Journal of Artificial Intelligence Research 49 (2014) 111-142 Submitted 10/13; published 02/14

Towards Minimizing Disappointment in Repeated Games

Jacob W. Crandall JCRANDALL@MASDAR.AC.AE
Masdar Institute of Science and Technology
Abu Dhabi, United Arab Emirates

Abstract

We consider the problem of learning in repeated games against arbitrary associates. Specifically, we study the ability of expert algorithms to quickly learn effective strategies in repeated games, towards the ultimate goal of learning near-optimal behavior against any arbitrary associate within only a handful of interactions. Our contribution is three-fold. First, we advocate a new metric, called disappointment, for evaluating expert algorithms in repeated games. Unlike minimizing traditional notions of regret, minimizing disappointment in repeated games is equivalent to maximizing payoffs. Unfortunately, eliminating disappointment is impossible to guarantee in general. However, it is possible for an expert algorithm to quickly achieve low disappointment against many known classes of algorithms in many games. Second, we show that popular existing expert algorithms often fail to achieve low disappointment against a variety of associates, particularly in early rounds of the game. Finally, we describe a new meta-algorithm that can be applied to existing expert algorithms to substantially reduce disappointment in many two-player repeated games when associates follow various static, reinforcement learning, and expert algorithms.

1. Introduction

Many real-world environments require machines to interact repeatedly with other independent, self-interested entities, including both people and other machines. These finitely repeated interactions endure for unknown periods of time ranging from minutes, to hours, days, months, or even years. To be successful in these interactions, machines must employ algorithms that quickly learn good strategies against arbitrary (likely adaptive) associates.

Many algorithms for repeated games have been developed over the last several decades, including reinforcement learning algorithms (e.g., Watkins & Dayan, 1992; Littman, 1994, 2001; Bowling & Veloso, 2002; Greenwald & Hall, 2003; Crandall & Goodrich, 2011), opponent modeling algorithms (e.g., Fudenberg & Levine, 1998; Ganzfried & Sandholm, 2011), algorithms for computing desirable equilibria (e.g., Littman & Stone, 2005; Cote & Littman, 2008; Johanson, Bard, Lanctot, Gibson, & Bowling, 2012), and expert algorithms (e.g., Auer, Cesa-Bianchi, & Fischer, 2002; de Farias & Megiddo, 2004; Auer, Cesa-Bianchi, Freund, & Schapire, 1995). While sometimes successful, these algorithms typically have one or more of the following shortcomings which preclude their use. First, many of these algorithms learn very slowly. They achieve successful behavior only after thousands of interactions, even in simple games (e.g., Crandall & Goodrich, 2011). Second, existing algorithms are often myopic. They fail to learn profitable strategies in long-term interactions. Third, many algorithms are successful only against a limited set of associates.

Our long-term goal is to identify algorithms that learn near-optimal behavior against any arbitrary associate within only a handful of interactions. In this paper, we focus on the potential of expert algorithms to achieve this goal in two-player normal-form games. In each round, an expert algorithm selects an expert from a predefined set of experts to dictate the agent's behavior in that round. This algorithmic structure has several potential strengths. First, it offers a simple and flexible design process. One can create experts that perform well in specific scenarios that the agent is likely to encounter (such as being paired with a particular associate) without worrying that any one expert must perform well in all scenarios. The expert algorithm is responsible for finding the best expert for the specific scenario during run time. Second, experts can be as complicated as necessary. Though we focus on normal-form games in this paper, experts that compute complex equilibria or execute sophisticated algorithms can also be derived for stochastic and dynamics games. These two advantages give rise to a third potential advantage: expert algorithms have the potential to learn effective strategies quickly, particularly when experts employ precomputed strategies.

An expert algorithm is evaluated based on its ability to learn to select successful experts. Several metrics have been defined and used in the literature to measure the success of an expert algorithm in this process, perhaps the most popular of which is regret (Foster & Vohra, 1999; Greenwald & Jafari, 2003; Bowling, 2004; Gordon, Greenwald, & Marks, 2008). Loosely, an expert algorithm's (external) regret is the difference between the payoffs the agent would have received had it always followed its best expert *against its associates' observed actions* and the payoffs it actually received. In the context of repeated games, regret is a desirable notion since it provides a simple generalizable benchmark of success. Unfortunately, minimizing regret does not always correspond to maximizing payoffs. In this paper, we advocate an alternative metric, called disappointment, which is equivalent to maximizing payoffs while still providing a simple generalizable benchmark of success.

While there are many algorithms that are guaranteed to achieve *no regret* (e.g., Bowling, 2004; Foster & Vohra, 1999; Gordon et al., 2008), we show that it is impossible for an algorithm to be guaranteed to have *no disappointment* against an unknown associate. However, it is possible for an algorithm to quickly achieve and maintain low disappointment against classes of algorithms in many repeated games. We first evaluate the effectiveness of several existing expert algorithms to achieve low disappointment when paired with various (1) non-adapting, (2) reinforcement learning, and (3) expert algorithms in two-player games. Finally, we describe a new meta-algorithm for enhancing expert algorithms. We show that it substantially reduces the disappointment of these algorithms against these same associates in both short- and long-term interactions.

In Section 2, we discuss the evaluation of expert algorithms. In so doing, we define disappointment and establish several theoretical results, from which we derive a research agenda. In Section 3, we define a method for generating an effective set of experts for repeated normal-form games. We then evaluate the ability of existing expert algorithms to select effective experts across ten different repeated games in Section 4. We propose a meta-algorithm for enhancing expert algorithms in Section 5, and evaluate its effectiveness in Section 6. We conclude and reflect in Section 7.

2. Evaluating Expert Algorithms

In this section, we review existing evaluation metrics for repeated games, define and discuss disappointment, and compare it to existing metrics. Finally, we formally state our research agenda.

2.1 Notation and Terminology

We consider two-player repeated normal-form games, which consist of a set of joint actions $A = A_1 \times A_2$, where A_i is player (or agent) i's action set, and a payoff function $M : A \to \mathbb{R}^2$. In each round t, each agent i independently selects an action $a_i^t \in A_i$. The resulting joint action $\mathbf{a}^t = (a_1^t, a_2^t)$ produces the payoff pair $\big(m_1(\mathbf{a}^t), m_2(\mathbf{a}^t)\big)$, where $m_i(\mathbf{a}^t)$ is the payoff to agent i. For simplicity, we assume that, $\forall \mathbf{a} \in A, m_i(\mathbf{a}) \in [0, 1]$, $\max_{\mathbf{a} \in A} m_i(\mathbf{a}) = 1$, and $\min_{\mathbf{a} \in A} m_i(\mathbf{a}) = 0$. Play repeats an unknown number of rounds. We refer to the two agents as i and $-i$.

We use the terms *policy* and *strategy* to refer to how agents select their actions. Agent i's policy is a probability distribution over its action set A_i. This probability distribution specifies the probability that agent i selects each action. A strategy is a rule that defines an agent's policies in each state of the world. In the context of the experts used in this paper, world states are typically defined by the previous joint action played by the agents in the game. We use the notation $u_i(\pi_i, \pi_{-i})$ to denote agent i's expected payoff in a round in which it plays policy π_i and its associate plays π_{-i}. We use the notation $\mu_i(\rho_i, \rho_{-i})$ to denote agent i's average expected per-round payoff over time when it perpetually plays strategy ρ_i and its associate perpetually plays strategy ρ_{-i}.

We make several assumptions. First, we initially assume that the game is played with perfect information: both players know each other's payoff matrix. We later relax this assumption to account for occasions in which the players incorrectly assess their associate's payoffs. Second, we focus on general-sum repeated games, but do not specifically address constant-sum games. However, many of the concepts discussed herein also apply to constant-sum games.

In each round t, an expert algorithm employed by agent i selects an expert ϕ_i^t from a set of experts Φ_i. This expert then dictates the policy executed by agent i in round t. For simplicity in analysis, the literature often only considers experts that always play a single action or policy. We make no such assumption in this paper. Experts can also be sophisticated automata or learning algorithms.

2.2 Metrics

Currently, there is no universally accepted metric for evaluating how well expert algorithms select experts in repeated games. Existing metrics typically define a desirable *performance benchmark* an algorithm should achieve and then compare the payoffs obtained by algorithms to this benchmark. To be reliable indicators of success, such metrics should be *payoff comparable*.

Definition 2.1 (*Payoff comparable*): *Let $\mathcal{A}$ and $\mathcal{B}$ be two distinct algorithms, and let $\mu_{\mathcal{A}}^{o,M,T}$ and $\mu_{\mathcal{B}}^{o,M,T}$ denote the average per-round payoff obtained by $\mathcal{A}$ and $\mathcal{B}$, respectively, against associate o in a repeated game of length T with payoff matrix M. An evaluation metric is* payoff comparable *if, for any scenario defined by o, M, and T, it rates $\mathcal{A}$ higher than $\mathcal{B}$ if and only if $\mu_{\mathcal{A}}^{o,M,T} > \mu_{\mathcal{B}}^{o,M,T}$.*

In words, an evaluation metric is payoff comparable if and only if success as defined by the metric implies success with respect to maximizing the agent's average per-round payoff.

We seek an evaluation metric that (1) defines a generalizable (and desirable) performance benchmark and (2) is payoff comparable. We discuss several metrics with respect to these two attributes.

2.2.1 Regret

Regret has become a popular performance metric for evaluating the effectiveness of learning rules in repeated games. This notion has been re-discovered independently several times under various names (Foster & Vohra, 1999). Several forms of regret have been formulated, including external regret and internal (or swap) regret. For simplicity of argument, we focus on external regret.

Formally, agent i's total external regret after round T is

$$\mathcal{R}_i^T = \max_{\phi \in \Phi_i} \sum_{t=1}^{T} u_i^t(\phi, a_{-i}^t) - \sum_{t=1}^{T} m_i(\mathbf{a}^t), \tag{1}$$

where $u_i^t(\phi, a_{-i}^t)$ is agent i's expected payoff in round t if, in each round $\tau \in \{1, \cdots, t\}$, agent i were to have followed expert ϕ while agent $-i$ played its observed action a_{-i}^τ. Agent i's *average external regret* through round T is

$$\bar{\mathcal{R}}_i^T = \frac{\mathcal{R}_i^T}{T}. \tag{2}$$

$\bar{\mathcal{R}}_i^T \leq 0$ means that the expert algorithm has done at least as well as it would have done had it always followed its best expert *given that the associate would have still played its observed actions*. Agent i is said to have *no regret* when $\lim_{T\to\infty} \bar{\mathcal{R}}_i^T \leq 0$. When all agents use no-regret learning rules, play converges to correlated equilibria (Greenwald & Jafari, 2003; Gordon et al., 2008).

The performance benchmark (the estimated payoffs of its best expert) used to calculate regret is simple and generalizable to any scenario. However, regret is not payoff comparable. The assumption that agent i's behavior does not affect agent $-i$'s future actions is clearly violated when agent $-i$ executes a learning algorithm or even a simple automata. This limiting assumption means that regret minimization does not imply payoff maximization. In fact, as we demonstrate in Section 2.3, low regret sometimes strongly correlates with low payoffs.

Several alterations to regret have been made in attempt to alleviate its shortcomings. For example, Chang (2007) proposed a modified form of regret which considers multi-period strategies. This modification provides more effective evaluations against simple automata such as tit-for-tat (at the expense of increased computation complexity), but still does not address the general deficiency that regret minimization does not imply payoff maximization when one's associate learns. Alternatively, Bowling (2004) embraced regret as a minimum criterion despite its limitations, but advocated for more: an algorithm should also converge or achieve negative regret in self play. Though this addition makes the metric stronger, it does not make the metric payoff comparable.

2.2.2 EXPERIENCED REGRET

An alternative metric devised by de Farias and Megiddo (2003, 2004), which we refer to as *experienced regret* (*e-regret*), compares the agent's average payoff over all rounds with the actual average payoff obtained by its most successful expert in rounds it was followed. Let $x_i^T(\phi)$ be the average payoff obtained by agent i in each round that it followed expert ϕ up to round T, given by[1]

$$x_i^T(\phi) = \frac{\sum_{t=1}^{T} I(\phi, \phi_i^t) m_i(\mathbf{a}^t)}{\sum_{t=1}^{T} I(\phi, \phi_i^t)}, \tag{3}$$

where $I(\phi, \phi_i^t)$ is an indicator function that returns 1 if $\phi = \phi_i^t$ and 0 otherwise. Then, i's *e-regret* is

$$\bar{\mathcal{E}}_i^T = \max_{\phi \in \Phi_i} x_i^T(\phi) - \frac{1}{T} \sum_{t=1}^{T} m_i(\mathbf{a}^t). \tag{4}$$

1. As an exception, if ϕ has never been played up to time T, then $x_i^T(\phi) = 0$.

As with regret (Eq. 1), the minuend of Eq. (4) is not independent of the subtrahend. In the general case, $x_i^T(\phi)$ can be different depending on the sequence of experts (and, hence, actions) that the agent plays. As such, $\max_\phi x_i^T(\phi)$ is not guaranteed to be a stable metric of success to which the agent's performance can be compared. Lower e-regret is not guaranteed to correspond to higher payoffs (and often does not; see Section 2.3). Thus, e-regret is not payoff comparable in general.

