Convergence to Pareto Optimality in General Sum Games via Learning Opponent's Preference

Dipyaman Banerjee
Department of Math & CS
University of Tulsa
Tulsa, OK, USA
dipyaman@gmail.com

Sandip Sen
Department of Math & CS
University of Tulsa
Tulsa, OK, USA
sandip@utulsa.edu

ABSTRACT

We consider the learning problem faced by two self-interested agents playing any general-sum game repeatedly where the opponent payoff is unknown. The concept of Nash Equilibrium in repeated games provides us an individually rational solution for playing such games and can be achieved by playing the Nash Equilibrium strategy for the single-shot game in every iteration. However, such a strategy can sometimes lead to a Pareto-dominated outcome for the repeated game. Our goal is to design learning strategies that converge to a Pareto-efficient outcome that also produces a Nash Equilibrium payoff for repeated two player n-action general-sum games. We present a learning algorithm, POSNEL, which learns opponent's preference structure and produces, under self-play, Nash equilibrium payoffs in the limit in all such games. We also show that such learning will generate Pareto-optimal payoffs in a large majority of games. We derive a probability bound for convergence to Nash Equilibrium payoff and experimentally demonstrate convergence to Pareto optimality for all structurally distinct 2-player 2-action conflict games. We also compare our algorithm with existing algorithms such as WOLF-IGA and JAL and showed that POSNEL on average, outperforms both the algorithms.

1. INTRODUCTION

It has been pointed out by researchers that a learning algorithm playing a repeated game should at least converge in self-play and ensure the safety or minimax value to both the players [2]. To achieve this, researchers have tried to develop algorithms that converges to the Nash Equilibrium of the single-stage version of the game that is being played repeatedly. We refer to such an equilibrium as Nash Equilibrium for Single Stage Game (NESSG). NESSG is a strategy profile such that no player has the incentive to unilaterally deviate from its own strategy and hence, playing one's part of NESSG is considered to be individually rational. Also playing the Nash Equilibrium strategy for the single stage game in each iteration of the repeated game is a Nash Equilibrium for the repeated game and it always guarantees safety value for a player. Due to these reasons, a number of learning mechanisms have been proposed which were proven to converge to NESSG under certain conditions [3, 2, 5, 8, 10]. There are two problems, however, with choosing NESSG as a solution concept for repeated games. First, there can exist multiple Nash Equilibria for a single-shot game and to reach an agreement among the players for choosing one of them is a non-trivial task. Secondly, for games such as the Prisoner's Dilemma, the only NESSG outcome is Pareto-dominated and playing that repeatedly will result in a Pareto-dominated outcome for the iterated version of the game as well[1]. To avoid these problems, instead of trying to converge to an NESSG, we set our learning goal to converge to a solution that is Pareto-efficient and produces Nash-Equilibrium payoffs for the repeated game. We refer to these outcomes as Pareto Outcomes Sustained by Nash Equilibrium (POSNE). As a first step towards this goal, we developed an algorithm called POSNEL (POSNE Learner) which is guaranteed to converge to an outcome that is not Pareto-dominated by any other pure strategy outcomes and produces Nash-equilibrium payoff for the repeated game. In section 2 we show how the Folk Theorem [6] can help us identify the POSNE outcomes of a repeated game.

Most previous learning mechanisms developed for game playing assume complete transparency of the payoffs [5, 7, 8, 11], i.e, players can observe the payoffs received by all the players after they make their action choices. This may not be always possible in real environments. A more realistic scenario is that of *incomplete information games* where players cannot observe opponent's payoffs[2]. In this paper we consider only incomplete information games. However, we assume that the players can observe the actions of all other players. Note that in such games a player cannot compute the opponent's best response to its own strategy as the best response will depend on the opponent's payoff structure. The lack of opponent's payoff information also precludes the players from computing the Nash Equilibria of a game where every player's strategy is a best response to their opponents' strategy. We believe however, that it is still possible to predict opponents preference rankings over different outcomes of the game by observing its history of action choices. POSNEL players attempt to learn these preference orderings and uses that knowledge to converge to an

[1]An outcome is Pareto-dominated by another if at least one player prefers the latter and no player prefers the former. An outcome An out come that is not Pareto-dominated is Pareto-efficient.

