Learning ε-Pareto Efficient Solutions With Minimal Knowledge Requirements Using Satisficing

Jacob W. Crandall and **Michael A. Goodrich**
Computer Science Department
Brigham Young University
Provo, UT 84602
crandall, mike@cs.byu.edu

Abstract

Many problems in multiagent learning involve repeated play. As such, naive application of Nash equilibrium concepts are often inappropriate. A recent algorithm in the literature (Stimpson & Goodrich 2003) uses a Nash bargaining perspective instead of a Nash equilibrium perspective, and learns to cooperate in self play in a social dilemma without exposing itself to being exploited by selfish agents. It does so without knowledge of the game or the actions of other agents. In this paper, we show that this algorithm likely converges to near pareto efficient solutions in self play in most nonzero-sum n-agent, m-action matrix games provided that parameters are set appropriately. Furthermore, we present a tremble based extension of this algorithm and show that it is guaranteed to play near pareto efficient solutions arbitrarily high percentages of the time in self play for the same large class of matrix games while allowing adaptation to changing environments.

Introduction

Many applications in which multiagent learning can be applied require agents to cooperate with and adapt to each other on an ongoing basis. Thus, inherently, multiagent learning involves repeated play. This suggests that in many applications, the Nash equilibrium perspective should be replaced by a Nash bargaining perspective (Nash 1950; 1951), since learning a one-shot best-response (locally optimal) solution is not always desirable when agents interact repeatedly. It is, therefore, important that multiagent learning algorithms be developed that learn bargaining-like solutions. This entails that the algorithms learn to play pareto efficient or near pareto efficient solutions high percentages of the time when it is feasible, and have a solid fallback position when it is not. Such an approach also dictates that solutions be "fair."[1]

Many real-world applications require agents to learn in non-stationary environments where only partial information about the world and/or other agents is available. This dictates that, in addition to learning near pareto efficient solutions, multiagent learning algorithms should require minimal knowledge/information and be able to adapt to changes in the environment and other learning agents. Moreover, the solutions learned should have stability or equilibrium characteristics.

In this paper, we discuss a multiagent learning algorithm (which we call the S-Algorithm) presented by Stimpson and Goodrich (2003) that uses satisficing. Stimpson showed how the S-Algorithm in a multiagent social dilemma (MASD)[2] successfully learns pareto efficient solutions with high probability in self play while avoiding being exploited by selfish agents. This result is made more significant since the algorithm requires no knowledge of the structure of the game or the actions of other agents. We show that this algorithm is applicable not only in the MASD but also in most nonzero-sum repeated play matrix games.

Additionally, we introduce a trembled based extension of the S-Algorithm. This extension of the algorithm, while maintaining the S-Algorithms minimal knowledge requirements, allows agents to adapt to changing environments. Also, it allows agents to play mixed-strategy solutions over time.

Related Literature

To date, most multiagent learning algorithms have focused on converging to Nash equilibrium and best response strategies. Various algorithms have been shown to reach the Nash equilibrium with various degrees of success. Examples include (Littman 1994; Fudenberg & Levine 1998; Bowling & Veloso 2001; Conitzer & Sandholm 2003). While learning to play Nash equilibrium solutions is effective in some games, it is not always successful in other games such as social dilemmas since the Nash equilibrium is not guaranteed to be pareto efficient.

In response, Littman and Stone (2001) developed a "leader" algorithm that teaches a class of best-response agents to learn to play pareto efficient solutions in two-agent repeated play games. Although their algorithm does not necessarily work in self play, they concluded from their results that best response approaches are not always sufficient in multiagent contexts. They extended their work (Littman & Stone 2003) by developing the concept of a repeated play Nash equilibrium and a method to compute it. Their algorithm, however, requires full knowledge of the structure of

[1] We discuss the way that we use the term *fair* later in the paper.

[2] The MASD is equivalent to public goods games (Camerer 2003).

the game and observability of the actions of the other agent.

Terms and Definitions

Before proceeding, we provide some terms and definitions that we use in this paper. The set of actions that agent i can choose from is denoted by A_i. Let $a = (a_1, a_2, \ldots, a_n)^T$, where $a_i \in A_i$, be a joint action for n agents, and let $A = A_1 \times \cdots \times A_n$ be the set of joint actions of the agents. Let $a_i^t \in A_i$ denote the action played by agent i at time t, and let $a^t = (a_1^t, a_2^t, \ldots, a_n^t)^T$ be the vector denoting the joint action played by the n agents at time t. The payoff received by agent i as a result of the joint action a being played is given by $u_i(a)$.

A *strategy* for agent i is a distribution π_i over its action set A_i. Such a strategy may be a *pure strategy* (where all probability is placed on a single action) or a *mixed strategy* (otherwise). The joint strategy played by the n agents is $\pi = (\pi_1, \pi_2, \ldots, \pi_n)^T$ and, thus, $u_i(\pi)$ is the expected payoff for agent i when the joint strategy π is played. A *solution* is a particular joint strategy.