As an exception, de Farias and Megiddo (2004) showed that minimizing e-regret *in the limit* as $T \to \infty$ can directly translate into maximizing payoffs against "flexible opponents," or associates against whom the agent's average payoff between rounds t and $t+s$ converges (as $s \to \infty$) to a limit that is independent of the history of play prior to round t. Against flexible opponents, if ϕ is followed for a sufficiently large number of rounds s (approaching infinity) when selected, $x_i^t(\phi)$ does not vary substantially depending on the sequence of experts chosen (and, hence, the minuend of Eq. (4) is independent of the subtrahend). In such circumstances, minimizing e-regret equates to maximizing payoffs. Using such reasoning, de Farias and Megiddo also established well-defined performance guarantees (in the limit) for the expert algorithm EEE against flexible opponents.

While the performance bounds provided by de Farias and Megiddo (2004) are appealing in this special case, we do not adopt e-regret as an evaluation metric in repeated games for two reasons. First, though the set of "flexible opponents" includes useful classes of algorithms, it does not include many algorithms an agent is likely to encounter, such as many expert algorithms (including EEE itself) and other learning algorithms. Second, the performance bounds of EEE against flexible opponents are true only in the limit. Since most interactions are not infinite, we are interested in a metric that can provide an accurate measure of success over any time interval.

In summary, regret and e-regret are desirable since they provide generalizable performance benchmarks for expert algorithms. Unfortunately, these evaluation metrics are not payoff comparable against arbitrary associates. Hence, we adopt an alternative (though related) metric for evaluating how effectively expert algorithms select experts in repeated games.

2.2.3 DISAPPOINTMENT

External regret and e-regret imply a simple notion of success: an expert algorithm should perform at least as well as it would have performed had it always followed its best expert. Disappointment targets this same notion, minus the assumption that agent i's actions do not impact agent $-i$'s future actions. Formally, let $\pi_{-i}^t(\phi)$ be the policy that agent $-i$ would have played in round t had agent i always followed expert ϕ up to round t. Agent i's total disappointment[2] up to round T is

$$\mathcal{D}_i^T = \max_{\phi \in \Phi_i} \sum_{t=1}^{T} u_i^t(\phi, \pi_{-i}^t(\phi)) - \sum_{t=1}^{T} m_i(\mathbf{a}^t), \tag{5}$$

where $u_i^t(\phi, \pi_{-i}^t(\phi))$ is agent i's expected payoff in round t if, in each round $\tau \in \{1, \cdots, t\}$, agent i had followed expert ϕ and agent $-i$ had acted according to $\pi_{-i}^\tau(\phi)$. Agent i's *average disappointment* up to round T is

$$\bar{\mathcal{D}}_i^T = \frac{\mathcal{D}_i^T}{T}. \tag{6}$$

2. After this paper was accepted for publication, we became aware of recent work defining *policy regret* (Arora, Dekel, & Tewari, 2012; Cesa-Bianchi, Dekel, & Shamir, 2013). Disappointment captures the same notion as policy regret, except that disappointment allows for experts that implement complex (even adaptive) algorithms rather than just actions or action sequences. While this generalization is somewhat trivial, the term *policy regret* does not seem to fit given such experts. Rather than continue to overload the term *regret*, we refer to this metric as *disappointment*.

	C	D
c	0.60, 0.60	0.00, 1.00
d	1.00, 0.00	0.20, 0.20

(a) Prisoners' Dilemma

	c	d
a	0.84, 0.84	0.33, 1.00
b	1.00, 0.33	0.00, 0.00

(b) Chicken

Table 1: Payoff matrices for the PD and Chicken. In each cell, the row player's payoffs are listed first, followed by the column player's.

Agent i is said to have *no disappointment* when $\lim_{T\to\infty} \bar{\mathcal{D}}_i^T \leq 0$.

We make several observations about Eq. (5). First, the minuend and subtrahend are independent. Unlike Eqs. (1) and (4), the computation of an agent's best expert as measured in Eq. (5) is independent of how agent i played. The minuend is simply a constant specifying how well agent i would have done had it always followed its best expert, and hence is a stable benchmark of success. An agent's disappointment is this benchmark minus its accumulated payoffs. Hence, disappointment is payoff comparable: algorithms that receive higher payoffs against a given associate achieve lower disappointment than algorithms that receive lower payoffs against that associate (and vice versa).

Second, when agent $-i$ is not influenced by agent i's actions, a_{-i}^t is a good approximation of $\pi_{-i}^t(\phi)$. In such cases, disappointment and external regret are essentially equivalent. This is desirable since minimizing regret is equivalent to maximizing payoffs in such cases.

Third, like external regret, disappointment can be negative. Negative disappointment indicates that an expert algorithm performed better than it would have performed had it always followed its best expert. However, while sometimes possible, achieving negative disappointment can be extremely difficult against unknown associates. Thus, as an immediate goal, we focus on finding expert algorithms that achieve (or come close to achieving) no disappointment.

Finally, a strength of both regret and e-regret is that they can be computed during run time in a repeated game against an unknown associate. Thus, regret can be used as part of an algorithm in addition to being an evaluation metric. Indeed, regret is used as an algorithmic tool by various no-regret algorithms. On the downside, since minimizing regret does not necessarily correspond to maximizing payoffs, the use of regret as an algorithmic tool can lead to low payoffs in some scenarios. On the other hand, the minuend of Eq. (5) cannot be computed during run time against an unknown associate. Thus, disappointment is limited to being a metric to evaluate algorithms in repeated games; it is not clear how it could be used as an algorithmic tool.

2.3 Examples: Regret vs. Disappointment

We illustrate differences between disappointment and regret with several examples. First, consider an expert algorithm playing a repeated prisoners' dilemma (PD; Table 1a) against tit-for-tat (TFT; Axelrod, 1984). The expert algorithm has at its disposal two experts: AC, an expert that always recommends *cooperate*, and AD, an expert that always recommends *defect*. Figure 1 shows the average payoff, average regret, and average disappointment after 20 rounds when the algorithm always cooperates or always defects. Since always following AC would produce a higher payoff than always following AD (0.6 as opposed to 0.24), AC is the best expert. As such, always cooperating has zero disappointment and always defecting has high disappointment. On the other hand, always cooperating has high regret, while always defecting has zero regret. Thus, in this scenario, minimizing regret does not correspond to maximizing payoffs, but minimizing disappointment does.

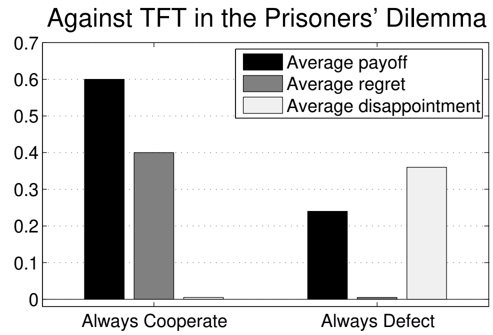


Figure 1: Comparison of average payoff, regret, and disappointment in the PD ($T = 20$).

Similar results are observed when learning algorithms play against each other in the repeated PD. For example, Figures 2a and 2c plot the average payoffs of six different learning algorithms (Exp3, UCB1, EEE, S, BR1, and BR2) against the algorithms' corresponding average regret and average disappointment, respectively, when paired with four different associates[3]. Figure 2a shows that the algorithms that achieved lower regret tended to have higher performance against BR1, WoLF-PHC, and Exp3, but not against S. Against S, algorithms with higher regret received substantially higher payoffs. On the other hand, Figure 2c shows that decreasing disappointment against each of the four algorithms resulted directly in higher payoffs in the PD.

Regret is even less indicative of performance in Chicken (Table 1b) against these associates. In this game, lower regret tends to lead to higher payoffs against WoLF-PHC, but not against Exp3, BR1, or S (Figure 2b). As in the PD, lower regret against S correlates with lower payoffs in Chicken, while lower disappointment always translates directly into higher payoffs (Figure 2d).

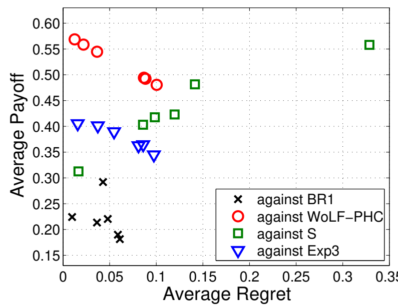
Figures 2e–2h demonstrate the deficiencies of e-regret as an evaluation metric in the PD and in Chicken against these same associates. After 1000 rounds, lower e-regret against BR1 in both the PD and in Chicken does not correspond with higher payoffs (Figures 2e and 2f). Discrepancies are even more pronounced by 50,000 rounds. In the PD, all algorithms have low e-regret against S, but with wildly different average payoffs (Figure 2g). Similar, though not identical, trends occur in Chicken (Figure 2h).

2.4 Research Agenda

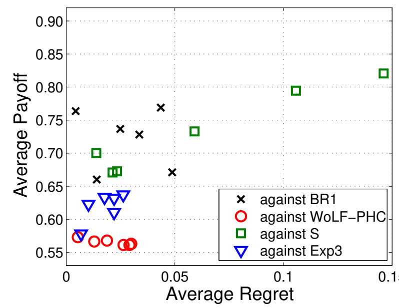
In the absence of a single, universally accepted, evaluation metric for repeated games, sets of performance criteria have been proposed (e.g., Powers & Shoham, 2005a; Crandall & Goodrich, 2011). Researchers advocating these agendas have argued that successful algorithms should simultaneously satisfy all criteria in the identified set. For example, Powers and Shoham (2005a) argued that successful algorithms should simultaneously satisfy three performance criteria:

- *Targeted Optimality*: Against any member of the target set of associates, the algorithm achieves within ε of the expected value of the best response to the associate's actual play.

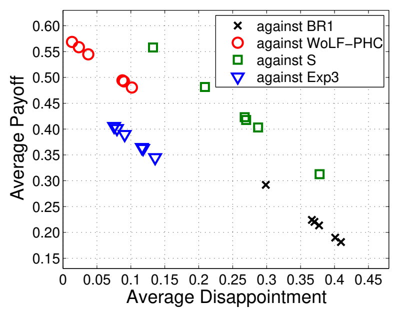
3. See Appendix A for implementation details. The experts used by Exp3, UCB, EEE, and S are described in Section 3. The regrets and disappointments of BR1 and BR2 are computed with respect to these same experts.



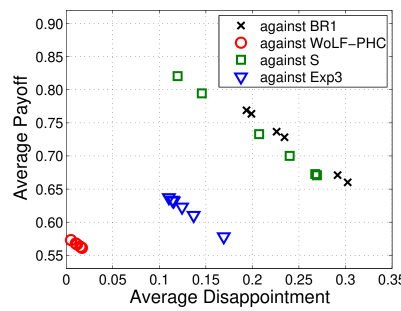
(a) PD: $\bar{\mathcal{R}}_i^{1000}$ vs. $\bar{\mu}_i^{1000}$



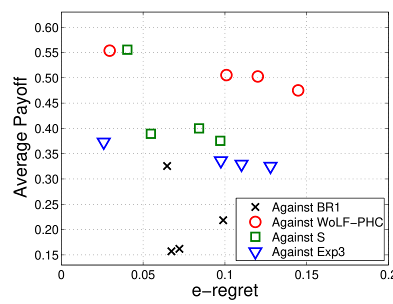
(b) Chicken: $\bar{\mathcal{R}}_i^{1000}$ vs. $\bar{\mu}_i^{1000}$



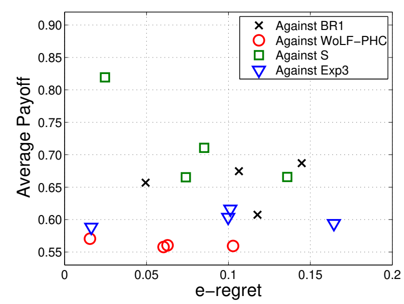
(c) PD: $\bar{\mathcal{D}}_i^{1000}$ vs. $\bar{\mu}_i^{1000}$



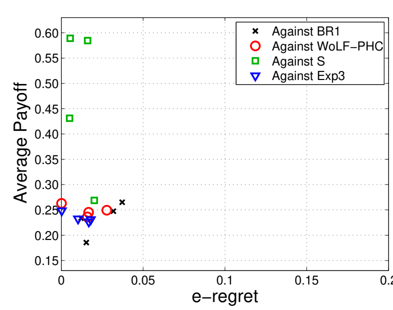
(d) Chicken: $\bar{\mathcal{D}}_i^{1000}$ vs. $\bar{\mu}_i^{1000}$



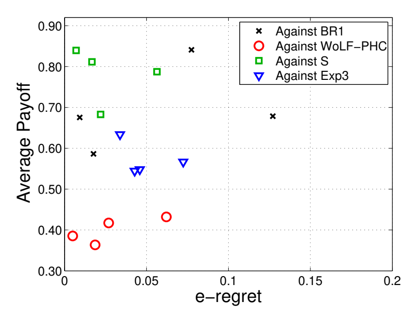
(e) PD: $\bar{\mathcal{E}}_i^{1000}$ vs. $\bar{\mu}_i^{1000}$



(f) Chicken: $\bar{\mathcal{E}}_i^{1000}$ vs. $\bar{\mu}_i^{1000}$



(g) PD: $\bar{\mathcal{E}}_i^{50000}$ vs. $\bar{\mu}_i^{50000}$



(h) Chicken: $\bar{\mathcal{E}}_i^{50000}$ vs. $\bar{\mu}_i^{50000}$

Figure 2: Average regret (a–b), disappointment (c–d), and e-regret (e–h) plotted against average payoffs ($\bar{\mu}_i^{1000}$) for various pairings in the PD and Chicken. Each point is an average of 50 trials. E-regret cannot be computed for BR1 and BR2, so those results are excluded in (e–h).

