Report on Satisfaction Equilibria [⋆]

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. 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. So far, most game theory approaches have assumed that this knowledge about the other agents was known by all agents and that observations could be made about all the agents' actions. In order to alleviate these assumptions, we propose a new game model in which all an agent knows is its own actions and rewards for certain specific outcomes it observes. We show that in this type of games, all an agent can do is reasoning about its own payoffs and that therefore, classical equilibria through deliberation are not possible. 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 and theoretical proofs 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.

A first game model proposed by Harsanyi [1] considering incomplete information are Bayesian games. Theses 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.

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

However, the concept of Nash equilibrium in these games can be troublesome if 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. In this approach however, 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 are satisfaction equilibria and that henceforth, cooperation and more optimal results can be achieved using this principle, instead of rational thinking.

In this report, we will first introduce the different game theory concepts needed to fully understand the rest of the report. Then we will 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. We also demonstrate by theoretical analysis and proofs, why these learning strategies work in some specific cases. Finally, we will conclude with future directions that can be explored in order to achieve better results.

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

2 Game Theory concepts

In this section we present some game theory concepts that will be used in the rest of the report. We first introduce the normal form game representation that we will use throughout this report. Then we explain the different strategies agents can take in this type of games, such as pure and mixed strategies. Afterwards we explain what are dominant and Pareto-optimal strategies and we finally conclude with Nash equilibria.

2.1 Normal form games

The normal form game is the most common and simple game representation. It is used to model the simultaneous interactions of agents with their associated rewards. A game is generally defined as a tuple $(n, A, R_1, R_2, \ldots, R_n)$. n defines the number of agents, A defines the joint action space of all agents, i.e. $A = A_1 \times A_2 \times \ldots \times A_n$ where A_i represents the set of actions agent i can do and R_i defines the reward function $R_i : A \to \mathbf{R}$ of agent i which takes in parameter the joint action of all agents and returns its associated reward. Graphically, a normal form game of n agents is represented by a n-dimensional matrix where each dimension represents an agent and each rows/columns within a dimension represents an action of the agent. The intersections of these rows/columns represents an outcome of the game where a certain payoff for each agent is attributed.

For example, we will consider the prisoner's dilemma, which is a game consisting of 2 agents that face an interrogatory. In this game, each agent has a choice between denouncing or not the other. We will refer to these actions as D for denouncing and C for cooperating. If both agents cooperate and do not denounce the other agent, then both agents only get 1 year of detention. Though, if one agent denounce the other agent while the other does not, then the agent who denounced the other agent will be free and the other agent will go in detention for 10 years. Although, if both agents denounce the other agent, then both agents will go in detention for 8 years. The normal form game representation of this game is given in figure 1.

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

Fig. 1. Prisoner's dilemma game matrix.

In this representation, the first number in parenthesis refer to row agent reward and the second corresponds to column agent reward.

When playing a normal form game, agents have the choice between playing one of the actions available to them, or to play a stochastic action under a certain probability distribution chosen by the agent. A pure strategy for agent

i, denoted by s_i, consists in playing a certain action $a_i \in A_i$ with probability 1. We also denote $s = (s_1, s_2, \ldots, s_n)$ the joint strategy of all agents. On the other hand, a mixed strategy for agent i, denoted p_i, consists in a probability distribution over its action set A_i. In this case, agent i will play stochastically with the probabilities defined in its strategy p_i. A pure strategy is therefore a special case of a mixed strategy when a probability of 1 is assigned to some action.

2.2 Best response

In order to analyze the prisoner's dilemma that we represented in figure 1, we will introduce the concept of best response. A best response defines what is the best strategy for an agent i under the joint strategy $s_{-i} = (s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_n)$ of the other agents. For example, in the prisoner's dilemma, if column agent plays C, then the best response for row agent is to play D, since it gets 0, which is better than -1. The same thing happens if column agent plays D, the best response for row agent is to play D. In this case, we see that for any strategy chosen by the other agent, the best response is to always play D.

2.3 Dominated and dominant strategies

Definition 1. *A strategy s'_i is said to be strongly dominated by another strategy s_i if for all joint strategies s_{-i} of the other agents, agent i gets a strictly greater reward by playing s_i instead of s'_i.*

$$R_i(s_i, s_{-i}) > R_i(s'_i, s_{-i}) \qquad \forall s_{-i}$$

Definition 2. *A strategy s'_i is said to be weakly dominated by another strategy s_i if for all joint strategies s_{-i} of the other agents, agent i gets at least an equal or better reward by playing s_i instead of s'_i.*

$$R_i(s_i, s_{-i}) \geq R_i(s'_i, s_{-i}) \qquad \forall s_{-i}$$

In the previous example presented in figure 1, we see that the strategy C was strongly dominated by strategy D for both agents. After eliminating dominated strategies, if only one strategy is left for an agent, we say that this strategy is dominant. So, D is a dominant strategy for both agents. In this case, if both agents act rationally, they will choose strategy D and end up with outcome (D, D). On the other hand, we see that outcome (C, C) would be more desirable since both agents would get better rewards if they could coordinate themselves on this strategy. What we realize is that thinking rationally on the payoff structure of the game does not always yield a desired outcome, even if the agent's strategy is dominant.

