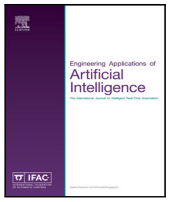
Engineering Applications of Artificial Intelligence 107 (2022) 104520

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai



An online learning algorithm to play discounted repeated games in wireless networks

Juan Parras *, Patricia A. Apellániz, Santiago Zazo

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

ARTICLE INFO

Keywords:
Repeated games
Online learning
Wireless networks
Folk theorem

ABSTRACT

Discounted repeated games are currently being used to model the conflicts that arise between the nodes in a wireless network, such as distributed resource allocation, interference management or defending the network against attacks. In current literature, it is frequent that authors devise a specific strategy that performs well only for their concrete problem, thus, it would be desirable to have a generic algorithm that allows learning strategies for such games. However, current learning algorithms focus on average payoff repeated games, and we show analytically that there are important differences that prevent us from using such algorithms for discounted repeated games. In this work, we aim to fill this gap and we propose LEWIS, a lightweight, online learning algorithm specifically designed for these games, that deals with imperfect and incomplete information and is able to return a good payoff. We test LEWIS on two settings based on current literature problems to show that it has a good performance, hence, being a promising method to learn how to play a discounted repeated game in wireless networks.

1. Introduction

Multiagent learning is a field in which several agents or players try to learn how to act optimally according to a certain reward function defined for each player that is coupled to what the rest of the players do. This model has a lot of interest in the field of networking, as each network node makes decisions that affect the rest of the nodes, such as whether to drop a packet or the communication channel used to transmit. A frequent mathematical foundation for this problem is game theory (Shoham et al., 2007), the branch of mathematics oriented to study the conflicts between different players that interact with possibly different targets each. Although multiagent learning can be studied using alternative tools (Stone, 2007), it is frequent to use games to model networks conflicts, as shown in Roy et al. (2010), Charilas and Panagopoulos (2010), Hoang et al. (2015), Moura and Hutchison (2018) and the references therein. This is due to the fact that game theory is a mature field with many important works covering different aspects of the theory (Fudenberg and Tirole, 1991; Basar and Olsder, 1999; Mailath and Samuelson, 2006; Mertens et al., 2015) that facilitate its applications on networking.

However, the use of game theory presents some important caveats. First, the most extended solution concept used in game theory, the Nash equilibrium, is computationally costly to obtain Daskalakis et al. (2009): this problem can be alleviated by using other solution concepts (Gilboa and Zemel, 1989). Second, in real environments a player may not know the objectives of the other players: this situation is known as incomplete information (Fudenberg and Tirole, 1991). Note that this may be the case in many network settings, as there might be nodes with different objectives that cause the game to range from extreme competition to extreme cooperation. And third, a player may not observe perfectly what other players do: this situation is known as imperfect information. These three problems arise frequently in wireless network environments, where the computational capabilities of each node may be constrained by its hardware and battery, where there might be malicious nodes in the network and where each node cannot possibly know what other players do. Motivated by this, we propose approaching the problem in an online learning fashion, where each player chooses what to do trying to maximize its own benefit.

In this article, we focus on repeated games (RGs), where a certain interaction among players is repeated a number of times, called stages. In each of these stages, the interests of the players are fixed, and hence, their rewards do not change along the game. RGs have been used to model a wide range of phenomena, such as macroeconomic policy (Alesina, 1987), supply chain interactions (Bao et al., 2020), lane changing in a freeway (Kang and Rakha, 2020) or optimal routing (La and Anantharam, 2002), to mention some.

Specifically, RGs are a very adequate tool to represent network interactions, as nodes in a network interact many times with a fixed target: as shown in Hoang et al. (2015), there are many network conflicts which have been modeled by making use of RGs, such as distributed resource allocation (Semasinghe and Hossain, 2016), interference management (Monsef and Saniie, 2015) or defense against

* Corresponding author.
E-mail address: j.parras@upm.es (J. Parras).

https://doi.org/10.1016/j.engappai.2021.104520
Received 9 February 2021; Received in revised form 23 September 2021; Accepted 17 October 2021
Available online 29 October 2021

attacks (Parras and Zazo, 2019). We can broadly classify RGs depending on how the total reward received by each agent (that we denote by payoff, preserving reward for the outcome received by each player at each stage) is obtained: we can average the rewards received in each stage (average payoff) or use a discount factor to weight the sum (discounted payoff). In network settings, discounted payoff schemes are more used (Hoang et al., 2015) due to being more realistic: the discount factor reflects the balance between present and future rewards and also, can be used as a measure on the uncertainty in the length of the game, whereas average payoff scheme assumes that the game duration is known. This might not be a realistic assumption in wireless networks (Konorski, 2006).

In the current literature in RGs applied to wireless networks, it is frequent that each work proposes a game model (i.e., the rewards for each player) and then proposes a certain algorithm that allows the players to obtain an equilibrium strategy (see, for instance, Qu et al., 2012; Semasinghe and Hossain, 2016; Monsef and Saniie, 2015 or Jaramillo and Srikant, 2007, to mention some). This has the advantage of finding a good strategy for that game, but that may not be applied to other games, even if similar.

We address all these issues by proposing a novel algorithm that can be used to obtain good payoffs in discounted RGs. Even though this area of research is intense (Hernandez-Leal et al., 2017a), most algorithms are valid only for average payoff RGs, but the difference introduced by the discount factor in the learning process cannot be skipped. As we show in this work, the difference between both payoff schemes acutes when the discount factor is low, i.e., is not close to 1. This has a strong impact on applications where low discount factors may be used, as in communication networks (Le Treust and Lasaulce, 2010; Xiao et al., 2012; Niyato and Hossain, 2008). Thus, our main contributions in this work are:

- We provide a detailed theoretical analysis of two important differences that arise between discounted and average payoff schemes that impact learning a strategy for an RG. As we have mentioned, discounted RGs are important in the networking field, so this is a significant gap.
- We propose a generic algorithm, that we call LEWIS (LEarning WIth Security) that is designed to learn a strategy that provides a good payoff in a discounted RG. It only requires knowing the discount factor and the rewards of the player (but not the rewards of the rest), and it is able to select actions in such a way that it is able to obtain good payoffs both in competition and cooperation situations. Its flexibility allows it to be used in many different problems, and it is computationally simple, which means that can be used in real wireless network deployments.

The rest of the article goes as follows. Section 2 introduces the mathematical background required for this work. Then, Section 3 analyzes theoretically two important differences between discounted and average payoff games, that have an impact on the learning process. Then, Section 4 introduces our own algorithm, specifically designed to learn in an online fashion and it is specifically designed for usual settings in wireless networks. We then check the performance of our algorithm using a distributed power control problem in Section 5 and a WBAN interference problem in Section 6, both of them taken from literature, where we compare our method to the specific strategy designed for each of these situations. Finally, we draw some conclusions and present some future work lines in Section 7.

2. Background

Let us introduce key concepts of Game Theory that will be used in this work.

2.1. Discounted payoff repeated games

We define a static game as follows (Basar and Olsder, 1999):

Definition 1 (*Static Game*)**.** A static game G is a triple $\langle N_p, A, r\rangle$, where:

- N_p is the number of players, numbered as $1, \dots, N_p$.
- A is the set of actions available to all players. The pure actions available to player i are denoted by a_i, with $a_i \in A_i$, being A_i the set of actions available to player i. A is defined as $A \equiv \prod_i A_i$. A mixed action is a distribution probability over actions.
- r is a function that gives the game rewards as:

$$r : \prod_i A_i \to \mathbb{R}^{N_p}.$$

We consider only discrete sets of actions (i.e., A_i are finite sets). If there are $N_p = 2$ players, then r_i can be expressed using a matrix R_i, whose dimensions are the number of actions of each player. A zero-sum game is a game in which the sum of the rewards of all players equals zero: $\sum_i r_i(a) = 0, \forall a \in A$. This means that the gains of some players are the loses of the others, and hence, zero-sum games model situations of extreme competition among players. The opposite situation happens when all players share the same rewards: $r_i(a) = r_j(a), \forall i, j \in N_p, \forall a \in A$. General sum games is the class of games that encompasses all possible reward functions.