- *Compatibility*: In self-play, the algorithm achieves at least within ε of the payoff of some Nash equilibrium that is not Pareto dominated by another Nash equilibrium.
- *Safety*: Against any associate, the algorithm always receives at least within ε of the security value of the game.

When the set of experts satisfies certain properties, an expert algorithm that is guaranteed to have no disappointment will also satisfy these desirable performance criteria. Let ϕ_i^{Safety}, ϕ_i^{Comp}, and $\phi_i^{\mathrm{TargetOpt}}$ be behavior rules/algorithms that would satisfy the *Safety*, *Compatibility*, and *Targeted Optimality* properties, respectively, if they were always followed. Then, an expert algorithm that has no disappointment will satisfy the *Safety*, *Compatibility*, and *Targeted Optimality* properties, respectively, if these experts are in the set of available experts Φ_i.

Proposition 2.1 *Let $\mathcal{A}$ be an expert algorithm, let $\phi_i^{\mathrm{Safety}}, \phi_i^{\mathrm{Comp}}, \phi_i^{\mathrm{TargetOpt}} \in \Phi_i$, and let $\mathcal{A}(\Phi_i)$ the algorithm that uses $\mathcal{A}$ to select experts from Φ_i. If $\mathcal{A}$ is guaranteed to have no disappointment, then $\mathcal{A}(\Phi_i)$ will satisfy the* Safety*,* Compatibility*, and* Targeted Optimality *performance criteria.*

Proof. The proof follows directly from Eq. (5). Against all associates in all possible games, we know that if $\mathcal{A}$ is guaranteed to have no disappointment, then $\mathcal{A}(\Phi_i)$'s average payoff will be at least as high in the limit as its security value minus ε, since $\mathcal{A}$ will do at least as well in the limit as it would have done if it had always followed ϕ_i^{Safety}. Thus, it satisfies the *Safety* criteria. Similarly, in self play, $\mathcal{A}(\Phi_i)$ will perform at least as well in the limit as ϕ_i^{Comp}, and it will do at least as well as $\phi_i^{\mathrm{TargetOpt}}$ against all associates from the targeted set of associates. Hence, it will also meet the *Compatibility* and *Targeted Optimality* performance criteria. $\square$

We argue that it is not difficult to find ϕ_i^{Safety}, ϕ_i^{Comp}, and $\phi_i^{\mathrm{TargetOpt}}$. For example, ϕ_i^{Safety} can be an expert that always plays the agent's maximin strategy. ϕ_i^{Comp} can be an expert that follows Littman's and Stone's Godfather strategy (Littman & Stone, 2005). $\phi_i^{\mathrm{TargetOpt}}$ could be any number of algorithms that derive a best response to memory bounded opponents, such as the algorithm described by Chakraborty and Stone (2010).

Similar arguments can show that an expert algorithm guaranteed to have no disappointment will also meet other performance criteria, such as the performance criteria advocated by Crandall and Goodrich (2011). Additionally, suppose that Φ_i consists of three experts: Expert 1 acts optimally against associates from behavior class X, Expert 2 acts optimally against associates from behavior class Y, and Expert 3 acts optimally against associates from behavior class Z. Then, if $\mathcal{A}$ is an expert algorithm guaranteed to have no disappointment, then $\mathcal{A}(\Phi_i)$ will learn to act optimally when playing against associates from behavior classes X, Y, and Z.

In short, if we can find (1) an expert algorithm that is guaranteed to have no disappointment and (2) a good set of experts, we will have an agent that performs very well in repeated games. Given these results, a tempting research agenda is to find an algorithm that always achieves *no disappointment*. Unfortunately, unless an agent is omniscient, such a goal is impossible.

Proposition 2.2 *Against an unknown associate, an expert algorithm can guarantee an average disappointment of no less than $1 - v_i^{\mathrm{mm}}$, where $v_i^{\mathrm{mm}} = \max_{\pi_i} \min_{\pi_{-i}} u_i(\pi_i, \pi_{-i})$ is its maximin (i.e., security) value.*

Proof. We adapt an example from de Farias and Megiddo (2003). Let σ^* be a particular string of actions of length 100. Consider agent i playing a repeated PD against an associate (agent $-i$) who has the following strategy. In rounds $t \leq 100$, always cooperate. For $t > 100$, always cooperate if agent i's actions matched σ^* in all rounds $t \leq 100$; otherwise, always play the attack policy $\pi_{-i}^{\text{attack}} = \arg\min_{\pi_{-i}} \max_{\pi_i} u_i(\pi_i, \pi_{-i})$ (defect). Suppose that agent i has an expert that plays the string σ^* in the first 100 rounds, and then defects in all rounds thereafter. For large T, this best expert (against this associate) would get an average payoff near 1. But, without omniscience, an expert algorithm cannot know that it should follow this expert for the first 100 rounds, and therefore is very unlikely to follow σ^* for all $t \leq 100$. Thus, the maximum payoff it can guarantee itself is no more than $v_i^{\text{mm}} = 0.2$. Thus, $\bar{\mathcal{D}}_i^T \geq 1 - v_i^{\text{mm}}$ as $T \to \infty$. □

A less ambitious, but still extremely challenging, research agenda is to find an algorithm that quickly achieves low disappointment when associating with any member of a target set of associates across many repeated games. The most common target set of algorithms has been memory-bounded algorithms (e.g., de Farias & Megiddo, 2004; Chakraborty & Stone, 2010; Arora et al., 2012; Cesa-Bianchi et al., 2013), since theoretical guarantees are easier to establish against such algorithms. However, we find it more appealing to target a set of associates that is broad enough to cover many state-of-the-art algorithms published in the literature, which would presumably be the set of likely associates an agent would face. For example, algorithms from the literature include static (memory-bounded) algorithms and automata, reinforcement learning algorithms, and expert algorithms. Thus, in this paper, we focus on finding expert algorithms that achieve low disappointment against associates from these three classes of algorithms.

A theoretical treatment of this aim is extremely challenging and is beyond the scope of this paper. Rather, as a starting point, we empirically evaluate the disappointment of algorithms against various static, reinforcement learning, and expert algorithms from the literature. In so doing, we seek an algorithm that quickly achieves low disappointment against each algorithm we consider. Let Θ be the set of opponent algorithms considered. Then, the *max disappointment* of algorithm $\mathcal{A}$ with respect to Φ and Θ after T rounds is

$$\hat{\mathcal{D}}_{\mathcal{A}}^T(\Theta, \Phi) = \max_{\theta \in \Theta} \bar{\mathcal{D}}_{\mathcal{A}}^T(\theta, \Phi), \tag{7}$$

where $\bar{\mathcal{D}}_{\mathcal{A}}^T(\theta, \Phi)$ is the average disappointment of algorithm $\mathcal{A}$ against associate $\theta \in \Theta$.

Though it is tempting to focus on asymptotic performance (as $T \to \infty$), interactions repeat no more than tens or hundreds of times in many realistic scenarios. As such, we seek to identify expert algorithms that *quickly* achieve low disappointment against static associates, reinforcement learning algorithms, and expert algorithms. Thus, we primarily focus on the disappointment achieved by algorithms over the first 1000 rounds, though we also consider longer-term performance.

In Section 4 we evaluate several existing algorithms with respect to disappointment. Before doing so, we describe a method for computing a good set of experts for repeated general-sum games.

3. Computing a Set of Experts for Arbitrary Repeated Games

The success of an agent employing an expert algorithm depends on both the expert algorithm's ability to select the most effective experts (and, thus, minimize disappointment) and the effectiveness of the set of experts Φ_i. Given the importance of Φ_i, we give considerable attention to computing a

	c	d
a	0.33, 0.33	0.67, 1.00
b	1.00, 0.67	0.00, 0.00

(a) Leader

	c	d
a	1.00, 1.00	0.00, 0.75
b	0.75, 0.00	0.50, 0.50

(b) Stag hunt

	c	d
a	0.84, 0.33	0.84, 0.00
b	0.00, 1.00	1.00, 0.67

(c) Security Game

	c	d
a	0.00, 0.00	0.00, 1.00
b	1.00, 0.00	0.00, 0.00

(d) Offset Game

	c	d
a	1.00, 1.00	0.00, 0.00
b	0.00, 0.00	0.50, 0.50

(e) Common Interest

	c	d
a	0.00, 0.00	0.67, 1.00
b	1.00, 0.67	0.33, 0.33

(f) Battle of the Sexes

	c	d
a	0.00, 1.00	1.00, 0.67
b	0.33, 0.00	0.67, 0.33

(g) Tricky Game

	d	e	f
a	0, 0	1, 0	0, 1
b	0, 1	0, 0	1, 0
c	1, 0	0, 1	0, 0

(h) Shapley's Game

Table 2: Payoff matrices for eight different games.

good set of experts Φ_i for repeated normal-form games. For each potential associate and game an agent might encounter, this set of experts should include at least one expert that will perform well.

Littman and Stone (2001) grouped algorithms for repeated games into two classes: *leaders* and *followers*. Leaders are typically effective when associating with follower algorithms, such as standard reinforcement learning and opponent modeling algorithms. Follower algorithms are typically more effective against leader strategies and other static algorithms. Thus, in order to have a high-performing expert for each scenario the agent might encounter, a good set of experts must contain both leader and follower experts.

3.1 Leader Experts

Based on the premise that the associate will play a best response, leader strategies are designed to play strategies that cause associates to play their portion of a desirable solution (sometimes called a *targeted pair*) (Littman & Stone, 2001). A *solution* $\mathbf{s}$ is a sequence of joint actions that the agents repeatedly play. Each solution $\mathbf{s}$ produces an expected payoff profile $\mathbf{v}(\mathbf{s}) = (v_i(\mathbf{s}), v_{-i}(\mathbf{s}))$. For example, the solution $\mathbf{s} = <(c, C), (d, C)>$ in the PD corresponds to the situation in which the column player always cooperates while the row player alternates between cooperating and defecting. It produces the expected payoff profile $\mathbf{v}(\mathbf{s}) = (0.8, 0.3)$ for the payoffs given in Table 1a.

Since it is often unclear which solution an agent should target, we define a set of potential target solutions Ω. Ω contains all solutions that satisfy the following three criteria. First, Ω consists only of solutions for which each agent's expected payoff v_i exceeds its maximin value v_i^{mm}. Second, in line with Occam's razor, Ω consists only of solutions with sequences of one or two joint actions. Solutions with longer sequences are likely to be too complex for many potential associates to identify, especially in interactions that last only tens of rounds, so we exclude them. Third, solutions in Ω must be enforceable. It must be possible to make playing the solution the associate's best response. Formally, when agent i plays strategy ρ_i, agent $-i$'s best response is

$$br_{-i}(\rho_i^t) = \arg\max_{\rho_{-i}} \mu_{-i}(\rho_i, \rho_{-i}). \tag{8}$$

Recall that $\mu_{-i}(\rho_i, \rho_{-i})$ is $-i$'s average per-round payoff when the strategies ρ_i and ρ_{-i} are played.

The cardinality of Ω ($|\Omega|$) varies from game to game. For example, in the PD, $|\Omega| = 6$, while $|\Omega| = 9$ in Shapley's Game (Table 2h). Furthermore, in a set of 50 randomly-generated 5-action games, we found $|\Omega|$ to vary between 18 and 187.

No.	Target solution (s)	$v_1(\mathbf{s})$	Strategy
1	$<(c,C)>$	0.60	If $g_2^t > 0$, play π_i^{attack}. Otherwise, play own portion of the current joint action in the sequence $\mathbf{s}$.
2	$<(d,C),(c,D)>$	0.50	
3	$<(c,C),(d,D)>$	0.40	
4	$<(d,D)>$	0.20	

Table 3: Leader experts generated for player 1 in the PD.

Though enforceable, some solutions can only be made an associate's best response if the associate conditions its strategy on many previous joint actions. Because of the curse of dimensionality, most algorithms designed to learn quickly condition their strategy on only a few previous joint actions. Thus, we form a separate leader expert (ϕ) only for those target solutions $\mathbf{s} \in \Omega' \subseteq \Omega$ for which $br_{-i}(\phi)$ requires the associate to remember only a single joint action. These leader experts incentivize associates to play their portion of the target solution $\mathbf{s}$ by punishing deviations from $\mathbf{s}$ using a similar mechanism to those used by previously defined leader strategies (e.g., Littman & Stone, 2005; Crandall & Goodrich, 2005). When the associate has conformed with $\mathbf{s}$ or has not benefited from deviating from it, the leader expert plays its portion of $\mathbf{s}$. However, if the associate has benefited from deviating from $\mathbf{s}$, the leader expert plays its attack policy π_i^{attack}.

Formally, each leader expert keeps track of a guilt parameter g_{-i}^t that defines how much its associate has benefited from deviating from $\mathbf{s}$. Initially, $g_{-i}^1 = 0$. Subsequently,

$$g_{-i}^{t+1} \leftarrow \begin{cases} 0 & \text{if } a_{-i}^t = b_{-i}^t \text{ and } g_{-i}^t = 0 \\ \max\left(0, g_{-i}^t + m_{-i}(\mathbf{a}^t) - v_{-i} + \delta_t\right) & \text{otherwise} \end{cases} \tag{9}$$

where b_{-i}^t is agent $-i$'s current action defined by the target solution $\mathbf{s}$ and δ_t is a small nonnegative value. We use $\delta_t = 0.1$ if $g_{-i}^t = 0$ and $\delta_t = 0.0$ otherwise. When $g_{-i}^t > 0$, agent $-i$ has recently benefited (or at least not been hurt) by deviating from $\mathbf{s}$. To discourage such behavior, the leader expert plays its attack policy π_i^{attack}. When $g_{-i}^t = 0$, the leader expert plays its portion of $\mathbf{s}$.