[2]Other multiagent systems literature that make the incomplete information assumption include work that requires no communication between agents, e.g., use of aspiration levels [14], and approaches that use some form of interagent communication external to the learning algorithms to facilitate concurrent learning, e.g., use of commitment sequences to enforce stationary environments [9], action revealation [13], trigger transition between different phases of learning [15], etc.

outcome that is not Pareto-dominated by any other pure strategy outcome and produces Nash Equilibrium payoff on average under self play for these incomplete information games.

This paper has been greatly motivated by the polynomial time algorithm developed by Littman and Stone [12]. In their work, Littman and Stone showed that for any general sum game, it is possible to find a POSNE outcome (in particular, the Nash bargaining solution) in polynomial time if the payoff structures of both the players are known. They used a centralized search through the payoff space to identify such an outcome. In this paper, we go one step forward and show that it is also possible to find such an outcome in a distributed manner even without the knowledge of opponent's payoff structure for most of the general sum games. We prove that, except for one case, two POSNEL agents when played against each other are able to find and converge to a POSNE outcome by learning opponent's preferences over the outcomes. In our previous work [1] we developed a new class of learner called a Conditional Joint Action Learner (CJAL) who tries to learn the correlation between actions taken by the players by computing the conditional probability of the opponent's actions given its own actions. It then uses these conditional probabilities to calculate the expected utilities of its different actions and plays that action which maximizes it. CJAL has been observed to converge to the mutual cooperation state in Prisoner's Dilemma under certain conditions but fails to reach Pareto-optimality in other general sum games. It continuously adopts its strategy depending on the opponent's expected behavior which makes it impossible for the opponent to learn its preference. We show that POSNEL follows a strategy that not only helps a player to learn opponent's preferences but also reveals its own preference structure to the opponent. However unlike CJAL and JAL, POSNEL does not immediately apply its learned knowledge to modify its strategy. Instead, it continues to follow this strategy throughout the learning phase to facilitate opponent's learning process. This in turn enables two POSNEL players to correctly learn each others preferences and converge to an outcome that is mutually preferred.

2. IDENTIFYING POSNE OUTCOMES

In this section we will describe how we can identify the POSNE outcomes for any infinitely repeated 2-player games using the Folk Theorem.

Let us consider the game of Battle of Sexes shown in Table 1. If we plot these payoff vectors in a 2-dimensional plane where the x and y axes represent the row and column players payoffs respectively we get the plot shown in figure 1. The points A, B, C and D represent the payoffs corresponding to all the pure strategy profiles of the single shot game. Note that points A and D are coincident at the origin (0,0) as the strategy profiles (F,O) and (O,F) produce zero payoff for both the players. According to the average payoff criterion for repeated games, any payoff corresponding to points A, B, C and D can be achieved in the repeated version of the game by simply repeating the action-pair that produces that payoff. Let us call this set of points P. We can now construct the convex hull for all such points in P as the smallest convex polygon C such that $\forall p \in P$, p is either on the boundary or inside C. The convex hull for the Battle of Sexes game is shown as the polygon ABC in figure 1. The convex hull for a set of points P includes all the points that can be generated using a convex combination of the points in P. In this case it means, ABC includes all the payoff vectors that are feasible as an expected payoff for some probability distributions over the pure strategy profiles in Q. Littman and Stone showed in

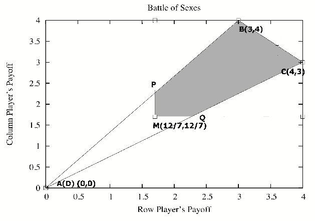

Figure 1: Possible payoffs for mixed strategies for the Battle of Sexes game.

	F	O
F	4,3	0,0
O	0,0	3,4

Table 1: Payoffs in the Battle of Sexes game.

their work [12] that any payoff vector inside the convex hull can also be realized as an average payoff of the corresponding repeated game.