Definition 1 - Pareto Dominated A solution π is strictly pareto dominated if there exists a joint action $a \in A$ for which $u_i(a) > u_i(\pi)$ for all i and weakly pareto dominated if there exists a joint action $a \in A$ for which $u_i(a) \geq u_i(\pi)$ for all i.

Definition 2 - Pareto Efficient A solution π is weakly pareto efficient (PE) if it is not strictly pareto dominated and strictly PE if it is not weakly pareto dominated. Unless specified, the former (i.e. weakly PE) terminology is implied.

Pareto efficiency should be a main goal of an agent who is involved in repeated play. However, it is sometimes difficult and impractical to guarantee. We relax this goal slightly by seeking near pareto efficiency.

Definition 3 - ε-Pareto Dominated A solution π is strictly ε-pareto dominated if there exists a joint action $a \in A$ for which $u_i(a) > u_i(\pi) + \varepsilon$ for all i and weakly ε-pareto dominated if there exists a joint action $a \in A$ for which $u_i(a) \geq u_i(\pi) + \varepsilon$ for all i.

Definition 4 - ε-Pareto Efficient A solution π is weakly ε-pareto efficient (ε-PE) if it is not strictly ε-pareto dominated and strictly ε-PE if it is not weakly ε-pareto dominated. The former (i.e. weakly ε-PE) terminology is implied unless specified otherwise.

We now give names to various regions of the reward space of n agents. Let $\mathbb{R}^n$ denote the real-valued reward space. Let D be the subset of $\mathbb{R}^n$ that is weakly pareto dominated. Let $P = \mathbb{R}^n - D$. That is, P is the subset of $\mathbb{R}^n$ that is strictly PE. Also, let D_ε be the subset of $\mathbb{R}^n$ that is strictly ε-pareto dominated. Finally, let $P_\varepsilon = D - D_\varepsilon$.

The *pareto boundary* divides set D from set P in the payoff space $\mathbb{R}^n$ of the game. If a vector $v \in D$, then v is said to be below the pareto boundary. If $v \in P$, v is said to be above the pareto boundary. Likewise, the ε-pareto boundary divides set D_ε from set P_ε. If a vector $v \in D_\varepsilon$ or $v \in P_\varepsilon$, then v is said to be below or above the ε-pareto boundary respectively.

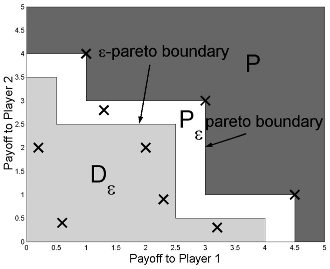


Figure 1: Diagram of the regions P, P_ϵ, and D_ϵ in the payoff space $\mathbb{R}^n$ for an arbitrary 3×3 matrix game with $\varepsilon = 0.5$.

Figure 1 shows (graphically) some of the above-defined terms for an arbitrary 3×3 matrix game with $\varepsilon = 0.5$. The x's depict the payoffs of the game.

It is also important to understand the way we use terms associated with satisficing. Let α_i^t be agent i's aspiration level at time t, and let $\alpha^t = (\alpha_1^t, \ldots, \alpha_n^t)$ be the joint aspirations of the agents at time t. A solution π is *satisficing* to agent i if $u_i(\pi) \geq \alpha_i^t$. Likewise, an agent is *satisfied* if the most recently played joint action was satisficing to it. A solution π is *mutually satisficing* at time t if $u_i(\pi) \geq \alpha_i^t$ for all i. Let S_i^t be the set of solutions that are satisficing to agent i at time t. Then, we have the *mutually satisficing set* (at time t) $S^t = S_1^t \cap S_2^t \cap \cdots \cap S_n^t$. The *satisficing region* is the region in $\mathbb{R}^n$ defined by S^t.

Nash bargaining is built on the concept of a fallback. In this context, we define a *fallback strategy* as a strategy an agent adopts when it is not satisfied with its payoffs. An agent's *fallback position* is the expected payoff associated with the agent's fallback strategy. The fallback position is important in determining the "fairness" of a solution. Thus, fallbacks are an important part of an agent's strategy.

The S-Algorithm in Nonzero-Sum Matrix Games

The S-Algorithm extends an algorithm developed by Karandikar *et al.* (1998) that guarantees that agents will converge to a cooperative solution in a small set of two-agent, two-action games. Stimpson and Goodrich (2003) extended the algorithm to an n-player, m-action ($n, m \geq 2$) multiagent social dilemma (MASD) and showed that it likely converges to a PE solution in this game. In this section we show the S-Algorithm can be guaranteed to converge with arbitrarily high probability to ε-PE solutions in most nonzero-sum games.