Table 3 lists the four leader experts generated for the PD. Note that no expert is created for the target solutions $<(c,C),(d,C)> \in \Omega$ and $<(c,C),(c,D)> \in \Omega$, since these solutions are not enforceable against an associate that learns a best response conditioned only on the last joint action.

3.2 Follower Experts

Followers learn to play a best response to the estimated strategy of their associate. We consider three types of follower experts, each of which estimates its associate's strategy in a different way. The first of these experts models its associate using the fictitious play assessment conditioned on the last joint action played. Let $\kappa_{-i}^t(\mathbf{a}, a_{-i})$ be the number of times that agent $-i$ has played a_{-i} given the previous joint action $\mathbf{a}$ up to round t. Then, the estimated probability that agent $-i$ plays a_{-i} given the previous joint action $\mathbf{a}$ is

$$\gamma_{-i}^t(\mathbf{a}, a_{-i}) = \frac{\kappa_{-i}^t(\mathbf{a}, a_{-i})}{\sum_{b_{-i} \in A_{-i}} \kappa_{-i}^t(\mathbf{a}, b_{-i})}. \tag{10}$$

The expert then computes the automaton that best responds to the agent's future discounted reward (we use discount factor $\gamma = 0.95$) given γ_{-i}^t. We refer to this expert as ϕ_i^*.

No.	Target solution (s)	$v_1(\mathbf{s})$	Strategy
1	(none)	$\mu_1(br_1(\gamma_2^t), \gamma_2^t)$	$br_1(\gamma_2^t)$
2	(none)	$v_1^{\text{mm}} = 0.20$	π_1^{mm}
3	$<(c, C), (d, C)>$	0.80	If $\mathbf{a}^{t-1}$ is in $\mathbf{s}$, play own portion of the next joint action in $\mathbf{s}$. Otherwise, randomly select a joint action from $\mathbf{s}$ and play own portion of that joint action.
4	$<(c, C)>$	0.60	
5	$<(d, C), (c, D)>$	0.50	
6	$<(c, C), (d, D)>$	0.40	
7	$<(c, C), (c, D)>$	0.30	
8	$<(d, D)>$	0.20	

Table 4: Follower experts generated for player 1 in the PD.

A second follower expert, called ϕ_i^{mm}, assumes that its associate is trying to exploit it. Thus, its best response is to play its maximin strategy (or policy), which is given by

$$\pi_i^{\text{mm}} = \arg\max_{\pi_i} \min_{\pi_{-i}} u_i(\pi_i, \pi_{-i}). \tag{11}$$

Finally, the associate could also be using a leader strategy, such as those computed in Section 3.1. Thus, we include in our set of experts a follower expert for each $\mathbf{s} \in \Omega$. These experts always play their part of $\mathbf{s}$. If a joint action in the solution was not played in the previous round, these experts randomly select a joint action from the solution sequence and play the agent's corresponding action.

Table 4 lists the eight follower experts generated for the PD by this method.

3.3 Set of Experts

The set of experts Φ_i used by the expert algorithms in the remainder of this paper consists of the follower and leader experts just described. In most scenarios we have encountered, at least one of these experts is capable of performing effectively. If an associate employs a follower algorithm, an expert algorithm can select any number of leader experts from Φ_i that could induce desirable behavior from the associate. Similarly, if the associate employs a leader algorithm, such as Godfather or Bully (Littman & Stone, 2001), the expert algorithm can select the corresponding follower expert. Additionally, the diversity of the experts in Φ_i is such that a good expert algorithm should be able to obtain high payoffs against other expert algorithms, including those that select from a similar set of experts. In this case, the expert algorithms must negotiate follower and leader roles.

An illustration of the performance of these experts against three different associates in several repeated games is provided in Appendix B. In each example, the set of experts contains at least one high-performing expert. However, these results also illustrate that identifying a best expert during run time can be extremely difficult. Sometimes the best performing expert must be followed consistently for many rounds before it produces high payoffs.

4. Results – Existing Expert Algorithms

Existing expert algorithms have typically been evaluated in terms of regret. In this section, we evaluate several of these algorithms in repeated normal-form games in terms of disappointment. Specifically, we analyze the average disappointment of four existing expert algorithms against twelve different algorithms across ten repeated games. Recall that our goal is to find an expert algorithm that quickly achieves and maintains low disappointment against static algorithms, reinforcement learning algorithms, and other expert algorithms.

The four expert algorithms are Exp3, UCB1, EEE, and S. Exp3 and UCB1 are well-known expert algorithms with well-defined regret bounds in multi-armed and adversarial multi-armed bandit problems, respectively. Both of these algorithms have been shown to perform effectively in some repeated games (Bouzy & Metivier, 2010). EEE is an ε-greedy expert algorithm designed for repeated games played against learning associates. As $T \to \infty$, it has been shown to have no e-regret (de Farias & Megiddo, 2003, 2004). S is an aspiration-based algorithm which has been shown to perform well as an expert algorithm (Bouzy, Metivier, & Pellier, 2011), though it was not originally designed as such. In the interest of space, we omit detailed overviews of these algorithms. Instead, we refer the reader to Appendix A, which provides references to descriptions and analysis of these algorithms, and also specifies the parameter values used to generate the results in this paper.

We compare the average disappointment of these four expert algorithms when paired against twelve representative algorithms: four static algorithms (A0, Random, Godfather, and Bully), four reinforcement learning algorithms (BR1, BR2, Q-learning, and WoLF-PHC), and each other. Implementation details for each of these twelve algorithms are also supplied in Appendix A. Comparisons are made across the ten games shown in Tables 1 and 2. These are well-studied games from the literature, each representing a different challenge. To be successful in all of these games, an algorithm must be able to learn to make and accept profitable compromises in many different situations.

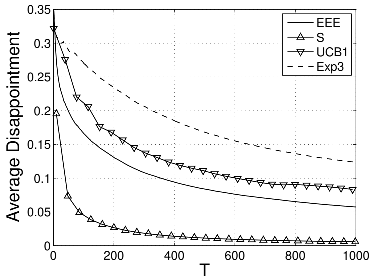
The average disappointment across all games for each pairing is shown in Figure 3. The figure shows that, over the first 1000 rounds, S typically has lower average disappointment against each associate than the other three expert algorithms. Additionally, despite its popularity and theoretical properties, Exp3 performs the worst of the four expert algorithms against each associate.

Results vary somewhat by associate. While all four expert algorithms eventually obtain low disappointment against each static algorithm, Figures 3a–3d show that only S reaches an average disappointment of less than 0.05 within 1000 episodes against all four static associates. The other expert algorithms are unable to do so, in large part due to the relatively large number of experts in Φ_i. S's mechanism for selecting experts allows it to find the best expert faster than the other algorithms. As an exception, S is not as effective against Random, especially in the long run, as S sometimes never does learn to play the best expert. For example, S does not learn to play a best response against Random in the PD, while the other three algorithms eventually do.

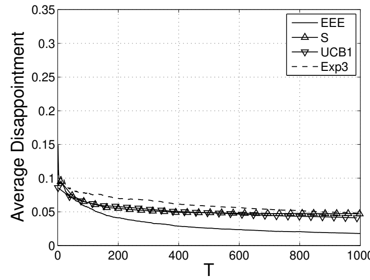
Because reinforcement learning algorithms adapt over time, achieving low average disappointment against these associates is more difficult than against static algorithms. This statement is confirmed by the performance of the four expert algorithms against BR1, BR2, WoLF-PHC, and Q-learning (Figures 3e–3h). Against these associates, none of the four expert algorithms achieves low disappointment during the first 1000 rounds. Their average disappointment quickly increases over the first 50-200 rounds, and then slowly decreases or plateaus thereafter.

The four expert algorithms also have high disappointment against each other (Figures 3i–3l). In self play, S manages to achieve an average disappointment of about 0.05, but none of the other three algorithms does so over the first 1000 rounds against any expert algorithm. In fact, against Exp3, the average disappointment of each expert algorithm increases throughout the first 1000 rounds.

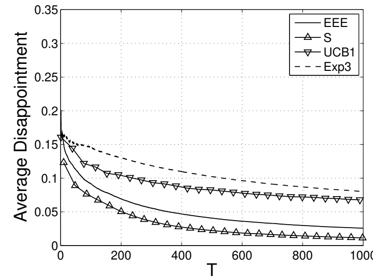
In short, none of the four expert algorithms quickly achieves low disappointment against static algorithms, reinforcement learning algorithms, and expert algorithms. Thus, in the next section, we describe a new meta-algorithm designed to enhance the performance of existing expert algorithms.



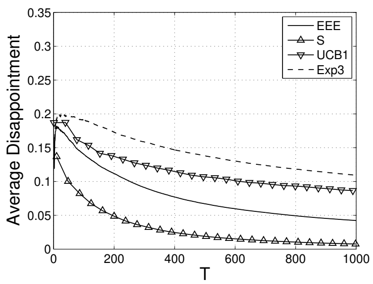
(a) Against A0



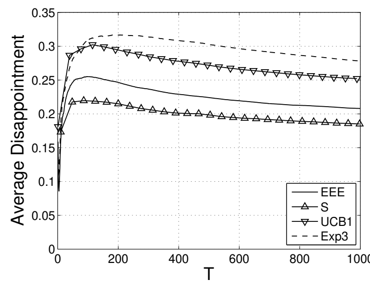
(b) Against Random



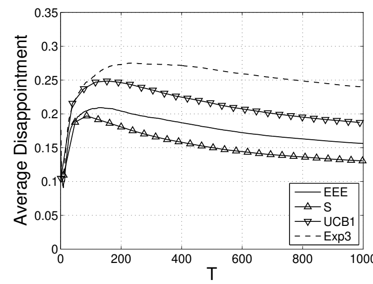
(c) Against Bully



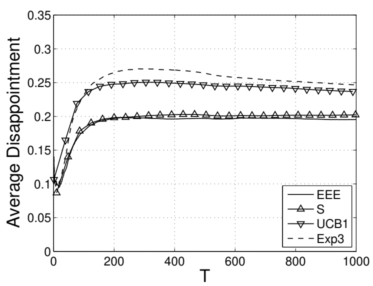
(d) Against Godfather



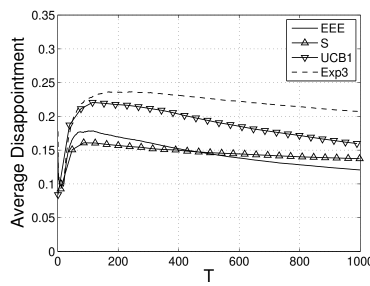
(e) Against BR1



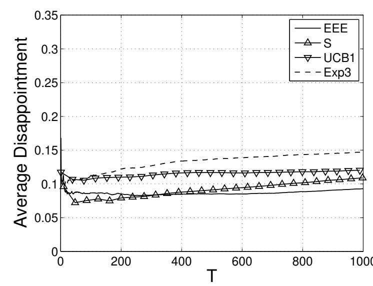
(f) Against BR2



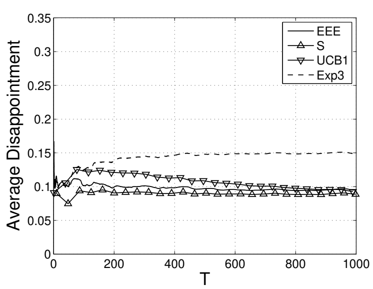
(g) Against WoLF-PHC



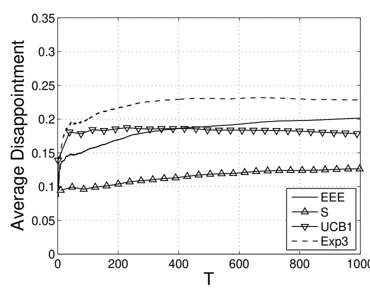
(h) Against Q-learning



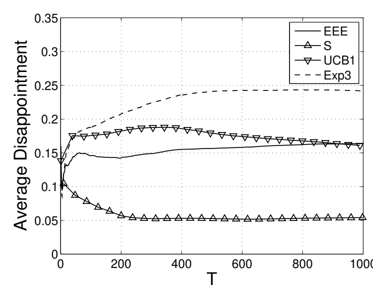
(i) Against Exp3



(j) Against UCB1



(k) Against EEE



(l) Against S

Figure 3: The average disappointment $\bar{\mathcal{D}}^T$ over time of four expert algorithms against twelve associates. Results are an average of 50 trials in each of the ten selected games.

In Section 6, we demonstrate that this meta-algorithm improves Exp3, UCB1, EEE, and S so that they consistently achieve much lower disappointment against these same associates.

5. Enhancing Existing Expert Algorithms

The failure of these expert algorithms to consistently achieve low disappointment against adapting agents appears to be tied to the algorithms' exploration strategies. Early in the game, these expert algorithms tend to spend many rounds following ineffective experts as they seek to determine from experience which experts are most effective. As a result, they receive low average payoffs in early rounds of the game. Furthermore, the frequent changes in behavior caused by cycling through many experts are incoherent to an outsider. Thus, associates are unlikely to determine how to coordinate behavior or strike mutually beneficial compromises. Against adaptive associates whose internal models are conditioned somewhat on their associate's behavior, this process often leads to low payoffs (and, hence, high disappointment) in both the short and the long term. In this section, we describe a rather simple meta-algorithm designed to overcome this deficiency.

