Satisfaction Equilibrium: Achieving Cooperation in Incomplete Information Games *

Stéphane Ross and Brahim Chaib-draa

Department of Computer Science and Software Engineering
Laval University, Québec (Qc), Canada, G1K 7P4
{ross,chaib}@damas.ift.ulaval.ca,
http://www.damas.ift.ulaval.ca

Abstract. So far, most equilibrium concepts in game theory require that the rewards and actions of the other agents are known and/or observed by all agents. However, in real life problems, agents are generally faced with situations where they only have partial or no knowledge about their environment and the other agents evolving in it. In this context, all an agent can do is reasoning about its own payoffs and consequently, cannot rely on classical equilibria through deliberation, which requires full knowledge and observability of the other agents. To palliate to this difficulty, we introduce the satisfaction principle from which an equilibrium can arise as the result of the agents' individual learning experiences. We define such an equilibrium and then we present different algorithms that can be used to reach it. Finally, we present experimental results that show that using learning strategies based on this specific equilibrium, agents will generally coordinate themselves on a Pareto-optimal joint strategy, that is not always a Nash equilibrium, even though each agent is individually rational, in the sense that they try to maximize their own satisfaction.

1 Introduction

Game theory provides a general framework for decision making in multi-agent environments, though, general game models assume full knowledge and observability of the rewards and actions of the other agents. In real life problems, however, this is a strong assumption that does not hold in most cases.

One game model proposed by Harsanyi [1] considering incomplete information are Bayesian games. These games allow the modelling of unknown information as different agent types and a Nature's move that selects randomly each agent's type according to some probability distribution before each play. The agent must choose the action that maximizes its reward considering the probabilities it associates to each of the other agents' types and the probabilities it associates to the actions of the other agents when they are of a certain type. However, the concept of Nash equilibrium in these games can be troublesome if

* This research was supported in part by the Natural Science and Engineering Council of Canada (NSERC).

not all agents have the same common beliefs about the probability distribution of all agents' types. Furthermore, a Nash equilibrium requires that each agent knows the exact strategy of the other agents, which is not always the case when an agent faces unknown agents[1].

Another recent approach based on Bayesian games is the theory of learning in games which relaxes the concept of equilibrium. Instead of considering an equilibrium as the result of a deliberation process, it considers an equilibrium as the result of a learning process, over repeated play, and it defines the concept of self-confirming equilibrium [2] as a state in which each agent plays optimally considering its beliefs and history of observations about the other agents' strategies and types. However, they showed that if an agent does not observe the other agents' actions, then the set of Nash equilibria and self-confirming equilibria may differ. While self-confirming equilibrium is a very interesting concept and worth consideration, we note that when an agent faces unknown agents and does not observe the other agents' actions, thinking rationally on possibly false beliefs may after all, not be optimal.

In order to address this problem, we consider here that an equilibrium is the result of a learning process, over repeated play, but we differ in the sense that we pursue an equilibrium that arises as the result of a learning mechanism, instead of rational thinking on the agent's beliefs and observations. To make this equilibrium possible, we introduce the satisfaction principle, which stipulates that an agent that has been satisfied by its payoff will not change its strategy, while an unsatisfied agent may decide to change its strategy. Under this principle, an equilibrium will arise when all agents will be satisfied by their payoff, since no agent will have any reason to change its strategy. From now on, we will refer to this equilibrium as a *satisfaction equilibrium*.

We will show that if the agents have well defined satisfaction constraints, Pareto-optimal joint strategies that are not Nash equilibria can be satisfaction equilibria and that henceforth, cooperation and more optimal results can be achieved using this principle, instead of rational thinking.

In this article, we will first introduce the game model we will use to take into account the constrained observability of the other agents' actions and rewards and we will also present the different concepts we will need to analyze a game in terms of satisfaction. Afterward, we will present different algorithms that converge towards satisfaction equilibria with experimental results showing their strengths and drawbacks in some specific games. Finally, we will conclude with future directions that can be explored in order to achieve better results.

2 Satisfaction Equilibrium

In this section, we will introduce the game model we will use to formalize a game where the agents do not know nor observe the actions and rewards of the other

[1] By "unknown agents", we mean that an agent does not know strategies, actions, outcomes, rewards, etc. of other agents.

agents. Afterward, we will formally define the satisfaction equilibrium based on the satisfaction function of the different agents.