Let us now define RGs:

Definition 2. A Repeated Game (RG) is built using a static game, also called stage game, which is played repeatedly over T periods. We consider RGs of infinite horizon, where $T = +\infty$. The main elements in an RG are the following, where superscript indicates time and subscript indicates the players:

1. The set of histories $\mathcal{H}^t \equiv A^t$. A history h^t is a list of actions played in periods $[0, \dots, t-1]$. In other words, a history contains the past actions.
2. A strategy for player i is a mapping from the set of all possible histories into the set of actions: $\sigma_i : \mathcal{H} \to A_i$. We denote by σ the strategy of all players.
3. The discounted payoff to player i is obtained with the infinite sequence of rewards $(r_i^0, r_i^1, \dots)$ as:

$$V_i(\sigma) = (1-\delta) \sum_{t=0}^{\infty} \delta^t r_i^t(a^t(\sigma)), \tag{1}$$

where $\delta \in (0, 1)$ is the discount factor.

Note that $a^t(\sigma)$ denotes that action $a^t = (a_i, a_{-i})$ is chosen following strategy $\sigma = (\sigma_i, \sigma_{-i})$, where the subscript i refers to the player and $-i$ refers to all players except player i. Also, we use r_i^t to denote rewards in the stage game, and V_i denotes the total discounted RG payoff. Note that if we considered mixed actions, we would have to take expectations in (1).

It is important to remark the role of the discount factor δ. It denotes how much the future rewards are valued, and hence, can be used as a measure of the player patience (Mailath and Samuelson, 2006). But δ can also be used when the temporal horizon is uncertain: in this case, the game may end at any stage with probability $1-\delta$ (Hoang et al., 2015), which means that the expected number of stages of the game is $(1-\delta)^{-1}$. Note that if $\delta = 0$, the expected number of stages is 1, as it coincides with the static case.

We can also define average payoff RGs, whose main difference with discounted RGs is that the total payoff now is as:

$$U_i(\sigma) = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{\infty} r_i(a^t(\sigma)), \tag{2}$$

where we use U_i for the average payoff and V_i for the discounted payoff (1). Note that the average payoff, intuitively, corresponds to the case in which $\delta = 1$, as it assigns the same importance to all rewards.

2.2. Equilibria concepts

In game theory, the solution to a game is known as equilibrium. There is not a unique equilibrium concept, but many (Mailath and Samuelson, 2006; Fudenberg and Tirole, 1991). In this Section, we work using the discounted payoff only, although equivalent concepts have been developed also for the average payoff.

The best known equilibrium concept is the Nash equilibrium (NE). When all players follow an NE strategy, no player can obtain a better payoff by a unilateral deviation. Mathematically,

$$V_i(\sigma_{i,NE}, \sigma_{-i}) \geq V_i(\sigma_i, \sigma_{-i}), \quad \forall \sigma_i, \quad \forall i \in N_p.$$

A key result is that any finite static game has at least one possibly mixed NE (Nash, 1951). It can be proved that the set of NEs in an RG includes the set of NEs of the stage game (Mailath and Samuelson, 2006). But an RG may have additional NE: this result is captured by the Folk Theorems (Mailath and Samuelson, 2006; Fudenberg and Tirole, 1991). Roughly speaking, the Folk Theorem states that in a repeated game, for a δ value sufficiently close to 1, better payoffs than the stage game NE payoff could be achieved. Note that the set of valid NEs in an RG depends on the discount factor value. There are several common strategies that take advantage of the Folk theorem, such as Nash reversion, tit-for-tat, grim or forgiving strategies (Mailath and Samuelson, 2006; Hoang et al., 2015). For instance, the grim strategy consists on cooperating (i.e., using a strategy that benefits all players), and in case that any player defects, a punishment strategy is followed forever.

Another frequent strategy is the minmax strategy (MS): the player finds a strategy such that it maximizes the worst reward that she could obtain:

$$\sigma_{i,MS} = \arg\max_{\sigma_i} \min_{\sigma_{-i}} V_i(\sigma_i, \sigma_{-i}).$$

The minmax strategy is the NE in zero-sum games: note that the minmax concept implicitly assumes that the other players are extreme competitors. For two players, two action games, the MS can be obtained using a linear program which depends only on the payoffs of one player (Aumann and Hart, 1992, Ch. 20).

However, both the MS and the NE have disadvantages. The MS, although simpler to compute, may yield very poor payoffs if the game is not of extreme competition, which is frequent in network settings. On the other hand, the NE may be computationally complex to obtain and requires knowing the reward functions of all players. A different approach consists on directly choosing strategies that provide a high payoff, regardless of whether they are equilibria or not. This is similar to no-regret strategies (Cesa-Bianchi and Lugosi, 2006; Zhou et al., 2016), as these strategies aim to provide a good payoff in terms of no-regret (i.e., a player does as good as possible), and we choose to use this approach in the algorithm we propose.

3. Discounted vs average payoffs

Before introducing our algorithm, we proceed in this Section to study two important effects that have an impact on learning schemes and that arise due to the discount factor, thus they only affect the discounted payoff scheme. For the sake of mathematical tractability, we consider in Sections 3.1 and 3.2 that the players use a fixed mixed strategy σ. That is, σ_i is a fixed probability distributions over A_i that does not change with time (for instance, it could be an NE). As a consequence, the sequence of actions $a_i^0, a_i^1, \ldots, a_i^t$ and rewards $r_i^0, r_i^1, \ldots, r_i^t$ for each player i is composed by independent and identically distributed random variables for each stage. Note also that the actions are independent among players, as each player samples their action sequence using their own σ_i.

3.1. Time to achieve a certain payoff

We have noted that the main difference between the average payoff U_i and the discounted payoff V_i, for the same rewards sequence $r_i^0, r_i^1, r_i^2 \ldots$, is that the discounted payoff is a weighted mean, with weights $(1-\delta), (1-\delta)\delta, (1-\delta)\delta^2, \ldots$. Since the weights are decreasing, the first difference between average and discounted payoff is that the discounted payoff scheme emphasizes the first several elements of the payoff sequence. Moreover, this effect depends on δ. Note that since we consider that player i uses a fixed strategy σ_i, then $\mathbb{E}[r_i^t]$, the expected reward for player i, is constant. Let us start by defining the metric t^M:

Definition 3. We define $t^M, M \in (0, 100)$ as the stage t of the repeated game in which the $M\%$ of the expected discounted payoff of player i has already been assigned in the discounted payoff case if player i uses a fixed strategy σ_i, that is,

$$t^M = \left\{ \min t \,\middle|\, \mathbb{E}\left[(1-\delta)\sum_{k=0}^{t} \delta^k r_i^k\right] \geq \frac{M}{100}\mathbb{E}\left[V_i\right] \right\}. \tag{3}$$

This definition of t^M leads to the following:

Theorem 1. *In a discounted, repeated game with infinite time horizon, with $\delta \in (0,1)$, if a fixed strategy σ is played in all stages, we can obtain t^M as*

$$t^M = \left\lceil \frac{\log\left(1 - \frac{M}{100}\right)}{\log(\delta)} - 1 \right\rceil, \tag{4}$$

where $\lceil x \rceil$ denotes that x is rounded up to the next integer

Proof. The problem we have to solve, using (3) and the definition of V_i (1), is to obtain the minimum t^M that satisfies

$$\mathbb{E}\left[(1-\delta)\sum_{k=0}^{t^M} \delta^k r_i^k\right] \geq \frac{M}{100}\mathbb{E}\left[(1-\delta)\sum_{k=0}^{\infty} \delta^k r_i^k\right],$$

which becomes

$$(1-\delta)\sum_{k=0}^{t^M} \delta^k \mathbb{E}\left[r_i^k\right] \geq \frac{M}{100}(1-\delta)\sum_{k=0}^{\infty} \delta^k \mathbb{E}\left[r_i^k\right],$$

and since σ_i is fixed, $\mathbb{E}\left[r_i^k\right]$ is constant, and hence,

$$(1-\delta)\sum_{k=0}^{t^M} \delta^k \geq \frac{M}{100}(1-\delta)\sum_{k=0}^{\infty} \delta^k. \tag{5}$$

Now, we can use the following expression for geometric sums

$$\sum_{t=t_0}^{t_1} \delta^t = \frac{\delta^{t_0} - \delta^{t_1+1}}{1-\delta}, \quad \delta \neq 1, \tag{6}$$

to manipulate (5) and obtain

$$1 - \delta^{t^M+1} \geq \frac{M}{100},$$

and the minimum t_M that solves this expression is (4). □

Note that t^M (4) can be used to study the part of the payoff that has been assigned on time stage t^M, which depends on the δ value as plot in Fig. 1 for $M = 99$. Small values of δ mean that the major part of the payoff is assigned in a short number of stages, whereas δ values close to 1 take more time stages to assign the payoff. Note that under average payoff, $t^M = \infty$ in the limit for $M > 0$. The impact that this has on a learning scheme is that under average payoff, the learning algorithm may converge in a long time stage t and it would not affect significantly the payoff. However, under a discounted payoff, the learning speed is key: a learning algorithm that converges slowly will yield poor payoffs.