5.1 A New Meta-Algorithm

The purpose of the meta-algorithm is to help the expert algorithm explore the effectiveness of its set of experts Φ_i more effectively. It does this by computing the highest expected payoff (or *potential*) of each expert, and then supplying the expert algorithm with the subset of experts whose potential meets some *performance threshold*. Experts with lower potential are only followed once experts with higher potentials have demonstrated an inability to meet their potentials.

The performance threshold in each round t is determined using aspiration learning (Karandikar, Mookherjee, R., & Vega-Redondo, 1998; Stimpson, Goodrich, & Walters, 2001; Chasparis, Shamma, & Arapostathis, 2010). In aspiration learning, agent i maintains an aspiration α_i^t. In our algorithm, α_i^1 is initialized to the potential (see Section 5.3) of the expert with the highest potential. Formally, let $z_i^t(\phi)$ denote the potential of expert ϕ in round t. Then,

$$\alpha_i^1 = \max_{\phi \in \Phi_i} z_i^1(\phi). \tag{12}$$

As with S (Appendix A), each round that a new expert is selected, α_i^t is updated as follows:

$$\alpha_i^t = \lambda^\tau \alpha_i^{t-\tau} + (1 - \lambda^\tau)\bar{r}_i^t, \tag{13}$$

where $\bar{r}_i^t$ is the average payoff received by agent i in the last τ rounds, τ is the number of consecutive rounds the expert was followed, and $\lambda \in [0, 1]$ is a learning rate (we use $\lambda = 0.99$).

The performance threshold in round t is the agent's minimum aspiration level α_i^k over some time interval $k \in [\zeta, t]$, where $1 \leq \zeta \leq t$. Each expert whose potential meets or exceeds this performance threshold is considered for selection in round t; the other experts are not.[4] Formally, in round t, experts are selected from the set

$$\Phi_i'(t) = \{\phi \in \Phi_i : z_i^t(\phi) \geq \min_{k \in [\zeta, t]} \alpha_i^k\}. \tag{14}$$

When $\Phi_i'(t)$ as defined in Eq. (14) is empty, a default expert is added to $\Phi_i'(t)$. For the set of experts defined in Section 3, we add the expert that plays a best response to the fictitious-play assessment conditioned on the previous joint action. That is, $\Phi_i'(t) = \{\phi_i^*\}$ for such cases.

4. As before, disappointment is always computed using the full set of experts Φ_i.

Algorithm 1 A meta-algorithm (for agent i) to enhance expert algorithms.

Input:
$\mathcal{A}$ (the expert algorithm), Φ_i (the set of experts), and M (the payoff matrix)
Initialize:
t = 1
Compute $z_i^t(\phi)$ for each $\phi \in \Phi_i$
Initialize $\alpha_i^t = \max_{\phi \in \Phi_i} z_i^t(\phi)$
repeat
Compute $\Phi_i'(t) = \{\phi \in \Phi_i : z_i^t(\phi) \geq \min_{\tau \in [\zeta, t]} \alpha_i^\tau\}$, where $0 \leq \zeta < t$
Execute (and update) $\mathcal{A}(\Phi_i'(t))$ for τ rounds, where τ is specified by $\mathcal{A}$. Observer $\bar{r}_i^{t+\tau}$.
$t = t + \tau$
Update $\alpha_i^t = \lambda^\tau \alpha_i^{t-\tau} + (1 - \lambda^\tau)\bar{r}_i^t$
Compute $z_i^t(\phi)$ for each $\phi \in \Phi_i$
until Game Over

The complete meta-algorithm is stated in Algorithm 1. We make two observations. First, the meta-algorithm embodies the optimism-in-uncertainty principle (Brafman & Tennenholtz, 2003). The aspiration level is initially set high, and each expert is presumed to be able to meet its highest potential. Only after experts fail to meet their highest potentials (which causes the aspiration level to fall; Eq. 13) does the algorithm consider selecting experts with lower potential. Second, while aspiration updates are similar in nature to previous work on aspiration learning (Karandikar et al., 1998; Stimpson et al., 2001; Chasparis et al., 2010), the aspiration level is used differently in this meta-algorithm. Rather than use α_i^t to determine whether an action or expert should be repeated in the next round, we use α_i^t to prune the set of selectable experts.

5.2 Properties of the Reduced Set $\Phi_i'(t)$

Selecting experts from $\Phi_i'(t)$ rather than from Φ_i leaves open the possibility that $\Phi_i'(t)$ might not contain the best expert. However, for certain parameter settings, our meta-algorithm will either obtain average payoffs no less than the best expert in the limit, or the best expert will be contained in $\phi_i'(t)$ for all $t > \tau$. Let $v_i^*(\phi)$ denote the highest possible average per-round payoff that agent i could ever obtain if it were to always follow expert ϕ, let ϕ' denote the best expert for the game and associate in question, and let μ_i^t be the average payoff obtained by agent i up to time t. Then we have the following proposition.

Proposition 5.1 *If $\zeta = 1$ and $\forall \phi \in \Phi_i, z_i^t(\phi) \geq v_i^*(\phi)$, then one of the following must hold:*
Condition 1: $\lim_{t \to \infty} \mu_i^t \geq z_i^t(\phi')$
Condition 2: $\exists \tau, \phi' \in \Phi_i'(t)$ for all $t \geq \tau$

Condition 1 equates with the expert algorithm having no disappointment, while Condition 2 says that ϕ' will eventually enter (and then perpetually remain in) $\Phi_i'(t)$. The proof of the proposition is straightforward. Since the aspiration level α_i^t is the fading average payoff of the agent, it must at some time τ fall below $z_i^t(\phi')$ if its average payoff μ_i^t is perpetually below $z_i^t(\phi')$. In which case, in accordance with Eq. (14), ϕ' would be in $\Phi_i'(t)$ thereafter.

As is typical in aspiration learning (Karandikar et al., 1998), we use $\zeta = t$ (i.e., the performance threshold is α_i^t) in generating the results shown in the next section. This means that our implementation does not technically satisfy the conditions of Proposition 5.1. However, we have observed that, in practice, the algorithm achieves similar results for both $\zeta = 1$ and $\zeta = t$.

5.3 Computing Potential

Most strategies and learning processes have a target value, such as the value of an equilibrium solution or the current expected value of the learning process. While not always conforming with the pre-condition of Proposition 5.1, such target values are often sufficient for determining the potential of experts in practice. We demonstrate how to define the potential of experts using our set of experts, which was defined in Section 3.

Let $\Phi_i^{\text{lead}} \subseteq \Phi_i$ denote the set of leader experts in Φ_i. Under the assumption of a rational (follower) associate, the highest expected payoff (or potential) that we can reasonably expect each leader expert $\phi \in \Phi_i^{\text{lead}}$ to obtain is the expected per-round payoff the agent will receive when its opponent plays a best response to its strategy. Formally, for all $\phi \in \Phi_i^{\text{lead}}$ and for all t,

$$z_i^t(\phi) = \mu_i(\phi, br_{-i}(\phi)). \tag{15}$$

Let $\Phi_i^{\text{follow}} \subseteq \Phi_i$ denote the set of follower experts in Φ_i excluding the maximin and best-response experts (ϕ_i^{mm} and ϕ_i^*, respectively). If the associate appears to be playing its portion of the target solution corresponding to expert $\phi \in \Phi_i^{\text{follow}}$, then ϕ's potential is the agent's expected payoff when the target solution corresponding to that expert is played. Let s_{-i}^t denote the observed strategy employed by agent $-i$ in round t. Then,

$$z_i^t(\phi) \leftarrow \begin{cases} \mu_i(br_i(s_{-i}^t), s_{-i}^t) & \text{if } \phi_i \in br_i(s_{-i}^t) \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

To determine which strategies its associate could be playing, agent i models agent $-i$'s actions given the previous joint action $\mathbf{a}$. Since agent $-i$ could change s_{-i}^t at any time, only a partial estimate of s_{-i}^t may be available. The agent remembers the last round k that agent $-i$ changed its action given any previous joint action $\mathbf{a} \in A$. All actions taken from round k onward define s_{-i}^t. Formally, agent $-i$'s estimated strategy given $\mathbf{a}$ is

$$s_{-i}^t(\mathbf{a}) \leftarrow \begin{cases} a_{-i}^j & \text{if } \exists j \in [k, t) : \mathbf{a}^j = \mathbf{a} \\ \wedge & \text{otherwise} \end{cases} \tag{17}$$

If agent i's estimate of s_{-i}^t is consistent with the leader strategy corresponding to $\phi_i \in \Phi_i^{\text{follow}}$, then the agent assumes it is playing that leader strategy when computing $z_i^t(\phi)$.

The potentials of the maximin and best-response experts are $z_i^t(\phi_i^{\text{mm}}) = v_i^{\text{mm}}$ and $z_i^t(\phi_i^*) = \mu_i(br_i(\gamma_{-i}^t), \gamma_{-i}^t)$, respectively.

5.4 Safety

Safety, the guarantee that expected average payoffs will not be substantially below the maximin value v_i^{mm}, is perhaps the oldest objective in repeated games (Fudenberg & Levine, 1998). An expert algorithm that is guaranteed to have no regret is guaranteed to have safety if some $\phi \in \Phi_i$ is safe. However, for those expert algorithms that do not have well-defined regret bounds, we add a mechanism to our meta-algorithm that ensures safety. We call this mechanism the *safety override* since it overrides Eq. (14) under certain conditions.

The safety override is adopted from a security mechanism described by Crandall and Goodrich (2011). If the sum of the agent's payoffs is ever less than a constant C_i below what it would have achieved had it always received its maximin value (i.e., if $\exists T \leq t$ such that $\sum_{\tau=1}^{T} m_i(\mathbf{a}^\tau) + C_i < T v_i^{\mathrm{mm}}$), then $\Phi_i'(t) = \{\phi_i^{\mathrm{mm}}\}$. This guarantees that the agent's expected average payoff will be no less than v_i^{mm} as $t \to \infty$, regardless of the game or associate. The proof of safety provided by this mechanism is given by Crandall and Goodrich (2011). We used $C_i = 100$ in our implementation.

5.5 Best Response

Previous work has shown the value of properly balancing optimistic, best response, and secure attitudes in repeated games (Crandall & Goodrich, 2011). Eqs. (12–14) induce an optimistic attitude, while the safety override is a secure attitude. Finally, we add a *best-response override*. The algorithm sets $\Phi_i'(t) = \{\phi_i^*\}$ when the following two conditions are met:

1. $\phi_i^* \in \Phi_i'(t)$; see Eq. (14) in conjunction with the safety override.
2. The historical average per-round payoff for playing ϕ_i^* is as high as that of any other expert. Formally, let $x_i^t(\phi)$ be the weighted average per-round payoff[5] received by agent i in each round that it has followed expert ϕ up to round t. Then, $\forall \phi \in \Phi_i'(t),\ x_i^t(\phi_i^*) \geq x_i^t(\phi)$.

This override is of the most use for algorithms like S, which learn effectively against many algorithms, but sometimes do not learn a best response against static agents (such as Random).

6. Results – Enhanced Expert Algorithms

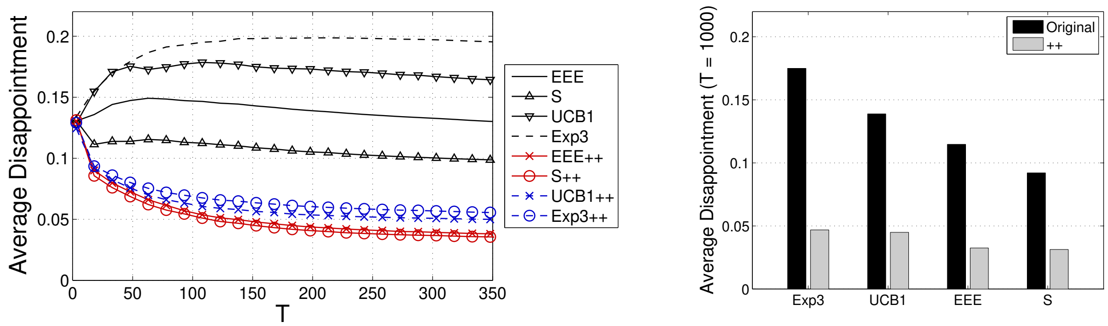
We enhanced the four expert algorithms evaluated in Section 4 with the meta-algorithm. We call the enhanced versions of these algorithms Exp3++, UCB++, EEE++, and S++, respectively. These algorithms are identical to the original algorithms except that they select experts from $\Phi_i'(t)$ rather than Φ_i. We first evaluate whether these expert algorithms consistently achieve low disappointment against the same twelve associates as before. We also compare the payoffs of these algorithms to top-performing algorithms from the literature in both perfect and imperfect information settings.

6.1 Against Static, Reinforcement Learning, and Expert Algorithms

The average disappointment of the enhanced and original algorithms against all twelve associates across all ten games is shown in Figure 4. Each enhanced algorithm has substantially less average disappointment than the original algorithm. The enhanced algorithms quickly reach low levels of disappointment (Figure 4a). By 1000 rounds, the average disappointment of each of the enhanced algorithms is below 0.05 (Figure 4b), which is substantially less than the original algorithms. Similar improvements are present in terms of max disappointment (Figure 5).