2.1 Game Model

The game model we will consider will be a modified repeated game in which we introduce an observation function and a modified reward function in order to let agents observe their rewards but not the other agents' actions.

Formally, we define the game as a tuple $(n, A, \Omega, O, R_1, R_2, \ldots, R_n)$; where n defines the number of agents, A defines the joint action space of all agents, i.e., $A = A_1 \times A_2 \times \ldots \times A_n$ and where A_i represents the set of actions agent i can do, Ω is the set of possible outcomes in the game observed by the agents, O the observation function $O : A \to \Omega$ which returns the observed outcome by the agents associated to the joint action played and finally R_i the reward function $R_i : \Omega \to \mathbf{R}$, which defines the reward of agent i given the outcome it observed. Each agent participating in the game only knows its own action set A_i and its reward function R_i. After each play, each agent is given the outcome $o \in \Omega$, corresponding to the joint action played, to compute its own reward. However, since the agents do not know the observation function O, they do not know which joint action led to this outcome.

2.2 Satisfaction function and equilibrium

To introduce the satisfaction principle in the game model previously introduced, we add a satisfaction function $S_i : \mathbf{R} \to \{0, 1\}$ for each agent i, that returns 1 if the agent is satisfied and 0 if the agent is not satisfied. Generally, we can define this function as follows:

$$S_i(r_i) = \begin{cases} 0 \text{ if } r_i < \sigma_i \\ 1 \text{ if } r_i \geq \sigma_i \end{cases}$$

where σ_i is the satisfaction constant of agent i representing the threshold at which the agent becomes satisfied, and r_i is a scalar that represents its reward.

Definition 1. *An outcome o is a satisfaction equilibrium if all agents are satisfied by their payoff under their satisfaction function and do not change their strategy when they are satisfied.*

$$\begin{array}{ll} (i) & S_i(R_i(o)) = 1 \; \forall i \\ (ii) & s_i^{t+1} = s_i^t \qquad \forall i, t : S_i(R_i(o_t)) = 1 \end{array}$$

s_i^{t+1} defines the strategy of agent i at time $t + 1$, s_i^t its strategy at time t and o_t the outcome observed at time t. Condition (i) states that all agents must be satisfied by the outcome o, and condition (ii) states that the strategy of an agent i at time $t + 1$ must not change if it was satisfied at time t. This is necessary in order to have an equilibrium. As a side note, this definition requires deterministic payoffs, because if $R_i(o)$ can be higher and lower than σ_i for the same observation o, then o will not be an equilibrium.

	C	D
C	-1,-1	-10,0
D	0,-10	-8,-8

$\sigma_i = -1 \Longrightarrow$

	C	D
C	1,1	0,1
D	1,0	0,0

Fig. 1. Prisoner's dilemma game matrix (left) and its satisfaction matrix (right).

We can now represent a satisfaction matrix by transforming a normal form game matrix with the satisfaction function of each agents. For example, the figure 1 shows the prisoner's dilemma game matrix with its transformed satisfaction matrix when both agents have a satisfaction constant set to -1.

While the game matrix and satisfaction matrix are not known to the agents, the satisfaction matrix is a useful representation to analyze the game in terms of satisfaction. Here, we can easily see that the only satisfaction equilibrium is the joint strategy (C, C), which is a Pareto-optimal strategy of the original game. This was the case in this example because we set both satisfaction constants to -1, which was the reward of the Pareto-optimal joint strategy of each agent. From this, we can conclude the following theorem 1.

Theorem 1. *In any game containing a Pareto-optimal joint strategy s, the outcome $O(s)$ and its equivalent outcomes*[2] *are the only satisfaction equilibria if $\sigma_i = R_i(O(s))\ \forall i$.*

Proof. see [3].

Therefore, we see that a major part of the problem of coordinating the agents on a Pareto-optimal joint strategy is to define correctly the satisfaction constants of each agent. While we have assumed so far that these constants were fixed at the beginning of the learning process, we will show an algorithm in the last section that tries to maximize the satisfaction constant of an agent such that it learns to play its optimal equilibrium under the other agents' strategies.