Borrowing from their analysis, we can define the *minimax point* as the point corresponding to the minimax payoffs for both the players. Such a point will always be either inside or on the convex hull. In figure 1 point M(12/7,12/7) is the minimax point. Folk Theorem tells us that any point inside the convex hull of feasible payoffs which Pareto-dominates the minimax point also produces Nash Equilibrium payoff for the repeated game according to the average payoff criterion [6]. Using the definition of Pareto-dominance we observe that a point is Pareto-dominated by all other points which are right and above of it (including the points that are either on the vertical or on the horizontal line passing through it). So, according to figure 1, any point in the shaded region will coincide with the average payoff that can be obtained by playing some Nash-Equilibrium strategy-profile of the repeated game. We name this shaded region as ADM (Area Dominating Minimax). We now define POSNE outcomes as the points inside ADM that are Pareto-optimal. Observe that, in figure 1 any point on the line segment BC is a POSNE outcome. BC is called the Pareto-frontier for the game, which is the portion of the convex hull that is inside ADM with all the points over it being Pareto-optimal. Using this definition, an outcome is POSNE if and only if it resides on the Pareto-frontier. Note that if there exists an ADM an outcome for a game is always POSNE if and only if it resides on the Pareto-frontier. In the particular case when no point dominates the minimax point, there exists no ADM. In this case the minimax-point will be a Nash-Equilibrium and also a POSNE outcome.

3. POSNEL ALGORITHM

In this section we will first describe the different phases of POSNEL algorithm and how it searches for a POSNE outcome. We will then provide the actual algorithm and highlight its features.

A POSNEL agent models the repeated game as a finite state automata without any terminal state. The states of this automata are the pure strategy outcomes Q of the stage game being played in every iteration and are represented by the action-pair corresponding to that outcome. Agents choose their actions in successive iterations to trigger state transitions. Formally, we assume a set $S = \{i, j\}$ of 2 agents, where each agent $i \in S$ has a set of actions A_i. The agents repeatedly play a stage game. In every iteration they simultaneously choose an action from their respective action sets and receive a payoff. We say that after t iterations agents are in state $s_t = \{a_i, a_j\}$ if the agents play the actions a_i and a_j respectively at the t^{th} iteration of the game. Let $Q = A_i \times A_j$ be the set of all such possible states in the world which is same as the set of outcomes of the stage game being played. We denote s_t^i as the action taken by i^{th} agent in t^{th} iteration. So, in the above case, $s_t^i = a_i$. s_t^i is also known as the action played from state es_{t-1}. We also define the payoff function of agent i as $U_i(s_t) : Q \rightarrow \mathbb{R}$, which gives the reward obtained by agent i in state s_t. We denote the minimax value that an agent i can achieve as mm_i. In the POSNEL algorithm, each agent maintains a history H^t of all the states visited up to t^{th} iteration. We denote $\hat{s}_t \in H^t$ as the last state visited before coming to s_t which is different from s_t. We refer to this state as the *predecessor* state of s_t.

Given these notations, we will now discuss the different phases of operations for a POSNEL agent. A POSNEL player operates in two phases, namely the *Learning Phase* and the *Application Phase*.

3.1 Learning Phase

In the learning phase POSNEL follows a simple strategy in order to learn opponent's preferences. Essentially, this strategy prescribes that after coming to a state POSNEL should change its action with a high probability only if the current state produces payoff less than its predecessor state or less than its safety (minimax) value, otherwise it repeats its last action with a high probability. To start the process, a player chooses its actions randomly for the first two iterations and then follows this strategy throughout the learning phase. Formally, this strategy $\pi_i : Q \times Q \rightarrow A_i$ followed by the i^{th} agent that decides its action from a state s_t after coming from a state $\hat{s}_t$ in the learning phase is given as below:

if $U_i(s_t) \geq max(U_i(\hat{s}_t), mm_i)$

$$\pi_i(s_t, \hat{s}_t) = \begin{cases} \text{play } s_t^i \text{with } 1 - \epsilon_1 \text{ probability} \\ \text{play any other action with probability } \frac{\epsilon_1}{|A_i|-1} \end{cases}$$

Otherwise,

$$\pi_i(s_t, \hat{s}_t) = \begin{cases} \text{play } s_t^i \text{with } \epsilon_2 \text{ probability} \\ \text{play any other action with probability } \frac{1-\epsilon_2}{|A_i|-1} \end{cases}$$

where $0 < \epsilon_1 = \epsilon_2 << 0.5$ are two constants which have the same value but have different connotations. ϵ_1 denotes the probability with the agent moves out from a state even when it yields better payoff than its predecessor. Let us call it as the entropy of the agent. On the other hand, ϵ_2 denotes the probability with which an agent wants to stay in a state even if it yields worse payoff than its predecessor. We call it as the inertia of the agent. We also name the action that is chosen with high probability as the strategic choice and any other action as the exploratory choice. Note that, any state can be reached from another state with at least $\frac{\epsilon_1^2}{|A_i| \times |A_j|}$ probability.