The S-Algorithm (shown in Table 1) uses an aspiration relaxation search to find pareto efficient solutions. An agent employing the S-Algorithm begins by setting its aspirations high. Thereafter, it moves its aspirations closer to the reward it receives at each iteration. If the agent is not satisfied with the reward it receives on a given iteration, it plays randomly on the subsequent iteration. If it is satisfied, it repeats the action it played previously. An agent employing this algorithm requires only the knowledge of the actions it can take. It does not need to know the actions of other agents or the payoff structure of the game.

```
1. Let $\lambda \in (0,1]$ be a learning rate. Initialize,
       $sat = \text{false} \qquad \alpha_i^0 = \text{high}$
2. Repeat
   (a) Select an action $a_i^t$ according to the following criteria:
           $a_i^t \leftarrow \begin{cases} a_i^{t-1} & \text{if } (sat) \\ \text{rand}(\mathcal{A}_i) & \text{otherwise} \end{cases}$
       where $\text{rand}(\mathcal{A}_i)$ denotes a random selection from the set $\mathcal{A}_i$.
   (b) Receive reward $r_t$ and update:
       i.  $sat \leftarrow \begin{cases} \text{true} & \text{if } (r_t \geq \alpha_i^t) \\ \text{false} & \text{otherwise} \end{cases}$
       ii. Update aspiration level
               $\alpha_i^{t+1} \leftarrow \lambda \alpha_i^t + (1-\lambda) r_t \qquad (1)$
```

Table 1: The S-Algorithm for agent i.

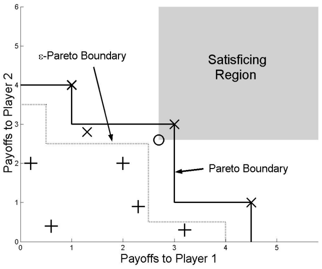


Figure 2: The payoff space of a 2-agent, 3-action game.

To better understand the S-Algorithm, consider Figure 2, which shows the payoff space of a 2-agent, 3-action game (shown with $\varepsilon = 0.5$). The payoffs to player 1 are plotted as x-coordinates, and the payoffs to player 2 are plotted as y-coordinates. In the figure, x's indicate ε-PE solutions, +'s indicate non-ε-PE solutions, and the circle (o) indicates the the current aspirations of the agents. The gray rectangle is the satisficing region. At the time shown in the figure, the only solution in the satisficing region is ε-PE. Once the solution in the satisficing region is played, it will be played every iteration thereafter. However, if it is not played by the agents, aspirations are likely to relax until the satisficing region contains a (pure strategy) solution that is not ε-PE, which would make convergence to a non-ε-PE solution possible.

The S-Algorithm incorporates a fallback strategy that endorses exploration. When an agent is not satisfied, it resorts to playing randomly (exploring). While exploring, an S-Algorithm agent modifies its aspirations (this allows compromise) until its aspirations are satisfied. In self-play, this generally results in convergence to solutions quite related to those Littman proposes (Littman & Stone 2003), but only in the pure-strategy domain. Against non-negotiating agents, the S-Algorithm eventually learns with high probability to cease negotiations and "defect" as well.

Table 1 specifies only that initial aspirations should be "high." We make the follow assumptions on initial aspirations $\alpha^0 = (\alpha_1^0, \ldots, \alpha_n^0)$. First, $\alpha^0 \in P$, which means that initial aspirations must be above the pareto boundary. Second, $\forall i \; \alpha_i^0 \geq \frac{1}{m^n} \sum_{a \in A} u_i(a)$, which means that each agent's initial aspiration must be at least as high as the average of all the payoffs it can receive. Third, $\alpha_i^0 \geq \min u_i(a), a \in P_\varepsilon$, which means that each agent's initial aspiration must be at least as high as the reward it would receive if any pure strategy ε-PE solution were played.

Convergence to ε-PE solutions

In this subsection, we give two theorems that apply to most nonzero-sum matrix games. The first guarantees that the S-Algorithm (in self-play) will converge (given appropriate initial aspirations) with at least probability q to an ε-PE solution. The second shows that the probability q can be made arbitrarily close to one by adjusting the learning rate (λ).

First, we provide a lemma that establishes a bound (δ) on the maximum change that can occur in an agent's aspirations during a single time step. Such a result is useful because it allows us to put a bound on the minimum (worst case) amount of time T_p that only ε-PE solutions will be mutually satisficing.