2.3 Satisfying strategies and other problematic games

Similarly to the concept of dominant strategies, we can define a satisfying strategy as a strategy s_i for agent i such that it is always satisfied when it plays this strategy. The existence of a satisfying strategy in a game can be problematic if by playing such a strategy, no satisfaction equilibrium is possible. Furthermore, other games with some specific payoff structure can also be troublesome. For example, we will consider the following 2 games presented in figure 2.

In the first game (left), we see that the row agent has a satisfying strategy A. Therefore, if row agent starts playing strategy A, then column agent will be forced to accept an outcome corresponding to joint strategy (A, A) or (A, B). This is problematic since none of these outcomes are satisfaction equilibria. In

[2] We consider that an outcome o' is equivalent to another outcome o if the rewards of all agents are the same in o and $o' : R_i(o) = R_i(o')\forall i$

	A	B
A	1,0	1,0
B	1,1	0,0

	A	B	C
A	1,1	0,1	0,1
B	1,0	1,0	0,1
C	1,0	0,1	1,0

$\sigma_i = 1 \Longrightarrow$

	A	B	C
A	1,1	0,1	0,1
B	1,0	1,0	0,1
C	1,0	0,1	1,0

Fig. 2. A game containing a satisfying strategy (left) and a problematic game (right).

	B	F
B	2,1	0,0
F	0,0	2,1

$\sigma_i = 1 \Longrightarrow$

	B	F
B	1,1	0,0
F	0,0	1,1

Fig. 3. Battle of sexes game matrix (left) and its satisfaction matrix (right).

the second game (right), there exists a unique Pareto-optimal joint strategy (A, A). With the satisfaction constants set to 1 for both agents, the corresponding satisfaction matrix is the same as the original game matrix. But, what we can see in this example is that we can never reach the satisfaction equilibrium (A, A) unless both agents starts with strategy A. Effectively, if one of the agent plays A but the other agent plays B or C, then the agent playing A will never be satisfied until it changes its strategy to B or C. This problem comes from the fact that an agent playing B or C will always be satisfied when the other agent plays A, and therefore, it will never change its strategy to A when the other agent plays A. Also, there is no joint strategy where both agents are unsatisfied that could allow a direct transition to joint strategy (A, A). From this, we conclude that if both agents do not start at the point of equilibrium (A, A), they will never reach an equilibrium since there exists no sequence of transitions that leads to this equilibrium. The effects of such payoff structures on the convergence of our algorithms will be showed with experimental results in the next sections.

2.4 Games with multiple Satisfaction Equilibria

In some games, more than one satisfaction equilibrium can exist depending on how the satisfaction constants are defined. For example, we can consider the battle of sexes, presented in figure 3 with satisfaction constants set to 1. What will happen when more than one satisfaction equilibrium exists is that both agents will keep or change their strategy until they coordinate themselves on one of the satisfaction equilibrium. From there, they will keep playing the same action all the time.

2.5 Mixed Satisfaction Equilibrium

In some games, such as zero sum games in which each agent either get the maximum or minimum reward, it is impossible to find a satisfaction equilibrium in pure strategy, unless we set the satisfaction constant to the minimum reward.

However, higher expected rewards could be obtained by playing mixed strategies. This can be achieved by playing a mixed satisfaction equilibrium.

Definition 2. *A mixed satisfaction equilibrium is a joint mixed strategy p such that all agents are satisfied by their expected reward under their satisfaction function and do not change their strategy when they are satisfied.*

$$\begin{array}{ll} (i) & S_i(E_i(p)) = 1 \ \forall i \\ (ii) & p_i^{t+1} = p_i^t \qquad \forall i, t : S_i(E_i(p)) = 1 \end{array}$$

$E_i(p)$ represents the expected reward of agent i under the joint mixed strategy p. Condition (ii), as in definition 2.5, ensures that an agent keeps the same mixed strategy when it is satisfied. This more general definition of the satisfaction equilibrium is also applicable in the case of stochastic payoffs, contrary to definition . However, the only way an agent will have to compute its expected reward will be to compute the average of the past n rewards it obtained under its current strategy, since it does not know the strategy of the other agents.

3 Learning the Satisfaction Equilibrium

We now present an algorithm that can be used by agents to learn over time to play the satisfaction equilibrium of a game.