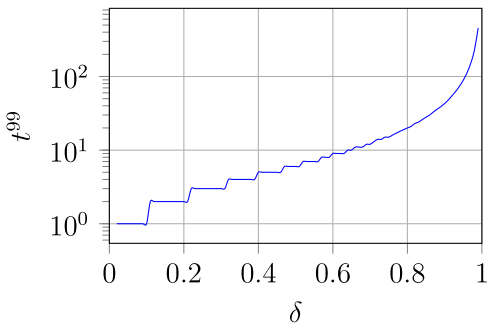


Fig. 1. Evolution of t^{99} as a function of δ values using (4). The horizontal axis represents the δ values, and in the vertical axis, we plot t^{99}, the number of stages needed to assign the 99% of the discounted payoff. Note that for low values of δ most of the payoff is assigned in a few stages. Also, note that we may use (4) in order to obtain the discount factor in a game where t^M is a given parameter.

3.2. Variance

A second difference between discounted and average payoffs is the variance of the total payoff when a mixed strategy is used: the variance depends on the discount factor, as the next Theorem shows.

Theorem 2. *In a discounted, repeated game with time horizon T and $\delta \in (0,1)$, if a fixed mixed strategy σ is played in all stages, the following expressions hold for the discounted payoff case:*

$$\begin{aligned} \mathbb{E}\left[V_i\right] &= (1-\delta^T)\mathbb{E}\left[r_i^t\right] \\ Var\left[V_i\right] &= \frac{1-\delta}{1+\delta}\left(1-\delta^{2T}\right) Var\left[r_i^t\right] \end{aligned}, \tag{7}$$

and for the average payoff case:

$$\begin{aligned} \mathbb{E}\left[U_i\right] &= \mathbb{E}\left[r_i^t\right] \\ Var\left[U_i\right] &= \frac{1}{T} Var\left[r_i^t\right] \end{aligned}, \tag{8}$$

where $\mathbb{E}\left[r_i^t\right]$ is the expected reward of player i and $Var\left[r_i^t\right]$ the variance of the reward of player i.

Proof. Since each player follows a mixed fixed strategy, the reward r_i^t of player i follows a distribution probability which depends on the product of probabilities for each player for each action vector a, because each player chooses her mixed action independently of the rest following her strategy. Thus, if we define $X_{t,i} = (1-\delta)\delta^t r_i^t$ as the random variable that models the reward that player i receives in stage t, we obtain:

$$\begin{aligned} \mathbb{E}\left[X_{t,i}\right] &= \mathbb{E}\left[(1-\delta)\delta^t r_i^t\right] = (1-\delta)\delta^t \mathbb{E}\left[r_i^t\right] \\ Var\left[X_{t,i}\right] &= Var\left[(1-\delta)\delta^t r_i^t\right] = (1-\delta)^2\delta^{2t} Var\left[r_i^t\right] \end{aligned}. \tag{9}$$

The expected discounted reward that a player obtains after T stages is

$$\mathbb{E}\left[V_i\right] = \mathbb{E}\left[\sum_{t=0}^{T-1} X_{t,i}\right] = \sum_{t=0}^{T-1}(1-\delta)\delta^t \mathbb{E}\left[r_i^t\right], \tag{10}$$

and thus, using (6), (9) and (10), and taking into account that the variance of the sum of independent random variables is the sum of the variances and that $\mathbb{E}\left[r_i^t\right]$ is constant, we obtain the following results for the discounted payoff (1):

$$\begin{aligned} \mathbb{E}_t\left[V_i\right] &= \mathbb{E}\left[\sum_{t=0}^{T-1}(1-\delta)\delta^t r_i^t\right] = (1-\delta^T)\mathbb{E}\left[r_i^t\right] \\ Var\left[V_i\right] &= Var\left[\sum_{t=0}^{T-1}(1-\delta)\delta^t r_i^t\right] \\ &= \sum_{t=0}^{T-1} Var\left[(1-\delta)\delta^t r_i^t\right] = \frac{1-\delta}{1+\delta}\left(1-\delta^{2T}\right) Var\left[r_i^t\right] \end{aligned}.$$

$$\begin{pmatrix} (1,-1) & (-1,1) \\ (-1,1) & (1,-1) \end{pmatrix}$$
Matching pennies (MP).

Fig. 2. Payoff matrices for MP. Player 1 is the row player and player 2 is the column player. In the matrix, the payoff entries for each pair of actions $a = (a_1, a_2)$ are $(r_1(a), r_2(a))$.

And solving for the average payoff case (2), we obtain:

$$\begin{aligned} \mathbb{E}\left[U_i\right] &= \mathbb{E}\left[\frac{1}{T}\sum_{t=0}^{T-1} r_i^t\right] = \mathbb{E}\left[r_i^t\right] \\ Var\left[U_i\right] &= Var\left[\frac{1}{T}\sum_{t=0}^{T-1} r_i^t\right] \\ &= \frac{1}{T^2}\sum_{t=0}^{T-1} Var\left[r_i^t\right] = \frac{1}{T} Var\left[r_i^t\right]. \quad \square \end{aligned}$$

Observe how in the average payoff case (8), the mean value of the total payoff coincides with the mean value of the reward, and the variance of the total payoff tends to zero with sufficiently long time stages. However, in the discounted payoff case (7), the mean value and variance of the total payoff depend on the δ value. For sufficiently large values of T such that $\delta^T \to 0$, the mean value of the total payoff coincides with the mean value of the reward, but the variance still depends on the δ value and only tends to 0 if $\delta \to 1$. In other words, the discount factor does also have an impact on the mean and variance of the total payoff. It is possible to understand this effect intuitively by realizing that the discount factor gives a higher weight to the firsts r_i^t values to obtain V_i: with a discount factor, say, $\delta = 0.1$, r_i^0 has a weight $(1-\delta)\delta^t = 0.9$, r_i^1 has a weight 0.09 and r_i^2 has a weight 0.009; that is, the first three r_i^t values concentrate the 0.999% of the total V_i. Hence, low δ values cause large variances because a small amount of samples dominates the payoff.

Thus, the variance dependence on the discount factor may also affect a learning procedure. First, because a larger variance means that a player needs to collect many rewards in order to evaluate whether a strategy being learned is good or not in terms of payoff. And second, because a player may obtain a poor payoff in a repeated game even if the strategy being used is an equilibrium of the game, simply due to the large variance that appears when using a low discount factor.

3.3. Illustration using Matching Pennies

Let us illustrate the effects described in Sections 3.1 and 3.2 by using a known game as Matching-Pennies (MP). MP is a zero-sum game with $N_p = 2$ players, where each player has 2 possible actions: the first player receives a positive reward if both players choose the same action, and a negative reward if both players choose different actions, while the second player receives a positive reward when both players choose different actions. The payoff matrix of the game is in Fig. 2.

The NE of a two-player, two-actions zero-sum game can be obtained using a linear program which depends only on the payoffs of one player (Aumann and Hart, 1992, Ch. 20). In the case of MP, there is a single mixed NE: $a_1 = a_2 = (1/2, 1/2)$, which yields each player an expected reward $r_1 = r_2 = 0$. That is, if both players select their action with probability 1/2, they will obtain an expected reward of 0. However, let us assume that player 1 does play $a_1 = 1/2$, while player 2 does not know the equilibrium a_2. Player 2 may use a learning scheme in order to obtain a_2 (see Section 4.4 for references to current algorithms). However, Theorem 1 shows that by the time the learning algorithm has converged to $a_2 = 1/2$, most of the payoff may have already been assigned. By using (4), we reach the results contained in Table 1: observe that if $\delta \leq 0.5$, the 99% of the total payoff is assigned in the first six stages at most, the 95% in the first four stages and the

Table 1
Evolution of t^{99}, t^{95} and t^{90} for different δ values, where (4) was used to obtain the values of t^M.