3.1.1 *Learning Opponent's Preference Ordering from Observation*

While playing its own strategy π_i, after every iteration $t+1$, a POSNEL agent i simultaneously estimates and updates the conditional probability of its opponent j playing the action s_{t+1}^j from the last state s_t given the predecessor state $\hat{s}_t$. The probability is denoted as $Pr^i(s_{t+1}^j | s_t, \hat{s}_t)$ and can be computed as below:

$$Pr^i(a_j | s_t, \hat{s}_t) = \frac{\text{No. of times } j \text{ plays } s_{t+1}^j \text{after transition } \hat{s}_t \rightarrow s_t}{\text{No. of } \hat{s}_t \rightarrow s_t \text{ transition}} \quad (1)$$

Now if a player observes its opponent to repeat the last action s_t^j with a high probability after coming to a state s_t from state $\hat{s}_t$, it learns that its opponent prefers s_t over $\hat{s}_t$. However note that, we can not conclude that a player prefers $\hat{s}_t$ over s_t if it frequently moves out from s_t after coming from $\hat{s}_t$. This is because a player can also move out from a state if it produces lower payoff than its safety value. In our algorithm, we say a state s_t *attracts* another state $\hat{s}_t$ only if both the player prefers s_t over $\hat{s}_t$ and if s_t is not attracted by some other states. We put the second criteria to avoid labeling a Pareto-dominated state as an attractor state.

Formally we can define an attractor state as below:

Definition 1: To an agent i, a state s_t attracts a state $\hat{s}_t$ if $U_i(s_t) \geq max(U_i(\hat{s}_t), mm_i)$ and $Pr^i(s_t^j | s_t, \hat{s}_t) > (1 - \delta)$ and there is no other state $s_k \in S$, such that s_k attracts s_t. State s_t is called an *attractor state* and state $\hat{s}_t$ is called a *distractor* state of s_t. $\epsilon_1 < \delta < 0.5$ is a constant.

Definition 2: To an agent i, a state s_t is called a *minimax-attractor* if $U_i(s_t) \geq mm_i$ and $Pr^i(s_t^j | s_t, \hat{s}_t) > (1 - \delta)$ and there is no other state $s_k \in S$, such that s_k attracts s_t.

Given the definitions, observe that an attractor state is always a minimax attractor. However, the reverse is not true. Minimax attractor states can exist even when there is no attractor states in the environment, We later show that a state is minimax attractor iff it resides inside ADM and is not Pareto-dominated by any pure strategy outcomes. We show that a POSNEL agent always converges to a minimax attractor state, if one exists.

We already said that in the learning phase after every state transition from s_t to s_{t+1}, a POSNEL player updates the probability $Pr^i(s_{t+1}^j | s_t, \hat{s}_t)$[3]. This learning phase terminates after P such transitions between all possible state-pairs. It then uses the above definitions to find the attractor and the minimax attractor states in the environment and converges to them in the application phase. The criteria for classifying a state as attractor or distractor involves the parameter δ, a probability threshold which signifies the opponents desire to stay in a state after transition from another state. If the opponent repeats its last action with probability more than $(1-\delta)$ our agent concludes that the opponent intends to stay in that state and it prefers the current state more than its predecessor state. On the other hand, the observed probability of playing the last action being less than δ signals its inclination to move out from that state after this transition. If the observed probability lies between δ and $1 - \delta$ our agent fails to make any conclusion.

3.2 Application Phase

[3]Alternatively, the probabilities can be updated, based on the total state transition counts, only at the end of the learning phase, and before the application phase begins.

