%
	2. Sarsa	0.642	2. **M-Qubed**	**33.4%**
	3. **M-Qubed**	**0.620**	3. CoLF	0.3%
Security game	1. **M-Qubed**	**0.617**	1. **M-Qubed**	**100%**
	2. Sarsa	0.607		
	3. S-alg	0.601		
Tricky game	1. **M-Qubed**	**0.737**	1. S-alg	51.8%
	2. CoLF	0.721	2. **M-Qubed**	**48.2%**
	3. Sarsa	0.719		
Offset game	1. FP	0.447	1. FP	100%
	2. GIGA	0.365		
	3. GIGA-WoLF	0.349		
	9. **M-Qubed**	**0.163**		
Prisoners' dilemma	1. **M-Qubed**	**0.301**	1. S-alg	98.6%
	2. Sarsa	0.297	2. **M-Qubed**	**1.4%**
	3. S-alg	0.294		
Battle of the sexes	1. Hyper-Q	0.941	1. Hyper-Q	97.2%
	2. FP	0.927	2. FP	2.8
	3. GIGA	0.911		
	10. **M-Qubed**	**0.800**		
Chicken	1. S-alg	0.920	1. S-alg	64.1%
	2. Exp3	0.886	2. **M-Qubed**	35.9
	3. WMA	0.853		
	8. **M-Qubed**	**0.763**		
Shapley's game	1. **M-Qubed**	**0.551**	1. **M-Qubed**	**100%**
	2. GIGA-WoLF	0.496		
	3. Sarsa	0.495		
Matching pennies	1. **M-Qubed**	**0.590**	1. **M-Qubed**	**38.9%**
	2. sFP	0.558	2. GIGA-WoLF	15.0%
	3. Sarsa	0.550	3. sFP	12.6%

Table 5 (*Continued*)

Game	Average round-robin payoffs		Population after 1000 generations	
Rock, paper, scissors	1. **M-Qubed**	**0.600**	1. **M-Qubed**	**34.8%**
	2. sFP	0.570	2. sFP	21.7%
	3. Sarsa	0.536	3. WPL	11.8%
Average in selected games	1. **M-Qubed**	**0.616**	1. **M-Qubed**	**34.2%**
	2. Sarsa	0.600	2. S-alg	24.5%
	3. CoLF	0.586	3. FP	18.5%

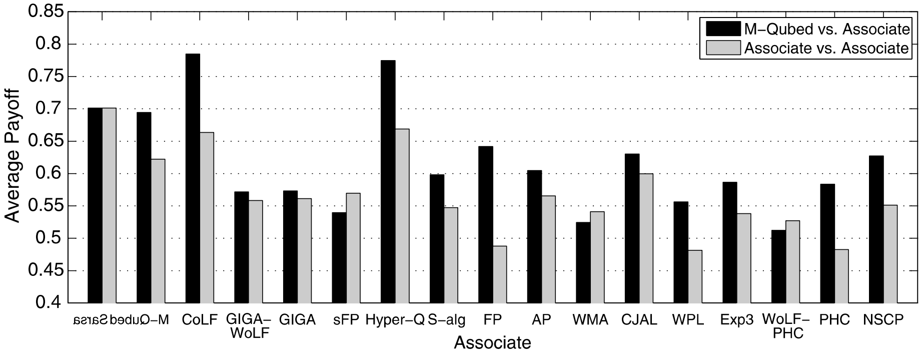


Fig. 6 M-Qubed's average payoffs across the twelve selected games (Table 4) played with each of the 17 algorithms. These payoffs are compared to associates' performance in self play. The algorithms are ordered so that the more successful algorithms are on the left

was the most successful learning algorithm across all twelve games in both the round-robin tournaments and the evolutionary tournaments. Averaged over all games, M-Qubed had the highest payoff per stage game (0.616) and held the largest population share after 1000 generations (34.2%). Pairwise statistical comparisons show a statistically significant difference between M-Qubed's payoffs across the selected games and the payoffs of each of the other algorithms ($p < 0.001$).

M-Qubed placed in the top three in eight out of the twelve round-robin tournaments, and nine of the twelve evolutionary tournaments. Furthermore, it performed well in each kind of game. In the common-interest game, M-Qubed finished third in the round-robin tournament, and second in the evolutionary tournament. In the constant-sum games (*Matching pennies* and *Rock, paper, scissors*), M-Qubed finished first in both tournaments. In games of conflicting interest, M-Qubed was the most consistent algorithm, scoring in the top three in five out of the nine round-robin tournaments, and six out of nine evolutionary tournaments.

M-Qubed performed particularly well in games that require agents to learn high-risk solutions. These games include the *Stag hunt*, the *Security game*, and the *Prisoners' dilemma*. M-Qubed's success in these games can be attributed to its ability to bound losses when associates are not apt to take risks, while learning mutually beneficial solutions when associates are willing to take risks.

On the other hand, M-Qubed performed the worst in games in which an agent can profit by "bullying" associates to play a particular solution. Two examples of these types of games

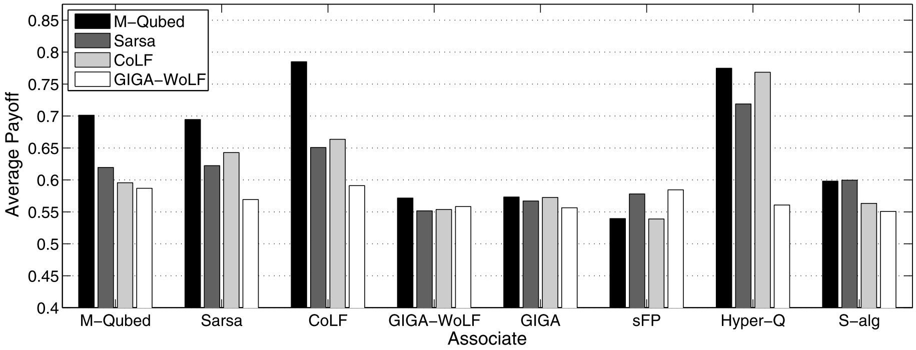


Fig. 7 Average payoffs of the four top-performing algorithms in selected games against other top performers

are the *Coordination game* and *Battle of the sexes*. In each case, the more successful algorithms were those that repeatedly played the action corresponding to their highest payoff.