2.4 Pareto-optimal strategies

Definition 3. *A joint strategy s Pareto-dominates another joint strategy s' if all agents get at least equal or better rewards by playing s instead of s' and there exists an agent who strictly gets a better reward in joint strategy s.*

$$(\forall i| : R_i(s) \geq R_i(s')) \wedge (\exists i| : R_i(s) > R_i(s'))$$

In the prisoner's dilemma (figure 1), we realize that joint strategy (C, C) Pareto-dominates joint strategy (D, D). In that case, (C, C) would effectively be more desirable for both agents than (D, D). Although, when we want to compare (C, C) to (D, C) for example, we can not say that (C, C) Pareto-dominates (D, C) since in this last joint strategy, one agent gets a better reward and the other gets a lower reward. When this happens, we say that (C, C) and (D, C) are not Pareto-comparable.

Definition 4. *A joint strategy s is Pareto-optimal if there does not exist any joint strategy s' that Pareto-dominates joint strategy s.*

Therefore, in the prisoner's dilemma, there are 3 Pareto-optimal strategies, which are (C, C), (D, C) and (C, D).

2.5 Nash Equilibrium

Definition 5. *A joint strategy s is said to be a strong Nash equilibrium if no agent gets a better or equal reward by changing its strategy s_i to another strategy s'_i when the other agents keep playing the same joint strategy s_{-i}.*

$$R_i(s_i, s_{-i}) > R_i(s'_i, s_{-i}) \qquad \forall i, s'_i$$

Definition 6. *A joint strategy s is said to be a weak Nash equilibrium if no agent gets a better reward by changing its strategy s_i to another strategy s'_i when the other agents keep playing the same joint strategy s_{-i}.*

$$R_i(s_i, s_{-i}) \geq R_i(s'_i, s_{-i}) \qquad \forall i, s'_i$$

For example, in the prisoner's dilemma (figure 1), the joint strategy (D, D) is a strong Nash equilibrium, since if row agent changes its strategy to C while column agent keeps playing D, then row agent gets a lower reward, and the same thing happens if column agent changes its strategy to C while row agent keeps playing D. All the other joint strategies are not Nash equilibrium since an agent playing C can always get a better reward by changing its strategy to D, if the other agent keeps playing the same strategy. From this we can deduce that no Nash equilibrium exists within a strongly dominated strategy. Though, weak Nash equilibrium can exists within weakly dominated strategies.

The most important result showed by Nash [3] is that all games contain at least one Nash equilibrium when we consider Nash equilibrium in mixed strategy. In a mixed Nash equilibrium, no agent has any advantage to modify its probability distribution over its action set.

Definition 7. *A joint mixed strategy p is said to be a mixed Nash equilibrium if no agent gets a better expected reward by changing its mixed strategy p_i to another mixed strategy p'_i when the other agents keep playing the same joint mixed strategy p_{-i}.*

$$E_i(p_i, p_{-i}) \geq E_i(p'_i, p_{-i}) \qquad \forall i, p'_i$$

In this definition, the E_i function returns the expected reward of agent i under joint mixed strategy (p_i, p_{-i}).

For example, we will consider the Matching Penny game illustrated in figure 2.

	H	T
H	1,-1	-1,1
T	-1,1	1,-1

Fig. 2. Matching penny game matrix.

In this game, we can see that no pure joint strategy leads to a Nash equilibrium. Although, there exists a mixed Nash equilibrium which consists in both agents playing 50/50 each action. When this happens, no agent can get a better expected reward by changing its probability distribution if the other agent keeps playing 50/50. On the other hand, if row agent plays H 60% of the time and T 40% of the time, and column agent plays 50/50, then this is not a mixed Nash equilibrium since column agent could get an expected reward of 0.2[2], which is better than 0, by always playing T. Therefore, the only mixed Nash equilibrium in this game is for both agents to play 50/50.

As we can see, the classical game theory concepts we have presented so far all need full knowledge of the actions and rewards of the other agents. Like mentioned earlier, this is a strong assumption that is not always applicable in real life problems since an agent must sometimes take decisions toward unknown agents. In that case, the previously presented concepts are unapplicable since the agent does not have knowledge of the game matrix. The next section will present new concepts to extend game theory in order to learn how to behave in these situations.

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

[2] By playing T, the expected reward of column agent is given by $E_{col}(p) = 0.6 \times 1 + 0.4 \times (-1) = 0.2$. In the later case, by playing 50/50, the expected reward of column agent is given by $E_{col}(p) = 0.5 \times (0.6 \times 1 + 0.4 \times (-1)) + 0.5 \times (0.6 \times (-1) + 0.4 \times 1) = 0$

3.1 Game Model

The game model we will consider will be a simplified repeated Bayesian game in which Nature fixes, once and for all, the types of all agents at the beginning of the learning process, such that only the joint action of the agents affect their payoffs. However, we will extend this game model with an observation function that takes in parameter the joint action and returns the outcome observed by the agents. We will use a modified reward function that takes an outcome and returns its associated reward. This is done in order to let the 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 of all agents and finally R_i the reward function $R_i : \Omega \to \mathbf{R}$ of agent i. Each agent participating in the game only knows its own action set A_i and its reward function R_i. To compute its reward, the agent is given the outcome of the game $o \in \Omega$ corresponding to the joint action of the agents. However, since the agents do not know the observation function O, they do not know which joint action led to this outcome.