3.1 Pure Satisfaction Equilibrium with fixed constants

The most basic case we might want to consider is the case where an agent tries to find a pure strategy that will always satisfy its fixed satisfaction constant.

Our algorithm 1 (called PSEL for Pure Satisfaction Equilibrium Learning) implements the satisfaction principle in the most basic way: if an agent is satisfied, it keeps its current action, else it chooses a random action in its set of actions to replace its current action.

Algorithm 1 PSEL: Pure Satisfaction Equilibrium Learning

```
Function PSEL(σ_i, K)
s_i ← ChooseAction()
for n = 1 to K do
  Play s and observe outcome o
  if R_i(o) < σ_i then
    s_i ← ChooseAction()
  end if
end for
return s_i
```

In this algorithm, the constant K defines the allowed number of repeated plays and the *ChooseAction* function chooses a random action uniformly within

the set of actions A_i of the agent. Under this learning strategy, once all agents are satisfied, no agent will change its strategy and therefore all agents reach an equilibrium. Once the agent has played K times, it returns its last chosen strategy. Evidently, in games where there exists no satisfaction equilibrium under the agents' satisfaction constants, those agents will never reach an equilibrium. Furthermore, if agent i has a satisfying strategy s_i, then we are not sure to reach a satisfaction equilibrium if s_i does not lead to an equilibrium (see figure 2 for an example).

3.2 Using an exploration strategy

While we have considered in our previous algorithm 1 that the *ChooseAction* function selects a random action within the set of actions of the agent, we can also try to implement a better exploration strategy such that actions that have not been explored often could have more chance to be chosen. To achieve this, the agent can compute a probability for each action, that corresponds to the inverse of the times it has chosen them, and then normalize the probabilities such that they sum to 1. Finally, it chooses its action according to the resulting probability distribution[3]. The results presented in section 3.3 will confirm that using this exploration strategy, instead of a uniform random choice, offers a slight improvement in the average number of plays required to converge to a satisfaction equilibrium.

3.3 Empirical results with the PSEL Algorithm

We now present results obtained with the PSEL algorithm in different games. We have used 2 standard games, i.e. the prisoner's dilemma with satisfaction constants set to -1 for both agents (see figure 1 for the corresponding satisfaction matrix) and the battle of sexes with satisfaction constants set to 1 for both agents (see figure 3 for the corresponding satisfaction matrix). We also tested our algorithm in a cooperative game and a bigger game to verify the performance of our algorithm when the joint strategy space is bigger. These games are presented in figure 4. Finally, we also present results with the 2 problematic games introduced in sections 2.3.

In the cooperative game, the satisfaction constants were set to 3 for both agents such that the only satisfaction equilibrium is joint strategy (C, C). In the big game, they were set to 5 for both agents and therefore, the only satisfaction equilibrium is joint strategy (E, D).

For each of these 6 games, we ran 5000 simulations, consisting of 5000 repeated plays per simulation, varying the random seeds of the agents each time. In figure 5, we present for each of these games the number of possible joint strategies, the number of satisfaction equilibria (SE), the convergence percentage to a SE and a comparison of the average number of plays required to converge to such an equilibrium (with 95% confidence interval) with the random and exploration strategies presented.

[3] A detailed presentation of this algorithm is available in [3]

	A	B	C
A	0,0	1,1	0,0
B	2,2	0,0	0,0
C	0,0	0,0	3,3

	A	B	C	D	E	F	G	H
A	0,0	0,0	-1,4	0,0	2,-2	0,0	3,0	0,0
B	1,2	0,0	0,0	3,0	0,0	0,0	0,3	1,1
C	0,0	3,3	0,0	1,1	0,0	0,0	2,2	0,0
D	4,4	0,0	5,1	0,2	2,2	1,4	0,0	0,0
E	0,1	0,0	0,0	5,5	0,0	0,0	2,1	0,0
F	0,4	2,2	0,2	0,0	0,0	3,3	0,0	4,4
G	0,0	5,3	3,0	0,0	-1,3	0,0	2,-1	0,0
H	0,0	2,4	1,1	0,0	0,0	-3,2	0,0	0,0

Fig. 4. Cooperative game matrix (left) and big game matrix (right).