M-Qubed's high average performance in both round-robin and evolutionary tournaments can be attributed in part to its ability to perform relatively well against both successful and unsuccessful algorithms. This is demonstrated in Fig. 6, which compares the average stage payoffs of each of the algorithms in self play, averaged over the twelve selected games, with M-Qubed's average stage payoff when it was paired with each algorithm. In the figure, the algorithms are sorted from left to right so that the more successful algorithms are on the left and less successful algorithms are on the right.

Figure 6 shows a couple of important trends. First, M-Qubed scored higher against 13 out of the 16 other algorithms than the algorithms performed against themselves. Only sFP, WMA, and WoLF-PHC performed slightly better in self play than M-Qubed performed against them. Second, M-Qubed performed very well against the top algorithms. As we reported previously, it performs very well in self play (its payoffs in self play are the highest among all algorithms), and it receives high payoffs when associating with the other top performing algorithms (Sarsa and CoLF).

This latter point is supported by Fig. 7, which shows the average performance of the four highest scoring algorithms (M-Qubed, Sarsa, CoLF, and GIGA-WoLF) against the eight highest scoring algorithms. The figure shows that M-Qubed's average payoff meets or exceeds the other three top performing algorithms' payoffs when associating with seven out of the eight top algorithms. Thus, when the population becomes more populated by successful algorithms in evolutionary tournaments, M-Qubed's performance stays high due to its ability to learn profitable compromises.

5.2.2 Random games

The previously presented results indicate that M-Qubed learns profitable compromises in a suite of repeated matrix games. To ensure that these results are representative of general-sum games, we also conducted tournaments in randomly generated games with between two and five actions. In particular, we conducted both round-robin and evolutionary tournaments in random common-interest games, random constant-sum games, and random games of conflicting interest.[5]

[5] Hyper-Q was not entered into these tournaments due to its computational inefficiencies in games with more than three actions.

Table 6 Average payoffs round-robin tournaments of random games with between two and five actions

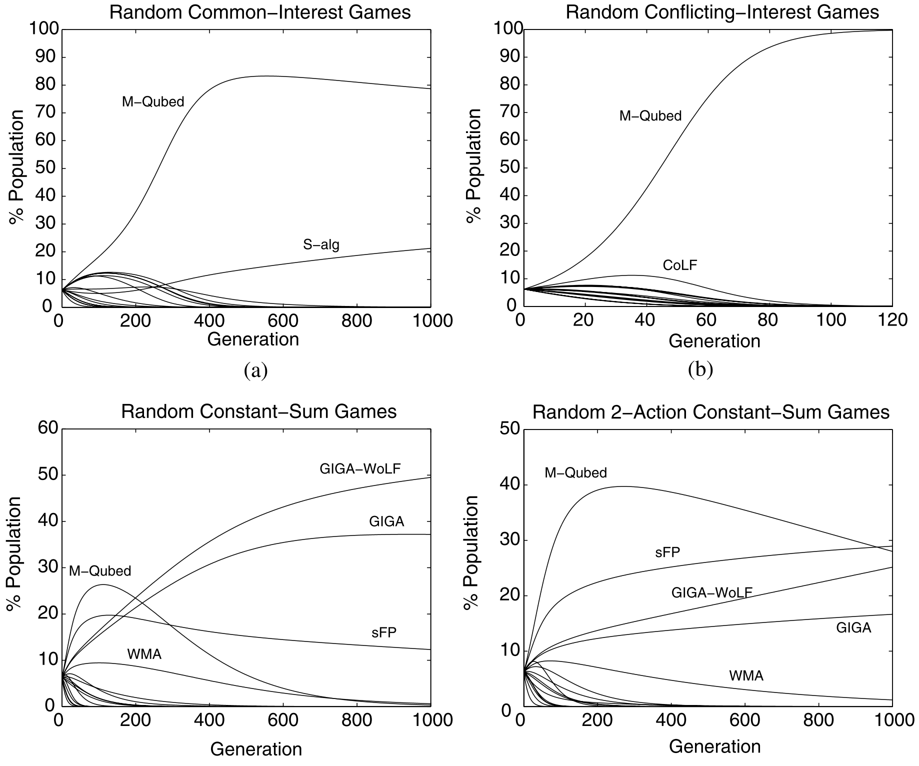
Common interest		Conflicting interest		Constant-sum	
1. M-Qubed	0.968	1. M-Qubed	0.828	1. M-Qubed	0.532
2. GIGA	0.965	2. CoLF	0.805	2. sFP	0.527
3. GIGA-WoLF	0.965	3. sFP	0.799	3. GIGA-WoLF	0.515
4. FP	0.965	4. AP	0.797	4. GIGA	0.514
5. sFP	0.964	5. GIGA	0.797	5. Sarsa	0.509
6. AP	0.963	6. GIGA-WoLF	0.796	6. WMA	0.509
7. WMA	0.957	7. FP	0.788	7. AP	0.507
8. CoLF	0.952	8. Sarsa	0.784	8. Exp3	0.503
9. S-alg	0.945	9. WMA	0.784	9. WPL	0.501
10. CJAL	0.944	10. S-alg	0.782	10. CoLF	0.500
11. Exp3	0.944	11. CJAL	0.776	11. WoLF-PHC	0.498
12. NSCP	0.936	12. Exp3	0.774	12. PHC	0.495
13. WPL	0.936	13. WPL	0.772	13. FP	0.488
14. PHC	0.930	14. NSCP	0.762	14. NSCP	0.474
15. WoLF-PHC	0.930	15. WoLF-PHC	0.761	15. S-alg	0.472
16. Sarsa	0.913	16. PHC	0.759	16. CJAL	0.453