After P transitions between all possible state pairs POSNEL identifies the *attractor* and *minimax-attractor* states and moves to the Application phase. We have proved that if P is sufficiently large, under self-play POSNEL will correctly identify the *attractor* and *minimax-attractor* states. Also, it is not necessary that different learners use the same value for this parameter. We now describe the strategy followed in the Application phase once these *attractor* states are identified. Let us assume a POSNEL player i is in state s_t at the beginning of the Application Phase. Depending on the nature of s_t we can have six following cases. We prescribe a rule for each of these cases that determines the subsequent action taken by the player.

Case 1: If s_t attracts its predecessor $\hat{s}_t$ then i continues to repeat the last action s_t^i as long as it stays in s_t.

Case 2: If s_t does not attract $\hat{s}_t$ and there exists a state $\bar{s}_t$ that attracts s_t, i plays the action $\bar{s}_t^i$. If s_t has multiple *attractor*s, i chooses that *attractor* which has maximum distractors. If multiple *attractor*s have the same maximum number of distractors, i chooses randomly among them.

Case 3: If s_t does not attract $\hat{s}_t$ and there exists no state $\bar{s}_t$ that attracts s_t, but there exists some other *attractor* states in the environment, the player continues to follow the strategy π_i of the learning phase.

Case 4: If there exists no *attractor* states and s_t is a *minimax-attractor* then i plays s_t^i.

Case 5: If there exists *minimax-attractor*s in the environment but no *attractor* states and s_t is not a *minimax-attractor* then i chooses a *minimax-attractor* state s randomly and plays s^i.

Case 6: If there exists no *attractor* states or *minimax-attractor* states in the environment i follows the minimax strategy.

In the Application phase, POSNEL uses these rules in every iteration to choose its action.

PROPOSITION 1. *If there exists a non-null set of* minimax-attractor *states S_{mm} and if both POSNEL players correctly find all these* minimax-attractor *states of the game during the learning phase, they will converge to a state $s \in S_{MM}$ in the Application phase.*

The proof of this and the remaining propositions have been omitted due to space constraints.

We can now present our algorithm more precisely. Algorithm 1 outlines the POSNEL algorithm skeleton which, given the current state s_t, previous state s_{t-1} and the predecessor of the previous state $\hat{s}_{t-1}$ returns an action for agent i and simultaneously learns its opponents preferences. The two functions *choose-action* and *update-probability* performs these functions respectively. The *choose-Action* function given in Algorithm 2 decides which action to be taken from the current state whereas the function *update-Probability* in Algorithm 3 observes the current action taken by the opponent and updates its conditional probability of taking different actions. It also locates the *attractor* and *minimax-attractor* states in the environment using the criteria given in Definitions 1 and 2.

From algorithm we can make the following observations:

Algorithm 1 POSNEL Algorithm

```
Input: $history$: list containing $s_t, \hat{s}_t, s_{t-1}, \hat{s}_{t-1}$
Input: $LA$, List of *attractor* state
Input: $LD$, List of distractor state
Input: $LM$, List of *minimax-attractor* state
Input: round, iteration No.
Input: count, 3 dimensional Array
Output: chosen action
count[$s_t^j$][$s_{t-1}$][$\hat{s}_{t-1}$]++;
**if** round $< 2$ **then**
  return a randomly chosen action
**end if**
**if** count of all possible transitions $> P$ **then**
  /* Application Phase */
  **if** $LA$ is empty **then**
    **if** $LM$ is empty **then**
      return action according to the minimax strategy
    **else**
      **if** $s_t \in LM$ **then**
        return $s_t^i$
      **else**
        choose a state $s$ randomly from $LM$
        return $s^i$
      **end if**
    **end if**
  **else if** $s_t \in$ attractor$(\hat{s}_t))$ **then**
    return $s_t^i$
  **else if** attractor$(s_t)$ is not empty **then**
    $s = arg\max_{s \in \text{attractor}(s_t)}$(size of distractor$(s)$)
    return $s^i$
  **else**
    choose-action()
  **end if**
**else**
  /* Learning Phase */
  update-probability()
  choose-action()
**end if**
```