While this model assumes that all agents observe the same outcome, we could always extend it with multiple observation functions, with one for each agent defining the outcomes they observe. Although, since the outcome is determinist and only required to compute the reward of the agent, we can get the same result by modifying the reward function of the agent who should not observe the same outcome to the reward it would get with the outcome it would normally observe. As a side note, it would be interesting to consider games where the outcome is stochastic and/or partially observable by the agents, but this will not be further discussed in this report.

3.2 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 8. *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.*

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

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.

While this definition is applied to the modified game model where agents observe a certain outcome of the game, we can also apply this definition to the general game model by replacing the outcome o by the joint strategy s of all agents.

Definition 9. *A joint strategy s 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.*

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

In this last definition, we use the reward function of the general game model, such that it takes a joint strategy instead of an outcome, and s^t represents the joint strategy at time t.

What is important to realize here is that, while the satisfaction equilibrium is applied to a modified game model in this report, it can also be applied to general games where we know and observe the other agents' rewards and actions. Throughout the rest of this report, we will use the former representation, but simply remember that all proofs and definitions presented can be applied to the general game model by replacing the outcome o with the joint strategy s.

We can now represent a satisfaction matrix by transforming a normal form game matrix with the satisfaction function of each agents. For example, we will take the prisoner's dilemma described earlier in figure 1 and then we consider that both agents have a satisfaction constant of -1. In this case, we can transform the original matrix into a satisfaction matrix represented in figure 3.

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

Fig. 3. Prisoner's dilemma satisfaction matrix.

While the agents do not know this satisfaction matrix, since it requires knowledge of all the satisfaction constants, rewards and actions of the other agents, this representation is useful to analyze the behavior of the agents and the satisfaction equilibria of the game.

We can easily see that, in figure 3, the only satisfaction equilibrium is the joint strategy (C, C). While some might argue that in this example the equilibrium would be weak, since row agent can play D and still be satisfied if column agent continues to play C, we recall that if an agent is satisfied, it would never change its strategy (see definition 8), and therefore, a satisfaction equilibrium is always a *strong equilibrium*. If row agent would decide to change its strategy to D, then column agent would not be satisfied by the outcome (D, C), which would eventually result in a change of strategy to D by column agent. The end result is that the initial strategy change by row agent eventually result in an unsatisfying outcome (D, D), which would eventually make it go back to C. This supports the fact that when agents are satisfied, they should not change their strategy.

As a second remark, if we look back at figure 1, we can observe that the satisfaction equilibrium (C, C) 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 agents. 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*[3] *are the only satisfaction equilibria if $\sigma_i = R_i(O(s))\ \forall i$.*

Proof. Let $G = (n, A, \Omega, O, R_1, \ldots, R_n)$ be a game such that it contains a Pareto-optimal joint strategy s. Now, let's define the satisfaction constant of all agents such that : $\sigma_i = R_i(O(s))\ \forall i$. To prove that $O(s)$ and its equivalent outcomes are the only satisfaction equilibria in game G, we will show that $O(s)$ is a satisfaction equilibria and that for all joint strategies s' where $\neg(O(s') \equiv O(s))$, $O(s')$ is not a satisfaction equilibrium.

First let's show that $O(s)$ is a satisfaction equilibrium. We have that for all agents i, $R_i(O(s)) = \sigma_i$. By definition of the satisfaction function, we conclude that $S_i(R_i(O(s))) = 1\ \forall i$. Therefore, the outcome $O(s)$ is a satisfaction equilibrium. Similarly, all equivalent outcomes will be satisfaction equilibrium since if $R_i(O(s)) = R_i(O(s'))\ \forall i$, then $S_i(R_i(O(s'))) = 1\ \forall i$ and we have that $O(s')$ is a satisfaction equilibrium.

To show that all other outcomes $O(s')$ such that $\neg(O(s') \equiv O(s))$ are not satisfaction equilibrium, we will consider 2 cases: either strategy s Pareto-dominates s' or either strategy s' is not Pareto-comparable to s. These are the only 2 cases we must consider since $\neg(O(s') \equiv O(s))$ and s is Pareto-optimal.

Case 1 : Strategy s Pareto-dominates strategy s'.

Since s Pareto-dominates s', there exists an agent i such that $R_i(O(s')) < R_i(O(s))$. Therefore, since $\sigma_i = R_i(O(s))$, we conclude that $S_i(R_i(O(s'))) = 0$. Consequently, $O(s')$ is not a satisfaction equilibrium because there exists an agent i that is not satisfied by the joint strategy s'.

[3] 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' : $R_i(o) = R_i(o')\forall i$

Case 2 : Strategy s' is not Pareto-comparable to strategy s.

Since s and s' are not Pareto-comparable, there exists an agent i such that $R_i(O(s')) > R_i(O(s))$ and an agent j such that $R_j(O(s')) < R_j(O(s))$. Therefore, since $\sigma_j = R_j(O(s))$, we conclude that $S_j(R_j(O(s'))) = 0$. Consequently, s' is not a satisfaction equilibrium because there exists an agent j that is not satisfied by the joint strategy s'.

Conclusion : If strategy s is Pareto-optimal, the outcome $O(s)$ and its equivalent outcomes are the only satisfaction equilibria if $\sigma_i = R_i(O(s))\ \forall i$. Q.E.D.

From this theorem, 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 algorithms 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.

3.3 Satisfying and unsatisfying Strategies

The concept of satisfying strategies is equivalent to the concept of dominant strategies when we consider the satisfaction matrix. But first, we will define the set of possible outcomes that an agent can observe by playing a certain strategy s_i. We will denote this set as Ω_{s_i} :

$$\Omega_{s_i} = \{o | (\exists s_{-i}| : O(s_i, s_{-i}) = o)\}$$

Basically, Ω_{s_i} contains a certain outcome o if there exists a joint strategy s_{-i} combined to strategy s_i that leads to this outcome. From this, we can define a satisfying strategy as follows:

Definition 10. *A satisfying strategy is a strategy s_i where agent i is always satisfied in all its possible outcomes:*

$$S_i(R_i(o)) = 1 \qquad \forall o \in \Omega_{s_i}$$

While a satisfying strategy may exists for a certain agent i, the agent will not generally know it unless it knows all the possible outcomes associated with each of its actions. Though, we must consider these strategies if we want to predict the outcome of the game, since once an agent starts playing a satisfying strategy, it will never change its strategy again, forcing the other agents to play a satisfaction equilibrium within the subset of outcomes Ω_{s_i}. This can lead to some problems in finding an equilibrium if no satisfaction equilibrium exists within Ω_{s_i}.

Definition 11. *An unsatisfying strategy is a strategy s_i where agent i is always unsatisfied in all its possible outcomes:*

$$S_i(R_i(o)) = 0 \qquad \forall o \in \Omega_{s_i}$$

An unsatisfying strategy will never contain a satisfaction equilibrium since one agent will always be unsatisfied. Moreover, if an agent knows that one of its strategy s_i is unsatisfying, then it should rationally never play this strategy again and the outcome of the game will never be within Ω_{s_i}.

To illustrate the consequences of these kind of strategies, we can construct a satisfaction matrix as in figure 4.

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

Fig. 4. Strategy A for row agent is satisfying and strategy B for column agent is unsatisfying.

In this game we see that row agent has a satisfying strategy A and that column agent has an unsatisfying strategy B. 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). In the case where column agent is aware that B is an unsatisfying strategy, we can predict that both agents will reach equilibrium[4] in point (A, A), even though this is not a satisfaction equilibrium. However, if column agent does not know that B is unsatisfying, then it will play both A and B continuously, never reaching an equilibrium, since both of these outcomes are unsatisfying for column agent. On the other hand, if we consider that the satisfaction constant of an agent can change over time, then depending on the payoff structure of the game, satisfying strategies can become non-satisfying strategies.

3.4 Games with multiple Satisfaction Equilibria

The examples we presented so far had only a specific outcome that lead to a satisfaction equilibrium. However, depending on how the satisfaction constants are defined, more than one outcome in a game can lead to a satisfaction equilibrium. To illustrate this problem, we will take the example of the battle of sexes represented in figure 5, that presents the game matrix and the satisfaction matrix where both agents have 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.

[4] While (A, A) is not a satisfaction equilibrium, (A, A) would be a weak Nash equilibrium in this case if we consider that agents can reason on their satisfying and unsatisfying strategy, since strategy A weakly dominates strategy B for both agents in the satisfaction matrix.

$$\begin{array}{|c|c|c|}\hline & B & F \\ \hline B & 2,1 & 0,0 \\ \hline F & 0,0 & 2,1 \\ \hline \end{array} \Longrightarrow \begin{array}{|c|c|c|}\hline & B & F \\ \hline B & 1,1 & 0,0 \\ \hline F & 0,0 & 1,1 \\ \hline \end{array}$$

$$(\forall i| : \sigma_i = 1)$$

Fig. 5. Battle of sexes game matrix (left) and satisfaction matrix (right).

3.5 Games with no pure strategy Satisfaction Equilibrium

In some games, such as zero sum games with no tie strategy[5], it is impossible to find a satisfaction equilibrium in pure strategy, unless we define the satisfaction constants to the minimum possible reward. Though, this is not really interesting since we definitely do not want our agents to be satisfied by the minimum reward they can get, since they are already sure to get at least this minimum from any strategy they could play. As an example, we will consider the Matching Penny game illustrated before in figure 2. In this game, the only possible way to get a pure strategy satisfaction equilibrium is to set the satisfaction constants to -1, which is the minimum possible reward in the game. On the other hand, since this is a zero sum game, we can presume that an agent would be satisfied if it gets at least 0. Figure 6 presents the resulting satisfaction matrix under satisfaction constants set to -1 and 0.

$$\begin{array}{|c|c|c|}\hline & H & T \\ \hline H & 1,1 & 1,1 \\ \hline T & 1,1 & 1,1 \\ \hline \end{array} \qquad \begin{array}{|c|c|c|}\hline & H & T \\ \hline H & 1,0 & 0,1 \\ \hline T & 0,1 & 1,0 \\ \hline \end{array}$$

$$(\forall i| : \sigma_i = -1) \qquad (\forall i| : \sigma_i = 0)$$

Fig. 6. Matching penny transformation with $\sigma_i = -1$ (left) and $\sigma_i = 0$ (right).

We can clearly see that no pure strategy satisfaction equilibrium exists if we set the constants higher than the minimum reward -1. However, if we allow mixed strategies, then both agents could achieve an expected reward of 0 if they both play H and T 50% of the time. We conclude from this that satisfaction equilibria can exist in such game if we base the satisfaction principle on the expected reward of the agents' mixed strategies.

[5] In zero sum games, a tie strategy is a pure joint strategy s such that the reward of all agents is 0.

3.6 Satisfaction Equilibrium in mixed strategy

Definition 12. *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.*

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

$E_i(p)$ represents the expected reward of agent i under the joint mixed strategy p. Condition (ii), as in definition 8, ensures that an agent keeps the same mixed strategy when it is satisfied.

While this works in theory, 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.

4 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.

4.1 Pure Satisfaction Equilibrium with fixed constants

The most basic case we might want to consider is the case where the satisfaction constant is fixed and the agent tries to find a pure strategy where it will always be satisfied under this constant.

Our algorithm 1 implements the satisfaction principle in the most basic way: if the agent is satisfied, it keeps its current action, else it chooses a random action in its set of actions to replace its current action.

Here, 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. Evidently, in games where there exists no satisfaction equilibrium under the agents satisfaction constants, then agents will never reach an equilibrium. Furthermore, if agent i have a satisfying strategy s_i, then we are not sure to reach a satisfaction equilibrium if no outcome within Ω_{s_i} is a satisfaction equilibrium (see figure 4 for an example).

4.2 Empirical results with the PSEL Algorithm

We now present results obtained with the PSEL algorithm in different games. We have used 2 usual games, i.e. the prisoner's dilemma with satisfaction constants set to -1 for both agents (see figure 3 for the corresponding satisfaction matrix) and the battle of sexes with satisfaction constants set to 1 for both agents (see figure 5 for the corresponding satisfaction matrix). We also used a cooperative game presented in figure 7 with satisfaction constants set to 3.

Algorithm 1 PSEL: Pure Satisfaction Equilibrium Learning

```
Function PSEL($\sigma_i$, $K$)

Returns: The chosen strategy after $K$ repeated plays.
Inputs: $\sigma_i$: The satisfaction constant of agent i.
        $K$: The number of times we want to repeat the game.
Statics: $A_i$: The set of actions of agent i.
         $R_i$: The reward function of agent i.

$s_i \leftarrow ChooseAction()$
for $n = 1$ to $K$ do
  Play $s$
  Observe outcome $o$
  $r_i \leftarrow R_i(o)$
  if $r_i < \sigma_i$ then
    $s_i \leftarrow ChooseAction()$
  end if
end for
return $s_i$
```

$$
\begin{array}{|c|c|c|c|}
\hline
 & A & B & C \\
\hline
A & 0,0 & 1,1 & 0,0 \\
\hline
B & 2,2 & 0,0 & 0,0 \\
\hline
C & 0,0 & 0,0 & 3,3 \\
\hline
\end{array}
\Longrightarrow
\begin{array}{|c|c|c|c|}
\hline
 & A & B & C \\
\hline
A & 0,0 & 0,0 & 0,0 \\
\hline
B & 0,0 & 0,0 & 0,0 \\
\hline
C & 0,0 & 0,0 & 1,1 \\
\hline
\end{array}
$$

$$(\forall i| : \sigma_i = 3)$$

Fig. 7. Cooperative game matrix (left) and satisfaction matrix (right).

In this cooperative game, we set the satisfaction constants to 3 for both agents such that the only satisfaction equilibrium is joint strategy (C, C). Finally, we also used a bigger game to verify the performance of our algorithm when the joint strategy space is bigger. This game is presented in the following figure 8.

	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. 8. Big game matrix.

In this big game, the satisfaction constants were set to 5 for both agents and therefore, the only satisfaction equilibrium was joint strategy (E, D).

For each of these four games, we ran 5000 simulations, consisting of 5000 repeated plays per simulation, varying the random seeds of the agents each time. In figure 9, we present for each of these games the convergence rate to a SE, the average number of steps required to converge to such an equilibrium (with 95% confidence interval), the number of SE present and the number of possible outcomes.

Fig. 9. Convergence rate and steps needed to converge to a SE in different games with the PSEL algorithm

Game	$\|A\|$	n_{SE}	conv. rate	Avg. steps
Prisoner's Dilemma	4	1	100%	8.67 ± 0.23
Battle of Sexes	4	2	100%	1.97 ± 0.04
Cooperative Game	9	1	100%	8.92 ± 0.23
Big Game	64	1	100%	67.95 ± 1.89

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 cases, we always converged to a SE within the allowed 5000 repeated plays. Therefore, we see from these results that when the satisfaction constants are well defined, we seem to eventually converge toward a Pareto-optimal satisfaction equilibrium[6] (POSE).

4.3 Improving the 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, our algorithm 2 presents an exploration strategy that can be used in the $ChooseAction$ function.

In this function, we first compute a probability for each action which corresponds to the inverse of the times we have chosen them. Then in the second loop, we normalize the probabilities such that it sums to 1. Finally, we use the Random function to select a random action according to the computed probability distribution. The following results presented in figure 10, using the same games and parameters as in figure 9, show a slight improvement in the average number of steps required to converge to a satisfaction equilibrium.

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

Algorithm 2 ChooseAction

```
Function ChooseAction()

Returns: An action within the set of actions $A_i$ of the agent
Statics: $A_i$: The set of actions of agent $i$.
         $N[0 \dots |A_i| - 1]$: Number of times we have chosen each action
                                 Initially at 1 for all actions.

$probDist[0 \dots |A_i| - 1] \leftarrow$ an array initialized with 0 values
$sum \leftarrow 0$
for $a = 0$ to $|A_i| - 1$ do
    $prob \leftarrow 1/N[a]$
    $sum \leftarrow sum + prob$
    $probDist[a] \leftarrow prob$
end for
for $a = 0$ to $|A_i| - 1$ do
    $probDist[a] \leftarrow probDist[a]/sum$
end for
$a \leftarrow Random(probDist)$
$N[a] \leftarrow N[a] + 1$
return $a$
```

Fig. 10. Average number of steps required to converge to a SE with different exploration strategies.

Game	Uniform distribution Avg. steps	Exploration strategy Avg. steps	Improvement
Prisoner's Dilemma	8.67 ± 0.23	6.72 ± 0.18	22.49%
Battle of Sexes	1.97 ± 0.04	1.95 ± 0.04	1.02%
Cooperative Game	8.92 ± 0.23	7.82 ± 0.19	12.33%
Big Game	67.95 ± 1.89	61.51 ± 1.65	9.48%

4.4 Problematic games and convergence problems

The previous results were quite satisfying provided that we reach a POSE in all cases. However, we will see that some specific payoff structures can pose problem to the efficiency of this algorithm. For example, let's suppose we have a game as in figure 11.

In this game, there exists a unique Pareto-optimal strategy (A, A). Now let's suppose we set the satisfaction constants of both agents to 1, the corresponding satisfaction matrix is the same as the game matrix, as shown in figure 11. But, what we can see in this example is that we will never reach the point of 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

$$
\begin{array}{|c|c|c|c|}
\hline
 & A & B & C \\
\hline
A & 1{,}1 & 0{,}1 & 0{,}1 \\
\hline
B & 1{,}0 & 1{,}0 & 0{,}1 \\
\hline
C & 1{,}0 & 0{,}1 & 1{,}0 \\
\hline
\end{array}
\Longrightarrow
\begin{array}{|c|c|c|c|}
\hline
 & A & B & C \\
\hline
A & 1{,}1 & 0{,}1 & 0{,}1 \\
\hline
B & 1{,}0 & 1{,}0 & 0{,}1 \\
\hline
C & 1{,}0 & 0{,}1 & 1{,}0 \\
\hline
\end{array}
$$

$$(\forall i| : \sigma_i = 1)$$

Fig. 11. A problematic game matrix (left) and satisfaction matrix (right).

the other agent plays A. Also, there is no point where both agents are unsatisfied that could allow a direct transition to joint strategy (A, A). From this, we conclude that if both agents does 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.

To illustrate this problem, we will represent the possible strategy change of the agents as a graph, where the nodes correspond to the joint strategies of the agents, and the oriented links between these nodes represent the possible strategy transitions when an agent is unsatisfied. For example, in the prisoner's dilemma, if we consider the corresponding satisfaction matrix presented in figure 3, we get the following transition graph presented in figure 12.

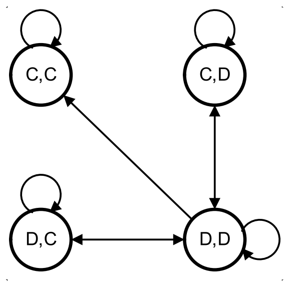

Fig. 12. Possible strategy transitions under the PSEL algorithm in the prisoner's dilemma

Since (D, D) is a joint strategy where all agents are unsatisfied, there are transitions from this node to all the nodes of the graph. When we consider the joint strategy (D, C), the agent playing D is satisfied but the other playing C is unsatisfied, so only the agent playing C can change its strategy, so we have 2 possible transitions, one to (D, D) and one to (D, C). The same applies to strategy (C, D). When we consider strategy (C, C), since this is a satisfaction equilibrium, the only possible transition is a transition to itself, since no agent will change its strategy.

Now let's consider the problematic game presented in figure 11. When we consider the transition graph corresponding to the satisfaction matrix of this game, we get the following graph presented in figure 13.

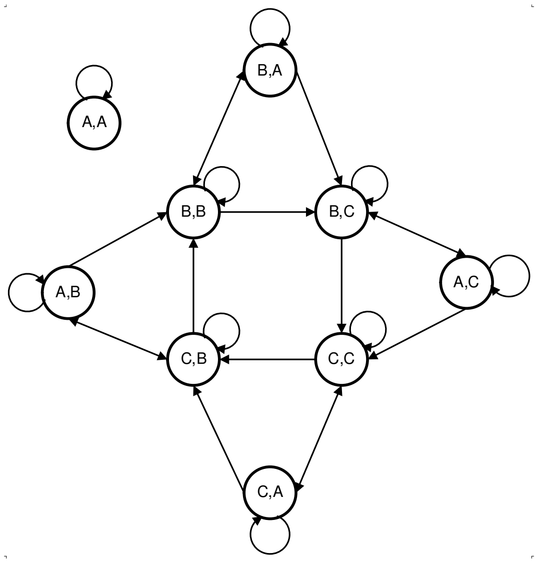

Fig. 13. Possible strategy transitions in the problematic game of figure 11

What we realize is that there exists no path to the node (A, A) from any other node of the graph. When this happens, we can see that the PSEL algorithm is incapable of providing a good solution, if the agents do not start on strategy (A, A). In this case, the only possible equilibrium would be to consider a mixed satisfaction equilibrium where the satisfaction constants would be set around 0.5, since a mixed equilibrium exists with an expected reward of 0.5 when both agents play 50/50 between B and C.

Another problem that can arise is when an agent has a satisfying strategy and that the satisfaction equilibrium of the game is not achieved using this strategy. In this case, even if there exists a path from the current joint strategy to the satisfaction equilibrium in the transition matrix, we are not sure to reach the satisfaction equilibrium since, if a transition to the satisfying strategy happens, then the other agent won't be able to force the agent playing its satisfying strategy to change its strategy, and therefore agents will never reach the equilibrium. For example, we will consider the satisfaction matrix of the game presented in figure 4. When we consider the possible strategy transitions of the different agents, we get the following transition graph presented in figure 14.

As we can see, when the agents play (A, A) or (A, B), they can't come back to the nodes (B, B) or (B, A). Therefore, we see that the PSEL algorithm would

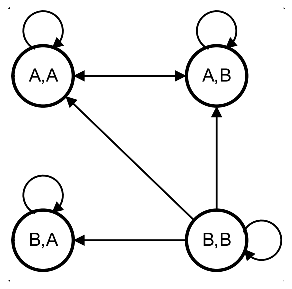

Fig. 14. Possible strategy transitions in a game with a satisfying strategy

be unable to converge to the satisfaction equilibrium (B, A) if the first agent starts playing A.

Empirical results over 5000 simulations in the last two problematic games presented effectively gives poor results as seen in figure 15.

Fig. 15. Convergence rate to a SE in problematic games with the PSEL algorithm

Game	$\|A\|$	conv. rate
Problematic Game	9	10.88%
Game with satisfying strategy	4	33.26%

While these problems seem to defeat the interest of pursuing a satisfaction equilibrium, one solution is to allow the satisfaction constant to vary according to the satisfaction of the agents. In some game structure, the previously mentioned problems only happen when the satisfaction constants of the agents are set to some specific values. For example let's consider the game matrix presented in figure 16.

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

Fig. 16. A problematic game matrix.

If both agents have a satisfaction constant set to 1, then we get the problematic satisfaction matrix presented in figure 11. Although, if we move up the satisfaction constant of both agents to 2, then strategy (C, C) becomes unsatisfying for both agents and it is therefore possible to reach the satisfaction equilibrium

(A, A), since both agents can always make a sequence of transitions to reach the joint strategy (C, C) and then make a transition to joint strategy (A, A). Although, in the game matrix presented in figure 11, there was no way to change the satisfaction constants such that convergence to joint strategy (A, A) was certain.

4.5 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.

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* $|A|/n_{SE}$*.*[7]

Proof. Let a game $G = (n, A, \Omega, O, R_1, \ldots, R_n)$ and the satisfaction constants of all agents be defined such that $S_i(R_i(o)) = S_j(R_j(o))\forall i,j,o$. We have that, by definition of the satisfaction constants, in any outcome o, all agents are all satisfied or all unsatisfied by outcome o. In the case where all agents are satisfied, the PSEL algorithm has converged to a satisfaction equilibrium and the agents will keep playing the same strategy. We will therefore consider the case where all agents are unsatisfied. In this situation, every agent may decide to change its strategy or not and, since all agents use a random uniform distribution to choose their new action, all the joint strategies are equally probable to be chosen. Therefore, the probability p that the agents immediately converge to a SE is given by $p = n_{SE}/|A|$. In the case where the agents do not choose the right joint strategy such that they are still all unsatisfied, the same process will repeat again with the same probabilities, until they reach a SE and converge. Therefore, we seek to find the probability that a first successful transition to a SE happens within K play. This is given by a geometric law where the probability of success is parameter $p = n_{SE}/|A|$. In a geometric law, the probability that it takes more than K trials to get a first success is given by $P(k > K) = q^K$ where $q = 1 - p$. Therefore the probability that we have a first success within K trials is given by $P(k \leq K) = 1 - P(k > K) = 1 - q^K$. Furthermore, the expected number of trials required to get a first success in a geometric law is given by $E = 1/p$. From these properties of the geometric law, we conclude that we will converge to a SE within K plays with probability $1 - q^K$ where $q = 1 - n_{SE}/|A|$ and that the expected number of plays required to converge is given by $|A|/n_{SE}$. Q.E.D

This will be always true in cooperative games[8] if we use the same satisfaction constant for all agents. In this case, since all agents have the same rewards

[7] $|A|$ represents the joint action space size and n_{SE} is the number of SE in the game.
[8] A cooperative game is a game where all agents have the same reward function

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.

5 Learning the Satisfaction Constant

While the *PSEL* algorithm has showed interesting performance in some games, it has the disadvantage that the satisfaction constant must correctly be 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.

5.1 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 decreases 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 rate 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 toward 0, by multiplying it by the constant $\gamma_i \in]0, 1[$ after each play. The matrix S keeps an history of the last n states of satisfaction for each action and the matrix Σ keeps, for each action, an 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 can consider that we have reach a satisfaction equilibrium since their satisfaction constant will be almost stable. While we are not guaranteed to converge toward a *POSE*, we will see that in practice, this algorithm yields almost a convergence rate of 100% toward the *POSE* in any non problematic games.

Algorithm 3 LHSL : Limited History Satisfaction Learning

```
Function LHSL($\delta_i$, $\gamma_i$, $n_i$)
Returns: A tuple $(s_i, \sigma_i)$ where
         $s_i$ : The chosen strategy by agent $i$ after convergence
         $\sigma_i$ : The learned satisfaction constant for agent $i$
Inputs: $\delta_i$: The increment/decrement of the satisfaction constant of agent $i$.
        $\gamma_i$: The constant multiplying $\delta_i$ after each play.
        $n_i$: The history size of agent $i$.
Statics: $A_i$: The set of actions of agent $i$.
         $R_i$: The reward function of agent $i$.
         $\epsilon_i$: The convergence threshold of agent $i$.
$\sigma_i \leftarrow min(r_i)$
$s_i \leftarrow ChooseAction()$
$S[0..|A_i|-1, 0..n-1] \leftarrow$ 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 $s_i$
  Observe outcome $o$
  $r_i \leftarrow R_i(o)$
  $lastStrategy \leftarrow s$
  $satisfied \leftarrow (r_i < \sigma_i)$
  if Not $satisfied$ then
    $s_i \leftarrow ChooseAction()$
    $tmp \leftarrow \sigma_i + \delta_i$
  else
    $tmp \leftarrow \sigma_i - \delta_i$
    for $j = 0$ to $n_i - 1$ do
      if $tmp \geq \Sigma[lastStrategy, j]$ and $S[lastStrategy, j] = false$ then
        $tmp \leftarrow \sigma_i$
      end if
    end for
  end if
  for $j = n_i - 1$ to $1$ do
    $\Sigma[lastStrategy, j] \leftarrow \Sigma[lastStrategy, j-1]$
    $S[lastStrategy, j] \leftarrow S[lastStrategy, j-1]$
  end for
  $\Sigma[lastStrategy, 0] \leftarrow \sigma_i$
  $S[lastStrategy, 0] \leftarrow satisfied$
  $\sigma_i \leftarrow tmp$
  $\delta_i \leftarrow \delta_i \cdot \gamma_i$
end while
return $(s_i, \sigma_i)$
```

5.2 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 rate to the *POSE* obtained with the best γ_i value and history sizes we have tested[9]. We also compare these results to the special case where we do not use an 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}.

Fig. 17. Convergence rate to a POSE in different games with the LHSL algorithm

		With history			Without history	
Game	$\|A\|$	conv. rate	γ_i	n_i	conv. rate	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

In all cases, the best results, showed in figure 17, were obtained with the exploration strategy we have presented in section 4.3. In most games, except the problematic game (figure 11, we were able to get a convergence rate near 100%. We can also see that the use of an history offers a significant improvement over the results we obtain without an history. As a side note, the convergence rate of the *LHSL* algorithm seems to vary a lot depending on the history sizes and gamma values. This is illustrated in figure 18.

The first graphic in figure 18 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. Although, in general, the more slowly we decrement δ and the more bigger the history size is, the better are the results. In the second graphic, we compare the convergence rate of the 2 different exploration approaches under different γ values for the prisoner's dilemma, in the case where no history was used. This graphic confirms that the exploration strategy presented in section 4.3 improves slightly the convergence rate of the *LHSL* algorithm.

6 Conclusion and future works

While this report covered a lot of new concepts, it laid out only the basic theoretical foundations of the satisfaction equilibrium. The algorithms we have pre-

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

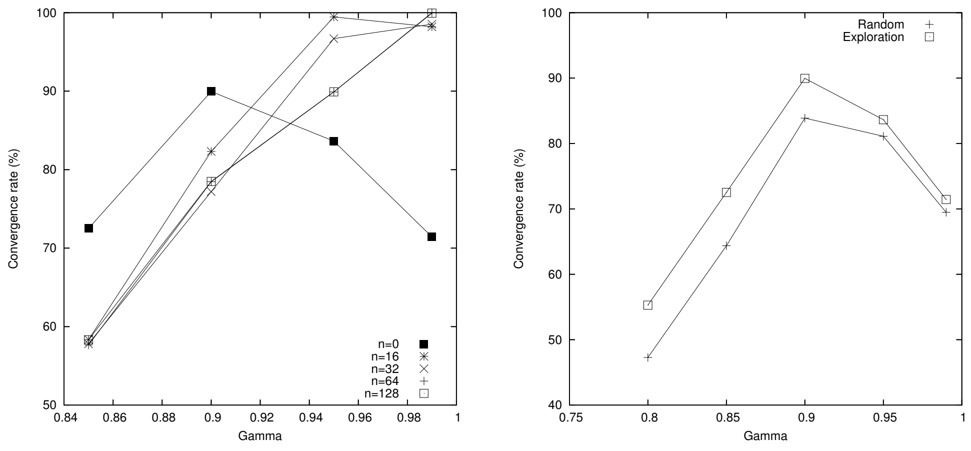

Fig. 18. Convergence rate to a *POSE* in the prisoner's dilemma under different γ values, history sizes and exploration strategies

sented have shown great performance in practice, but we have seen some games with specific payoff structures that could pose problems or rend 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 to these problems. Although, what we may realize 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 converge 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. Nash, J.: Non-cooperative games. Annals of Mathematics **54** (1951) 286–295