The results of these round-robin tournaments, shown in Table 6, mirror the results from the selected games. M-Qubed's average payoffs were highest for all three sets of random games. Pairwise statistical comparisons show that M-Qubed's payoffs were statistically higher than each of the other algorithms in both conflicting-interest and constant-sum games ($p < 0.001$ in each case). In the random common-interest games, M-Qubed's payoffs were not statistically different from those of GIGA, GIGA-WoLF, FP, sFP, and AP, but they were statistically higher than the payoffs of the other algorithms.

Interestingly, the difference between M-Qubed and the next highest scoring algorithm was greatest in games of conflicting interest, in which learning good compromises is typically the most difficult. This can be traced to M-Qubed's ability to make good compromises with a wide-range of associates by balancing its cautious, best-response, and optimistic learning biases.

M-Qubed also gained the highest population share in random common- and conflicting-interest evolutionary tournaments (Fig. 8(a) and 8(b), respectively). In common-interest games, the ending population consisted of agents employing M-Qubed and the satisficing learning algorithm (S-alg). The population eventually converged so that approximately 72% of the population used M-Qubed and 28% of the population used the S-alg. In games of conflicting interest, M-Qubed quickly gained and held the entire population share.

The results of the random constant-sum evolutionary tournaments show a surprising and interesting trend (Fig. 8(c)). While M-Qubed initially began to dominate the population, its population share began to decrease after about 100 generations, eventually fading to zero by about 1000 generations. M-Qubed was initially successful in this tournament since its use of recurrent state allowed it to model and exploit insecure algorithms. However, when paired with some of the more successful algorithms in these games (such as GIGA and GIGA-WoLF), M-Qubed experienced a small constant lost, bounded by L_i^{tol}, over the duration of the repeated game. Thus, as the insecure algorithms became extinct and M-



(a) (b)

(c) (d)

Fig. 8 Population evolution in random games with between two and five actions

Qubed associated more frequently with these successful algorithms, M-Qubed's population share slowly dissolved.

These results stand in contrast to those observed in the games *Matching pennies* and *Rock, paper, scissors*, in which M-Qubed maintained a significant share of the population after 1000 generations (Table 5). The difference can be traced to M-Qubed's behavior as the number of actions in the game increases. Since M-Qubed increased its risk tolerance L_i^{tol} as the number of actions in the game increased in order to more effectively coordinate and cooperate (Table 7), it experienced greater loss against GIGA and GIGA-WoLF in games with more actions. This resulted in faster deterioration to its population share in the random constant-sum games. In two-action random constant-sum games, M-Qubed maintained a higher share of the population after 1000 generations (Fig. 8(d)).

These results suggest two things. First, the need to balance cautious, best-response, and optimistic learning biases comes at a cost. No algorithm is "optimal" in all situations (Axelrod 1984). When the competitive nature of the game is known, the way that the cautious, best-response, and optimistic learning biases are balanced should be adjusted. For example, in constant-sum games, it would have been desirable for M-Qubed to use a smaller risk tolerance L_i^{tol} to balance its learning biases, while in common- and conflicting-interest games, it is typically best for M-Qubed to use a high risk tolerance.

Second, while M-Qubed is successful in repeated general-sum games with two agents and only a few actions, it does not necessarily scale well to games with more players and actions. Thus, identifying ways to effectively scale reinforcement learning algorithms for large multi-agent environments is an important topic of future work. The results of this paper suggest that these future algorithms should encode and balance competing learning biases such as caution, best-response, and optimism.