δ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
t^{99}	1	2	3	5	6	9	12	20	43
t^{95}	1	1	2	3	4	5	8	13	28
t^{90}	1	1	1	2	3	4	6	10	21

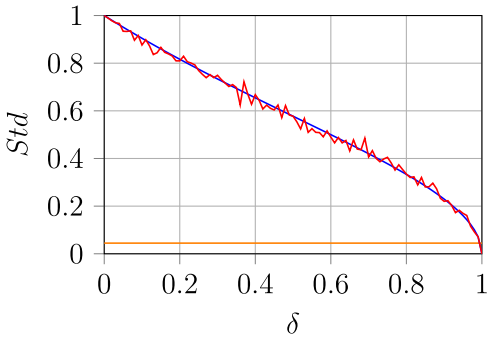


Fig. 3. Results of the standard deviation comparison simulation using MP. The horizontal axis represents the δ values, and in the vertical axis, we plot the standard deviation of the payoff. Orange line is for the theoretical average payoff case, using (8). Blue line is for the theoretical discounted payoff case, using (7). Red lines are the empirical standard deviation obtained under simulation. Note how the standard deviation depends on δ under the discounted payoff case. Also, note that the average payoff case gives in general lower deviations, except when $\delta \to 1$.

90% in the first three. Thus, any learning algorithm that we may use must be extremely fast in order to ensure that the agent obtains a good payoff if the discount factor is low, otherwise, by the time the player has obtained a good strategy, the amount of payoff to be assigned may be negligible, and hence, the player may obtain a poor payoff.

Also, the variance of the payoffs depend on the discount factor, as Theorem 2 states. If both players know the NE and play it, $\mathbb{E}[r_i^t] = 0$ and $Var[r_i^t] = 1$. We simulated 100 different repeated games with $T = 500$ stages using 101 equispaced values of $\delta \in [0, 1]$ and obtained the empirical standard deviation of the payoff for each δ value. Also, we obtained the theoretical standard deviation value using (7). For comparison purposes, we added the theoretical standard deviation under the average payoff scheme using (8). The results can be observed in Fig. 3, where we can notice that the standard deviation in the discount payoff scheme depends on the δ value and is larger than the standard deviation under the average payoff scheme except when $\delta \approx 1$. Note that in MP, if the discount factor is low, the payoff obtained by each player may be far from the expected value of 0, as the variance is large.

4. LEWIS algorithm

We have shown that discounted RGs are of interest to wireless networks, and also shown that any algorithm that pretends learning how to play them must be aware of the effects that arise due to the discount factor. We proceed to describe a generic algorithm that can be used to play in these situations, which we call LEWIS, as an acronym of LEarning WIth Security. It is a distributed algorithm that learns as players interact: each player first computes a security strategy, that provides a worst-case, possibly poor payoff, and then they try to improve that payoff if possible. Some LEWIS highlights are:

- LEWIS tries to improve the security payoff, thus, implicitly it makes use of the Folk Theorem.
- LEWIS is designed for secure online learning in discounted setups, taking into account the effects presented in Section 3.
- LEWIS can be used in games of incomplete information, as each player needs not knowing the reward functions of the others, but only knowing their own reward function.

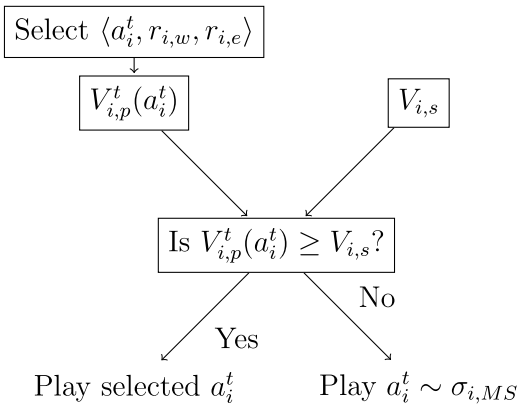


Fig. 4. LEWIS block diagram.

- LEWIS can be used in games of imperfect information, as each player only needs to know its own actions and the reward that receives at each stage.

Instead of relying on the concept of equilibria, LEWIS modifies its strategy in an online fashion depending on the rewards received by the player, and hence, it is similar in spirit to no-regret techniques. The basic idea is playing a certain action a_i^t if and only if (1) the expected payoff by using a_i^t is larger than a security payoff and (2) the worst case payoff of using a_i^t does not fall below a certain threshold. A block diagram of LEWIS can be found in Fig. 4, which we now explain.

4.1. Action selection block

The first key component of LEWIS is the action selection block, which returns a triple $\langle a_i^t, r_{i,w}, r_{i,e} \rangle$ formed by a recommended action a_i^t, the worst reward that could be obtained by playing action a_i^t, $r_{i,w}$, and the expected reward of playing action a_i^t, $r_{i,e}$. We propose selecting actions based on the past rewards received by agent i. For each discrete action $a_i \in A_i$, we keep a measure of the reward that player i has obtained by playing this action in the past, $\hat{r}_i^t(a_i)$. The index t is used because the estimation is updated in each stage t as

$$\hat{r}_i^t(a_i) = \begin{cases} (1-\alpha)\hat{r}_i^{t-1}(a_i) + \alpha r_i^t & \text{if} \quad a_i = a_i^t \\ \hat{r}_i^{t-1}(a_i) & \text{if} \quad a_i \neq a_i^t \end{cases}, \tag{11}$$

where $\alpha \in [0, 1]$ is a parameter that controls how much weight we give to the current reward. Note that (11) obtains an exponential weighted average of the received payoffs, where the exponential decay is controlled by α. Larger α values provide a faster update, but also larger variance. We initialize the estimation optimistically to facilitate exploration (Stimpson et al., 2001), by setting $\hat{r}_i^{-1}(a_i) = \max_{a_{-i}} r_i(a_i, a_{-i})$, that is, the estimation is initialized to the maximum reward value for each action.

At each stage t, this block recommends the action with a larger reward estimation, that is, $a_i^t = \arg\max_{a_i} \hat{r}_i^{t-1}(a_i)$ and $r_{i,e} = \hat{r}_i^{t-1}(a_i^t)$. The worst case reward is the minimum reward that player i would obtain by playing a_i^t, that is, $r_{i,w} = \min_{a_{-i}} r_i(a_i^t, a_{-i})$. LEWIS may decide to use a_i^t or not, and then a reward r_i^t would be received, which allows updating $\hat{r}_i^t(a_{i,t})$ using (11) and the actual action chosen by LEWIS.

4.2. Security condition

LEWIS also has a security property, as it guarantees a minimum payoff for the player by comparing a payoff estimation with a security payoff. Since each player knows only her own payoff, it is not possible for her to compute the NE of the game and use this as a security payoff. However, the player could use a worst-case value as security value: the minmax values, as the minmax payoff maximizes the worst reward that

the player could obtain and hence, the MS can be used as a security strategy to guarantee a payoff $V_{i,MS}$ at worst, regardless of what other players do.

However, as mentioned, V_{MS} may yield a poor payoff, as it assumes extreme competition among players. As we want LEWIS to provide good payoffs when possible, there needs to be a compromise between the security payoff and the ability to cooperate with other players. We model this compromise by using a parameter $\epsilon \geq 0$ to define $V_{i,s}$, the security payoff for player i, as

$$\begin{aligned} V_{i,s} &= \mathbb{E}\left[V_{i,MS} - \epsilon\right] = \mathbb{E}\left[(1-\delta)\sum_{t=0}^{\infty}\delta^t r_{i,MS}\right] - \epsilon \\ &= \mathbb{E}\left[r_{i,MS}\right] - \epsilon, \end{aligned} \tag{12}$$

where use (1) and (6). Note that in (12), if $\epsilon = 0$, player i would always use $\sigma_{i,MS}$ in order to guarantee herself a security payoff equal to the minmax payoff. However, if ϵ is a positive value, player i could be willing to use actions that do not follow $\sigma_{i,MS}$ if the worst case payoff provided by these actions is larger than the security payoff, which now is smaller than the MS payoff by ϵ.