Lemma 1. *There exists a δ that bounds the absolute change in an agent's aspirations from time t to time $t+1$.*

Proof. The absolute change in an agent's aspirations is given by $|\alpha_{t+1} - \alpha_t|$. The relationship between α_{t+1} and α_t is given in (1) and is the greatest when the difference between α_t and r_t is the greatest. This occurs when $\alpha_t = h_\$$ and $r_t = l_\$$, where $h_\$$ and $l_\$$ are the highest and lowest payoffs an agent can receive from the game. Plugging these values into (1), we get $\alpha_{t+1} = \lambda h_\$ + (1-\lambda) l_\$$. Thus, $\delta = |\alpha_{t+1} - \alpha_t| = |(\lambda h_\$ + (1-\lambda) l_\$) - \alpha_t|$. Substituting $h_\$$ for α_t and simplifying, we have

$$\delta = |(1-\lambda)(l_\$ - h_\$)| = (1-\lambda)(h_\$ - l_\$). \qquad (2)$$

□

Since λ can be set to any value between 0 and 1, Equation (2) indicates that δ can be made as small as needed. Thus, λ can be set to guarantee that only ε-PE solutions will be mutually satisficing for at least a minimum amount of time T_p.

Lemma 2. *The minimum (worst case) amount of time T_p that only ε-PE solutions will be mutually satisficing is given by*

$$T_p \geq \frac{\varepsilon}{\delta}. \qquad (3)$$

Proof. Since all solutions within ε of the pareto boundary are ε-PE, the bound follows trivially from Lemma 1. □

We note that $h_\$$ and $l_\$$ may not be known. When this is the case, these values can be determined (at least with high probability) during an exploratory period at the beginning of play. The necessary length of the exploratory period depends on the number of agents n and the number of actions m in the game. To assure that this exploratory period

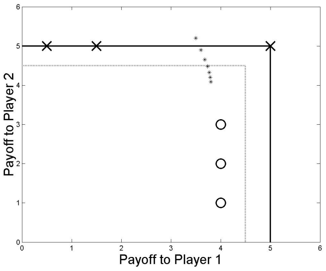


Figure 3: Example of a problematic game for the S-Algorithm. The aspiration trail (*'s), the pareto boundary (solid lines) and the ε-pareto boundary (doted lines) are also shown.

does not interfere with the time steps of T_p, initial aspirations should be set sufficiently above the pareto boundary. We leave further exploration of this subject to future work.

In order to show that the S-Algorithm will converge to an ε-PE solution with at least probability q, it must be shown that a solution in the satisficing region may be played with some nonzero probability during some subset of the time steps of T_p. This, however, cannot be achieved for every nonzero-sum matrix games as demonstrated by the game shown in Figure 3. In the figure, solutions of the matrix game are marked with X's and O's. The aspiration trail is shown by *'s. In this game, agent 1 can select between the actions x and o. If agent 1 selects action x, then one of the solutions marked X will be played. If agent 1 plays o, then one of the actions marked O will be played. If aspirations begin above the pareto boundary as shown, once agent 1 plays action o, it will always be satisfied with the payoffs it receives (and, thus, will continue to play action o), independent of what agent 2 does. Since none of the "O" solutions are ε-PE (if $\varepsilon < 1$), an ε-PE solution will never be played. Thus, we cannot guarantee that the S-Algorithm will converge to an ε-PE solution in this game.

Below we give a lemma that bounds the minimum number of time steps T that a) only ε-PE solutions are mutually satisficing and b) one of these mutually satisficing solutions has a nonzero probability of being played.

Lemma 3. *Let ρ be the percentage of time steps of T_p that a mutually satisficing ε-PE solution has a nonzero probability of being played while $\alpha^t = (\alpha_1^t, \ldots, \alpha_n^t) \in P_\varepsilon$ (i.e., the aspirations of the agents are ε-PE but not PE). Then we have*

$$T \geq \frac{\rho\varepsilon}{\delta}. \tag{4}$$

Proof. This bound follows trivially from Lemma 2. □

When $\rho = 0$, such as is posibble in games such as the one shown in Figure 3, we possibly have a game in which we can make no guarantee about the convergence of the S-Algorithm (in self play) to an ε-PE solution. This generally occurs when an agent has an action that always yields a constant, relatively high (non-ε-PE) payoff regardless of the actions taken by other agents playing the game. Additionally, seemingly uninteresting (rare) games may include cycles which also may cause ρ to be zero. In general, the agents still learn when $\rho = 0$ to play solutions that yield good payoffs (payoffs that are at least as high as their fallback positions). However, the solutions are not ε-PE, so agents could do better by learning a different solution. Changing the initial aspiration vector changes whether these games have a ρ greater than zero. All matrix games have initial joint aspiration vectors that cause ρ to be nonzero for some learning rate λ.

Since we cannot guarantee an initial aspiration vector that guarantees $\rho > 0$, we deal in this paper only with the large class of non-zero sum games in which $\rho > 0$ for all legal initial joint aspirations vectors, and leave analysis of the S-Algorithm in the other small class of non-zero sum games to future work. The set of games in which $\rho > 0$ for all legal inital joint aspirations vectors includes, among others, important games such as the prisoners' dilemma, battle of the sexes, and chicken. In these games, the learning rate λ can be set such that the S-Algorithm (in self play) is guaranteed to converge to an ε-PE solution with probability q. We state and prove this below.