5.3 What makes M-Qubed successful?

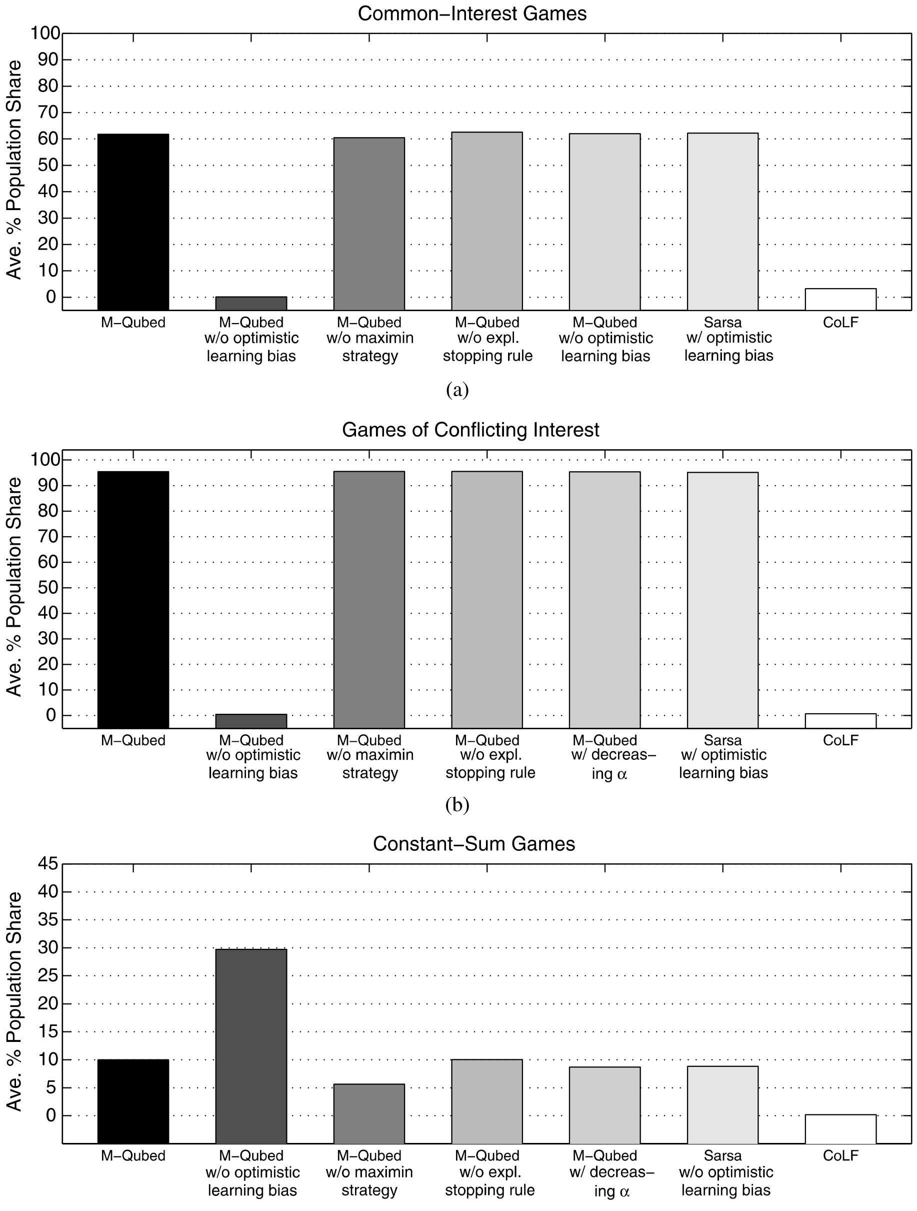
To better understand which of M-Qubed's characteristics cause it to learn successful compromises when associating with other learners, we compared M-Qubed with the five variants described in Sect. 4.3 in the random evolutionary tournaments. To do this, we ran separate evolutionary tournaments replacing M-Qubed with each of these variants. We then compared M-Qubed's average population share over 1000 generations with the average population share that each of the variants would have held. We also compare these average population shares with those of CoLF, which differs from *Sarsa with optimistic learning bias* only in that it uses two constant learning rates to update its Q-values.

Figure 9 shows that the optimistic learning bias had a large impact on M-Qubed's performance. If M-Qubed had not encoded an optimistic learning bias, its average population share would have been near zero in common- and conflicting-interest evolutionary tournaments (Figs. 9(a) and 9(b), respectively). This result indicates that optimism is a key component of coordination and compromise when associating with other learners in repeated games. On the other hand, the optimistic learning bias negatively affected M-Qubed's performance in constant-sum games (Fig. 9(c)), since being open to compromise exposed M-Qubed to additional risk. However, M-Qubed was still one of the more successful algorithms in these constant-sum games.

Figure 9 also shows that the cautious learning bias had a small effect on M-Qubed's performance in constant-sum games, as the absence of this learning bias cut M-Qubed's average population share in half in constant-sum games. However, the effect is not as dramatic as might be expected. In fact, a decreased learning rate (i.e., *Sarsa with optimistic learning bias*) nearly compensated for not encoding the maximin strategy. Thus, while an all-knowing seer agent is able to continually exploit an M-Qubed variant that does not encode the maximin strategy (Fig. 1), learning algorithms with knowledge restrictions similar to M-Qubed are often unable to exploit a reinforcement learning algorithm that does not encode the maximin strategy.

The best-response learning bias employed by M-Qubed is also an important part of M-Qubed's success when it associates with many kinds of learning algorithms. This is demonstrated by S-alg and CoLF, both of which encode similar optimistic learning biases. However, while both algorithms performed reasonably well in the tournaments, they were not as robust as M-Qubed. Given the S-alg's simple representation, it does not encode a best-response learning rule. As such, it is not as successful as M-Qubed in learning good compromises when paired with algorithms that are not apt to coordinate and cooperate. Likewise, the variable learning rate employed by CoLF, which is selected based on whether CoLF's payoffs change suddenly or not, causes its Q-values to be less likely to represent its true payoffs. This, diminishes CoLF's ability to play a best response, and results in near-zero average population shares in all three types of games (Fig. 9). Thus, when associating with many kinds of learning algorithms, the best-response learning bias has a large impact on an algorithms effectiveness.

Together these results suggest that encoding and balancing cautious, best-response, and optimistic learning biases is an effective method for learning good compromises in repeated general-sum games played with learning associates.



(a)

(b)

(c)

Fig. 9 Average population share over 1000 generations in (**a**) random common-interest games, (**b**) random games of conflicting interest, and (**c**) random constant-sum games

6 Conclusions and future work

In this paper, we have addressed the challenge of learning good compromises in repeated general-sum games in which an agent can observe the actions but not the payoffs of its associates. We have advocated that a successful learning algorithm for these games should have three properties of compromise. First, an algorithm should be secure, meaning that its average payoffs should meet or exceed its maximin value in the limit. Second, an algorithm should learn to coordinate its behavior in *common-interest* games so that it achieves its highest possible payoff. Third, in games of *conflicting interest*, an algorithm should learn to cooperate, meaning that its payoffs should approach or exceed its NBS value.