Thus, at each stage t, LEWIS first obtains a recommended action a_i^t and then it decides whether to play this recommended action or not. In order to make that decision, LEWIS obtains the worst predicted payoff for a_i^t, $V_{i,p}^t(a_i^t)$, as

$$\begin{aligned} V_{i,p}^t(a_i^t) =& \mathbb{E}\left[(1-\delta)\left(\sum_{k=0}^{t-1}\delta^k r_i^k + \delta^t r_{i,w} + \sum_{k=t+1}^{\infty}\delta^k r_{i,MS}\right)\right], \\ =& (1-\delta)\sum_{k=0}^{t-1}\delta^k r_i^k + (1-\delta)\delta^t r_{i,w} + \delta^{t+1}\mathbb{E}\left[r_{i,MS}\right] \end{aligned} \tag{13}$$

where $V_{i,p}^t(a_i^t)$ is the expected payoff that player i would obtain if she plays a_i^t and obtains the worst possible reward for this action, $r_{i,w}$, and in the rest of the game, player i follows its minmax strategy. Note that this idea is similar to the one-shot deviation principle (Mailath and Samuelson, 2006), widely used to find strategies that yield good payoffs, which is what LEWIS tries to do. By using (12) and (13), we define a secure action a_i^t as follows:

Definition 4. In a discounted repeated game with infinite time horizon, with $\delta \in (0,1)$, and $V_{i,s}$ and $V_{i,p}^t(a_i^t)$ defined as in (12) and (13) respectively, an action a_i^t is secure if

$$V_{i,p}^t(a_i^t) \geq V_{i,s}.$$

This condition means that the worst predicted payoff needs to be greater than or equal to the security payoff. If that condition is not satisfied, then, the action a_i^t is not considered secure and LEWIS follows the security strategy.

4.3. Algorithm overview

An overview of LEWIS is in Algorithm 1. As input, each player i needs only her own payoff function r_i, the discount factor δ and the ϵ value that will be used to set the security payoff (note that each player may use its own δ and ϵ value). At each stage t, player i obtains a_i^t and checks whether this action is secure or not using Definition 4. If a_i^t is secure, then the player plays it, otherwise, it follows the security strategy. After that, the reward is observed and the strategy block is updated using (11).

Regarding computational complexity, note that LEWIS is fast and efficient. The more computationally complex part consists on computing the minmax strategy, which can be done using linear programming (Aumann and Hart, 1992, Ch. 20). The action selection block update (11) implies a constant complexity with the number of actions, as at each time step, the reward for a single action is updated, and the security condition from Definition 4 is a single comparison. Thus, LEWIS is fast and efficient, and could be implemented in low resources devices.

Algorithm 1 LEWIS algorithm for player i

Input: δ, r_i, ϵ, α

```
Obtain the minmax values: r_{i,MS} and σ_{i,MS}
for t = 0, 1, 2, ... do
    Obtain ⟨a_i^t, r_{i,w}, r_{i,e}⟩
    Obtain V_{i,p}^t(a_i^t) using (13)
    if V_{i,p}^t(a_i^t) is secure using Definition 4 then
        Play a_i^t
    else
        Play a_i^t ~ σ_{i,MS}
    Observe the actions and rewards
    Update action selection block using (11)
```

A word is required regarding convergence. We give no guarantee that LEWIS will converge to an equilibrium, as we only focus on achieving a payoff as large as possible. Moreover, LEWIS provides no guarantees of payoff other than the security payoff: its capacity to improve this payoff depends on the kind of game and the value of ϵ. In other words, LEWIS does not learn a strategy σ, but rather, it decides at each stage whether to play a recommended action or the minmax action, and this choice is done purely in terms of payoff. In RGs, there might be many possible strategies that lead to different action sequences that provide similar payoffs (Nachbar and Zame, 1996). This fact is used by LEWIS by choosing only actions that provide the player with a certain payoff larger than the security payoff. Finally, note that the lack of guarantees on the maximum payoff and the strategy is a consequence of working on incomplete information settings: if a player knows the rewards of the others, they might use a different strategy as security strategy, although we only consider minmax strategies in this work.

Finally, we note that LEWIS does take into account that the first stages have a larger impact on the payoff (Theorem 1), as all the security conditions take into account the discount factor. However, we do not incorporate the dependence of the payoff with the variance (Theorem 2). This could be incorporated in LEWIS by making use of a more complex security payoff that could take this variance into account. We leave this for future work.

4.4. Similar works

Even though there are many proposed algorithms for learning how to play in an RG, these do not satisfactorily address the problems that we have explained in Section 3. A recent survey notes more than 20 algorithms specifically addressed to RGs (Hernandez-Leal et al., 2017a), yet most of them do not take into account the discount factor in the learning process. Even though many learning algorithms based on Q-learning use a discount factor in the algorithm updates, they end up using the average reward as total payoff. We have already noted that the main difference between using average and discounted payoff is that under the average paradigm, all rewards equally contribute to the total payoff (i.e., all the rewards have the same weight on the total payoff), whereas under a discounted payoff, the first rewards have a larger weight on the total payoff (Theorem 1). As we have noted, discounted payoffs are dominant in network settings, but most of the current learning algorithms are not designed to deal with discounted payoffs (Bowling, 2005; de Cote et al., 2006; Abdallah and Lesser, 2008; Kaisers and Tuyls, 2010; Bloembergen et al., 2010; Hernandez-Leal et al., 2017b; Abdallah and Kaisers, 2016; Banerjee and Peng, 2004; Powers et al., 2007; Conitzer and Sandholm, 2007; Crandall and Goodrich, 2011; Damer and Gini, 2017; Chakraborty and Stone, 2014; Hernandez-Leal et al., 2014, 2017c) or (Wunder et al., 2012). An exception is Peski (2014), which introduces an algorithm designed for repeated games with discounted payoff, which however, is only valid

when there is a sufficiently large discount factor, whereas LEWIS does not have such limitation.

Some existing algorithms have ideas that may be used for learning on discounted RGs. One possible idea is learning as fast as possible, which is done by FAL-SG (Elidrisi et al., 2014), which applies to stochastic games and does not study the effects of the discount factor. Note that learning fast may mean learning in a few stages if the discount factor is low, as most of the payoff may be assigned in the firsts stages as shown in Fig. 1. Thus, a fast algorithm may be a solution only if the discount factor is sufficiently large, but may not be a good solution for very low discount factors. A different way to cope with this problem could be using security conditions, which allow choosing strategies in such a way that a minimum payoff is guaranteed, as LEWIS does. Some algorithms that use this idea are GIGA-WoLF (Bowling, 2005), ReDVaLeR (Banerjee and Peng, 2004), M-Qubed (Crandall and Goodrich, 2011) and RSRS (Damer and Gini, 2017). However, the security in these algorithms is related to the average payoff concept, and some of them also require observing the actions of the rest of the players (as M-Qubed), whereas LEWIS does not need observing them. A final possibility, used in Parras and Zazo (2020), consists on having a phase prior to the game in which all players agree on the strategy they are going to follow such that all of them receive a satisfactory payoff, but this algorithm requires a first phase dedicated to negotiation, whereas LEWIS learns in an online fashion.

5. Application to distributed power control

Let us now show how LEWIS can be applied to different situations in wireless networks, studying its advantages and disadvantages. To this purpose, we select two different problems, which are distributed power control in this Section, and interference control in Section 6. The first application that we present is distributed power allocation on wireless networks. In this field, game theory is an approach that has been proposed to deal with the conflicts that naturally arise between the network agents involved, as can be seen for instance in Ma et al. (2012), Shahid et al. (2014), Al-Imari et al. (2015) or Semasinghe and Hossain (2016): note that a node transmitting with more power maximizes its own throughput, but increases the interference to other network nodes, and hence, a compromise should be reached so that the total network throughput is optimized. In the remainder of this Section, we apply LEWIS to a setting based on the distributed resource allocation of power problem described in Semasinghe and Hossain (2016) for a wireless network.