Fig. 5. Convergence percentage and plays needed to converge to a SE in different games with the PSEL algorithm

Game	$\|A\|$	n_{SE}	conv. %	Random Avg. plays	Exploration Avg. plays	Improvement[4]
Prisoner's Dilemma	4	1	100%	8.67 ± 0.23	6.72 ± 0.18	22.49%
Battle of Sexes	4	2	100%	1.97 ± 0.04	1.95 ± 0.04	1.02%
Cooperative Game	9	1	100%	8.92 ± 0.23	7.82 ± 0.19	12.33%
Big Game	64	1	100%	67.95 ± 1.89	61.51 ± 1.65	9.48%
Problematic Game	9	1	10.88%	-	-	-
Game with satisfying strategy	4	1	33.26%	-	-	-

In each of these games, the *SE* were corresponding to Pareto-optimal joint strategies and the satisfaction constants were set according to theorem 1. In all non problematic games, we always converged to a *SE* within the allowed 5000 repeated plays. Therefore, we see from these results that, in non problematic games, when the satisfaction constants are well defined, we seem to eventually converge toward a Pareto-optimal satisfaction equilibrium[5] (*POSE*). However, in the problematic games, we see that the convergence percentage of the *PSEL* algorithm is dramatically affected. We note that in such games, the convergence of the algorithm is highly dependant on the initial joint action chosen by the agents, since some initial choices can never reach a SE. This is not the case of the other non problematic games where a SE is always reachable by doing a certain sequence of joint strategy transitions.

3.4 Convergence of the PSEL algorithm

While we have already showed that the *PSEL* algorithm does not work in all games, there is a specific class of games where we can easily define the convergence probability of the *PSEL* algorithm according to theorem 2.

[4] The improvement corresponds to the percentage of gain in average plays required to converge to a SE with the exploration strategy : $\frac{Avg(Random)-Avg(Exploration)}{Avg(Random)} * 100\%$

[5] We define a Pareto-optimal satisfaction equilibrium as a joint strategy that is a satisfaction equilibrium and also Pareto-optimal.

Theorem 2. *In all games where all agents have the same satisfaction in all outcomes, i.e. ($S_i(R_i(o)) = S_j(R_j(o))\forall i, j, o$), the PSEL algorithm, using a uniform random exploration, will converge to a SE within K plays with probability $1-q^K$ where $q = 1 - n_{SE}/|A|$ and the expected number of plays required to converge is given by $|A|/n_{SE}$.*

Proof. see [3].

Here, $|A|$ represents the joint action space size and n_{SE} is the number of SE in the game. This theorem will always be applicable to identical payoffs games[6] if we use the same satisfaction constant for all agents. In this case, since all agents have the same rewards and satisfaction constants, they will always have the same satisfaction in all outcomes. From theorem 2, we can conclude that in such games, as $K \to \infty$, the convergence probability will tend toward 1. In practice, for the cooperative game (figure 4) where theorem 2 applies, we see that the the expected number of plays required to converge is 9 and the probability to converge within 50 plays is around 99.7%.

4 Learning the Satisfaction Constant

While the *PSEL* algorithm has showed interesting performance in some games, it has the disadvantage that the satisfaction constant must be correctly set in order to achieve good results. To alleviate this problem, we present a new learning strategy that tries to maximize the satisfaction constant while staying in a state of equilibrium.

4.1 Limited History Satisfaction Learning (LHSL) Algorithm

In order to achieve this, we present an algorithm (called *LHSL* for Limited History Satisfaction Learning) that implements the strategy of increasing the satisfaction constant when the agent is satisfied and decreasing the satisfaction constant when it is unsatisfied. We also decrease the increment/decrement over time in order to converge to a certain fixed satisfaction constant. This will be achieved by multiplying the increment by a certain factor within the interval $]0, 1[$ after each play. Moreover, we keep a limited history of the agent's experience in order to prevent it from overrating its satisfaction constant, by checking whether it was unsatisfied by its current strategy in the past when its satisfaction constant was higher than a certain threshold. We will see in the results, that this technique really helps the convergence percentage of the algorithm compared to the case where we do not prevent this, as in the special case where the history size will be 0.