Algorithm 2 choose-action

```
Function: choose-action
Input: $history$, List containing $s_t, \hat{s}_t, s_{t-1}, \hat{s}_{t-1}$
**if** $U_i(s_t) \geq max(U_i(\hat{s}_t), mm_i)$ **then**
  choose $s_t^i$ with probability $1 - \epsilon_1$
  choose any action other than $s_t^i$ with probability $\frac{\epsilon_1}{|A_i|-1}$
**else**
  choose $s_t^i$ with probability $\epsilon_2$
  choose any action other than $s_t^i$ with probability $\frac{1-\epsilon_2}{|A_i|-1}$
**end if**
```

- Our agent maintains three global lists LA, LD and LM which contains all the discovered *attractor*, distractor and minimax attractor states respectively.
- A POSNEL agent i also maintain a list of states which after every t^{th} iteration contains the current state s_t, the last state s_{t-1} and the predecessor of the last state $\hat{s}_{t-1}$. Observe that, if $s_t \neq s_{t-1}$, $\hat{s}_t = s_{t-1}$ otherwise, $\hat{s}_t = \hat{s_{t-1}}$.
- After every iteration t the variable count$[s_t^j][s_{t-1}][\hat{s}_{t-1}]$ is incremented which denotes the number of times j played action s_t^j from state s_{t-1} with the predecessor state as $\hat{s}_{t-1}$. The probability $Pr^i(s_t^j|s_{t-1}, \hat{s}_{t-1})$ is also updated using equation 1.
- Each state $s \in Q$ is associated with an *attractor* list $attractor(s)$ and a distractor list $distractor(s)$ that contains the states which attracts s and the states that are attracted by s respectively.
- A state can never be both an *attractor* and a distractor. We put a state in LA only if it is not in LD. On the other hand, if a state is already in LA and later found to be a distractor, we remove it from LA and clear the contents of its distractor list. We also remove this state from the *attractor* lists of its distractors. So, if there are 4 states with payoffs as (1,1), (2,2), (3,3) and (4,4) then only (4,4) is recognized as the *attractor* and all other states as its distractors.
- An agent terminates its learning phase if the number of transitions between all possible state-pairs occurs P times. As the number of transitions between any two states s_i and s_j is same for both the agents, their learning phase terminates at the same point.

4. CONVERGENCE OF POSNEL

We now discuss the convergence property of POSNEL. We claim that under self-play POSNEL will eventually converge to an outcome that produces Nash-Equilibrium payoff on average and will not be Pareto-dominated by any other pure strategy outcome.

The proof of convergence is presented in three steps. First, we prove that a state is *minimax-attractor* if and only if it resides in ADM and is not Pareto-dominated by any other pure strategy outcomes. We then point out the different cases that can arise and use Proposition 1 to prove that if POSNEL discovers all the *minimax-attractor* states in the learning phase POSNEL will always converge to a *minimax-attractor* state. We also show that, when there exists no *minimax-attractor*, the minimax point is a Nash Equilibrium outcome and POSNEL converges to it. Finally we prove that with sufficient explorations all the *minimax-attractor* states will be discovered. In this proof we would be only interested in the values of ϵ_1 and ϵ_2, which we assume to be identical to avoid notational complications and henceforth, we will refer to them as ϵ. We present only the propositions here as space constraints preclude including their proofs.

PROPOSITION 2. *A state s is a* minimax-attractor *for both the players iff s resides inside ADM and is not Pareto-dominated by any other pure strategy outcomes (states) in the game.*

PROPOSITION 3. *If all the* attractor *and the* minimax-attractor *states are correctly discovered by both the players, then under self-play POSNEL will converge to a state that resides inside the ADM*