Theorem 1. *In an n-agent, m-action*[3] *repeated play game, if $\alpha^0 \in P$, $\rho > 0$, and the bound on the learning rate (δ; see Lemma 1) satisfies*

$$\delta \leq \frac{\rho\varepsilon \ln\left(1 - \frac{1}{m^n}\right)}{\ln(1 - p_n)},$$

then the S-Algorithm (in self play) is guaranteed to converge to an ε-PE solution with at least probability q.

Proof. Three different things can happen. In case 1, we have $\alpha^t \in P$ for all t. This can only occur if the average payoffs $v = (v_1, v_2, \ldots, v_n)^T$, where v_i is the average payoff (i.e., fallback position) of agent i, is such that $v \in P$. This means that the mixed strategy solution caused by each agent playing its fallback is PE, so the S-Algorithm has converged to a PE mixed strategy solution and we are finished. In case 2, we have that α (the joint aspirations) falls below the pareto boundary, but then rises above it again. If this happens, we have case 1, unless α again falls below the pareto boundary at some future time (in which case we have case 3). In case 3, α falls below the pareto boundary and stays there until a mutually satisficing solution is found. It is with this case that we are concerned.

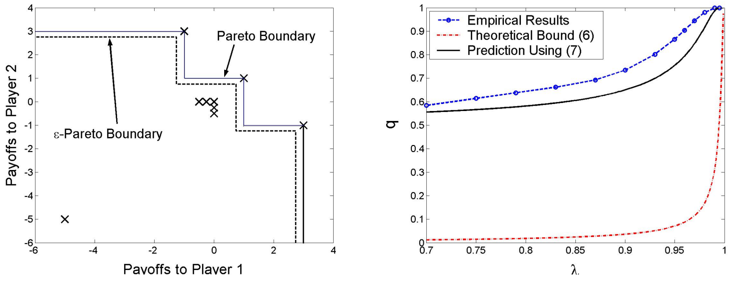
As mentioned previously, there exists at least $T \geq \frac{\rho\varepsilon}{\delta}$ time steps for which a) only pure strategy solutions that are ε-PE are mutually satisficing and b) the probability that one of these solutions will be played (p_p) is nonzero. We cannot be sure how many of such solutions are present, however, we know that there is at least one. Therefore, at each time step

[3] In many games, the number of actions is different for each agent. We assume that the number of actions for each agent in the game is the same. The proofs in this paper for games in which this assumption does not hold follow trivially.

Chicken (Extended)

	C	D	O
C	(1,1)	(-1,3)	(0,-0.25)
D	(3,-1)	(-5,-5)	(0,-0.5)
O	(-0.25,0)	(-0.5,0)	(0,0)

(a)



(b)

(c)

Figure 4: (a) Game matrix (b) Game's payoff space (c) Relationship between λ and q

of T, $p_p \geq \frac{1}{m^n}$.[4] From this it is straight-forward to observe that the probability $q' = 1 - q$ that an ε-PE solution will not be chosen during the T time steps has an upper bound of

$$q' = (1 - p_p)^T = \left(1 - \frac{1}{m^n}\right)^T .$$

Substituting $T \geq \frac{\rho\varepsilon}{\delta}$ and $q' = 1 - q$ and solving for q yields

$$q \geq 1 - \left(1 - \frac{1}{m^n}\right)^{\frac{\rho\varepsilon}{\delta}} .$$

Since all the variables are fixed on the right-hand side of the equation except δ, the lower bound on q is determined by δ. Solving for δ we have

$$\delta \leq \frac{\rho\varepsilon \ln\left(1 - \frac{1}{m^n}\right)}{\ln(1-q)}. \tag{5}$$

This means that for any nonzero-sum matrix game in which $\rho > 0$, there exists a δ that bounds q. □

Theorem 2. *If $\rho > 0$, the probability q that the S-Algorithm (in self play) converges to an ε-PE solution can be made arbitrarily close to one by setting the learning rate λ appropriately.*

Proof. The result follows trivially from (2) and (5), and yields

$$\lambda \geq 1 - \frac{\rho\varepsilon \ln\left(1 - \frac{1}{m^n}\right)}{\ln(1-q)(h_\$ - l_\$)}. \tag{6}$$

Thus, q can be made arbitrarily close to one. □

The above proofs serve to illustrate that the S-Algorithm can be guaranteed to converge with arbitrarily high probability to ε-PE solutions in self play. Additionally, they show that λ must approach one exponentially as n increases and polynomially as m increases. However, the bounds they establish (see Equations (5) and (6)) are tight bounds only for a worst case game. They are quite pessimistic for many other games. Depending on the game, various assumptions can be made to better approximate the relationship between λ and q. First, for many practical games, the bound on δ given in Theorem 3 can be replaced by the average estimate $\delta = (1-\lambda)(h_\$ - \text{fb})$, where fb is an agent's fallback position and can be estimated as $\text{fb} = \frac{1}{m^n}\sum_{a \in A} u_i(a)$. Second, the minimum distance γ from the pareto boundary to any non-ε-PE solution is likely greater than ε (and never smaller). Thus, if γ is known, it can be used in place of ε to tighten the bound. Third, in most games, there is a high probability p_a that the S-Algorithm will settle on a ε-PE solution even if it does not do so during the time interval T. A reasonable modeling assumption is $p_a = 0.5$. These assumptions yield

$$q = 1 - (1 - p_a)\left(1 - \frac{1}{m^n}\right)^T , \tag{7}$$

where $T = \frac{\rho\gamma}{(1-\lambda)(h_\$ - \text{fb})}$.