5.1. Setup description

Let us assume that we have 2 small cells with full duplex users and two Base Stations (BSs), where for simplicity, we assume that each BS serves a single user (U), thus, there are two Us. We index each U and its BS with $k = \{1, 2\}$. We consider that there are two communication channels: one for the uplink (UL, i.e., from U_k to the BS_k) and another for the downlink (DL, i.e., from BS_k to U_k). Since all BSs and Us are close physically, they cause a non-negligible interference on the communications of the rest of the network. In order to characterize the quality of each communication link, we can use the signal to noise and interference ratio ($SINR$), which can be computed for the UL (i.e., at each BS) as follows:

$$SINR_k^{UL} = \frac{p_k^{UL} l_{BS_k,U_k}}{N_0 + p_j^{UL} l_{BS_k,U_j} + \gamma(p_k^{DL} + p_j^{DL} l_{BS_k,BS_j})} \tag{14}$$

where j indexes the other U/BS in the network (i.e., j and k index the two BSs/Us in the network), p_k^{UL} denotes the transmission power of U_k, p_k^{DL} is the transmission power of BS_k, $l_{a,b}$ denotes the signal attenuation between the positions of the Us / BSs a and b, N_0 is the thermal noise level at the receiver (we assume the same N_0 for all BSs and Us), and γ is a parameter that accounts for the attenuation between the UL and DL communication channels (i.e., co-channel interference). Equivalently, the $SINR$ for the DL at each U_k is:

$$SINR_k^{DL} = \frac{p_k^{DL} l_{BS_k,U_k}}{N_0 + p_j^{DL} l_{BS_j,U_k} + \gamma(p_k^{UL} + p_j^{UL} l_{U_k,U_j})} \tag{15}$$

As in Semasinghe and Hossain (2016), we can define a game between both BSs, where each BS must adjust its transmission power p_k^{DL} in order to optimize the interference caused, as it affects to the quality of the communication of the whole network. The authors in Semasinghe and Hossain (2016) model this situation by using a discounted repeated game whose stage game $G = \langle N_p, A, r \rangle$ (see Definition 1) is defined as follows:

- $N_p = 2$, as the players are the two BSs, and thus, the Us transmit using a fixed power. It could also make sense considering that the Us can also modify their powers, and hence, the game would become a four players game. Although LEWIS may deal with this situation, which speaks of its flexibility, we follow the original problem formulation in order to make a fair comparison with their strategy, which we use as baseline
- The set of actions A_k available to each player is a set of predefined transmission power levels. We assume that both players have the same action set, thus $A_1 = A_2$.
- The reward function for each player is:

$$r_k(a_1, a_2) = \log(SINR_k^{UL}) + \log(SINR_k^{DL}) \tag{16}$$

where, according to (14) and (15), both $SINR$ terms depends on a_1 and a_2, i.e., the downlink transmission power of each BS.

The authors in Semasinghe and Hossain (2016) show that the only NE of the stage game consists on transmitting with the maximum power available, which also means experiencing the maximum interference. In order to alleviate this problem, they use a repeated discounted game by making use of the Folk Theorem: there are better payoffs for both players if they are patient enough, i.e., when δ is sufficiently close to 1. To achieve these desirable payoffs, they propose using a Grim strategy: all players compromise to transmit with a certain power level that allows the SINR to be controlled. In case that any player deviates, which could be detected by an increase in the interference level, then all players switch to their NE strategy and transmit with their maximum power indefinitely. This Grim strategy turns out to be cheat-proof, as no player has any incentive to deviate, because its total payoff would decrease. Finally, it is necessary to reach an agreement on the initial transmission powers among all the players: the authors in Semasinghe and Hossain (2016) propose using a distributed algorithm based on perturbed Markov chains, in which each player tries different power transmission levels until it is satisfied with the payoff obtained. This algorithm is run during several iterations prior to the actual play of the game, during the so-called learning phase.

Table 2
Parameters used in the distributed power control problem, where $d_{a,b}$ is the distance between a and b.

Parameter	N_0	p_k^{UL}	γ	$l_{a,b}$
Value	0.001 W	10 W	0.001	$d_{a,b}^{-4}$

5.2. Empirical results

We can use LEWIS as an alternative way to address the distributed power control game. In order to test the performance of LEWIS, we use the setup described in Section 5.1, with the parameters contained in Table 2 and consider that the action space (i.e., transmission power options) of each BS is $A_k = \{5, 10, 15, 20, 25, 30\}$ W. We place our BSs and Us in the following (x, y) plane points: $(x_{BS_1}, y_{BS_1}) = (10, 10)$, $(x_{BS_2}, y_{BS_2}) = (0, 0)$, $(x_{U_1}, y_{U_1}) = (1, 8)$, $(x_{U_2}, y_{U_2}) = (5, 5)$. We use as

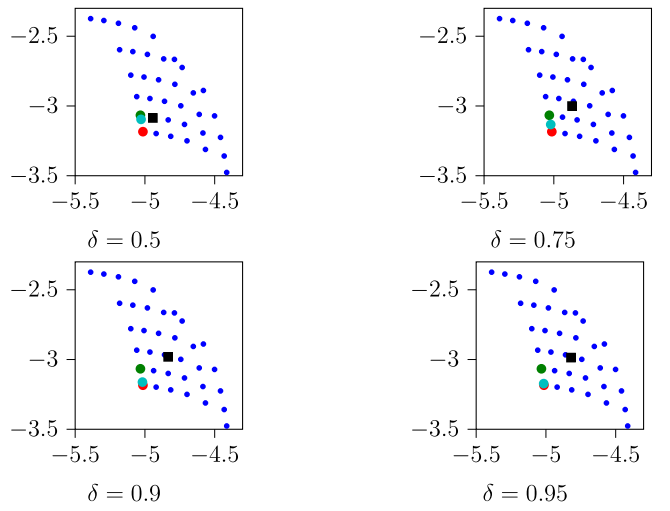


Fig. 5. Payoffs obtained for different values of the discount factor δ in the distributed power control problem. The horizontal axis is V_1, and the vertical axis V_2. The small blue dots are the possible payoffs provided by all combinations of $A_1 \times A_2$. The red point represents the NE payoff $V_{n,k}$: note that there are better payoffs for both players, i.e., points such that $V_k > V_{n,k}, \forall k = 1,2$. The green and cyan points have been obtained with the baseline, without deviation (green) and with the player 2 deviating in the second stage (cyan). The black square is the payoff obtained using LEWIS. Note how LEWIS is capable to obtain better payoffs than the baseline, and the improvement grows as δ increases.

baseline the algorithm proposed in Semasinghe and Hossain (2016), where the learning phase lasts 500 iterations. We compare the baseline strategy and the case in which each player uses LEWIS for different values of the discount factor $\delta = \{0.5, 0.75, 0.9, 0.95\}$, for $T = 500$ stages, as $t^{99} = 89$ (4) in the worst case (i.e., we set enough stages to deliver most of the payoff). In all cases, we set $\alpha = 0.5$. The results obtained can be observed in Fig. 5 and Table 3, where we set $\epsilon = 0.1$ for LEWIS (we tested for $\epsilon = \{0.01, 0.1, 1\}$, but they all provided very similar results).

The results in Fig. 5 let us draw several interesting conclusions. Firstly, we observe that the NE strategy provides a poor payoff $V_{n,k}$ for both players, as there are other actions (i.e., transmission powers) that provide both players with larger payoffs (i.e., payoffs such that $V_k > V_{n,k}, \forall k = 1,2$). Second, it is possible to check the cheat-proof property of the baseline used, as the payoff that player 2 obtains when deviates are always smaller than the payoff she could obtain by not deviating. And thirdly, LEWIS outperforms the baseline algorithm, specially as $\delta \to 1$, and provides payoffs which are better for both players (see also Table 3). This payoff increase is not the only advantage of LEWIS over the baseline in this problem. The baseline also has a training phase, which must last enough iterations to allow all players to find an adequate value for their transmission powers. In a real deployment, this would consume time and resources: LEWIS does not need such a training phase, and hence, it saves time and power. And a final advantage of LEWIS consists on its flexibility, as changes in the payoff function (i.e., using a metric different than $SINR$ in (16)) may affect the performance of the baseline, as the baseline was obtained as an equilibrium for this concrete reward function (Semasinghe and Hossain, 2016). However, as LEWIS needs minimal information about the problem (i.e., the payoff function of each player and the discount factor), it can also adapt to other problems.