Algorithm 3 update-probability

```
Function: update-probability
Input: $history$, list containing $s_t, \hat{s}_t, s_{t-1}, \hat{s}_{t-1}$
Input: $LA$, Global List of attractor states
Input: $LD$, Global List of distractor states
Input: $LM$, List of minimax-attractor state
Input: count, 3 dimensional Array
$transitions = \sum_{a_j \in A_j}$ count$[a_j][s_{t-1}][\hat{s}_{t-1}]$
$Pr^i(s_t^j|s_{t-1}, \hat{s}_{t-1}) =$ count$[s_t^j][s_{t-1}][\hat{s}_{t-1}]/transitions$
if $transitions = P$ then
  if $U_i(s_{t-1}) \geq mm_i$ and $Pr^i(s_{t-1}^j|s_{t-1}, \hat{s}_t) > (1-\delta)$ then
    if $s_{t-1} \notin LD$ then
      $LM \leftarrow LM \cup s_{t-1}$
    end if
  end if
  if $U_i(s_{t-1}) \geq max(U_i(\hat{s}_{t-1}), mm_i)$ and $Pr^i(s_{t-1}^j|s_{t-1}, \hat{s}_t) > (1-\delta)$ then
    if $s_{t-1} \notin LD$ then
      $LA \leftarrow LA \cup s_{t-1}$
      attractor$(\hat{s}_{t-1}) \leftarrow$ attractor$(\hat{s}_{t-1}) \cup s_{t-1}$
      distractor$(s_{t-1}) \leftarrow$ distractor$(s_{t-1}) \cup \hat{s}_{t-1}$
      $LD \leftarrow LD \cup \hat{s}_{t-1}$
      if $\hat{s}_{t-1} \in LM$ then
        remove $\hat{s}_{t-1}$ from $LM$
      end if
      if $\hat{s}_{t-1} \in LA$ then
        remove $\hat{s}_{t-1}$ from $LA$
        for $s \in$ distractor$(\hat{s}_{t-1})$ do
          remove $\hat{s}_{t-1}$ from attractor$(s)$
        end for
        remove all elements of distractor$(\hat{s}_{t-1})$
      end if
    end if
  end if
end if
```

and is not Pareto-dominated by any other pure strategy outcomes (states) in the game.

PROPOSITION 4. *In the limit, POSNEL will correctly find all the* attractor *states if the exploration probability $\epsilon > 0$.*

Corollary: The probability that POSNEL will find all the attractor states increases inversely with ϵ, if ϵ and P is greater than zero.

Observe that for a given P and ϵ, n also increases with the number of actions available to an agent. So, given the number of actions, one needs to choose the value of ϵ judiciously to trade-off learning accuracy with convergence speed.

5. EXPERIMENTAL RESULTS

We analytically proved that POSNEL is guaranteed to converge to an outcome that is not Pareto-dominated by any other pure strategy outcome and generates Nash-equilibrium payoff on average for any two-player general sum games. To show this experimentally, we used all possible structurally distinct two-player two-action conflict games as a testbed for POSNEL. In a conflict game, there exists no outcome that produces the maximum possible payoff for both the players. Steven Brams showed that there can be 57 of such game matrices with ordinal payoffs [4]. We used these set of games as a testbed to empirically verify convergence behavior of POSNEL.

Two POSNEL players repeatedly played against each other in all 57 games and we observed their convergence behavior. We used the following criteria to evaluate the performance of POSNEL:

Average Social Welfare: Sum of the payoffs obtained by the two players in their converged state, averaged over 57 games.

Average Product of Payoffs: Product of the payoffs obtained by two players in their converged state, averaged over 57 games.

Success Rate: Percentage of games, out of the 57 games in which the players converge to a POSNE outcome.

We compared our results with CJAL, WOLF-PHC and JAL using these evaluation criteria, who were also tested on these 57 games under self-play. The results are tabulated in Table 2. We used the following experimental parameters: max # of iterations in learning phase = 3000, $P = 20$, $\epsilon = 0.1$, $\delta = 0.5$, # of runs = 20. The results are then averaged over all the 57 games and are presented in Table 2. The first two columns represent the average social welfare and the product of the payoffs respectively. The third column represents the proportion of games in which the algorithms have converged to a POSNE outcome. We also compared our results with the average Nash Equilibrium payoffs for all of these single shot games. We can observe that POSNEL outperforms the other strategies on all these metrics. We also observe that POSNEL converges to a POSNE outcome in 95% of the games, whereas WOLF-PHC converges in only 75% of the games and CJAL and JAL converges in 86% and 81% of the games respectively. Also note that, in 75% of time the single stage Nash Equilibrium solutions are POSNE which is same as the success rate of WOLF-PHC which has been proved to converge to an NESSG.