When the payoffs of associates are unknown, no algorithm in the literature satisfies all three of these properties. M-Qubed, the algorithm discussed and analyzed in this paper, encodes and balances cautious, optimistic, and best-response learning biases to take a step in this direction. M-Qubed is provably secure, regardless of the payoff structure of the game and the behavior of its associates. Furthermore, our empirical results show that M-Qubed achieves high degrees of coordination and cooperation when associating with itself and other learners in a wide-range of two-player repeated general-sum games. These attributes make M-Qubed a robust learning algorithm for these games, as attested by its robust performance in a series of round-robin and evolutionary tournaments involving 17 learning algorithms from the literature.

M-Qubed's robust learning behavior in these games requires the ability to encode and balance cautious, best-response, and optimistic learning biases. Each of these learning biases has been proposed and analyzed individually in the multi-agent literature over the last couple of decade. While individually useful in particular contexts, M-Qubed demonstrates that an effective combination of these learning biases allows an agent to overcome the conflict between competition and cooperation across the range of general-sum games played with a variety of (unknown) learning associates.

The empirical results presented in this paper are limited to two-player repeated matrix games with particular knowledge requirements and restrictions. Future work should explore how cautious, best-response, and optimistic learning biases should be encoded and balanced in other situations, including stochastic games and games with more players.

Acknowledgement We would like to thank two anonymous reviewers for invaluable suggestions and criticisms that greatly enhanced this paper.

Appendix A: Determining the learning rate α

To help ensure a proper balance among its best-response, cautious, and optimistic learning biases, M-Qubed should set its learning rate α so that $\max_a Q_i(s,a) \to \frac{v_i^{\text{MM}}}{1-\gamma}$ as $L_i^{\text{accum}}(t) \to L_i^{\text{tol}}$. While the way that Q-values change over time is also dependent on the actual payoffs received in the game, α can be estimated using three assumptions.

First, since an agent plays its maximin action π_i^{MM} in the cautious strategy rule only when its average payoff per stage ($v_i^{\text{avg}}(t) = \frac{1}{t}\sum_{\tau=1}^{t} r_i^{\tau}$) is less than its maximin value v_i^{MM}, we assume $r_i^t = v_i^{\text{MM}} - \zeta$, where $\zeta > 0$ is some small constant, for all t. Given this assumption and the requirement that M-Qubed's highest payoff is one and its lowest payoff is zero, we can reduce each payoff in M-Qubed's payoff matrix by $v_i^{\text{MM}} - \zeta$. In so doing, $v_i^{\text{MM}} = \zeta$ and the highest payoff is $1 - v_i^{\text{MM}} + \zeta$. Second, we assume that all state-action pairs are visited uniformly. Third, we assume that $V^t(s^{t+1}) \approx Q^t(s^t, a)$.

While these assumptions are likely to be violated, they provide a rough estimate for determining how Q-estimates will change over time. Specifically, given these assumptions and (6), we have that:

$$Q^t(s,a) = Q^0(s,a)(1-\alpha+\alpha\gamma)^{\frac{t}{|A_i||H(\omega)|}}. \tag{17}$$

We are interested in finding α such that $Q^t(s,a)$ falls to the discounted sum of the maximin value, which is $\frac{\zeta}{1-\gamma}$, when $L_i^{\text{accum}}(t) = L_i^{\text{tol}}$. Since $L_i^{\text{accum}}(t) = \zeta t$ under our assumptions, we have that $t = \frac{L_i^{\text{tol}}}{\zeta}$ and $Q^0(s,a) = \frac{1-v_i^{\text{MM}}+\zeta}{1-\gamma}$. Plugging these values into (17) and solving for α, we find that α should be set as follows:

$$\alpha = \frac{1 - \left(\frac{\zeta}{1-v_i^{\text{MM}}+\zeta}\right)^{\frac{\zeta|A_i||H(\omega)|}{L_i^{\text{tol}}}}}{1-\gamma}. \tag{18}$$

Appendix B: Algorithmic parameters

Unless specifically stated otherwise, the parameters used by the various algorithms in the empirical studies reported in this paper are given in Table 7. Given the importance of parameter settings on the performance of the algorithms, we describe our justifications for the various parameter settings that we selected. We first discuss M-Qubed's parameters, followed by a discussion of the parameters used for the other algorithms.

B.1 M-Qubed's parameters

The parameters used by M-Qubed are as follows:

1. γ—The discount factor, which determines the weight of future rewards on M-Qubed's Q-values. γ should be set high in repeated games to better approximate long-term future payoffs. For the experiments reported in this paper, we selected $\gamma = 0.95$ since it has empirically been shown to work well, though other high values are also effective (Crandall and Goodrich 2005).
2. η—The exploration rate, which determines how often M-Qubed acts randomly when it believes that further exploration could be profitable. We use a relatively low exploration rate η that decays over time. This worked well in practice, though other exploration rates can be used with success.
3. ω—The number of previous joint actions used to define M-Qubed's state. We used $\omega = 1$ since it is sufficient for most games, and allows M-Qubed to learn more quickly. However, increasing ω can increase M-Qubed's performance in some games substantially (Fig. 4).
4. L_i^{tol}—M-Qubed's loss tolerance, which defines how quickly M-Qubed triggers to its cautious learning bias. Higher values of L_i^{tol} cause M-Qubed to switch to its cautious learning bias more slowly when it is being exploited. We made L_i^{tol} a function of the size of the agent's state and action spaces to allow M-Qubed sufficient time to learn before it switched to its cautious learning bias.
5. α—The learning rate, which defines how quickly M-Qubed's Q-values change when exposed to new experiences. As outlined in Appendix A, the learning rate is determined by the parameters ζ and L_i^{tol}. As indicated in Appendix A, ζ should be a small positive constant. For the experiments reported in this paper, we randomly set ζ in the range $[0.05, 0.1]$ in each game. This range worked well in practice, but future work could better explore its affect on M-Qubed's performance.