6. Application to WBAN interference

Let us now present a second application related to Wireless Body Area Network (WBAN) interference control. A WBAN is formed by several sensors deployed around the human body that send their data to a central coordinator which controls the communication. Each sensor measures health indicators, and hence, the data collected by the sensors may be processed with medical purposes. However, in crowded environments, two WBAN may interfere with each other, and this may be a real threat to these networks which deal with delicate and sometimes very urgent data (Hanlen et al., 2010). In order to deal with this problem, Monsef and Saniie (2015) proposes using discounted repeated games to minimize the interference, which means not only saving battery, but also possibly saving lives if a WBAN needs to transmit a very urgent packet. In the remainder of this Section, we apply LEWIS to a setting based on the WBAN interference problem described in Monsef and Saniie (2015).

6.1. Setup description

In Monsef and Saniie (2015), the authors pose the interference problem between two WBANs as a discounted repeated game whose stage game $G = \langle N_p, A, r \rangle$ is defined as:

- The players are the two WBANs that are interfering, and hence, $N_p = 2$.
- Both players have the same two actions, hence, $A_1 = A_2$. The actions to transmit in the next time slot (T) or not to transmit (NT). That is, the central coordinator of each WBAN will allow its sensors to transmit or not. Note that if both WBANs transmit at the same time, there will be interference among the networks, and hence, the packets will be lost.
- As each player has only two actions, the payoffs of the stage game can be posed in matrix form as:

$$\begin{pmatrix} (0,0) & (-1,1) \\ (1,-1) & (-\lambda,-\lambda) \end{pmatrix} \tag{17}$$

  where player 1 (i.e., the first WBAN) is the row player and player 2 (i.e., the second WBAN) is the column player. In the matrix, the payoff entries for each pair of actions $a = (a_1, a_2) = (NT, T)$ are $(r_1(a), r_2(a))$. Note that if no player transmits, there is no gain nor loss, and if both transmit at the same time, there is a loss $-\lambda$, where λ is related to the time loss when there is a collision. Of course, if one network transmits, and the other does not, the one that transmits has a gain, while the one that does not has a loss. Note that the stage game has a single NE, which is $a_1 = a_2 = T$, which provides a payoff of $r_1 = r_2 = -\lambda$. Again, as in the case of the distributed power control problem, repeating the game will allow players to obtain better payoffs.

The authors in Monsef and Saniie (2015) also consider that each WBAN may have different emergency to transmit (for instance, due to limited battery life or very urgent health signals). They model this by assigning a different discount factor to each WBAN: the one that has more urgency to transmit has a lower discount factor than the other. Without loss of generality, we assume that player 1 has the urgency to transmit, and hence, $\delta_1 < \delta_2$.

In order to find an equilibrium for this repeated game, Monsef and Saniie (2015) proposes the following two-phase strategy:

- During the first t_k stages, player 1 uses $a_1 = T$ and player 2 uses $a_2 = NT$.
- From iteration $t_k + 1$ onwards, player 1 uses $a_1 = NT$ and player 2 switches between $a_2 = NT$ and $a_2 = T$ with probability $1/2 - e$ and $1/2 + e$ respectively, where $e \in [0, 1/2]$ is a parameter that controls the mixed strategy of player 2.
- Any deviation from the previous strategy causes the other player to always play $a = T$ (i.e., the NE strategy). Note that this is again a Grim strategy.

where:

$$t_k \leq \frac{\log\left(\frac{1-\lambda}{e+3/2}\right)}{\log \delta_2}, \quad e \leq \frac{2\delta_1 - 1}{\delta_1 - \frac{\lambda}{\lambda+1}} - \frac{3}{2} \tag{18}$$

Table 3
Payoffs obtained in the distributed power control problem, $\epsilon = 0.1$, where each pair of values account for the payoffs (V_1, V_2). Note that LEWIS is able to provide the best payoffs across several δ values: while the baseline provides a fixed payoff, LEWIS provides better results as the players become more patient, as there are more chances to achieve cooperation.

δ	0.5	0.75	0.9	0.95
NE payoff	$(-5.01, -3.18)$	$(-5.01, -3.18)$	$(-5.01, -3.18)$	$(-5.01, -3.18)$
Baseline payoff (without deviation)	$(-5.03, -3, 07)$	$(-5.03, -3, 07)$	$(-5.03, -3, 07)$	$(-5.03, -3, 07)$
Baseline payoff (with deviation)	$(-5.03, -3, 07)$	$(-5.02, -3, 13)$	$(-5.02, -3, 16)$	$(-5.02, -3, 17)$
LEWIS payoff	$(-4.94, -3, 09)$	$(-4.87, -3, 00)$	$(-4.83, -2.98)$	$(-4.82, -2.99)$

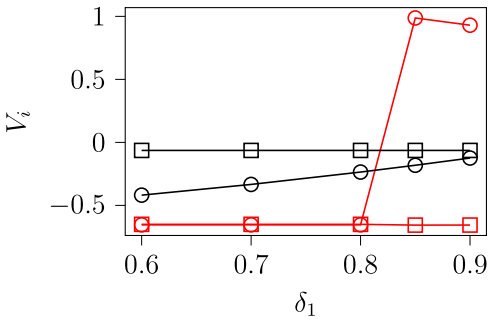


Fig. 6. Payoff obtained for player 1, V_1 (circled points) and player 2, V_2 (squared points), when using the baseline strategy described in Monsef and Saniie (2015) (in red) and LEWIS (in black). The baseline strategy provides the NE payoff, which is $V_1 = V_2 = -\lambda \approx -0.65$ in several cases except when condition (19) is fulfilled, that is, $\delta_1 > 0.81$. Note that the baseline gives player 2 a payoff very close to its Nash payoff, and allows player 1 to obtain a significantly higher payoff. Also, note how LEWIS allows, in all cases, to obtain a higher payoff for both players than their NE payoff.

The strategy explained relies on the fact that the more patient player may postpone its transmissions to later. The strategy is proved to be a valid one in Monsef and Saniie (2015) for the values of t_k and e shown in (18). However, this strategy presents an important problem, and it is the fact that the discount factor for the player 1 may be too restrictive. As shown in Monsef and Saniie (2015), δ_1 must fulfill the following:

$$\delta_1 \geq \frac{1 - \lambda(e + 1/2)}{(1 + \lambda)(1/2 - e)} \tag{19}$$

Thus, note that not all values of δ_1 can be used. This, in turn, means that the strategy proposed may find a restrictive application when certain values of δ_1 (i.e., urgency) are needed, as we are going to show in the next Section.

6.2. Empirical results

We can use LEWIS as an alternative way to solve the WBAN interference game, as LEWIS is able to deal with heterogeneous discount factors. In order to test the performance of LEWIS, we use the setup described in Section 6.1, with $\lambda = 628/960$, as in Monsef and Saniie (2015).

First, let us find the minimum value of δ_1. We compute the first derivative of (19) with respect to e and we obtain that the sign of the derivative depends on $(1 - \lambda)/(1 + \lambda)$. As we have that $\lambda < 1$, then δ_1 monotonously increases with the value of e, and hence, the minimum value of δ_1 corresponds to $e = 0$ (i.e., the minimum value of e), which means that $\delta_{1,min} = (2 - \lambda)/(1 + \lambda) \approx 0.81$, from (19). This in turn means that player 1 must expect that the game lasts at least $(1 - \delta)^{-1} \approx 5.36$ stages, that is, the first WBAN expects to have more than 5 transmission attempts. In case that the WBAN has more urgent data to transmit, then δ_1 would have to be smaller, and in that case, the strategy described in the previous Section would not be valid, as condition (19) would not be satisfied. Yet LEWIS is not affected by such restrictions, hence, we proceed to compare LEWIS and the cooperative strategy described in the previous Section as baseline.