The accuracy of (7) for a 2-agent, 3-action game of chicken is shown in Figure 4. The matrix game, shown in Figure 4(a), is the same as the traditional 2-action chicken game except that agents can also "opt out" (O). Figure 4(c) compares the relationship between λ and q as predicted by (6) and (7) with the actual relationship obtained from empirical results. For the plots, we assumed $\rho = 1$. While the theoretical bound from (6) calls for pessimistic values of λ in many games, (7) gives a more reasonable estimate.

Discussion on the S-Algorithm

We showed in the previous subsection that the S-Algorithm, in self play, converges to ε-PE solutions with high probability if parameters are set appropriately. Furthermore, it is able to do so without knowledge of the game structure or the actions of other agents. On the downside, the S-Algorithm does not have the ability to adapt to changing environments. This is because a changing environment means, essentially, a new game, so the S-Algorithm must begin its aspirations above the pareto boundary in order to be able to guarantee a high probability of convergence to ε-PE solutions. Also, the S-Algorithm generally only learns pure strategy solutions.

Satisficing with Trembles

Karandikar *et al.* (1998) offer an alteration to their algorithm that incorporates trembles in aspirations. In their approach,

[4] Note that since m^n is the number of pure strategy solutions in the game, $p_p \geq \frac{1}{m^n}$.

```
1. Let $\lambda \in (0, 1]$ be a learning rate and let $\eta$ be the probability
    of trembling. Initialize,
      $sat = \text{false} \qquad \alpha_i^0 = \text{random}$
2. Repeat
    (a) Select an action $a_i^t$ according to the following criteria:
          $a_i^t \leftarrow \begin{cases} a_i^{t-1} & \text{if } (sat) \\ \mathrm{rand}(\mathcal{A}_i) & \text{otherwise} \end{cases}$
        where $\mathrm{rand}(\mathcal{A}_i)$ denotes a random selection from the set $\mathcal{A}_i$.
    (b) Receive reward $r_t$ and update:
        i.  $sat \leftarrow \begin{cases} \text{true} & \text{if } (r_t \geq \alpha_i^t) \\ \text{false} & \text{otherwise} \end{cases}$
        ii. Update aspiration level
              $\alpha_i^{t+1} \leftarrow \lambda\alpha_i^t + (1 - \lambda)r_t$
            if $0 \leq (r_t - \alpha_i^{t+1}) < \phi$, then, with probability $\eta$
                    $\alpha_i^{t+1} \leftarrow \alpha_i^{t+1} + \beta$                (8)
```

Table 2: The S-Algorithm with trembles (SAwT) for agent i.

an agent's aspirations are trembled according to some probability distribution function. By doing this, they showed that agents played cooperatively most of the time regardless of initial aspirations.

Thus, a natural extension of the S-Algorithm is to add trembles in aspirations. This allows initial aspirations to be anything (they do not have to be above the pareto boundary). Trembles in aspirations also help prevent premature selection of local maxima and allow adaptation to changing environments, such as non-stationary worlds and those with other learning agents. We call this algorithm the S-Algorithm with Trembles (SAwT).

The SAwT algorithm is shown in Table 2. Like the S-Algorithm, SAwT performs a relaxation search. However, SAwT differs from the S-Algorithm in two ways. First, with SAwT, aspirations may be initialized to any value.[5] Second, SAwT agent i trembles its aspirations with probability η only when a) it is satisfied (i.e., $(r_t - \alpha_i^{t+1}) \geq 0$) and b) α_i^{t+1} is close to r_t (i.e., $(r_t - \alpha_i^{t+1}) < \phi$). This means that if ϕ[6] and η are quite small, trembles in aspirations will only (generally) occur when agents are mutually satisfied (i.e., have settled on a solution).