The meta-algorithm consistently produces substantial decreases in disappointment against all associates and in all ten games (not shown). Figure 6 shows the average disappointment of EEE++ and Exp3++ against each associate across all games. Against static associates (Figures 6a–6d), EEE++ and Exp3++ both achieve very low average disappointment well before 1000 rounds. The meta-algorithm produces even greater decreases in average disappointment against the reinforcement learning algorithms (Figures 6e–6f) and the expert algorithms (Figures 6i–6l). The algorithms

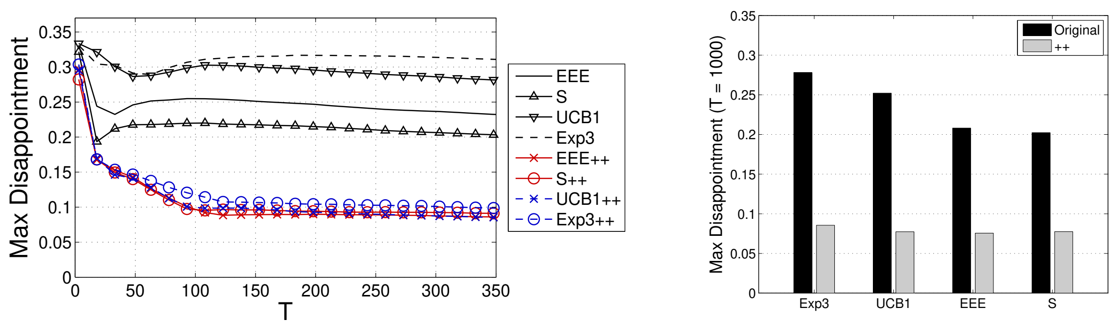
5. Initially, $x_i^1(\phi) = 1$. $x_i^t(\phi)$ is updated after each round that ϕ is followed: $x_i^t(\phi) = \beta_i^t x_i^{t-1}(\phi) + (1 - \beta_i^t) m_i(\mathbf{a}^t)$, where $\beta_i^t = \max(1/\kappa_i^t(\phi), 2(1-\lambda))$ and $\kappa_i^t(\phi)$ is the number of times agent i has played ϕ up to round t.



(a) Average disappointment over time

(b) Average disappointment (T=1000)

Figure 4: Average disappointment $\bar{\mathcal{D}}^T$ across all selected games and associates.



(a) Max disappointment over time

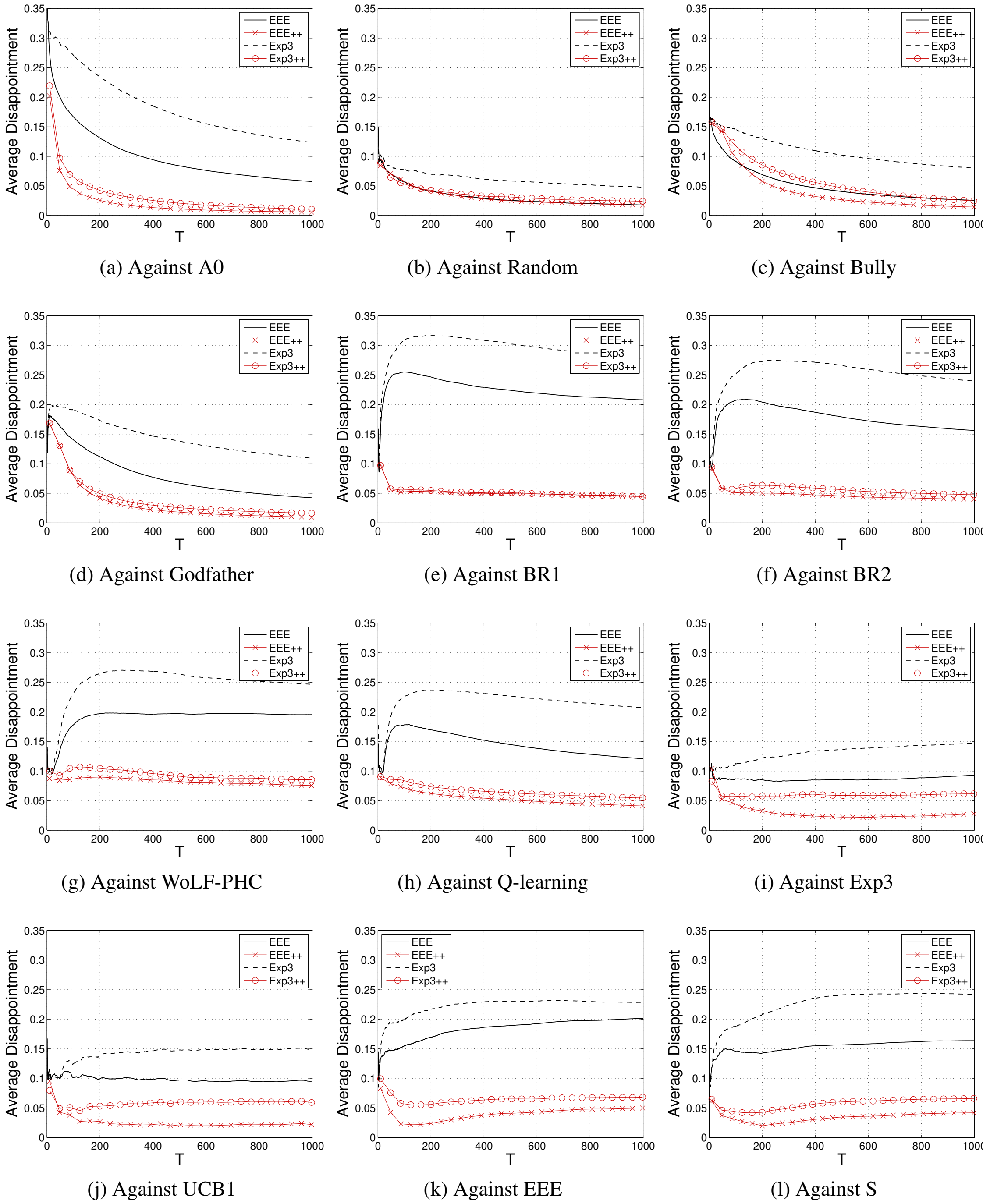
(b) Max disappointment (T=1000)

Figure 5: Max disappointment $\hat{\mathcal{D}}^T$ (Eq. 7).

enhanced by the meta-algorithm also achieve higher payoffs in self play than do the original algorithms (Figure 7).

One interesting exception to the previously stated trend is that EEE performs better against Bully over the first 100 rounds than does EEE++ (Figure 6c). Against Bully, the best-performing experts (see Figure 11 in Appendix B) tend to not have high potential in some games, while experts with high potential often do not achieve high payoffs well. Hence, it takes many rounds before the agent's aspiration level falls far enough for the best expert to be in $\Phi'_i(t)$. This causes EEE++ to have higher disappointment in early rounds. Once the best expert appears in $\Phi'_i(t)$, its average disappointment quickly decreases.

Finally, Figure 8 shows the average payoffs obtained by the algorithms against these associates over 50,000 rounds. Even after 50,000 rounds, the enhanced expert algorithms all outperform the original algorithms. While UCB1 has the best performance of the four original algorithms after 50,000, all the enhanced algorithms have substantially higher payoffs. Thus, not only do the enhancements improve the algorithms in the near term, but also in the long term.



(a) Against A0 (b) Against Random (c) Against Bully

(d) Against Godfather (e) Against BR1 (f) Against BR2

(g) Against WoLF-PHC (h) Against Q-learning (i) Against Exp3

(j) Against UCB1 (k) Against EEE (l) Against S

Figure 6: Average disappointment $\bar{\mathcal{D}}^T$ over time against each associate across the selected games. Results are an average of 50 trials in each of the ten selected games.

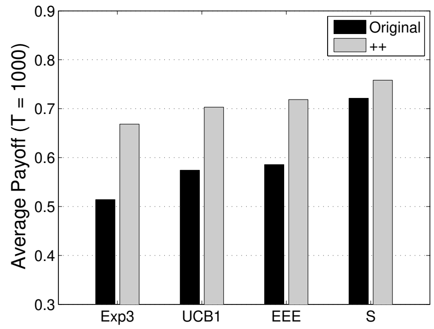


Figure 7: Average payoffs in self play over 1000 rounds across all selected games.

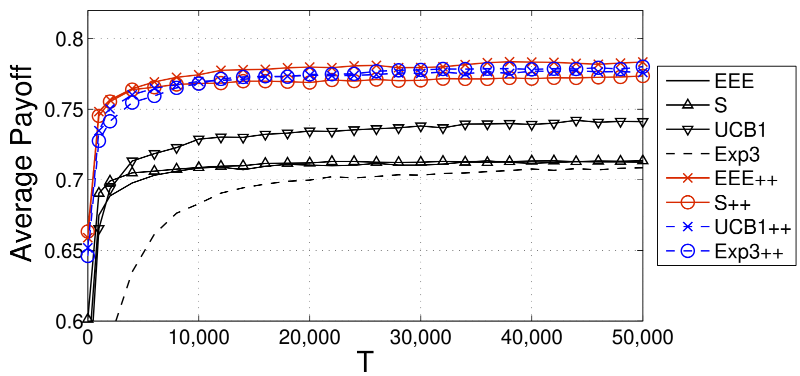
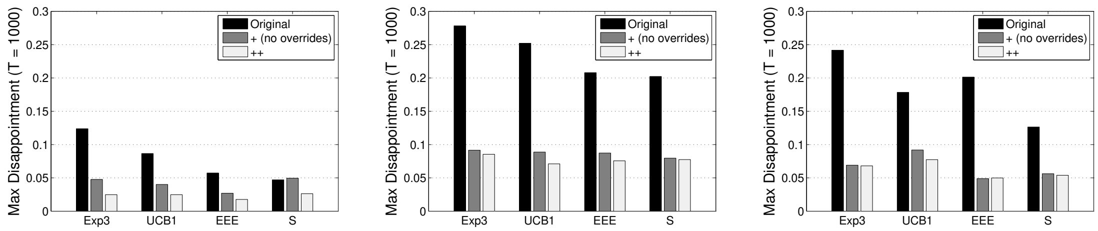


Figure 8: Average payoffs over time across all selected games and associates.

6.2 Why the Meta-Algorithm Works

The meta-algorithm has two different components: (1) a method for distinguishing experts that are likely to be successful from those that are not and (2) best response and safety overrides. The combined impact of the overrides on the max disappointment of the enhanced algorithms is shown in Figure 9. While the overrides typically lower the algorithms' max disappointment by a small margin, only in a small number of cases (e.g., S against static agents) are the overrides responsible for substantial improvements. In many scenarios, the overrides are not invoked.

Thus, the meta-algorithm improves the original algorithms primarily via its mechanism for distinguishing successful experts from unsuccessful experts. The original expert algorithms tend to spend many rounds following ineffective experts as they seek to determine from experience which experts are most effective. The resulting frequent changes in behavior caused by cycling through many experts are incoherent to associates, making it difficult for the agents to coordinate behavior or strike mutually beneficial compromises. On the other hand, the enhanced algorithms select from fewer experts in early rounds of the game, which produces more predictable behavior that associates can more easily model and adapt to. This, in turn, allows them to quickly find mutually beneficial compromises, which lead to higher payoffs both in the short and the long term.



(a) Against static algorithms (b) Against RL algorithms (c) Against expert algorithms

Figure 9: Max disappointment $\hat{\mathcal{D}}^{1000}$ after 1000 rounds against each class of associates.

6.3 Comparison to Top-Performing Algorithms

We have shown that our meta-algorithm helps expert algorithms quickly achieve and maintain low disappointment against a variety of static, reinforcement learning, and expert algorithms across many repeated games. To further illustrate the value of this contribution, we now compare the average payoffs of these enhanced algorithms to that of top-performing algorithms in repeated games.

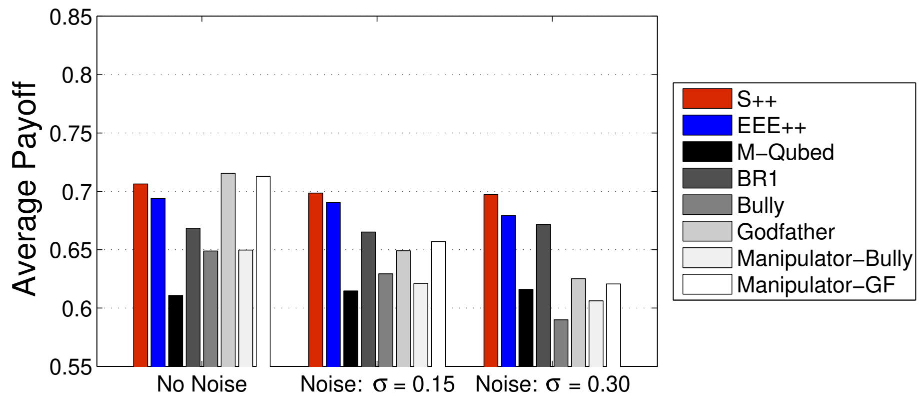
To compare our algorithms with state-of-the-art learning algorithms, we ran several round-robin tournaments involving eight algorithms: S++, EEE++, and six algorithms from the literature: M-Qubed, Manipulator-Bully, Manipulator-Godfather, BR1, Godfather, and Bully (see Appendix A). These algorithms were chosen due to their elite attributes. For example, Bully and Godfather are leader algorithms with well-understood equilibrium characteristics (Littman & Stone, 2001, 2005). These algorithms precompute desirable behaviors, and are good standards of performance in early rounds of repeated games. M-Qubed is a reinforcement learning algorithm that has demonstrated superior asymptotic performance in several empirical studies (Crandall & Goodrich, 2011; Bouzy & Metivier, 2010). This makes it a good standard of comparison for long-term (or asymptotic) performance. Manipulator combines Godfather or Bully with BR1 and the maximin strategy π_i^{mm} to provide theoretical guarantees with respect to targeted optimality, safety, and compatibility (Powers & Shoham, 2005a).