Table 7 Algorithms and their parameter values, unless stated otherwise. $\kappa^t(s)$ is the number of times state s has been visited up to time t, and $\kappa^t(s,a)$ is the number of times that action a has been taken in state s up to time t

Algorithm name	Parameters
M-Qubed	$\gamma = 0.95$, $\eta(t) = \frac{(0.04)\cdot 1000}{1000+\max_s \kappa^t(s)}$, $\omega = 1$, $L_i^{\text{tol}} = 500 \cdot \lvert A_i\rvert \cdot \lvert H(\omega)\rvert$, α set as in (18) with $\zeta \in [0.05, 0.1]$
Sarsa (Rummery and Niranjan 1994)	$\alpha = \frac{1.0}{10.0+\frac{\kappa^t(s,a)}{100}}$, $\gamma = 0.95$, ϵ-greedy w/$\epsilon = \frac{(0.04)\cdot 1000}{1000+\max_s \kappa^t(s)}$; $Q(s,a) \in [0, \frac{1}{1-\gamma}]$ $\forall(s,a)$, state is the previous joint action ($\omega = 1$)
CoLF (de Cote et al. 2006)	$Q(s,a) = \frac{r_i^{\max}}{1-\gamma}$ for all (s,a), $\alpha_{NS} = 0.1$, $\alpha_S = 4 * \alpha_S$, $\gamma = 0.95$, $\lambda = 0.1$ state is the previous joint action ($\omega = 1$) ϵ-greedy w/$\epsilon = \max(0, 0.2 - 0.00006 * t)$;
GIGA-WoLF (Bowling 2005)	$\eta_i = \frac{1}{\sqrt{t}}$
GIGA (Zinkevich 2003)	$\eta_i = \frac{1}{\sqrt{t}}$
PHC (Bowling and Veloso 2002)	$\alpha = \frac{1}{100+\frac{t}{10000}}$, $\pi^{\text{MM}} = \frac{1}{20000+t}$, $Q(s,a) = 0$ for all (s,a) ε-greedy with 5% exploration stateless
WoLF-PHC (Bowling and Veloso 2002)	$\alpha = \frac{1}{100+\frac{t}{10000}}$, $\pi^{\text{MM}} = \pi_w^{\text{MM}} = \frac{1}{20000+t}$, $\pi_l^{\text{MM}} = 4\pi_w^{\text{MM}}$, $Q(s,a) = 0$ for all (s,a), ε-greedy with 5% exploration We used $\pi_l^{\text{MM}} = 4\pi_w^{\text{MM}}$ for increased learning speed
WMA or Hedge (Freund and Schapire 1995)	$\beta_i = 0.999$
Exp3 (Auer et al. 1995)	$\eta_i = \frac{\lambda}{\lvert A_i\rvert}$ ($\lvert A_i\rvert$ is the number of actions for player i), $g_i = 300000$
WPL (Abdallah and Lesser 2008)	$\eta_i = 0.002$, $\epsilon = 0.01$; with knowledge of the agent's payoff matrix, the gradient is determined by the payoff matrix rather than Q-values
FP—Fictitious Play (Fudenberg and Levine 1998)	$\kappa^0(a) = 0$ for all a, no exploration
sFP—Stochastic Fictitious Play (Fudenberg and Levine 1998)	$\kappa^0(a) = 0$ for all a. Boltzmann exploration with temperature parameter $\tau = \frac{1}{\frac{t}{100}}$
AP—Adaptive Play (Young 1993)	Uses 100 of the previous 200 actions to create the opponent model
CJAL (Banerjee and Sen 2007)	$N = 400$, $\epsilon = 0$

Table 7 (*Continued*)

Algorithm name	Parameters
Hyper-Q (Tesauro 2004)	Hyper-Q/Bayes formulation, with $\mu = 0.005$, and a uniform prior. Belief of opponent strategy formulated using previous $\frac{1}{\mu}$ actions. $Q(x, y)$ initialized to the expected payoff of playing strategy x against y. Discretization (for types): $N = 20$ for 2-action games, $N = 10$ for 3-action games (per dimension), $\alpha = \frac{1}{100+\frac{t}{10000}}$, ϵ-greedy with 1% exploration
NSCP (Weinberg and Rosenschein 2004)	$\alpha = \frac{1}{100+\frac{t}{10000}}$, $\beta = 0.95$, $Q(s, a_1, a_2) = 0$ for all (s, a_1, a_2) ϵ-greedy exploration with 5% exploration
S-alg (Stimpson and Goodrich 2003)	$\lambda = 0.99$