The variable β in (8) determines how much aspirations are trembled. We only tremble aspirations upward, so $\beta > 0$. Also, so that aspirations remain "reasonable," $\alpha_i^{t+1} + \beta$ should not be significantly greater than the maximum payoff an agent can receive. There are several ways to handle this. In this paper, we always tremble aspirations to slightly higher than the highest payoff an agent can receive (let this value be $h_\$$).[7]

Since SAwT eliminates the need for high initial aspirations by trembling its aspirations upward periodically, it can deal with changing environments as long as aspirations are trembled above the pareto boundary. Since SAwT trembles its aspirations to the highest payoff it has seen so far, it will tremble its aspirations so that the joint aspirations of the agents will be above the pareto boundary as long as it has seen at some point the highest payoff (or some higher payoff) of the current game/environment. Additionally, SAwT generally trembles its aspirations so that the other two initial aspiration requirements of the S-Algorithm are met as well. This means that SAwT can be applied to the same large set of matrix games as the S-Algorithm. Since trembles in aspirations generally cause them to meet the initial aspiration requirements of the S-Algorithm, SAwT can be made (by setting the learning rate λ appropriately) to settle with arbitrarily high probability on ε-PE solutions, even in changing environments.

Guaranteed ε-Pareto Efficiency for Nonzero-Sum Games

In this section, we provide a theorem that says that SAwT can be guaranteed to play ε-PE solutions arbitrarily high percentages of the time if parameters λ, ϕ, and η are set appropriately. In the interest of space, we provide only an outline of its proof.

Theorem 3. *In n-agent, m-action repeated-play games, there exist values of λ and η such that SAwT (in self play) will play ε-PE solutions arbitrarily high percentages of the time.*

Proof. (*Outline*) - A repeated play game with SAwT agents consists of a series of *epochs*. Three events occur in an epoch that divide it into two stages. The *search stage* begins when one of the agents in the game trembles its aspirations, proceeds while aspirations are relaxed, and concludes when the agents settle on a solution. The search stage may include agents trembling their aspirations before settling on a solution. Again, however, if ϕ and η are sufficiently small, such occurrences will generally be rare. The *convergence stage* begins when the agents settle on a solution and ends when one of the agents trembles its aspirations. Let t_s denote the length of the search stage and let t_n denote the length of the convergence stage. Thus, the length of an epoch is $t_c + t_s$.

For any repeated-play game, let b_ε be the percentage of the time that SAwT agents play ε-PE solutions (in self play). Since a repeated-play game (if long enough) consists of a sequence of epochs, b_ε is the same as the average probability that ε-PE solutions are played over an epoch. Let p_c be the probability that a solution played during the convergence stage is ε-PE, and let p_s be the probability that a solution played during the search stage is ε-PE. So, we have

$$b_\varepsilon = \frac{p_c t_c + p_s t_s}{t_c + t_s} \tag{9}$$

where t_c is the average length of a convergence stage and t_s is the average length of a search stage. In Theorems 1 and 2 we discussed the probability q that the S-Algorithm (and, thus, SAwT) settle on ε-PE solutions. The search stage of SAwT is essentially the S-Algorithm minus the slight possibility of "pre-convergence" trembles so q places a bound on p_c. Additionally, if we assume that no ϵ-PE solutions are

[5] A good way of doing this in practice is to set aspirations to just slightly higher than the first payoff received.

[6] For the results in this paper, $\phi = 0.01$

[7] Making $h_\$$ slightly higher than the highest payoff received is done to avoid premature convergence.

played in the search stage, we have $p_s = 0$. We need only find bounds, then, for t_c and t_s. We give two lemmas to place bounds on these variables.

Lemma 4. *The average amount of time spent in the convergence stage of an epoch (t_c) has the lower bound*

$$t_c \geq \frac{1}{1-(1-\eta)^n}, \tag{10}$$

where n is the number of agents and η is the tremble rate.

Lemma 5. *The average amount of time t_s that SAwT agents (in self play) spend playing non-ε-PE solutions in the search stage of an epoch is (provided that $p_r \neq 0$), in the worst case matrix game, bounded by*

$$t_s \leq \frac{\ln\left(\frac{\varepsilon}{h_\$ - l_\$}\right)}{\ln \lambda} + \frac{m^n}{p_r}, \tag{11}$$

where p_r is the probability that all agents will play randomly on a given iteration after aspirations fall below the pareto boundary.

Plugging all of these bounds into Equation (9) we get the bound

$$b_\varepsilon \geq \frac{q}{1 + t_s[1-(1-\eta)^n]}. \tag{12}$$

Observe from Equation (12) that if $\eta = 0$, then SAwT is equivalent to the S-Algorithm and $b_\varepsilon = q$. Otherwise, $b_\varepsilon < q$. Thus, in order for b_ε to be arbitrarily close to one, q must be made arbitrarily close to one and b_ε must be made arbitrarily close to q. Since there exists a λ that makes q anything between zero and one (see Theorem 2), q can be made arbitrarily close to one. Since b_ε approaches q (from below) as η approaches zero, b_ε can be made arbitrarily close to q. Thus, there exists values of λ and η such that SAwT (in self play) will play ε-PE solutions arbitrarily high percentages of the time.