	Social Welfare	Product of Payoffs	Success Rate
JAL	6.1	9.13	81%
CJAL	6.14	9.25	86%
WOLF-PHC	6.03	9.01	75%
POSNEL	6.4	10.11	95%
Nash	6.05	9.04	75%

Table 2: Comparison of JAL, WOLF-PHC, CJAL and POSNEL on Conflict Games

6. CONCLUSION AND FUTURE WORK

In this paper we developed a multi-agent algorithm called POSNEL that converges to an outcome that is not Pareto-dominated by any other pure strategy outcome and produces Nash Equilibrium payoff for 2-player n-action general sum games under self-play. In case there exists no such outcome it converges to the minimax equilibrium of the game. Our agents follows a strategy that reveals its preference structure to the opponent. At the same time it tries to learn opponent's preference by observing its action sequence. We showed that under self-play with sufficient exploration two POSNEL players accurately learns each other preferences and converges to an outcome that is mutually beneficial and guarantees safety value. We have proved our claim both empirically and analytically.

The goal of this research is to find a generic multi-agent learning strategy that will always produce and output on the Pareto-frontier. POSNEL is a first step towards it. In future we would like to improve our algorithm so that it can always guarantee such a convergence. Also, we believe that our algorithm can be modified to operate in an environment where opponent's actions are also unobservable. We would like to work on that aspect in future. We would also like to understand and observe POSNEL's behavior among a heterogeneous population of more than one agents who use different learning strategies.

Acknowledgments: This work has been supported in part by an NSF award IIS-0209208.

7. REFERENCES

[1] Dipyaman Banerjee and Sandip Sen. Reaching pareto optimality in prisoner's dilemma using conditional joint action learning. *Journal of Autonomous Agents and Multiagent Systems*, 2006. (to appear).

[2] Michael Bowling and Manuela Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136:215–250, 2002.

[3] Michael H. Bowling and Manuela M. Veloso. Existence of multiagent equilibria with limited agents. *Journal of Artificial Intelligence Research (JAIR)*, 22:353–384, 2004.

[4] Steven J. Brams. *Theory of Moves*. Cambridge University Press, Cambridge: UK, 1994.

[5] Vince Conitzer and Tuomas Sandholm. AWESOME: A general multiagent learning algorithm that converges in self-play. In *Twentieth International Conference on Machine Learning*, pages 83–90, San Francisco, CA, 2003. Morgan Kaufmann.

[6] D. Fudenberg and K. Levine. *The Theory of Learning in Games*. MIT Press, Cambridge, MA, 1998.

[7] Amy Greenwald and Keith Hall. Correlated-Q Learning. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 242–249. Morgan Kaufmann, 2003.

[8] Junling Hu and Michael P. Wellman. Nash q-learning for general-sum stochastic games. *Journal of Machine Learning Research*, 4:1039–1069, 2003.

[9] S. Kapetanakis, D. Kudenko, and M. Strens. Learning of coordination in cooperative multi-agent systems using commitment sequences. *Artificial Intelligence and the Simulation of Behavior*, 1(5), 2004.

[10] Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 157–163, San Mateo, CA, 1994. Morgan Kaufmann.

[11] Michael L. Littman. Friend-or-foe q-learning in general-sum games. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 322–328, San Francisco: CA, 2001. Morgan Kaufmann.

[12] M.L. Littman and P. Stone. A Polynomial-time Nash Equilibrium algorithm for repeated games. *Decision Support Systems*, 39:55–66, 2005.

[13] Sandip Sen, Rajatish Mukherjee, and Stephane Airiau. Towards a pareto-optimal solution in general-sum games. In *Proceedings of the Second Intenational Joint Conference on Autonomous Agents and Multiagent Systems*, pages 153–160, New York, NY, 2003. ACM Pres.

[14] Jeff L. Stimpson, Michael A. Goodrich, and Lawrence C. Walters. Satisficing and learning cooperation in the prisoner's dilemma. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 535–540, 2001.

[15] K. Verbeeck, A. Nowé, T. Lenaerts, and J. Parent. Learning to reach the pareto optimal nash equilibrium as a team. In *LNAI vo.2557: Proceedings of the 15th Australian Joint Conference on Artificial Intelligence*, pages 407–418. Springer-Verlag, 2002.