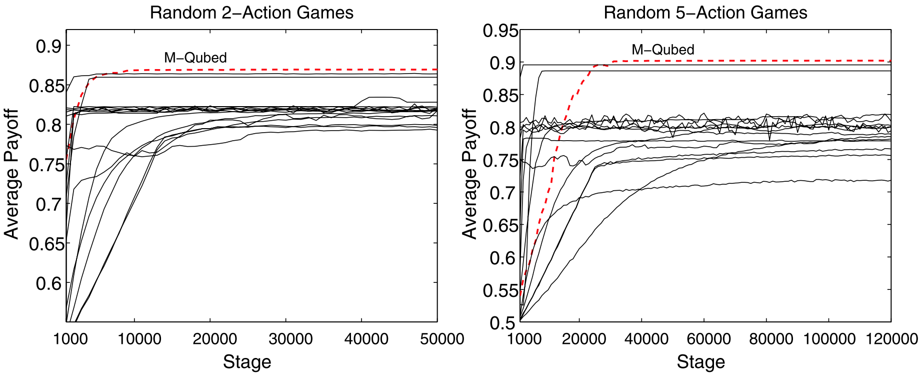


Fig. 10 Average payoffs of the various algorithms in self-play in (*left*) 50 two-action general-sum games and (*right*) 50 five-action general-sum games. After the time-window shown in the figures, none of the algorithms increased their performance substantially

B.2 Parameters chosen for other algorithms

We used two guiding principles to set the parameter values shown in Table 7. First, where possible, we used the parameter settings derived and used by the authors of the algorithms in the published works. This was done to preserve the basic philosophies advocated in the literature and to draw on the expertise of the creators of these algorithms.

Second, the parameters were set so that each algorithms' performance converged long before the completion of a repeated game (300,000 stages). Fig. 10 depicts the average learning speed of each of the 17 algorithms in self play in two- and five-action random general-sum matrix games. The figure shows that the performance of each of the algorithms converged long before 300,000 stages were completed. None of the algorithms increased its performance substantially after the time-windows shown in the figures.

While the performance of each algorithm plateaued long before the end of the repeated game, Fig. 10 shows that the algorithms did learn at different rates and reach different performance levels. This introduces the question of whether or not algorithms that converged faster performed better than algorithms that converged slower, or vice-versa. We conducted a Pearson correlation test to determine whether learning speed affected the average performance of the algorithms in round-robin tournaments. There was no statistically significant ($\alpha = 0.05$) correlation between convergence speed and average performance in the random conflicting-interest and random constant-sum games. However, there was a statistically significant trend in the random common-interest games ($r = -0.707$, $N = 17$, $p = 0.002$), as faster algorithms tended to outperform slower algorithms in these games.

We note that M-Qubed had the ninth fastest convergence speed in the random general-sum matrix games (Fig. 10).

Appendix C: Evolutionary tournaments

Formally, let Ω be the set of algorithms in the population, and let $P^\tau(o)$ be the percentage of the population employing algorithm $o \in \Omega$ in generation τ. Also let, $S(o_i, o_j)$ be the average payoff achieved by algorithm $o_i \in \Omega$ when playing $o_j \in \Omega$ in a 300,000-stage repeated game. Then, an evolutionary tournament proceeds as follows:

```
1. For all $o \in \Omega$, set $P^1(o) = \frac{1}{|\Omega|}$, and set $\tau = 1$
2. While ($\tau \leq 1000$)
3.     For all $o \in \Omega$, $u^\tau(o) = \sum_{o' \in \Omega} P^\tau(o') S(o, o')$
4.     For all $o \in \Omega$, $P^{\tau+1}(o) = P^\tau(o) \frac{u^\tau(o)}{\sum_{o' \in \Omega} u^\tau(o')}$
5.     $\tau = \tau + 1$
```

References

Abdallah, S., & Lesser, V. (2008). A multi-agent learning algorithm with non-linear dynamics. *Journal of Artificial Intelligence Research*, *33*, 521–549.

Auer, P., Cesa-Bianchi, N., Freund, Y., & Schapire, R. E. (1995). Gambling in a rigged casino: the adversarial multi-armed bandit problem. In *Proceedings of the 36th annual symposium on foundations of computer science* (pp. 322–331). Los Alamitos: IEEE Comput. Soc.

Aumann, R. J. (1974). Subjectivity and correlation in randomized strategies. *Journal of Mathematical Economics*, *1*, 67–96.

Axelrod, R. (1984). *The evolution of cooperation*. New York: Basic Books.

Banerjee, D., & Sen, S. (2007). Reaching pareto-optimality in prisoner's dilemma using conditional joint action learning. *Autonomous Agents and Multi-Agent Systems*, *15*, 91–108.

Borgs, C., Chayes, J., Immorlica, N., Kalai, A. T., Mirrokni, V., & Papadimitriou, C. (2008). The myth of the folk theorem. In *STOC'08: proceedings of the 40th annual symposium on theory of computing* (pp. 365–372). New York: ACM.

Bowling, M. (2000). Convergence problems of general-sum multiagent reinforcement learning. In *Proceedings of the 17th international conference on machine learning* (pp. 89–94).

Bowling, M. (2005). Convergence and no-regret in multiagent learning. In *Advances in neural information processing systems* (Vol. 17, pp. 209–216).

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

Brafman, R. I., & Tennenholtz, M. (2003). R-max—a general polynomial time algorithm for near-optimal reinforcement learning. *The Journal of Machine Learning Research*, *3*, 213–231.

Chang, Y.-H. (2007). No regrets about no-regret. *Artificial Intelligence*, *171*(7), 434–439.

Chang, Y.-H., & Kaelbling, L. P. (2005). Hedge learning: regret-minimization with learning experts. In *Proceedings of the 22nd international conference on machine learning* (pp. 121–128).

Chvatal, V. (1983). *Linear programming*. New York: Freeman.

Conitzer, V., & Sandholm, T. (2003). AWESOME: a general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. In *Proceedings of the 20th international conference on machine learning* (pp. 83–90).

Crandall, J. W. (2005). *Learning successful strategies in repeated general-sum games*. Ph.D. thesis, Brigham Young University, Provo, UT.