In this algorithm, the satisfaction constant σ_i is initialized to the minimum reward of agent i and the constant δ_i is used to increment/decrement this satisfaction constant. More precisely, δ_i is decremented over time, such that it tends

[6] An identical payoffs game is a game where all agents have the same reward function

Algorithm 2 LHSL : Limited History Satisfaction Learning

Function LHSL(δ_i, γ_i, n_i)
$\sigma_i \leftarrow min(r_i)$; $s_i \leftarrow ChooseAction()$
$S[0..|A_i|-1, 0..n-1] \leftarrow$ a matrix initialized with true values
$\Sigma[0..|A_i|-1, 0..n-1] \leftarrow$ a matrix initialized with $min(r_i)$ values
while $\delta_i > \epsilon_i$ **do**
  Play s_i and observe outcome o
  $lastStrategy \leftarrow s_i$; $satisfied \leftarrow (R_i(o) < \sigma_i)$; $tmp \leftarrow 0$
  if not $satisfied$ **then**
    $s_i \leftarrow ChooseAction()$; $tmp \leftarrow -\delta_i$
  else if not unsatisfied with s_i and $\sigma_i + \delta_i$ in history **then**
    $tmp \leftarrow \delta_i$
  end if
  If $n > 0$ add $satisfied$ and σ_i in history of $lastStrategy$ and remove oldest values
  $\sigma_i \leftarrow \sigma_i + tmp$; $\delta_i \leftarrow \delta_i \cdot \gamma_i$
end while
return (s_i, σ_i)

toward 0, by multiplying it by the constant $\gamma_i \in]0,1[$ after each play. The matrix S keeps a history of the last n states of satisfaction for each action and the matrix Σ keeps, for each action, a history of the last n satisfaction constants when the agent played these actions. This history is used to check, before incrementing the satisfaction constant, whether or not the agent was unsatisfied by its current strategy in the past when its satisfaction constant was below its new satisfaction constant. Finally, after each play, we update the history of the agent. We consider that the algorithm has converged to the optimal satisfaction constant when δ_i is lower than a certain constant $\epsilon_i \simeq 0$. At this point, the algorithm returns the satisfaction constant and the last strategy chosen by agent i. When all agents have converged, if they are all satisfied by their strategy, then we have reach a satisfaction equilibrium since their satisfaction constant will be stable[7]. While we are not guaranteed to converge toward a *POSE*, we will see that in practice, this algorithm yields a convergence percentage of almost 100% toward the *POSE* in any non problematic games.

4.2 Empirical results with the LHSL Algorithm

To test the *LHSL* algorithm, we have used the same 6 games we have presented for the results with the *PSEL* algorithm and we now try to learn the *POSE* without giving a priori its value to set accordingly the satisfaction constant. The results were obtained over 5000 simulations and we show the convergence percentage to the *POSE* obtained with the best γ_i value and history sizes we

[7] The satisfaction constants become stable when the floating point precision is insufficient to account for the change caused by the addition of δ_i. Therefore, we must choose ϵ_i such that $\sigma_i \pm \delta_i = \sigma_i$ when $\delta_i \leq \epsilon_i$. In fact, we could use $\sigma_i \pm \delta_i = \sigma_i$ as our convergence criteria.

Fig. 6. Convergence percentage to a POSE in different games with the LHSL algorithm

Game	$\|A\|$	With history conv. %	γ_i	n_i	Without history conv. %	γ_i
Prisoner's Dilemma	4	100%	0.99	64	89.96%	0.90
Battle of Sexes	4	100%	0.90	16	97.60%	0.80
Cooperative Game	9	99.66%	0.995	128	97.62%	0.95
Big Game	64	99.66%	0.995	16	93.88%	0.99
Problematic Game	9	9.86%	0.95	128	7.88%	0.50
Game with satisfying strategy	4	98.06%	0.95	128	38.78%	0.95