We repeat the game (17) during 100 stages for the values $\delta_1 = \{0.6, 0.7, 0.8, 0.85, 0.9\}$, and we keep $\delta_2 = 0.95$ in all cases (again, $t^{99} = 89$ (4) in the worst case). Note that this means that the first WBAN expects that the number of stages in each case is $2.5, 3.33, 5, 6.67$ and 10, while the second WBAN expects 20 stages: note that the chosen values of δ_1 reflect different levels of urgency in the transmission, but note that only the last two values of δ_1 allow for the cooperative baseline: in case that $\delta_1 < 0.81$, the baseline resorts to using the NE strategy. We represent the payoffs obtained for each player in Fig. 6, where we have used $\epsilon = 0.1$ as this value maximizes the payoff of both players (we also tested for $\epsilon = \{0, 0.01, 0.1, 1\}$). In all cases, we set $\alpha = 0.5$.

The results obtained in Fig. 6 let us draw several conclusions. First, in case of using the baseline strategy where condition (19) is fulfilled (i.e., $\delta_1 = \{0.85, 0.9\}$), we note that the payoff obtained by player 2 is near the NE payoff, whereas the payoff obtained by the first player is significantly larger. This is due to the choice of e and t_k with the extreme values of the expressions (18). Note that there are several combinations of e and t_k possibles, and they would benefit the payoff of player 2 at the expense of the first player payoff. In other words, there should be a negotiation mechanism that enforces fairness, as player 2 is receiving a payoff which is very close to its Nash payoff. Second, note how LEWIS gives a better payoff to both players, as they receive a similar increase in payoff from the NE payoff and player 1 obtains a more significant payoff gain. LEWIS is thus competitive in terms of payoffs with a strategy designed specifically for this game which, as we have mentioned, is too restrictive in terms of δ_1. Note that LEWIS did not need a detailed knowledge of the game, as it only needs the payoff of each player, providing very good results in this situation in which each player has a different δ value.

7. Conclusions

This article has focused on proposing a generic algorithm for discounted RGs, which are becoming increasingly frequent in wireless communications to model the conflicts that arise between the network nodes. In these games, the discount factor accounts for uncertainty in the length of the game and thus, puts a larger weights on the first rewards in order to compute the total payoff. On the one hand, this is an idea consistent with wireless interactions, where the discount factor may account for mobility of the nodes, a finite battery or the urgency in data transmission. On the other hand, it also means that current learning algorithms, based on the average payoff scheme, should not be used where we have a discounted RG. The usual workflow in most papers where discounted RGs are applied to wireless networks consists on defining the game and finding a certain strategy that provides good payoffs. Thus, it is of interest trying to find an algorithm that can learn a good strategy in such games.

We have first shown theoretically that using discounted payoffs instead of average payoffs in an RG has an impact both in the speed of learning and in the variance of the total payoff achieved by the players. The first aspect appears due to the fact that most of the payoff is assigned in the first stages of the game (Theorem 1), and the second is due to the fact that the total payoff variance depends on the discount factor (Theorem 2).

These differences mean that current algorithm for learning RGs, mostly devoted to the average payoff case, are not adequate for discounted RGs. In order to fill this gap, we develop LEWIS, an online learning algorithm specifically designed for discounted RGs, suited for wireless network settings as it is computationally light, it is adequate

for incomplete and imperfect information settings (i.e., where a node does not know the interests of the other nodes and cannot observe their actions) and tries to reach a compromise between obtaining a good payoff and guaranteeing a minimum payoff at the same time, where this compromise is controlled by the parameter ϵ. We have tested LEWIS on two different wireless network settings taken from the literature, where it has shown to have a good performance. Hence, we believe that it can be an alternative way to address wireless network discounted RGs settings, as it only needs minimum information to work (i.e., the discount factor δ and the reward function of the player).

There are several future work lines that also arise from this work:

- First, regarding LEWIS, note that we have used a single action selection block based on the payoff obtained in the past (11), but there might be other action selection blocks that might help LEWIS to cooperate, such as a prosocial action selection block (Albrecht and Stone, 2018). It is important observing that the action selection blocks may take advantage of other information available to the player. For instance, an action selection block may use past observations in order to select an action (Crandall and Goodrich, 2011) given that the agent observes the actions of other players.
- Note that LEWIS takes into account the first theoretical difference between discounted and average RGs, as it considers that the first rewards have more weight on the total payoff (Theorem 1). However, LEWIS does not deal with the variance dependence with the discount factor (Theorem 2). This may be included in future versions of LEWIS, for instance, by modifying the action selection block to return a percentile of the expected value rather than the mean.
- Even though we have limited to use as security payoff the minmax payoff, note that different values could be proposed if we had more information about the game (i.e., a NE for instance).
- Finally, LEWIS could be tested in different situations where RGs are used in order to study its concrete performance and how it could be improved.

CRediT authorship contribution statement

Juan Parras: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing – original draft, Writing – review & editing, Visualization. **Patricia A. Apellániz:** Conceptualization, Software, Validation, Investigation, Writing – review & editing, Visualization. **Santiago Zazo:** Conceptualization, Validation, Investigation, Resources, Writing – review & editing, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

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

References

Abdallah, Sherief, Kaisers, Michael, 2016. Addressing environment non-stationarity by repeating Q-learning updates. J. Mach. Learn. Res. 17 (1), 1582–1612.
Abdallah, Sherief, Lesser, Victor, 2008. A multiagent reinforcement learning algorithm with non-linear dynamics. J. Artificial Intelligence Res. 33, 521–549.
Al-Imari, Mohammed, Ghoraishi, Mir, Xiao, Pei, Tafazolli, Rahim, 2015. Game theory based radio resource allocation for full-duplex systems. In: 2015 IEEE 81st Vehicular Technology Conference. VTC Spring. IEEE, pp. 1–5.
Albrecht, Stefano V., Stone, Peter, 2018. Autonomous agents modelling other agents: A comprehensive survey and open problems. Artificial Intelligence 258, 66–95.
Alesina, Alberto, 1987. Macroeconomic policy in a two-party system as a repeated game. Q. J. Econ. 102 (3), 651–678.
Aumann, Robert J., Hart, Sergiu, 1992. Handbook of Game Theory with Economic Applications, vol. 2. Elsevier.
Banerjee, Bikramjit, Peng, Jing, 2004. Performance bounded reinforcement learning in strategic interactions. In: National Conference on Artifical Intelligence, pp. 2–7.
Bao, Binshuo, Ma, Junhai, Goh, Mark, 2020. Short-and long-term repeated game behaviours of two parallel supply chains based on government subsidy in the vehicle market. Int. J. Prod. Res. 58 (24), 7507–7530.
Basar, Tamer, Olsder, Geert Jan, 1999. Dynamic Noncooperative Game Theory, vol. 23. SIAM.
Bloembergen, Daan, Kaisers, Michael, Tuyls, Karl, 2010. Lenient frequency adjusted Q-learning. In: Belgium-Netherlands Conference on Artificial Intelligence. BNAIC. pp. 11–18.
Bowling, Michael, 2005. Convergence and no-regret in multiagent learning. In: Advances in Neural Information Processing Systems. pp. 209–216.
Cesa-Bianchi, Nicolo, Lugosi, Gábor, 2006. Prediction, Learning, and Games. Cambridge University Press.
Chakraborty, Doran, Stone, Peter, 2014. Multiagent learning in the presence of memory-bounded agents. Auton. Agents Multi-Agent Syst. 28 (2), 182–213.
Charilas, Dimitris E., Panagopoulos, Athanasios D., 2010. A survey on game theory applications in wireless networks. Comput. Netw. 54 (18), 3421–3430.
Conitzer, Vincent, Sandholm, Tuomas, 2007. AWESOME: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. Mach. Learn. 67 (1–2), 23–43.
de Cote, Enrique Munoz, Lazaric, Alessandro, Restelli, Marcello, 2006. Learning to cooperate in multi-agent social dilemmas. In: International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 783–785.
Crandall, Jacob W., Goodrich, Michael A., 2011. Learning to compete, coordinate, and cooperate in repeated games using reinforcement learning. Mach. Learn. 82 (3), 281–314.
Damer, Steven, Gini, Maria, 2017. Safely using predictions in general-sum normal form games. In: Conference on Autonomous Agents and MultiAgent Systems, pp. 924–932.
Daskalakis, