

A Distributed Algorithm to Obtain Repeated Games Equilibria With Discounting

Juan Parras*, Santiago Zazo

*Information Processing and Telecommunications Center
Universidad Politécnica de Madrid
ETSI Telecomunicación, Av. Complutense 30, 28040 Madrid (Spain)*

Abstract

We introduce a distributed algorithm to negotiate equilibria on repeated games with discounting. It is based on the Folk Theorem, which allows obtaining better payoffs for all players by enforcing cooperation among players when possible. Our algorithm works on incomplete information games: each player needs not knowing the payoff function of the rest of the players. Also, it allows obtaining Pareto-efficient payoffs for all players using either Nash or correlated equilibrium concepts. We explain the main ideas behind the algorithm, explain the two key procedures on which algorithm relies on, provide a theoretical bound on the error introduced and show empirically the performance of the algorithm on four well-known repeated games.

Keywords: Repeated games, Folk Theorem, Average Discounted Payoff, Nash equilibrium, Correlated equilibrium, Multiagent learning

1. Introduction

Multi agent learning (MAL) is a field which deals with the problem of multiple agents trying to learn while interacting with the rest of players. This problem has been extensively studied using game theory tools: the learning environment is modeled as a game and each of the agents is a player trying to maximize its own payoff, which depends on the actions of the rest of the players.

A general framework used to model the MAL problem is based on stochastic games [1]. In these games, each player tries to solve a Markov Decision Process, whose transitions and payoffs are coupled to the actions of the rest of the players. Several algorithms have been proposed for these games, such as Minimax-Q [2], WoLF [3], CE-Q [4], OPVar-Q [5] and Pepper [6]. A problem which arises

*Corresponding author

Email addresses: j.parras@upm.es (Juan Parras), santiago.zazo@upm.es (Santiago Zazo)

with stochastic games is the dimensionality: a large number of states significantly hardens the problem of learning. To avoid this, several algorithms based on approximations have been used, such as FAL-SG [7], AlphaGo Zero [8] or Libratus [9]. The recent advances in deep learning have also brought several new algorithms to the MAL problem [10].

However, many MAL problems can be solved using repeated games, which are stochastic games with only one state (i.e., the payoff functions of the players do not change along the game). In a repeated game, there is a static game, called stage or one-shot game, which is repeated a certain number of times [11]. The payoff obtained in each stage game by each player are averaged in order to obtain the total expected payoff of the game using a discount factor or not. If a discount factor is used, the payoff at the first stages have a larger weight on the total payoff. The simplest strategy for a repeated game consists in playing the equilibrium of the static game. Yet it might also be possible for all players to obtain higher payoffs by using different strategies: this fact is collected by the Folk Theorems [12, 11]. Hence, when learning a repeated game, it is possible either to learn the static equilibrium or learning a possibly better equilibrium by using the Folk Theorem.

The topic of repeated games is nowadays subject to an intense research from different perspectives as recent works show. To mention some, it has been studied from a physics point of view [13], [14], from a social and natural sciences perspective [15] and it has also been studied under a computer science perspective [16]. This amount of research has produced many algorithms to learn the stage game equilibrium, i.e., without using the Folk Theorem, such as Regret Matching [17] [18], ReDVaLeR [19] or AWESOME [20]. There are also algorithms that explicitly use the Folk Theorem without discounting, as [21] or [22]. M-Qubed [23] makes an implicit use of the Folk Theorem by setting a bound on the maximum losses that the players are willing to take.

The algorithms mentioned so far are based on the idea of online learning: players learn how to play based on the previous behavior and rewards obtained. In this approach, guaranteeing a certain performance bound while learning is important, as ReDVaLeR or M-Qubed do for games without discounting. However, in games with discounting, the learning process can cause that the players obtain a low payoff, as the first stages of the game are the ones that weight the most in the total payoff as well as the ones in which the players have not yet learned how to act optimally. A different approach is based in negotiation: prior to play, the agents interchange messages in order to negotiate how to play. This approach is used, for instance, in [24].

In this paper, we introduce Communicate & Agree (CA), a novel algorithm that aims to fill a gap in current MAL algorithms: CA is a MAL algorithm that computes equilibria of repeated game with discounting using the Folk Theorem and negotiation. Some of CA highlights are:

- CA is a fully distributed algorithm, which does not need a central entity to control the negotiation process.
- CA explicitly uses the Folk Theorem, and hence, it can obtain better

payoffs than the ones obtained by simply repeating the static game equilibrium.

- When there are several payoffs that could be achieved by making use of Folk Theorem tools, CA selects payoffs that are Pareto efficient.
- CA can be applied to compute different kind of equilibria, such as Nash or Correlated equilibria.
- CA can solve games of incomplete information, as each player needs not knowing the payoff functions of the others. Yet we assume that all players can observe the actions of the other players, that is, we assume perfect monitoring [12].

When compared to previous approaches, we observe that none of them has all of CA features: none of these algorithms is able to deal with discounted payoffs using the Folk Theorem, nor it is fully distributed nor it is based on negotiation.

The rest of the paper goes as follows: in Section 2, a brief introduction to repeated game theory and equilibria concepts is given, then in Section 3, CA algorithm is introduced and described. Section 4 provides a theoretical bound on the error that CA induces. Section 5 provides a testbench of CA algorithm performance using some well-known games. Finally, Section 6 presents some concluding remarks.

2. Repeated games background

2.1. Static games

A static game G is a tuple $\langle N, A, u \rangle$, where:

- N denotes the number of players, numbered as $1, \dots, i, \dots, N$.
- $a_i \in A_i$ are the actions available to player i , being A_i the set of actions available to player i . $A \equiv \prod_i A_i$ is the set of actions available to all players, and $a \in A$ is a vector of actions of all players.
- u is a function $u : A \rightarrow \mathbb{R}^N$ that gives the game payoffs as a function of the actions of the players. Each component of u is the payoff function for player i : $u_i : A \rightarrow \mathbb{R}$.

If an action set is finite, each of its component can be denoted as pure action. Mixed actions are distribution probabilities for each player that map each pure action of the player with the probability that the player plays that action. We will denote pure actions a^p by using the superscript p and mixed actions a without superscript.

Observe that the payoff functions denote the nature of the game. It can be purely competitive: the gains of some players are the losses of the others and thus, $\sum_i u_i = 0$ (zero-sum games). It can be purely cooperative, if all players

share the same payoff function: $\sum_i u_i = Nu_i$. It can finally range between these two extreme cases: these games are known as general-sum or non-zero sum games, on which we focus.

2.2. Repeated games of perfect monitoring

A repeated game is built by playing repeatedly over T periods a static game, called stage game. We assume repeated games of infinite duration (i.e., $T = +\infty$). We define a repeated game of perfect monitoring $G(D)$ as a tuple $\langle N, A, u, \sigma, \mathcal{H}^t, D \rangle$ where:

- N, A and u are defined as in the static game.
- $\mathcal{H}^t \equiv A^t$ denotes the set of histories. A history $h^t \in \mathcal{H}^t$ is a list of t -action profiles played in periods $t = [0, \dots, t-1]$. h^t is the past actions profile.
- A strategy for player i is a mapping from the set of all possible histories into the set of actions: $\sigma_i : \mathcal{H} \rightarrow A_i$.
- Average discounted payoff to player i is given by:

$$U_i(\sigma) = (1 - \delta) \sum_{t=0}^{\infty} \delta^t u_i(a^t(\sigma)) \quad (1)$$

where $\delta \in [0, 1)$, $\delta \in D$ is the discount factor. Observe that we will use U_i (capitalized) to refer to the average payoff of the repeated game (1) for player i and u_i to denote the payoff of the static game for player i .

We focus on games of perfect monitoring [12]: the history h^t is known to all players. We also focus in observable mixed actions: if all players randomize their actions (i.e., use mixed actions), the output of their randomizing devices is also observed by other players [25].

2.3. Equilibrium of the game

A Nash equilibrium (NE) [26] of an N -player game is an action vector such that no player can gain by deviating unilaterally. Mathematically, for a static game, an action vector a is a Nash ϵ_i -equilibrium of the game G , where $\epsilon_i \geq 0, \forall i \in N$ if:

$$u_i(a_i, a_{-i}) \geq u_i(a'_i, a_{-i}) - \epsilon_i, \quad \forall i, \forall a'_i \neq a_i \quad (2)$$

where a_i denotes the possibly mixed action of player i and A_{-i} the action of all players but player i . When $\epsilon_i = 0, \forall i$, we have a Nash equilibrium. A non-zero sum game is guaranteed to have at least one NE [27].

The correlated equilibrium (CE) concept was introduced by Aumann [28]. A correlated ϵ_i -equilibrium of a game G is a distribution probability ϕ over the set

of all pure actions $a^p \in A^p = \prod_i A_i^p$ such that no player can gain by deviating unilaterally [11]:

$$\begin{aligned} \sum_{a_{-i}^p \in A_{-i}^p} \phi(a_{-i}^p | a_i^p) u_i(a_i^p, a_{-i}^p) &\geq \\ \sum_{a_{-i}^p \in A_{-i}^p} \phi(a_{-i}^p | a_i^p) u_i(a_i^{p'}, a_{-i}^p) - \epsilon_i, &\quad \forall i, \forall a_i^{p'} \in A_i^p, a_i^p \neq a_i^{p'} \end{aligned} \quad (3)$$

When $\epsilon_i = 0, \forall i$, we have a CE. The set of NE is contained in the set of CE. Hence, NE are particular cases of CE, thus a non-zero sum game always has at least one CE. The CE set is easier to compute than the NE set [29].

For repeated games, the equilibrium concepts are defined similarly. A strategy profile σ is a Nash ϵ_i equilibrium of the repeated game $G(D)$ if:

$$U_i(\sigma) \geq U_i(\sigma_i', \sigma_{-i}) - \epsilon_i, \quad \forall i, \forall \sigma_i' \neq \sigma_i \quad (4)$$

Similarly, a distribution probability ϕ is a ϵ_i -correlated equilibrium of the repeated game $G(D)$ if:

$$\begin{aligned} \sum_{a_{-i}^p \in A_{-i}^p} \phi(a_{-i}^p | a_i^p) U_i(a_i^p, a_{-i}^p) &\geq \\ \sum_{a_{-i}^p \in A_{-i}^p} \phi(a_{-i}^p | a_i^p) U_i(a_i^{p'}, a_{-i}^p) - \epsilon_i, &\quad \forall i, \forall a_i^{p'} \in A_i^p, a_i^p \neq a_i^{p'} \end{aligned} \quad (5)$$

In each case, when $\epsilon_i = 0, \forall i$, we have an NE or a CE. The main difference between the static and the repeated game equilibria concepts is that the equilibria of the repeated games are defined in terms of the averaged discounted payoff (1). In the repeated game, the equilibrium concept is refined by imposing additionally the sequential rationality requirement: the behavior followed by the players must be optimal in all circumstances. This gives place to the Subgame Perfect Equilibrium (SPE) concept [12].

The equilibria of the stage game (static equilibria) will also be equilibria of the repeated game. But additionally, the Folk Theorems state that for sufficiently high values of δ (i.e., δ is close enough to 1), there will be even more SPE in the game [12]. There are many possible strategies σ that are potential candidates to be SPE. In this paper we will focus on grim trigger strategy, because it is a simple strategy that gives good results [12]. However, our approach is not specific for grim trigger, thus it can be adapted to other strategies as well. Grim trigger has two actions profiles, a_o and a_p . The latter is a punishment strategy, which must be an equilibrium of the game. We consider a_p to be an equilibrium of the stage game, which will provide player i with payoff $U_i(a_p)$. The strategy a_o is such that provides each player a payoff $U_i(a_o) \geq U_i(a_p)$. Also, a_o must satisfy that no player can improve her payoff by a unilateral deviation from a_o . This condition is checked by using the punishment strategy a_p : since all players can observe the actions of other players (we assume perfect monitoring), if

any of them deviates, the rest will switch to the punishment strategy forever. Mathematically, using Bellman equation to express (1) in the case of NE (4), that means:

$$\begin{aligned} (1-\delta)u_i(a_o) + \delta V_i(a_o) &\geq \\ (1-\delta) \max_{a'_i} u_i(a'_i, a_{-i,o}) + \delta V_i(a_p) - \epsilon_i, & \quad a'_i \neq a_{i,o}, \forall i \in N \end{aligned} \quad (6)$$

where $V_i(a_i)$ denotes the expected future payoff that player i will achieve by playing action a_i . The strategy a_o will be an equilibrium to player i if she cannot do better by unilaterally deviating. Grim trigger is an unforgiving strategy: once a player deviates, all players will switch forever to the punishment strategies, thus: $V_i(a_p) = u_i(a_p)$. While no player deviates, they play using a_o , which means that $V_i(a_o) = u_i(a_o)$. Then, (6) can be rewritten as:

$$u_i(a_o) \geq (1 - \delta) \max_{a'_i} u_i(a'_i, a_{-i,o}) + \delta u_i(a_p) - \epsilon_i, \quad a'_i \neq a_{i,o}, \forall i \in N \quad (7)$$

We can proceed similarly for the case of CE. Using again Bellman equation to express (1), the condition in case of CE (5) becomes:

$$\begin{aligned} \sum_{a^p_{-i} \in A^p_{-i}} \phi(a^p_{-i} | a^p_i) \left\{ (1 - \delta) \left(u_i(a^{p'}, a^p_{-i}) - u_i(a^p_i, a^p_{-i}) \right) \right. \\ \left. + \delta (V_i(a_p) - V_i(\phi)) \right\} \leq \epsilon_i, \quad \forall i, \forall a^{p'}_i \in A^p_i, a^p_i \neq a^{p'}_i \end{aligned} \quad (8)$$

where ϕ denotes the equilibrium distribution (equivalent to a_o in Nash equilibrium case). Again, we use grim trigger strategy, and we assume that the punishment strategy a_p is a static equilibrium, hence $V_i(a_p) = u_i(a_p)$. Also, $V_i(\phi) = \sum_{a^p \in A^p} \phi(a_p) u_i(a^p)$ is the payoff expected if all players follow the correlated equilibrium ϕ (where $\phi(a^p)$ denotes the probability assigned to each pure action vector a^p). Hence, (8) becomes:

$$\begin{aligned} \sum_{a^p_{-i} \in A^p_{-i}} \phi(a^p_{-i} | a^p_i) \left\{ (1 - \delta) \left(u_i(a^{p'}, a^p_{-i}) - u_i(a^p_i, a^p_{-i}) \right) \right. \\ \left. + \delta \left(u_i(a_p) - \sum_k \phi_k u_i(a^p_k) \right) \right\} \leq \epsilon_i \quad \forall i, \forall a^{p'}_i \in A^p_i, a^p_i \neq a^{p'}_i \end{aligned} \quad (9)$$

Observe than in a CE, each player equilibrium condition will be a certain number of linear equations (9). The number of equations that each player has to solve depends on the number of pure actions that she has.

The Folk Theorem asserts that grim trigger strategy will be able to reach, for sufficiently high values of δ , any achievable payoff such that $U_i(a_o) \geq U_i(a_p)$. In other words, a repeated game may have infinitely many valid equilibrium points. Selecting one of these equilibria is the problem of equilibrium selection, also known as bargaining in literature [30, 31, 32]. All of these solutions are Pareto-efficient. A vector v is said to be Pareto-efficient if there is no other vector

u such that $u_k \geq v_k$ for all the k components of the vector. In other words, a vector is Pareto-efficient if there is no other vector that provides a higher value for all of its components. In our problem, a vector of payoffs for a certain strategy a , $U(a)$, is said to be Pareto-efficient if there is no other strategy a' such that all players have higher payoffs, i.e., $\nexists a' \in A : \{U_i(a') \geq U_i(a), \forall i \in N\}$.

3. The CA algorithm

In this section, we introduce our novel algorithm. It allows obtaining a Pareto-efficient NE or CE in a repeated game of N players. We call it CA: Communicate and Agree. CA requires the following inputs for each player:

- The discount factor δ .
- The payoff function for player i , u_i . Each player does not need to know other players payoff function, which in turn means that the player does not know what kind of opponents she is facing (i.e., it is a game of incomplete information). CA is able to work in a wide variety of environments: from extreme competition to extreme cooperation games, without needing a priori knowledge of the kind of game. Each player also needs to know na , the dimension of the action vectors $a \in A$. This parameter is required to sample A .
- The punishment action for player i , $a_{i,p}$ and its punishment payoff, $u_{i,p}$. This is the payoff that CA will try to improve by using repeated game tools. We use a static equilibrium which can be obtained, for instance, using Regret-Matching (RM) algorithm [17]. This requirement appears in other MAL algorithms, such as [20].
- The number of players of the game, N .
- N_c , the maximum number of communications allowed. This parameter allows controlling the negotiation time.

We assume that all players know the equilibrium concept (Nash or correlated) and the strategy that all players will use, which in our case is grim trigger. CA could be modified to use other strategies, by changing the equilibrium conditions (7) or (9).

CA assumes that players communicate among them, as in [24]. We will use this communication for players to interchange strategies that they are willing to follow: player i proposes a strategy that leads her to an equilibrium, and if this strategy is also an equilibrium to the other players, then it constitutes an equilibrium of the repeated game.

CA proceeds in two steps. The first step is called action space sampling. All players sample the set of actions A and interchange messages in order to obtain A_s , the set of sampled actions which are valid equilibria for all players. In order to do this, each player samples actions profiles and tests them to check

which yield an equilibrium for her (i.e., using (7) or (9)). When a player finds an equilibrium strategy a , it communicates it to other players, and they test whether a is also an equilibrium for them or not. If it is an equilibrium for all players, all players do $A_s = A_s \cup a$. This is repeated a certain number of times, controlled by N_c .

The second step is called Pareto pruning: the players must choose one strategy from A_s . In order to do so, they coordinate to randomly choose a strategy $a_c \in A_s$. Then, all players discard all strategies $a \in A_s$ so that $U_i(a) < U_i(a_c)$ (i.e., they discard all strategies that are Pareto dominated by strategy a_c). Hence, A_s is pruned by eliminating all Pareto-dominated strategies. This procedure is repeated until A_s is a single strategy. By construction, this strategy is guaranteed to be Pareto-efficient, because if it were Pareto-dominated, it would have been pruned.

Algorithm 1 CA algorithm for each player i

Input: $\delta, u_i, a_{i,p}, u_{i,p}, N, N_c, na$
1: $A_s \leftarrow \text{sample} - \text{actions}(\delta, u_i, a_{i,p}, u_{i,p}, N, N_c, na)$
2: **if** $A_s = \emptyset$ **then**
3: $A_s = a_{i,p}$
4: **else**
5: **while** $|A_s| > 1$ **do**
6: $A_s \leftarrow \text{pareto} - \text{prune}(A_s, u_i)$
7: **end while**
8: **end if**
Output: A_s

A schematic description of CA can be found in Algorithm 1, where $|A_s|$ denotes the number of elements in the set A_s . Each subroutine is detailed in the following sections, as well as in Algorithms 2 and 3.

3.1. Action space sampling

Action space sampling is the first component of CA algorithm. Roughly speaking, we try to obtain the subset of actions that leads all players to an equilibrium (other approaches try to find the achievable set of payoffs, [33]). More formally, in case of NE:

$$A_s : \left\{ a \in A, u_i(a) \geq (1 - \delta) \max_{a'_i} u_i(a'_i, a_{-i}) + \delta u_{i,p}, a'_i \neq a_{i,o} \right\}$$

which are the conditions from (7). Using (9) we reach the condition for CE; in this case, the sampling applies to ϕ distributions instead of actions a :

$$A_s : \left\{ \phi \in \Phi, \text{so that (9) is satisfied} \right\}$$

In general, A_s is not easy to obtain. We propose using an independent sampling scheme to approximate A_s : each player uses a possibly different sampling

method to obtain \tilde{A}_i , a sampled version of A (the subscript i emphasizes that each player might obtain a different \tilde{A}_i set). Then, player i checks which action profiles $a \in \tilde{A}_i$ that are equilibria for her. Then, player i shares her equilibrium points with the rest of the players. If this point is a valid equilibrium for all players, then they all add a to A_s , that is, $A_s = A_s \cup a$. Note that for NE, we sample actions a , and for CE, ϕ distributions.

Observe that each player can use a different sampling schema. We propose three different sampling methods. The first is equispaced sampling on the action space A or the ϕ space. The second is random sampling: each player randomly obtains a sampled space \tilde{A}_i following a certain distribution. Those two methods are brute-force ones. The third method we propose is using an intelligent sampling method, based on optimization. Each player samples A trying to maximize her payoff. We define the following reward function for each player i :

$$f_i(a) = \lambda r_i(a) + (1 - \lambda) r_{-i}(a) \quad (10)$$

where $r_i(a)$ is a function that measures how good action a is to player i , $r_{-i}(a)$ does the same for the rest of the players and $\lambda \in [0, 1]$ is a parameter that allows modeling how much $f_i(a)$ takes into account player i reward and the rest of the players. We use:

$$r_i(a) = \begin{cases} \|u_i(a) - u_{i,p}\| & \text{if } u_i(a) \geq u_{i,p} \text{ and} \\ & a \text{ is an equilibrium} \\ -\|u_i(a) - u_{i,p}\| & \text{if otherwise} \end{cases} \quad (11)$$

$$r_{-i}(a) = \sum_{j \neq i} r_j(a)$$

where $\|x\|$ is the Euclidean norm of vector x . Our definition of $r_i(a)$ is positive only if the payoff that action a provides to player i is higher than the punishment payoff - and the highest this payoff is, the highest $r_i(a)$ will be. Also observe that $r_{-i}(a)$ provides an average on the payoff gain of the rest of the players; other metrics, such as the minimum payoff gain, could be used as well. Observe that in (11), each player computes $r_i(a)$ and then shares it to the rest of the players. If this is not desired or possible, we set $\lambda = 1$, and thus, $f_i(a) = r_i(a)$: each player does not take into account the rest of the players.

The intelligent sampling proposed is based on each player maximizing (10). Intuitively, we sample A so that we maximize (10): this sampling is intelligent because it finds actions a (or ϕ distributions) with high probabilities to be equilibria to all players. As optimization algorithm, we will use Simultaneous Optimistic Optimization (SOO) [34]. SOO is a non-convex optimization algorithm that allows maximizing a deterministic function when the function is smooth around one of its global maxima, using a limited number of evaluations.

SOO is a very adequate algorithm for this sampling method. First, because our objective function (10) is deterministic and possibly unknown: each player only knows her own payoff (and hence, the term $r_i(a)$), but she does not know the payoff of the other players. This means that, unless $\lambda = 1$, each player does

not know a term of the objective function (10). Second, because it allows finding an approximation to a maximum with a finite number of evaluations: this means that SOO will try to find a maximizer as good as possible with a fixed number of samples. And third, because SOO does not require the objective function (10) to be convex, but only to be locally smooth around a local maximum, which (10) and (11) satisfy.

We limit the maximum number of communications that each player can initiate to N_c - i.e., each player can ask up to N_c times whether an action is a valid equilibrium or not to other players. This assumes that the cost of evaluating whether a point is an equilibrium or not is negligible when compared to the cost of communicating. If that were not the case, we can limit the maximum number of points sampled. We set this limit in order to control the time that CA takes in execute.

Observe that our sampling scheme also allows to exploit the heterogeneity of the players: some players might have a higher computational capacity than the rest. The more computationally powerful players might sample using more complicated schemes than the other players, which benefits them and also may benefit the other players. Also, observe that the aim of the players is to distributedly obtain A_s . They are only allowed to ask other players whether an action vector is a valid equilibrium for them. In this way, each player needs not knowing what is the payoff function of other players and hence, CA works for incomplete information games.

Finally, it might happen that no equilibrium point is found (i.e., $A_s = \emptyset$). This might occur for two reasons: either the sampling is not fine enough or there is no possibility to obtain a better payoff than the punishment payoff. The former case could be solved by performing a denser sampling, which increases the computational cost. The latter is the case in which the static equilibrium used as punishment cannot be improved, such as in zero-sum games or in cases where δ values are not high enough to satisfy the Folk Theorem. When $A_s = \emptyset$, each player makes $A = a_{i,p}$. Hence, CA guarantees to return a strategy that provides all players with a payoff equal or higher to the punishment payoff.

The action space sampling procedure is summarized in Algorithm 2. Note that players may simultaneously question others about a point a and at the same time, be asked about other point. Due to this, we have put the tasks of asking, answering and updating A_s as separate threads.

3.2. *Pareto pruning*

The second component of CA algorithm is Pareto pruning. This mechanism selects one of the valid equilibria found in the action space sampling stage. Hence, the problem is distributedly choosing a strategy from A_s for all players. CA algorithm assumes that all players seek to optimize their own payoff functions selfishly. Hence, choosing an equilibrium $a \in A_s$ is not straightforward: each player may prefer different equilibria and no player should dominate others when choosing.

We solve this by using a jointly controlled lottery [35]. A jointly controlled lottery is a procedure that allows obtaining a random outcome distributedly. In

Algorithm 2 Action space sampling for player i

Input: $\delta, u_i, a_{i,p}, u_{i,p}, N, N_c, na$

```

1: {Questioning thread}
2: Initialize  $A_s = \emptyset$ 
3: for  $N_c$  iterations do
4:    $a = \text{obtain} - \text{sample}(na)$  {Use the sampling scheme desired}
5:   if  $a$  is an equilibrium ((7) or (9)) and  $u_i(a) \geq u_{i,p}$  then
6:     Ask other players if  $a$  is a valid equilibrium
7:     if All player answer 'YES' then
8:        $A_s = A_s \cup a$ 
9:       Tell all players that  $a$  is a valid equilibrium
10:    end if
11:  end if
12: end for
13: {Answering thread}
14: for all  $a$  that player is asked about do
15:   if  $a$  is an equilibrium ((7) or (9)) and  $u_i(a) \geq u_{i,p}$  then
16:     Answer 'YES'
17:   else
18:     Answer 'NO'
19:   end if
20: end for
21: {Updating thread}
22: for all  $a$  that other players tell as valid equilibria do
23:    $A_s = A_s \cup a$ 
24: end for

```

Output: A_s

[12], the following join controlled lottery is proposed for two players: each player simultaneously chooses a random number that follows a uniform distribution in $[0, 1]$, that is, $w_i \sim \text{unif}(0, 1)$. Then, we obtain w as:

$$w = \begin{cases} w_1 + w_2 & \text{if } w_1 + w_2 < 1 \\ w_1 + w_2 - 1 & \text{if } w_1 + w_2 \geq 1 \end{cases} \quad (12)$$

and w will follow a uniform distribution in $[0, 1]$. If one player chooses w_i deterministically, still w will follow a uniform random distribution. Thus, player i ensures a random w by simply using a random w_i .

Now, let us assume that A_s has a certain indexing equal for all players. If we have L actions in A_s , where each action has an index $l \in \{1, 2, \dots, L\}$, players select the action with index l :

$$l = \lfloor 1 + (L - 1)w \rfloor \quad (13)$$

where $\lfloor x \rfloor$ denotes the integer part of x . The action index l follows a uniform distribution in the interval $[1, L]$ and hence, a random action a_l will be selected.

Algorithm 3 Pareto pruning

Input: A_s, u_i

```

1: while  $|A_s| > 1$  do
2:   Run jointly-controlled lottery (e.g., use (12))
3:   Obtain random action  $a_l \in A_s$  (e.g., use (13))
4:   for all  $a \in A_s, a \neq a_l$  do
5:     if  $u_i(a) < u_i(a_l)$  then
6:        $A_s = A_s \setminus a$ 
7:       Inform other players that  $a$  is not a valid equilibrium
8:     end if
9:   end for
10:  {Listening thread}
11:  for all Actions  $a$  that other players communicate as non valid equilibrium
12:  do
13:     $A_s = A_s \setminus a$ 
14:  end for
15: end while

```

Output: A_s

It is important to remark that a_l has been randomly selected among all valid actions in A_s to avoid a player dominating this choice. This procedure can be extended to more than two players.

After choosing a_l , each player prunes the actions $a \in A_s$ such that $u_i(a) < u_i(a_l)$. That is, each player eliminates the actions that are Pareto-dominated by a_l . When a player erases an action a , it communicates to other players, so that all players can update A_s by erasing a as well. After each pruning procedure, it may happen that $|A_s| = 1$ (i.e., $A_s = \{a_l\}$), which means that a_l is an action that Pareto-dominates the rest of actions $a \in A_s$ and hence, it is Pareto-efficient and returned as the grim trigger strategy. Otherwise, $|A_s| > 1$ means that there is another action that Pareto-dominates a_l . In that case, the process starts again: a new jointly controlled lottery is performed, a new action a_l is chosen and the set A_s is pruned again, until $|A_s| = 1$.

A description of the Pareto-pruning procedure is in Algorithm 3. Note that in each pruning, players may simultaneously inform and be informed: due to this, we have separated the questioning and the listening tasks as a separate threads. Also, all players must wait each other to have fully pruned A_s before checking whether $|A_s| > 1$ and prune again or not.

Observe that CA scalability depends on two aspects: the ability of each player to sample the actions space and check if a point is a valid equilibrium (computational cost, which increases with the number of players and actions) and also, on the efficiency of the communications among players, which depend on the network topology and protocols used. If we assume that the former cost is negligible compared to the communications cost, we can model the scalability of CA by observing that it is an example of the atomic-commitment problem

[36]. The atomic-commitment problem appears in a distributed system, in which different subsystems have to apply an operation if and only if all subsystems apply it successfully; otherwise, the operation is reversed. In our case, the operation is checking whether a joint-action vector a is an equilibrium for player i and adding it to A_s if and only if a is an equilibrium for all players. In [36], it is shown that in absence of communication failures, there is an efficient (polynomial time) algorithm that minimizes this cost. Thus, the total cost of CA depends on N_c for the action space sampling and it is polynomial for the Pareto pruning, assuming that the computational cost is negligible when compared to the communication cost among players.

4. Error bounds in CA algorithm

4.1. General theoretical bounds

In this section, we study the error introduced by CA algorithm. We start with the NE error, whose equilibrium condition is (7). Observe that the error comes from the sampling method: if we were able to sample all points in A (that is, $A = \tilde{A}$), there would be no error. Let us assume that each player samples with a method that guarantees that the maximal distance between two actions for player i in \tilde{A}_i is Δa_i . Thus, for two sampled actions $\tilde{a}, \tilde{a}' \in \tilde{A}_i$, with $\tilde{a}_k = \tilde{a}'_k$ if and only if $k \neq i$ - i.e., the two actions vectors differ only in the action of player i :

$$\max \|\tilde{a}_1 - \tilde{a}_2\| \leq \Delta a_i \quad (14)$$

Now, observe (7). CA algorithm checks this equilibrium condition after sampling, which means that the equilibrium condition that each player computes, for two actions $\tilde{a}, \tilde{a}' \in \tilde{A}_i$, with $\tilde{a}_k = \tilde{a}'_k$ if and only if $k \neq i$ is:

$$(1 - \delta)u_i(\tilde{a}') + \delta u_i(a_p) - u_i(\tilde{a}) \leq 0, \quad \forall \tilde{a}' \quad (15)$$

We can simplify by observing that the equilibrium condition for player i assumes that the actions of the other players are fixed, hence, it is a condition that only affects the actions of player i , a_i :

$$(1 - \delta)u_i(\tilde{a}'_i) + \delta u_i(a_{p,i}) - u_i(\tilde{a}_i) \leq 0, \quad \forall \tilde{a}'_i \quad (16)$$

However, due to sampling, there might be an action a_i , which has not been sampled, such that $u_i(a_i) > \max_{\tilde{a}'_i} u_i(\tilde{a}'_i)$. We define $\Delta u_i = u_i(a_i) - u_i(\tilde{a}'_i)$, as the difference in payoffs. This would mean that the equilibrium that CA algorithm computes would not be anymore an NE, but a Nash ϵ_i -equilibrium, when $\Delta u_i > 0$. Hence, (16) would become:

$$(1 - \delta)[u_i(\tilde{a}'_i) + \Delta u_i] + \delta u_i(a_{p,i}) - u_i(\tilde{a}_i) \leq \epsilon_i, \quad \forall \tilde{a}'_i \quad (17)$$

Using (16) and (17), we obtain the following lower bound for ϵ_i :

$$\epsilon_i = (1 - \delta)\Delta u_i \quad (18)$$

In order to bound Δu_i , we will assume that u_i functions are Lipschitz-continuous in the action set A , that is:

$$\|u_i(a_1) - u_i(a_2)\| \leq C_i \|a_1 - a_2\|, \forall a_1, a_2 \in A \quad (19)$$

where C_i is the Lipschitz constant for the function $u_i(x)$. The lowest C_i that satisfies (19) is called the best Lipschitz constant, and will be denoted by C_i^* .

Let us assume that in (19), $a_1 = \tilde{a}_i$ is a sampled action and $a_2 = a_i$ is an action which was not sampled. In the worst case, according to (14), $\|\tilde{a}_i - a_i\| = \Delta a_i$ and hence, using (19), $\|u_i(\tilde{a}_i) - u_i(a_i)\| \leq C_i^* \Delta a_i$. Since the function u_i is a function returning real numbers, $\|u_i(\tilde{a}_i) - u_i(a_i)\| = |u_i(\tilde{a}_i) - u_i(a_i)| = \Delta u_i$ if $u_i(a) > u_i(\tilde{a}')$. Hence, $\Delta u_i \leq C_i^* \Delta a_i$. Thus, we can bound the error as:

$$\epsilon_i = (1 - \delta) C_i^* \Delta a_i \quad (20)$$

Observe that Δu_i measures the error induced by the possible actions a_i which are not sampled and which cause that the sampled action \tilde{a}_i is not an equilibrium, but an ϵ_i -equilibrium. Hence, the case when $u_i(a) < u_i(\tilde{a}')$ is not of interest to us, because the action not sampled returns a lower payoff than the sampled and hence, (16) holds for $\epsilon_i = 0$.

The result in (20) means that there are three factors that contribute to the error in the equilibrium obtained. The first one is the discount factor: as δ tends to 1, the error decreases. It also depends on C_i , which is an upper bound on the variation of the function as can be observed in (19). Since C_i is an upper bound, the tightest ϵ_i will be achieved with the lowest C_i possible, which is C_i^* . Finally, the last component is the maximal distance between a sampled and a not sampled action. As the number of actions sampled tends to infinity, this term will tend to 0 and hence, $\epsilon_i \rightarrow 0$, which means that CA algorithm finds a NE asymptotically.

In the case of CE, CA does not introduce any error. Note that for any value of $\tilde{\phi}$, if (9) holds, it will be a CE, with no error. In CE, we sample distributions $\tilde{\phi}$ instead of actions, thus, when a sampled $\tilde{\phi}$ distribution satisfies the equilibrium condition (9), it will be an exact CE.

Finally, we note that we do not provide any guarantees on whether CA will be able to find a repeated game equilibrium. This depends on the discount factor, the kind of game (zero-sum or general sum) and the sampling density. But we do assure that when CA finds a Nash SPE, it will be an ϵ_i -equilibrium for each player bounded by (20) and when CA finds a correlated SPE, it will contain no error.

4.2. Theoretical bounds on a 2 player, 2 action game, equispaced sampling

In this section, we particularize (20) for the case in which there are $N = 2$ players and each player has 2 pure actions. We denote the pure actions payoffs that player i receives as $u_i^p(j, k)$, where i denotes the player, j denotes the pure action of player 1 and k denotes the pure action of player 2. Note that $i, j, k = \{1, 2\}$. We denote the mixed action of player 1 as $(y, 1 - y)$, where

y is the probability that player 1 assigns to her pure action 1 and $1 - y$ the probability assigned to her pure action 2. The mixed action of player 2 is defined equivalently by $(z, 1 - z)$. Thus, the mixed action space of the game A is the unit square $A = A_1 \times A_2 = A = [0, 1] \times [0, 1]$, where one axis are the y values and the other, the z values.

We sample A using equispaced sampling, with K_i samples in each dimension. Hence, for each player, the sampled mixed actions are $\{0, \frac{1}{K_i-1}, \frac{2}{K_i-1}, \dots, 1\}$. The maximum distance between a sampled action and a not sampled one will take place when the not sampled action lies in the middle of two sampled actions, hence, $\Delta a_i = \frac{1}{2(K_i-1)}$.

The payoff function u_i has the following form:

$$\begin{aligned} u_i(y, z) &= yzu_i^p(1, 1) + y(1 - z)u_i^p(1, 2) + (1 - y)zu_i^p(2, 1) + (1 - y)(1 - z)u_i^p(2, 2) \\ &= A_i yz + B_i y + C_i z + D_i \end{aligned} \quad (21)$$

where

$$\begin{aligned} A_i &= u_i^p(1, 1) - u_i^p(1, 2) - u_i^p(2, 1) + u_i^p(2, 2) \\ B_i &= u_i^p(1, 2) - u_i^p(2, 2) \\ C_i &= u_i^p(2, 1) - u_i^p(2, 2) \\ D_i &= u_i^p(2, 2) \end{aligned} \quad (22)$$

Since the payoff functions $u_i(y, z)$ are polynomials, they are continuous, derivable and with bounded derivatives in A , which is a convex subset of \mathbb{R}^2 . A continuous function $f(x)$ with bounded derivatives is Lipschitz-continuous with $C^* = \sup_x \|\nabla f(x)\|$ (see [37, Lemma 2.18]).

Now, here there are two possible approaches. Remark that player i is interested in computing ϵ_i , a bound on the error she is committing when she evaluates an NE. Since she knows the actions of other players, player i can fix the actions of other players in u_i , so that u_i becomes a one variable function that only depends on a_i . In that case:

$$C_i^* = \max_{a_i} \left\{ \left. \frac{du_i(a_i, a_{-i})}{da_i} \right|_{a_{-i}} \right\}$$

Yet this implies that player i bound, ϵ_i , depends on a_{-i} , the actions of the other players. A second option is the worst case one by using $C_i^* = \sup_{(a_i, a_{-i})} \|\nabla u_i(a_i, a_{-i})\|$. This option yields a higher Lipschitz constant and hence, a less tight ϵ_i value, but it provides an upper bound for ϵ_i independent of the actions of other players. Using (21), we obtain $\nabla u_i = (A_i z + B_i, A_i y + C_i)$ and $\|\nabla u_i\| = \sqrt{(A_i z + B_i)^2 + (A_i y + C_i)^2}$. Hence, we obtain the following ϵ_i bound using (20):

$$\epsilon_i = \frac{1 - \delta}{2(K_i - 1)} \left\{ \max_{0 \leq y \leq 1, 0 \leq z \leq 1} \sqrt{(A_i z + B_i)^2 + (A_i y + C_i)^2} \right\} \quad (23)$$

$$\begin{array}{ll}
 \begin{pmatrix} (1, -1) & (-1, 1) \\ (-1, 1) & (1, -1) \end{pmatrix} & \begin{pmatrix} (2, 2) & (-1, 3) \\ (3, -1) & (0, 0) \end{pmatrix} \\
 \text{(a) Matching pennies (MP).} & \text{(b) Prisoner's dilemma (PD).} \\
 \begin{pmatrix} (2, 1) & (0, 0) \\ (0, 0) & (1, 2) \end{pmatrix} & \begin{pmatrix} (-10, -10) & (1, -1) \\ (-1, 1) & (0, 0) \end{pmatrix} \\
 \text{(c) Battle of sexes (BS).} & \text{(a) Chicken game (CG).}
 \end{array}$$

Figure 1: Payoff matrices for the four games proposed. Player 1 is row player, and player 2 is column player, hence, the first row stands for pure action 1 of player 1, and row 2 for her pure action 2. The first column contains the pure action 1 of player 2, and the second column, her pure action 2. In each matrix, the payoff entries for each pair of pure actions are (u_1, u_2) .

5. Testbench

5.1. Games description

In order to test the performance of CA, we have tested it on four repeated games with $N = 2$ players with 2 pure actions each. We choose games with very different characteristics, whose payoff matrices are in Figure 1. Remark that a static NE for these games will have the form $a_1 = (y, 1 - y)$, $a_2 = (z, 1 - z)$ and each static NE is also a CE of the form $\phi = (yz, y(1 - z), (1 - y)z, (1 - y)(1 - z))$.

The first game is matching pennies (MP), a zero-sum game. This means that $u_1 = -u_2$ and hence, the gains of one player are the losses of the other. This game has only one static NE: $a_1 = a_2 = (1/2, 1/2)$ (the equivalent CE is $\phi = (1/4, 1/4, 1/4, 1/4)$), which yields each player a payoff of $U_1 = U_2 = 0$. No gain in payoffs can be achieved by repeating the game with respect to the static equilibrium.

The second game is the prisoner's dilemma (PD), which is a non-zero sum game, used frequently to illustrate the Folk Theorem [12]. There is only one static NE, which is $a_1 = a_2 = (0, 1)$, which provides each player with a payoff of $U_1 = U_2 = 0$ (the equivalent CE is $\phi = (0, 0, 0, 1)$). However, when the game is repeated for a sufficiently high value of δ , new equilibria arise, and in this case, it is possible to achieve a payoff as high as $U_1 = U_2 = 2$ using grim trigger strategy (for a theoretical analysis, see [12, Ch. 2, 3]).

The third game is the Battle of sexes (BS), which is a non-zero sum game with three different static NE, namely, $a_1 = (2/3, 1/3)$, $a_2 = (1/3, 2/3)$, which yields the players a payoff of $U_1 = U_2 = 2/3$, $a_1 = (1, 0)$, $a_2 = (1, 0)$, which yields payoffs $(U_1, U_2) = (2, 1)$, and $a_1 = (0, 1)$, $a_2 = (0, 1)$, which yields them payoffs $(U_1, U_2) = (1, 2)$. The two pure action equilibria are Pareto-efficient.

The fourth game is the chicken game (CG), which is a non-zero sum game with three different static NE, namely, $a_1 = a_2 = (1/10, 9/10)$, which yields the players a payoff of $U_1 = U_2 = -1/10$, $a_1 = (1, 0)$, $a_2 = (0, 1)$, which yields payoffs $(U_1, U_2) = (1, -1)$, and $a_1 = (0, 1)$, $a_2 = (1, 0)$, which yields them payoffs $(U_1, U_2) = (-1, 1)$.

5.2. Simulation 1

Firstly, we obtain an empirical taste of the differences among the sampling methods proposed. We use the PD game with $\delta = 0.9$. The analytical solutions to PD game can be found in [12]. For the δ value we are using, the Pareto-efficient region is:

$$U_p = \begin{cases} U_2 = \frac{-U_1+8}{3} & \text{if } U_1 \in [0, 2] \\ U_2 = -3U_1 + 8 & \text{if } U_1 \in [2, 8/3] \end{cases} \quad (24)$$

We define ξ as the distance between a payoff and the Pareto frontier:

$$\xi = \min \|U(a) - U_p\| \quad (25)$$

where $U(a)$ is the repeated game payoff vector to all players by playing action vector a and U_p is the Pareto region (24). We use ξ to compare the performance of the three sampling methods we proposed: equispaced, random and SOO. In order to show empirically the advantages of using SOO, we will limit in equispaced and random sampling the number of communications to N_c , and in SOO, we only sample N_c times (see section 3.1). We test for $N_c \in [5, 200]$.

First, we obtain a static equilibrium using regret matching (RM) algorithm [17], which provides a static equilibrium for all players. We use 10^3 iterations for RM. The equilibrium that RM returns is used as punishment for CA algorithm.

Then, we run CA using all the sampling scheme proposed. In case of NE, the mixed action space is a square $A = [0, 1] \times [0, 1]$. For equispaced sampling, we used $K_i = 50$ samples in each dimension. In the case of CE, we equispacedly sample a simplex of dimension 3: note that $\sum_{k=1}^4 \phi_k = 1$. We used approximately the same number of points that for the NE case, 2500 - i.e., we test approximately 2500 ϕ distributions.

Then, we tested using random sampling, following a uniform distribution. In the case of NE, each player randomly generates a pair of actions following a uniform distribution between 0 and 1, that is, $(y, z) \sim (\text{unif}(0, 1), \text{unif}(0, 1))$. We limit the maximum number of actions tested to 10^4 for each player: if no equilibrium is found, the sampling procedure is exited. For the CE concept, each player samples uniformly in a simplex and again, we limit to 10^4 the maximum number of samples if no equilibrium is found.

Finally, we test SOO sampling method. We use (10) and (11), with $N_c = 10$ and $\lambda = \{0.5, 1\}$, for both NE and CE.

In order to test in NE whether there are profitable deviations or not (see (7)), we sample $a'_i \in [0, 1]$ equispacedly using 50 samples. This is used to check the NE deviation condition every time that is needed.

For each value of N_c (N_s in SOO case), we run 100 times CA, and the resulting payoff errors were computed using (25) and (24). The results can be observed in Figure 2, where we observe that CA approaches the Pareto frontier as the number of communications increases. We also observe that SOO sampling method outperforms the others, even though it has a stricter limitation (number of samples instead of number of communications). Thus, SOO intelligent sampling presents a clear advantage over the other methods proposed.

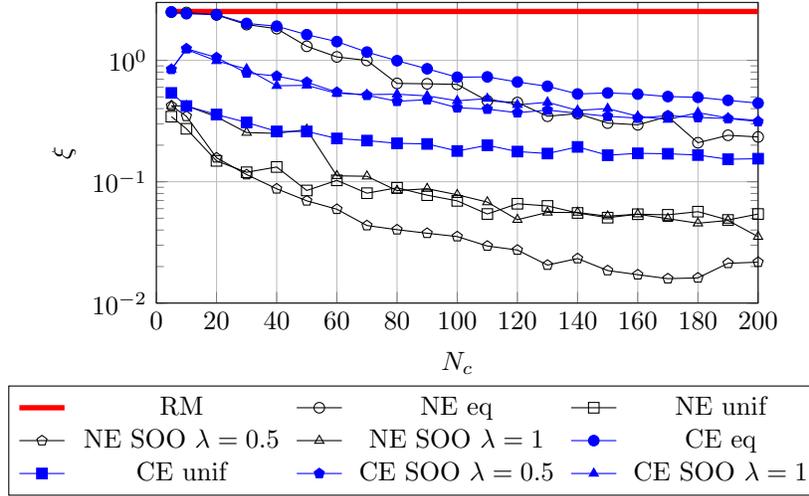


Figure 2: Average values of ξ for simulation 1. Equispaced sampling is 'eq', random uniform sampling is 'unif' and 'RM' stands for regret-matching results. We observe that, as we increase the number of communications allowed N_c , the error ξ decreases. Recall that ξ measures how far the CA results are from the theoretical Pareto frontier (see (25)). Thus, lower is better, as it implies that the players achieve a payoff closer to the Pareto frontier. Note that a greater N_c allows getting closer to the Pareto frontier. The sampling methods order, from the worst to the best performance, are equispaced, random uniform and SOO. Even though in SOO we use a stricter limitation (samples taken instead of communications), it outperforms the other sampling methods.

5.3. Simulation 2

We also simulated the performance of CA in the four games described above. We use three possible values for the discount factor: $\delta = \{0.1, 0.5, 0.9\}$. For each δ value and each game, we run 100 different repetitions: in each repetition we use the same 9 algorithms that we used in Simulation 1: regret matching and 8 different instances of CA, 4 for NE and other 4 for CE. We use the same parameters as in Simulation 1, except we fix $N_c = 100$ for equispaced and random sampling, and $N_s = N_c$ for SOO sampling.

We run the simulations and obtain the region of payoffs for each algorithm and the payoff that, in average, is obtained in each setup. The results can be observed in Figure 3, where we plot the payoff increase in the four games described that CA yields for the static equilibrium that RM provides. For MP, BS and CG, it is possible to observe that there is no significant increment in payoffs between CA and RM, as we expected. MP is a zero-sum game and hence, the Folk Theorem does not apply. In BS and CG, RM returns a static, Pareto-efficient payoff: since the payoff is already efficient, CA does not find a better one.

The case of PD is the most important and interesting one, because it is a game in which both players can benefit of repeating the game. It is possible to observe that CA does not improve the theoretical payoff for $\delta = 0.1$, because

Game	$\delta = 0.1$	$\delta = 0.5$	$\delta = 0.9$
MP	0.0509	0.0283	0.0057
PD	0.0569	0.0316	0.0063
BS	0.0509	0.0283	0.0057
CG	0.2558	0.1421	0.0284

Table 1: Comparison of theoretical ϵ_i values for the Nash equilibrium concept, when using equispaced sampling, according to (20), where $K_i = 50$. In all cases, $\epsilon_1 = \epsilon_2$, that is, both players had the same bound.

with that discount factor, the only equilibrium of the repeated game is the static one. Yet as δ increases, new equilibria arise and the gains of using CA appear.

As an example, in Figure 4 we include some of the payoff regions returned by CA algorithm. We observe that in case of PD game, as δ increases, a whole region of new payoffs appears: these payoffs can be achieved by repeating the game following the grim trigger strategy proposed. Although this needs not be the case in all games: in the case of BS game, the static equilibrium used as punishment is already Pareto-efficient and hence, CA cannot improve it. We also show how SOO provides similar results in terms of Pareto-efficient payoffs, with significantly fewer samples and equilibria evaluations.

In Table 1, it is possible to observe the different ϵ_i values obtained, using (20), for equispaced sampling and NE. It is possible to observe that the ϵ_i values are low, except in the game of the chicken, due to the higher values of the derivatives in this payoff matrix.

Recall that in all simulations, we used as CA input the RM equilibrium, in order to be used as punishment equilibrium and explore the possibility of obtaining a better payoff in the repeated game by using grim trigger strategy. In our game testbench, that was the case only of PD, and in that case, CA algorithm outperforms clearly RM, because it exploits the new equilibria that appear when repeating the game.

Yet this increment came at no cost. Observe that for games where the best possible equilibrium is the static one, CA algorithm does not improve RM. Hence, CA does not make sense if we know for sure that we can obtain no gain by repeating the game. Yet if we know or hope that new equilibria may arise by repeating the game, CA may find equilibria which yield better payoff for all players, such as in PD. Recall that CA needs no a priori information on the kind of game being played and hence, it works on imperfect information games.

Also, in the case of PD, observe that there are differences between the different implementations of CA. In general, NE yields higher payoff than CE, because the region of Nash equilibria is smaller - a square in the plane - than the correlated equilibria region - a simplex with 3 dimensions. Some aspects that may help in practice are:

- First, think whether CE makes sense in our game setup. This means that either we have access to a correlating device or to a jointly-controlled

lottery [35]. CE has two advantages over NE: first, the region of CE always contains NE, so there might be games in which there are CE that yield a better payoff than NE. And secondly, CA algorithm guarantees that any CE found will be exact, whereas NE have a bounded error, according to (20) (see Table 1). Yet since CE region has a higher dimensionality than NE region, the sampling schemes perform poorer.

- The computational capacity for sampling purposes. Using SOO and limiting the number of samples allows performing fewer sampling operations, (see Figure 4). Yet it implies implementing SOO algorithm [34].
- A final aspect is related to the ability of detecting a deviation. Grim trigger strategy needs to detect deviations immediately. In the case of NE, this means having access to the mixed actions of each player. In the case of CE, since a deviation is not following the recommendation ϕ , if we have an entity that sends the ϕ recommendation to each player, this device can detect whether a player deviates or not instantaneously, i.e., if player i does not play the pure action recommended. Thus, CE eases detecting a deviation.

6. Conclusions

We introduce CA, a novel MAL negotiation-based algorithm that allows computing equilibria of repeated games of perfect monitoring using the averaged discounted payoff criterion under a grim trigger strategy. CA is based in the idea that players can communicate each other the strategies they are willing to use. This allows, if possible, to reach a repeated game equilibrium.

CA is a powerful and flexible algorithm, with plenty of positive features: is completely distributed, is valid for N players, it is valid for imperfect information games, improves when possible an input static equilibrium, chooses Pareto-efficient payoffs, works both using NE or CE and may be adapted to different strategies by modifying the equilibrium conditions (7) or (9), can use intelligent sampling methods and finally, CA takes advantage of heterogeneous computational capacity of each player in the sampling stage.

CA requires that each player knows the discount factor, her payoff function, a punishment equilibrium - which CA tries to improve - and a sampling method for the action space. In case of NE, it returns an ϵ_i -equilibrium for all players, where ϵ_i is bounded. In case of CE, it returns an exact equilibrium. Hence, CA is a powerful and flexible algorithm that takes advantage of the new equilibria that may arise by repeating the game, according to the Folk Theorem, in order to improve the payoff that all players can obtain in a repeated game.

7. Acknowledgements

This work was supported by a Ph.D. grant given to the first author by Universidad Politécnica de Madrid, as well as by the Spanish Ministry of Science and Innovation under the grant TEC2016-76038-C3-1-R (HERAKLES).

8. References

References

- [1] L. S. Shapley, Stochastic games, *Proceedings of the national academy of sciences* 39 (10) (1953) 1095–1100.
- [2] M. L. Littman, Markov games as a framework for multi-agent reinforcement learning, in: *Machine Learning Proceedings 1994*, Elsevier, 1994, pp. 157–163.
- [3] M. Bowling, M. Veloso, Multiagent learning using a variable learning rate, *Artificial Intelligence* 136 (2) (2002) 215–250.
- [4] A. Greenwald, K. Hall, R. Serrano, Correlated q-learning, in: *ICML*, Vol. 3, 2003, pp. 242–249.
- [5] N. Akchurina, Multi-agent reinforcement learning algorithms., Ph.D. thesis, University of Paderborn (2010).
- [6] J. W. Crandall, Just add pepper: extending learning algorithms for repeated matrix games to repeated markov games, in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems—Volume 1*, International Foundation for Autonomous Agents and Multiagent Systems, 2012, pp. 399–406.
- [7] M. Elidrisi, N. Johnson, M. Gini, J. Crandall, Fast adaptive learning in repeated stochastic games by game abstraction, in: *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, International Foundation for Autonomous Agents and Multiagent Systems, 2014, pp. 1141–1148.
- [8] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al., Mastering the game of go without human knowledge, *Nature* 550 (7676) (2017) 354.
- [9] N. Brown, T. Sandholm, Superhuman ai for heads-up no-limit poker: Libratus beats top professionals, *Science* (2017) eaao1733.
- [10] M. Lanctot, V. Zambaldi, A. Gruslys, A. Lazaridou, J. Perolat, D. Silver, T. Graepel, et al., A unified game-theoretic approach to multiagent reinforcement learning, in: *Advances in Neural Information Processing Systems*, 2017, pp. 4193–4206.
- [11] D. Fudenberg, J. Tirole, *Game theory*, MIT press Cambridge, MA, 1991.
- [12] G. J. Mailath, L. Samuelson, *Repeated games and reputations: long-run relationships*, Oxford university press, 2006.
- [13] G. Szabó, G. Fath, Evolutionary games on graphs, *Physics reports* 446 (4-6) (2007) 97–216.

- [14] M. Perc, J. J. Jordan, D. G. Rand, Z. Wang, S. Boccaletti, A. Szolnoki, Statistical physics of human cooperation, *Physics Reports* 687 (2017) 1–51.
- [15] M. Perc, A. Szolnoki, Coevolutionary games—a mini review, *BioSystems* 99 (2) (2010) 109–125.
- [16] P. Hernandez-Leal, M. Kaisers, T. Baarslag, E. M. de Cote, A survey of learning in multiagent environments: Dealing with non-stationarity, arXiv preprint arXiv:1707.09183 (2017).
- [17] S. Hart, A. Mas-Colell, A simple adaptive procedure leading to correlated equilibrium, *Econometrica* 68 (5) (2000) 1127–1150.
- [18] S. Hart, A. Mas-Colell, Simple adaptive strategies: from regret-matching to uncoupled dynamics, Vol. 4, World Scientific, 2013.
- [19] B. Banerjee, J. Peng, Performance bounded reinforcement learning in strategic interactions, in: *AAAI*, Vol. 4, 2004, pp. 2–7.
- [20] V. Conitzer, T. Sandholm, Awesome: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents, *Machine Learning* 67 (1-2) (2007) 23–43.
- [21] M. L. Littman, P. Stone, A polynomial-time nash equilibrium algorithm for repeated games, *Decision Support Systems* 39 (1) (2005) 55–66.
- [22] E. M. De Cote, M. L. Littman, A polynomial-time nash equilibrium algorithm for repeated stochastic games, arXiv preprint arXiv:1206.3277 (2012).
- [23] J. W. Crandall, M. A. Goodrich, Learning to compete, coordinate, and cooperate in repeated games using reinforcement learning, *Machine Learning* 82 (3) (2011) 281–314.
- [24] Y. Hu, Y. Gao, B. An, Multiagent reinforcement learning with unshared value functions, *IEEE transactions on cybernetics* 45 (4) (2015) 647–662.
- [25] D. Abreu, On the theory of infinitely repeated games with discounting, *Econometrica: Journal of the Econometric Society* (1988) 383–396.
- [26] J. Nash, Equilibrium points in n-person games’, proceedings of the national academy of sciences of the united states, 36, 48-9, *INTERNATIONAL LIBRARY OF CRITICAL WRITINGS IN ECONOMICS* 67 (1996) 52–53.
- [27] T. Basar, G. J. Olsder, Dynamic noncooperative game theory, Vol. 23, Siam, 1999.
- [28] R. J. Aumann, Subjectivity and correlation in randomized strategies, *Journal of mathematical Economics* 1 (1) (1974) 67–96.

- [29] N. Nisan, T. Roughgarden, E. Tardos, V. V. Vazirani, Algorithmic game theory, Vol. 1, Cambridge University Press Cambridge, 2007.
- [30] J. F. Nash Jr, The bargaining problem, *Econometrica: Journal of the Econometric Society* (1950) 155–162.
- [31] E. Kalai, M. Smorodinsky, Other solutions to nash’s bargaining problem, *Econometrica: Journal of the Econometric Society* (1975) 513–518.
- [32] E. Kalai, Proportional solutions to bargaining situations: interpersonal utility comparisons, *Econometrica: Journal of the Econometric Society* (1977) 1623–1630.
- [33] M. Dermed, L. Charles, Value methods for efficiently solving stochastic games of complete and incomplete information, Ph.D. thesis, Georgia Institute of Technology (2013).
- [34] R. Munos, Optimistic optimization of a deterministic function without the knowledge of its smoothness., in: *NIPS*, 2011, pp. 783–791.
- [35] R. J. Aumann, M. Maschler, R. E. Stearns, Repeated games with incomplete information, MIT press, 1995.
- [36] O. Wolfson, A. Segall, The communication complexity of atomic commitment and of gossiping, *SIAM Journal on Computing* 20 (3) (1991) 423–450.
- [37] J. K. Hunter, B. Nachtergaele, Applied analysis, World Scientific Publishing Co Inc, 2001.

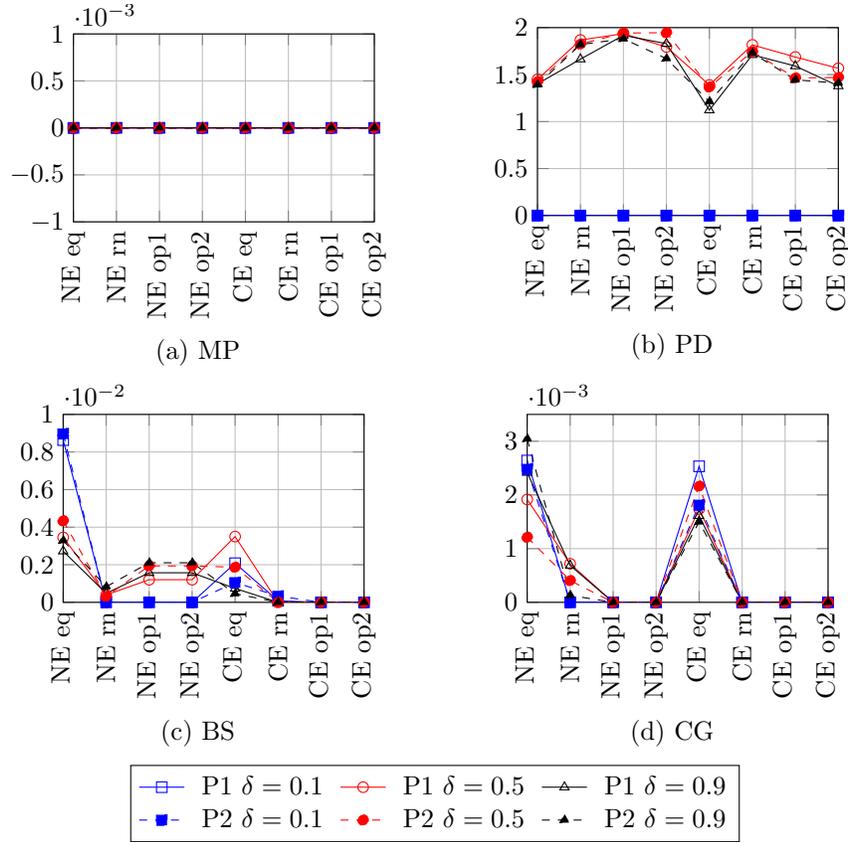


Figure 3: Payoff results: for each game, we represent the average payoff increment ΔU_i between CA and RM for different values of δ . Thus, higher is better, as it means that CA provides better payoffs than RM. We use four sampling methods for CA: equispaced (eq), random uniform (rn), SOO with $\lambda = 0.5$ (op1) and SOO with $\lambda = 1$ (op2), for NE and CE. When CA takes advantage of the Folk Theorem, it outperforms RM, as happens in PD. And when using the Folk Theorem provides no advantage in payoffs, as in MP, BS and CG, CA is not worse than RM, as expected.

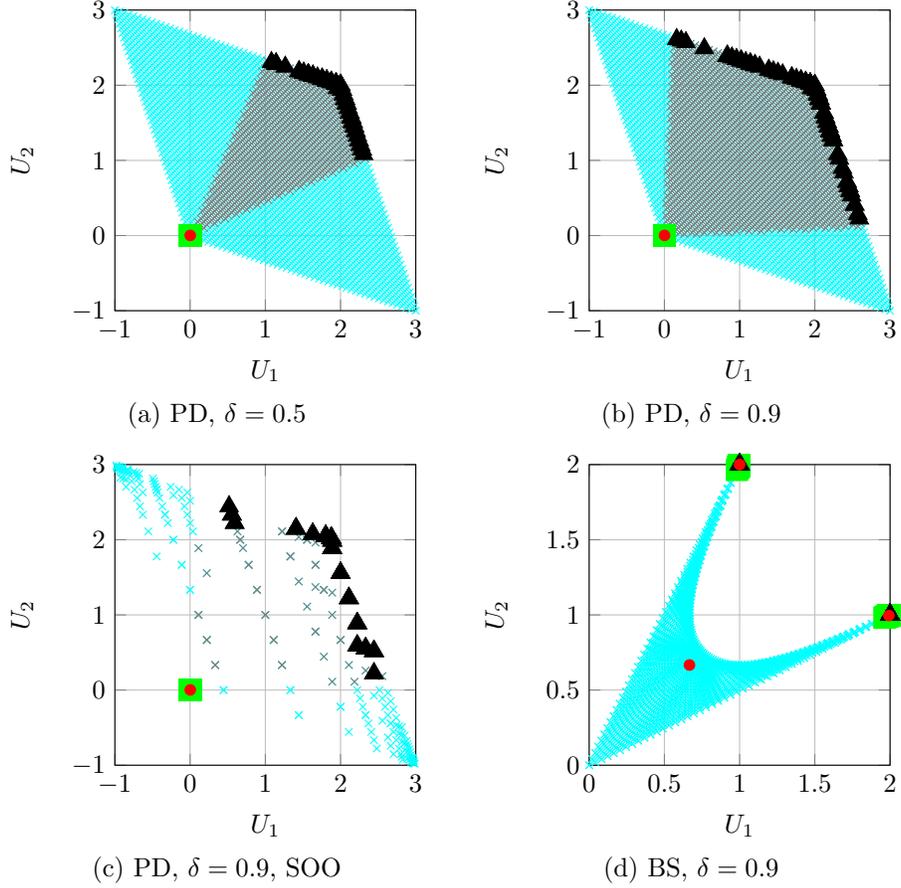


Figure 4: Comparison of NE payoff regions in PD and BS games. In light blue, we observe the possible payoff region, the gray darker region is the set of payoff equilibria in the repeated game. The red circles are the theoretical static payoff equilibria, the green squares are the payoff equilibria returned by RM and the black triangles are the payoff equilibria returned by CA. Note that RM always provides a static equilibrium payoff. Sampling in regions (a), (b) and (d) is equispaced with 2500 samples, whereas region (c) was sampled using SOO with $\lambda = 1$. We note that (1) increasing δ might provide a larger payoff equilibria region, as the Folk Theorem says: compare (a) and (b); (2) if a static equilibrium is already Pareto-efficient, CA cannot improve it, as shown in (d); (3) SOO provides similar equilibria to equispaced sampling taking much fewer samples: compare (b) and (c). Thus, CA with SOO sampling produces the best results both in terms of payoffs and samples taken.