To this point, we have assumed perfect information, wherein each player has perfect knowledge of the other player's payoffs. We now relax this assumption and consider scenarios in which the players have normally distributed errors in their assessments of the other player's payoffs. Thus, we conducted three separate round-robin tournaments: one with perfect information (*No Noise*), and two with different sizes of errors in payoff assessment ($\sigma = 0.15$ and $\sigma = 0.30$, respectively).

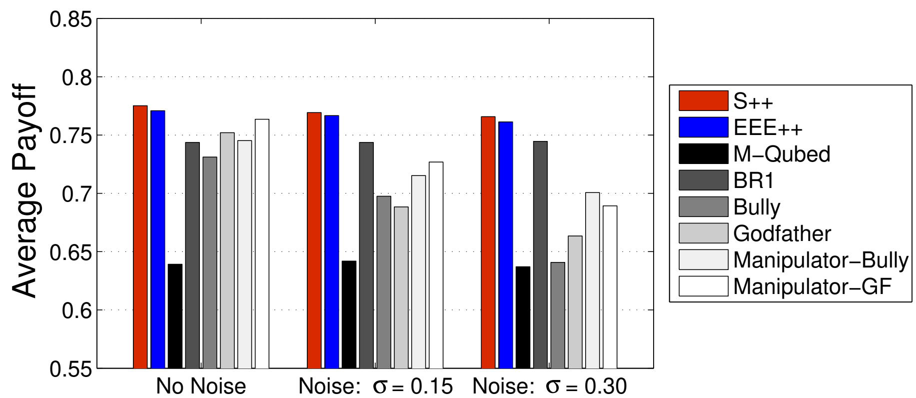
In each tournament, each algorithm was paired with itself and the other seven algorithms in 50 random 3-action repeated games. Each game consisted of 50,000 rounds. We compare the algorithms by their average per-round payoff over each of these eight pairings in both the short term (over the first 100 and 1000 rounds) and asymptotically (over the last 1000 rounds). While random games tend to produce less variation in average payoffs than do selected games[6], we use random games as a guard against overfitting the selected games.

The results of the tournaments are summarized in Figure 10.

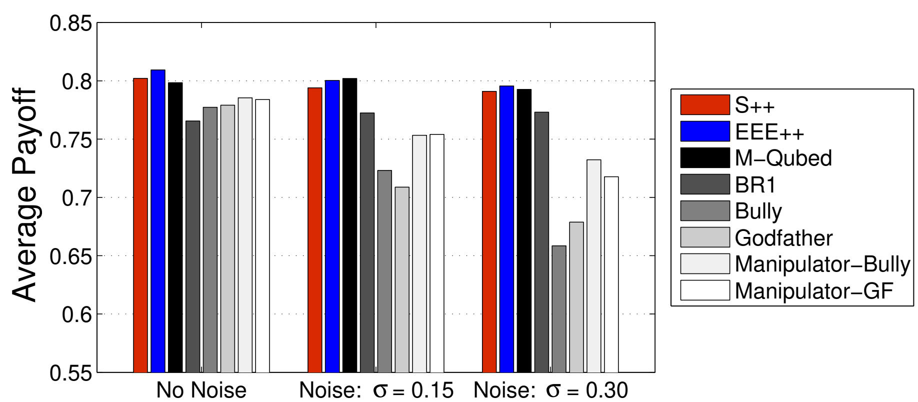
6. One comparison of 16 algorithms showed just a 0.07 difference in the average payoffs between the best- and worst-performing algorithms (Crandall & Goodrich, 2011)



(a) Averaged over the first 100 rounds



(b) Averaged over the first 1000 rounds



(c) Averaged over rounds 49,001 – 50,000

Figure 10: Average payoffs in perfect (No Noise) and imperfect (Noise: $\sigma = 0.15$ and $\sigma = 0.30$, respectively) information round-robin tournaments involving 50 random 3-action games.

6.3.1 PERFECT INFORMATION

Unsurprisingly, Godfather and Manipulator-GF (which are identical in the first 100 rounds) had the highest average payoffs over the first 100 rounds (Figures 10a). However, the average payoffs of S++ and EEE++ under perfect information were not far behind. T-tests show that the difference in average performance between S++ and Godfather was not statistically significant after 100 rounds ($p = 0.194$), though the difference between EEE++ and Godfather was ($p = 0.004$). The payoffs of the other algorithms were substantially lower. By 1000 rounds, both S++ and EEE++ outperformed all the other six algorithms under perfect information (Figure 10b). Through 1000 rounds, the difference between S++ and EEE++ was not statistically significant, though the differences between S++ and each of the other six algorithms was ($p < 0.010$). Thus, the short-term performances of S++ and EEE++ were very favorable compared with these other algorithms under perfect information.

The long-term performance of S++ and EEE++ matches that of M-Qubed in these perfect information games (Figure 10c). Given M-Qubed's top asymptotic performance in previous studies (Crandall & Goodrich, 2011; Bouzy & Metivier, 2010), these results speak to the effectiveness of the enhanced algorithms.

6.3.2 IMPERFECT INFORMATION

Both Godfather and Bully compute equilibrium strategies based on knowledge of the payoffs of associates. By using precomputed strategies, these algorithms quickly achieve high payoffs. M-Qubed and BR1, on the other hand, seek to learn effective behaviors from experience. While M-Qubed is eventually quite effective, its learning processes often takes many rounds. S++, EEE++, Manipulator-Godfather, and Manipulator-Bully each do some of both. They combine the computation of equilibria before the game begins with learning from experience. As a result, they tend to perform effectively both in the short-term and the long-term under perfect information.

However, since computing (and agreeing upon) an equilibrium sometimes requires perfect information, these algorithms are potentially limited given that perfect knowledge of an associate's payoffs is uncommon. However, it is not uncommon for agents to have estimates of the payoffs of others, though sometimes these estimates are in error. We model such scenarios in the imperfect information tournaments. The results of these tournaments are also shown in Figure 10.

The figure shows that in both early and later rounds of the game, Godfather, Bully, Manipulator-Godfather, and Manipulator-Bully are all negatively impacted by errors in their assessments of their associates' payoffs. Their average payoffs fall substantially (and statistically significantly) for both medium errors in assessment ($\sigma = 0.15$) and large errors in assessment ($\sigma = 0.30$). On the other hand, S++ and EEE++ are only slightly affected by both moderate and high errors in their assessments. Thus, they continue to perform on par with M-Qubed asymptotically (Figure 10c), while maintaining the highest performance in early rounds of the game.

While the Manipulator algorithms, S++, and EEE++ all utilize precomputed strategies and learn from their experiences, S++ and EEE++ are not substantially affected by imperfect information, while the Manipulator algorithms are. There are two primary differences between the Manipulator algorithms and the enhanced expert algorithms that cause this difference. First, the Manipulator algorithms each compute a single equilibrium strategy, while the enhanced expert algorithms compute many equilibria strategies (the various experts). Second, the Manipulator algorithms execute experts serially. These algorithms first execute their respective leader strategy. If the respective leader strategy fails to produce desired payoffs, they switch to following BR1, and so on. On the other hand, S++ and EEE++ continue to evaluate multiple experts (essentially in parallel). As a results, these algorithms are much more robust to errors in the assessments of their associate's payoffs.

7. Conclusions and Discussion

In this paper, we introduced a new metric, called disappointment, for evaluating expert algorithms in repeated games. Disappointment is similar to regret, except that disappointment is not built on the assumption that an agent's actions do not influence its associates' future behavior. As a result, minimizing disappointment is always equivalent to maximizing accumulated payoffs in a repeated game, whereas this is not the case with regret. While we showed that it is impossible to create an algorithm that is guaranteed to have no disappointment in all scenarios without omniscience, it is possible to create algorithms that quickly achieve (and maintain) low disappointment against many algorithms in many repeated games.

To accomplish this goal, we presented a new meta-algorithm that can be used to enhance existing expert algorithms. This algorithm reduces the set of selectable experts by combining aspiration learning and equilibrium computation. We showed that the resulting algorithms quickly achieve and maintain low disappointment when associating with various static algorithms, reinforcement learning algorithms, and expert algorithms. We also showed that these expert algorithms, given a good set of experts, outperform top-performing algorithms from the literature.

7.1 Reflections on Learning Using Experts

The meta-algorithm we presented for enhancing expert algorithms alters the order in which experts are selected. It does this by computing the highest expected payoff (or *potential*) of each expert, and then supplying the expert algorithm with the subset of experts whose potential meets some *performance threshold*, which is determined by aspiration learning. Experts with lower potential are only selected once experts with higher potential have demonstrated an inability to meet their potentials. This application of the optimism-in-uncertainty principle (Brafman & Tennenholtz, 2003) allows expert algorithms to learn effective strategies very quickly while ensuring that the expert algorithm has access to the best expert in the long term.

Several previously proposed algorithms (e.g., Powers & Shoham, 2005a, 2005b; Knobbout & Vreeswijk, 2011) select experts in a serial fashion, beginning with the expert with the highest potential. Our results under imperfect information advocate for a more integrated approach in which the experts are evaluated in parallel. This allows the agents to better negotiate leader and follower roles, which in turn leads to better chances for profitable cooperation and compromise.

7.2 Extensions to Stochastic Games

The expert algorithms we have discussed and analyzed can also be applied to two-player repeated stochastic games in at least two different ways. First, Pepper (Crandall, 2012) can be used to extend any algorithm designed for repeated normal-form games to repeated stochastic games. Pepper uses a separate instance of a learning algorithm designed for normal-form games in each stage game of the stochastic game. Future work should determine how quickly our enhanced expert algorithms learn in stochastic games when extended with Pepper.

Second, experts can be as complex as is necessary. Several algorithms for computing equilibrium strategies in stochastic games have recently been developed (e.g., Cote & Littman, 2008;

Johanson et al., 2012). These and other equilibrium strategies could define the set of experts used by the enhanced expert algorithms in stochastic games. Future work involves identifying an effective set of experts for stochastic games.

Acknowledgments

I would like to thank three anonymous reviewers who provided detailed and constructive feedback. Their suggestions, questions, and comments greatly improved the paper.

Appendix A. Specification of Algorithms

Table 5 states the parameters and settings used by each algorithm used in the paper. Each algorithm was carefully analyzed to ensure that it behaved as defined. Parameters for the algorithms were selected by balancing two objectives. First, we desired the algorithms to perform as they were intended by the algorithms' authors. Second, we sought to optimize each algorithm's short-term performance (i.e., the first 1000 rounds) while not compromising long-term performance.

Since our implementation of S as an expert algorithm is not provided in the literature (though we maintain the general principles of the algorithm), we provide further details of that algorithm. Though not originally designed as an expert algorithm, S has been shown to be effective when learning to select among learning experts in repeated games (Bouzy et al., 2011). S learns an *aspiration level*, and then searches for an expert that obtains payoffs that meets this aspiration.

S sets its initial aspiration α_i^1 to one, its highest payoff. It then randomly selects some expert $\phi_i \in \Phi_i$, and follows ϕ_i until a joint action is played twice. This is determined by comparing the latest joint action with H_c, the set of joint actions played so far. α_i^t is then updated as follows:

$$\alpha_i^t \leftarrow (\lambda_i)^{|H_c|}\alpha_i^{t-|H_c|} + (1-(\lambda_i)^{|H_c|})\bar{r}_i^t, \tag{18}$$

where $\lambda_i \in (0,1)$ is the learning rate and $\bar{r}_i^t$ is the average payoff obtained by agent i since the last expert was selected. After updating α_i^t, S selects a new expert:

$$\phi_i^t \leftarrow \begin{cases} \phi_i^{(t-|H_c|)} & \text{with prob. } f(\alpha_i^t, \bar{r}_i^t) \\ \text{random}(\Phi_i) & \text{otherwise} \end{cases} \tag{19}$$

Here, random(Φ_i) denotes a random selection from Φ_i, and $f(\alpha_i^t, \bar{r}_i^t)$ is the agent's inertia given by

$$f(\alpha_i^t, \bar{r}_i^t) = \min(1, \left(\bar{r}_i^t/\alpha_i^t\right)^{|H_c|}). \tag{20}$$

Eq. (20) specifies that the agent selects the expert that it played in the previous episode if $\bar{r}_i^t$ meets or exceeds α_i^t. If not, it randomly selects a new expert with probability $\left(\bar{r}_i^t/\alpha_i^t\right)^{|H_c|}$. H_c is then reset to include only the last joint action played, and the process repeats.

Appendix B. Performance of Individual Experts

To illustrate the effectiveness of the experts defined in Section 3, we plot the running average payoff of each of the experts against three different associates in Chicken, PD, Offset, and Tricky (Tables 1 and 2). These results are shown in Figure 11. We make several observations. First, in each scenario,

Algorithm	Description and Parameters
A0	An agent that always selects its first action.
Random	Randomly selects its action from the uniform distribution over the action set A_i.
Bully	The leader strategy (Section 3.1) for the solution $\mathbf{s}^* = \arg\max_{\mathbf{s}\in\Omega'} v_i(\mathbf{s})$.
Godfather	The leader strategy (Section 3.1) for the solution $\hat{\mathbf{s}} = \arg\max_{\mathbf{s}\in\Omega'}(v_i(\mathbf{s}) - v_i^{\mathrm{mm}})(v_{-i}(\mathbf{s}) - v_{-i}^{\mathrm{mm}})$.