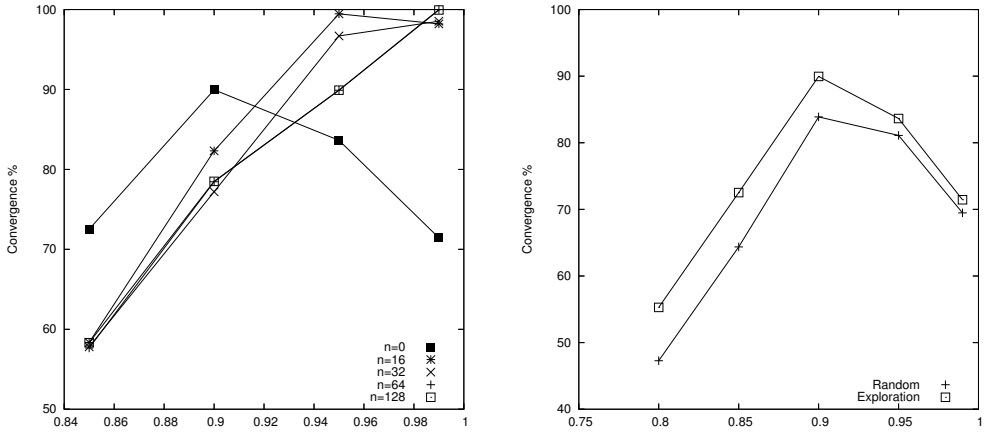

Fig. 7. Convergence percentage to a *POSE* in the prisoner's dilemma under different γ values, history sizes and exploration strategies

have tested[8]. We also compare these results to the special case where we do not use a history, i.e., $n = 0$. In all cases, δ_i was set to 1 and the convergence threshold ϵ_i was set to 10^{-20}.

In all cases, the best results, showed in figure 6, were obtained with the exploration strategy we have presented in section 3.2. In most games, except the problematic game (figure 2), we were able to get a convergence percentage near 100%. We can also see that the use of a history offers a significant improvement over the results we obtain without a history. As a side note, the convergence percentage of the *LHSL* algorithm seems to vary a lot depending on the history sizes and gamma values. This is illustrated in figure 7.

The first graphic in figure 7 compares the results with different history sizes and γ values. We can see that the bigger the history size, the closer to 1 γ must be in order to achieve better performances. While, in general, the more slowly we decrement δ and the bigger the history size is, the better are the

[8] In these results, γ_i, δ_i, ϵ_i, σ_i and the history size were the same for all agents

results, we see that small histories can also lead to very good results when γ is well defined. Since the closer γ is to 1, the more repetition will be needed for δ to reach ϵ, we can conclude that if we have only a few plays to learn the equilibrium, than it is better to use a small history, since it can achieve better convergence percentage when the number of repeated play is small. In the second graphic, we compare the convergence percentage of the 2 different exploration approaches under different γ values for the prisoner's dilemma, in the case where no history was used ($n = 0$). This graphic confirms that the exploration strategy presented in section 3.2 improves slightly the convergence percentage of the *LHSL* algorithm.

5 Conclusion and future works

While this article covered a lot of new concepts, it laid out only the basic theoretical foundations of the satisfaction equilibrium. The algorithms we have presented have shown great performance in practice, but we have seen some games with specific payoff structures that could pose problems or render impossible the convergence to a satisfaction equilibrium. We have identified possible solutions, such as allowing mixed satisfaction equilibrium and trying to maximize the satisfaction constant, that could sometimes palliate these problems. Although, what we may discover is that in some games it might not always be possible to converge to a satisfaction equilibrium, or to a *POSE*. What we might want to do in these games is to converge toward a Nash equilibrium. If convergence to a Nash equilibrium is always possible, then we may try to find an algorithm that converges in the worst case to a Nash equilibrium, and in the best case, to a Pareto-optimal satisfaction equilibrium. In order to achieve this goal, the next step will be to develop an algorithm that can converge to a Pareto-optimal mixed satisfaction equilibrium. Also, a lot of theoretical work needs to be done to prove and/or bound the efficiency of the presented algorithms and identify clearly in which cases the algorithms will converge or not to a satisfaction equilibrium. Afterward, another long term goal is to apply the satisfaction equilibrium to stochastic games in order to allow agents to learn a Pareto-optimal joint strategy without knowing anything about the other agents in these type of games.

References

1. Harsanyi, J.: Games of incomplete information played by bayesian players. Management Science **14** (1967) 159–182, 320–334, and 486–502
2. Dekel, E., Fudenberg, D., Levine, D.K.: Learning to play bayesian games. Games and Economic Behavior **46** (2004) 282–303
3. Ross, S., Chaib-draa, B.: Report on satisfaction equilibria. Technical report, Laval University, Department of Computer Science and Software Engineering, http://www.damas.ift.ulaval.ca/∼ross/ReportSatisfactionEquilibria.pdf (2005)