Crandall, J. W., & Goodrich, M. A. (2005). Learning to compete, compromise, and cooperate in repeated general-sum games. In *Proceedings of the 22nd international conference on machine learning*, Bonn, Germany (pp. 161–168).

de Cote, E. M., Lazaric, A., & Restelli, M. (2006). Learning to cooperate in multi-agent social dilemmas. In *Proc. of the 5th int. joint conf. on autonomous agents and multiagent systems* (pp. 783–785).

de Farias, D., & Megiddo, N. (2004). How to combine expert (or novice) advice when actions impact the environment. In *Advances in neural information processing systems* (Vol. 16).

de Farias, D., & Megiddo, N. (2005). Exploration–exploitation tradeoffs for expert algorithms in reactive environments. *Advances in Neural Information Processing Systems*, *17*, 409–416.

Foster, D. P., & Vohra, R. (1999). Regret in the on-line decision problem. *Games and Economic Behavior*, *29*, 7–35.

Frank, R. H. (1988). *Passions within reason: the strategic role of the emotions*. New York: Norton.

Freund, Y., & Schapire, R. E. (1995). A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the 2nd European conference on computational learning theory* (pp. 23–37). Los Alamitos: IEEE Comput. Soc.

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

Fulda, N., & Ventura, D. (2007). Predicting and preventing coordination problems in cooperative Q-learning systems. In *Proceedings of the 20th joint international conference on artificial intelligence*.

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

Greenwald, A., & Hall, K. (2003). Correlated Q-learning. In *Proceedings of the 20th international conference on machine learning* (pp. 242–249).

Hu, J., & Wellman, M. P. (1998). Multiagent reinforcement learning: theoretical framework and an algorithm. In *Proceedings of the 15th international conference on machine learning* (pp. 242–250).

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, *4*, 237–277.

Karandikar, R., Mookherjee, D., Ray, D., & Vega-Redondo, F. (1998). Evolving aspirations and cooperation. *Journal of Economic Theory*, *80*, 292–331.

Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th international conference on machine learning* (pp. 157–163).

Littman, M. L. (2001). Friend-or-foe: Q-learning in general-sum games. In *Proceedings of the 18th international conference on machine learning* (pp. 322–328).

Littman, M. L., & Stone, P. (2001). Leading best-response strategies in repeated games. In *IJCAI workshop on economic agents, models, and mechanisms*, Seattle, WA.

Littman, M. L., & Stone, P. (2005). A polynomial-time Nash equilibrium algorithm for repeated games. *Decision Support Systems*, *39*, 55–66.

Moody, J., Liu, Y., Saffell, M., & Youn, K. (2004). Stochastic direct reinforcement. In *AAAI spring symposium on artificial multiagent learning*, Washington, DC.

Nash, J. F. (1950). The bargaining problem. *Econometrica*, *28*, 155–162.

Powers, R., & Shoham, Y. (2005). Learning against opponents with bounded memory. In *Proceedings of the 19th international joint conference on artificial intelligence* (pp. 817–822).

Qiao, H., Rozenblit, J., Szidarovszky, F., & Yang, L. (2006). Multi-agent learning model with bargaining. In *Proceedings of the 2006 winter simulation conference* (pp. 934–940).

Rummery, G. A., & Niranjan, M. (1994). *On-line Q-learning using connectionist sytems* (Technical Report CUED/F-INFENG-TR 166). Cambridge University, UK.

Sandholm, T. W., & Crites, R. H. (1996). Multiagent reinforcement learning in the iterated prisoner's dilemma. *Biosystems*, *37*, 147–166.

Shapley, L. S. (1953). Stochastic games. *Proceedings of National Academy of Science*, *39*, 1095–1100.

Shoham, Y., Powers, R., & Grenager, T. (2007). If multi-agent learning is the answer, what is the question? *Artificial Intelligence*, *171*(7), 365–377.

Skyrms, B. (2004). *The stag hunt and the evolution of social structure*. Cambridge: Cambridge University Press.

Stimpson, J. R., & Goodrich, M. A. (2003). Learning to cooperate in a social dilemma: a satisficing approach to bargaining. In *Proceedings of the 20th international conference on machine learning* (pp. 728–735).

Stimpson, J. R., Goodrich, M. A., & Walters, L. C. (2001). Satisficing and learning cooperation in the prisoner's dilemma. In *Proceedings of the 17th international joint conference on artificial intelligence* (pp. 535–544).

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

Taylor, P. D., & Jonker, L. (1978). Evolutionarily stable strategies and game dynamics. *Mathematical Biosciences*, *40*, 145–156.

Tesauro, G. (2004). Extending Q-learning to general adaptive multi-agent systems. In *Advances in neural information processing systems* (Vol. 16). Cambridge: MIT Press.

Wang, X., & Sandholm, T. (2003). Reinforcement learning to play an optimal Nash equilibrium in team Markov games. *Advances in Neural Information Processing Systems*, *15*, 1571–1578.

Watkins, C. J., & Dayan, P. (1992). Q-learning. *Machine Learning*, *8*, 279–292.

Weinberg, M., & Rosenschein, J. S. (2004). Best-response multiagent learning in non-stationary environments. In *Proceedings of the third international joint conference on autonomous agents and multiagent systems*, Washington, DC, USA (pp. 506–513). Los Alamitos: IEEE Comput. Soc.

Young, H. P. (1993). The evolution of conventions. *Econometrica*, *61*(1), 57–84.

Zinkevich, M. (2003). Online convex programming and generalized infinitesimal gradient ascent. In *20th inter. conf. on machine learning* (pp. 228–236).