Equation (12) gives a pessimistic lower bound on b_ε for most games since it deals with worst case games. As an example, consider the 3-agent, 3-action MASD with $\varepsilon = 0.5$ and $q = 0.99$ (which defines the learning rate λ). The actual percentage of time that ε-PE solutions are played is plotted in Figure 5 against the lower-bound shown in Equation (12). □

Fairness

In the previous section we discussed how SAwT can be made to play ε-PE solutions arbitrarily high percentages of the time in most nonzero-sum games. In this section, we briefly discuss how SAwT learns to play "fair" ε-PE solutions, which can be either pure or mixed strategy solutions.

Fairness is based on the notion of fallback positions. An agent retreats to its fallback position when it is not content with its payoffs. SAwT's fallback is to play randomly when it is not satisfied. Thus, its fallback position (in self play) is, in general, approximately the random payoff $\frac{\sum_{a \in A} u_i(a)}{m^n}$. However, if one agent has more good payoffs than another, it has more satisficing options, which means that play is not exactly random. Since an agent that has more good payoffs has more power, it should possibly receive higher payoffs. Essentially, we use the term *fair* to indicate solutions that give both agents payoffs determined by their respective bargaining powers, which is largely determined by the strengths of their fallback positions.

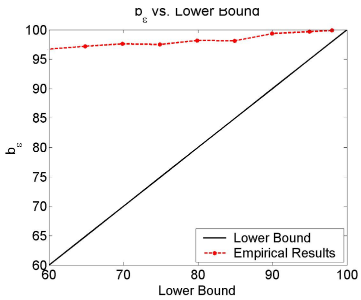


Figure 5: Shows the lower bound of b_ε plotted against actual empirical results from a 3-agent, 3-action MASD, with $\varepsilon = 0.5$.

Littman and Stone (2003) use the notion of a fallback to compute a desirable (PE and fair) solution for the agents to play. Their technique can be used in two-agent matrix games in which the entire payoff matrix is known. SAwT learns to play similar solutions to those computed by Littman and Stone's algorithm without knowledge of the payoff matrix or the actions played by other agents. Additionally, SAwT learns ε-PE and "fairness" in n-agent ($n \geq 2$) games.

As examples, we compare the solutions proposed by Littman and Stone's algorithm with the solutions learned by SAwT in self play in two games: a 2-agent, 4-action MASD and the battle of the sexes (BotS) game. For both examples, $\lambda = 0.99$ and $\eta = 0.001$.

In the 2-agent, 4-action MASD, Littman proposes that the "fair" result is for each agent to fully cooperate, or give everything to the group. SAwT settles on this solution 86.6% of the time, settles on two other closely related PE solutions 8.4% of the time, and settles on a non-PE solution 5% of the time. Thus, in this game, SAwT mostly converges to the "fair" solution.

The BotS matrix game is shown in Table 3. At the top of each cell in the table (each cell corresponds to a pure strategy solution of the game) is the joint payoff. Below the joint-payoff and to the left is the percent of time that Littman proposes that a solution be played; to the right is the percent of time that SAwT settles on the solution. As can be seen, SAwT settles on each solution the same percentage of time that Littman and Stone's algorithm proposes.

Discussion

We have advocated that multiagent learning algorithms (for repeated play games) should converge to near pareto efficient (and "fair") solutions. Furthermore, these algorithms should be able to learn when minimal information about the environment and the other agents in the game are available and be able to adapt to non-stationary environments. We have described a multiagent learning algorithm (SAwT) that

	C	D
c	(1, 1) 0 | 0	(3, 4) 50 | 50
d	(4, 3) 50 | 50	(2, 2) 0 | 0

Table 3: Payoff matrix and results for the Battle of the Sexes (BotS) game.

achieves these goals in self play.

References

Bowling, M., and Veloso, M. 2001. Multiagent learning using a variable learning rate. In *Artificial Intelligence*.

Camerer, C. F. 2003. *Behavioral Game Theory*. Princeton University Press.

Conitzer, V., and Sandholm, T. 2003. Awesome: A gerneral multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. In *Proccedings of the Twentieth International Conference on Machine Learning (ICML-2003)*.

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

Karandikar, R.; Mookherjee, D.; Ray, D.; and Vega-Redondo, F. 1998. Evolving aspirations and cooperation. In *Journal of Economic Theory*, volume 80, 292–331 (Article No. ET972379).

Littman, M. L., and Stone, P. 2001. Leading best-response strategies in repeated games. In *IJCAI Workshop on Economic Agents, Models, and Mechanisms*.

Littman, M. L., and Stone, P. 2003. A polynomial-time nash equilibrium algorithm for repeated games. In *2003 ACM Conference on Electronic Commerce (EC '03)*.

Littman, M. L. 1994. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning (ICML-94)*.

Nash, J. F. 1950. The bargaining problem. In *Econometrica*, volume 28, 155–162.

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

Stimpson, J. R., and Goodrich, M. A. 2003. Learning to cooperate in a social dilemma: A satisficing approach to bargaining. In *International Conference on Machine Learning (ICML'03)*.