(a) **Static algorithms**

Algorithm	Description and Parameters
BR1	A model-based reinforcement learning algorithm that encodes its state as the previous joint action. The algorithm estimates the associate's behavior using the fictitious-play assessment conditioned on the current state (Eq. 10), and then computes a best response using value iteration (discount factor $\gamma = 0.95$). It uses ε-greedy exploration, $\varepsilon = \frac{1}{10+t/10}$.
BR2	Identical to BR1 except that it encodes state as the previous two joint actions.
WoLF-PHC	See Bowling and Veloso (2002). For Figure 2, $\alpha = \frac{1}{100+t/10000}$, $\varepsilon = 0.05$, $\delta_l = \frac{4}{20000+t}$, $\delta_w = \frac{1}{20000+t}$. Otherwise, $\alpha = \frac{1}{10+t/100}$, $\varepsilon = \frac{1}{10+t/100}$, $\delta_l = \frac{2}{100+t/100}$, $\delta_w = \frac{1}{100+t/100}$
Q-learning	A model-free reinforcement learning algorithm proposed by Watkins (1992). Our implementation encodes state as the previous joint action and uses ε-greedy exploration. Q-values are initialized to the highest possible value $\frac{1}{1-\gamma}$. $\varepsilon = \frac{1}{10+t/10}$, $\gamma = 0.95$, $\alpha = \frac{1}{10+t/100}$

(b) **Reinforcement learning algorithms**

Algorithm	Description and Parameters
Exp3	An expert algorithm (Auer et al., 1995) with well-defined regret bounds for the multi-armed bandit problem. It was shown to be effective in some repeated games (Bouzy & Metivier, 2010; Crandall & Goodrich, 2011; Chang & Kaelbling, 2005). Our implementation evaluates an expert for $\omega = \|A_i\|\|A_{-i}\|$ rounds before selecting a new expert. $\eta = \gamma/\|\Phi_i\|$, $\gamma = \sqrt{\|\Phi_i\| ln(\|\Phi_i\|)/(e-1)T_0}$, where T_0 is the expected number of rounds in the game.
UCB1	An expert algorithm (Auer et al., 2002) with well-defined regret bounds for the adversarial multi-armed bandit problem. It has proved effective in some repeated games (Bouzy & Metivier, 2010). Our implementation evaluates an expert for $\omega = \|A_i\|\|A_{-i}\|$ rounds before selecting a new expert.
EEE	An ε-greedy expert algorithm designed for repeated games played against adaptive associates (de Farias & Megiddo, 2004). $\omega = \|A_i\|\|A_{-i}\|$, $\varepsilon = \frac{1}{10+t/10}$
S	An aspiration-based algorithm original proposed by Karandikar et al. (1998), and also analyzed by Stimpson et al. (2001). See the text of Appendix A. $\lambda_i = 0.99$

(c) **Expert algorithms**

Algorithm	Description and Parameters
M-Qubed	An RL algorithm with the highest asymptotic performance in several studies (Crandall & Goodrich, 2011; Bouzy & Metivier, 2010). Parameters set as in the work of Crandall and Goodrich (2011).
Manipulator-Bully	A la Powers and Shoham (2005a), this algorithm serially follows three experts. For the first 150 rounds, it follows Bully. Thereafter, if its payoffs become lower than expected, it switches to BR1. After 300 rounds, if its payoffs ever drop below $v_i^{\mathrm{mm}} - \varepsilon$, it plays ϕ_i^{mm} thereafter.
Manipulator-Godfather	A la Powers and Shoham (2005a), this algorithm serially follows three experts. For the first 150 rounds, it follows Godfather. Thereafter, if its payoffs become lower than expected, it switches to BR1. After 300 rounds, if its payoffs ever drop below $v_i^{\mathrm{mm}} - \varepsilon$, it plays ϕ_i^{mm} thereafter.

(d) **Other algorithms: standards of comparison**

Table 5: Algorithmic parameters used in the paper.

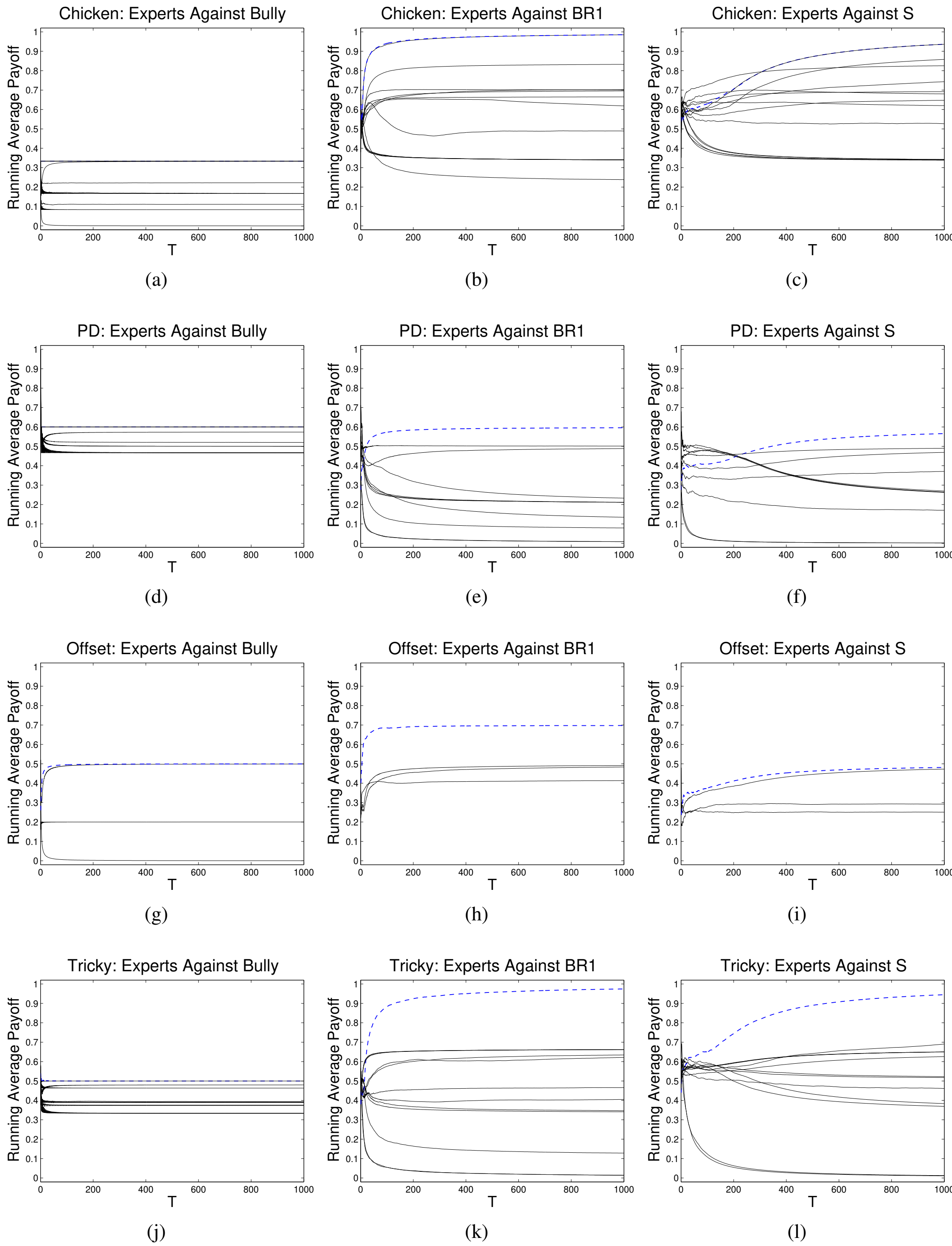


Figure 11: Running average payoffs of each expert against BR1, S, and Bully in four different games. Results are an average of 50 trials.

there is at least one expert that performs well against that opponent. For example, in Chicken, an ideal algorithm should be able to eventually achieve an average payoff near 1 against both BR1 and S after many rounds. Several experts achieve this performance level (Figures 11b and 11c). Against Bully, the best possible average payoff is 0.33, which several experts achieve (Figure 11a). Furthermore, in the PD, there is an expert that eventually obtains the average payoff of mutual cooperation (0.6) against each associate (Figures 11d–11f).

Second, Figure 11 illustrates that different payoffs are possible against the different associates. For example, in both Chicken and Tricky, the best expert against Bully has substantially lower payoffs (despite playing optimally) than the best experts against BR1 and S. We note that disappointment normalizes performance in each of these cases so that an average disappointment of zero indicates effective play relative to this set of experts.

Third, reaching the highest possible payoff sometimes requires a lot of patience. For example, against S in Chicken (Figure 11c), the best expert after 100 rounds is not the best expert after 1000 rounds. Only after repeatedly following the same expert for many rounds does the associate learn to accept the equilibrium offered by this expert. This illustrates how difficult it is for expert algorithms to maintain low disappointment over time.

References

Arora, R., Dekel, O., & Tewari, A. (2012). Online bandit learning against an adaptive adversary: from regret to policy regret. In *Proceedings of the 29th International Conference on Machine Learning*, pp. 1503–1510.

Auer, P., Cesa-Bianchi, N., & Fischer, P. (2002). Finite-time analysis of the multi-armed bandit problem. *Machine Learning*, *47*, 235–256.

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 Symposium on the Foundations of Computer Science*, pp. 322–331.

Axelrod, R. (1984). *The Evolution of Cooperation*. Basic Books.

Bouzy, B., & Metivier, M. (2010). Multi-agent learning experiments in repeated matrix games. In *Proceedings of the 27th International Conference on Machine Learning*, pp. 119–126.

Bouzy, B., Metivier, M., & Pellier, D. (2011). Hedging algorithms and repeated matrix games. In *ECML Workshop on Machine Learning and Data Mining in and Around Games*, Athens, Greece.

Bowling, M. (2004). Convergence and no-regret in multiagent learning. In *Advances in Neural Information Processing Systems 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. *Journal of Machine Learning Research*, *3*, 213–231.

Cesa-Bianchi, N., Dekel, O., & Shamir, O. (2013). Online learning with switching costs and other adaptive adversaries. In *Advances in Neural Information Processing Systems 26*, pp. 1160–1168.

Chakraborty, D., & Stone, P. (2010). Convergence, targeted optimality, and safety in multiagent learning. In *Proceedings of the 27th International Conference on Machine Learning*, pp. 191–198.

Chang, Y., & Kaelbling, L. P. (2005). Hedge learning: Regret-minimization with learning experts. In *Proceedings of the 22nd International Conference on Machine Learning*, pp. 121–128.

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

Chasparis, G., Shamma, J., & Arapostathis, A. (2010). Aspiration learning in coordination games. In *Proceedings of the 49th IEEE Conference on Decision and Control*, pp. 5756–5761.

Cote, E. M. D., & Littman, M. L. (2008). A polynomial-time Nash equilibrium algorithm for repeated stochastic games. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence*, pp. 419–426.

Crandall, J. W. (2012). Just add Pepper: extending learning algorithms for repeated matrix games to repeated markov games. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, pp. 399–406.

Crandall, J. W., & Goodrich, M. A. (2011). Learning to compete, coordinate, and cooperate in repeated games using reinforcement learning. *Machine Learning*, *82*(3), 281–314.

Crandall, J. W., & Goodrich, M. A. (2005). Learning to teach and follow in repeated games. In *AAAI workshop on Multiagent Learning*, Pittsburgh, PA.

de Farias, D., & Megiddo, N. (2003). How to combine expert (or novice) advice when actions impact the environment. In *Advances in Neural Information Processing Systems 16*.

de Farias, D., & Megiddo, N. (2004). Exploration–exploitation tradeoffs for expert algorithms in reactive environments. In *Advances in Neural Information Processing Systems 17*, pp. 409–416.

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

Fudenberg, D., & Levine, D. K. (1998). *The Theory of Learning in Games*. The MIT Press.

Ganzfried, S., & Sandholm, T. (2011). Game theory-based opponent modeling in large imperfect-information games. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems*, pp. 533–540.

Gordon, G. J., Greenwald, A., & Marks, C. (2008). No-regret learning in convex games. In *Proceedings of the 25th International Conference on Machine Learning*, pp. 360–367.

Greenwald, A., & Jafari, A. (2003). A general class of no-regret learning algorithms and game-theoretic equilibria. In *Proceedings of the 16th Annual Conference on Computational Learning Theory*, pp. 2–12.

Greenwald, A., & Hall, K. (2003). Correlated Q-learning. In *Proceedings of the 20th International Conference on Machine Learning*, pp. 242–249.

Johanson, M., Bard, N., Lanctot, M., Gibson, R., & Bowling, M. (2012). Efficient Nash equilibrium approximation through Monte Carlo counterfactual regret minimization. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, pp. 837–846.

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

Knobbout, M., & Vreeswijk, G. A. (2011). Sequential targeted optimality as a new criterion for teaching and following in repeated games. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems*, pp. 517–524.

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.

Powers, R., & Shoham, Y. (2005a). Learning against opponents with bounded memory. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pp. 817–822.

Powers, R., & Shoham, Y. (2005b). New criteria and a new algorithm for learning in multi-agent systems. In *Advances in Neural Information Processing Systems 17*, pp. 1089–1096.

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

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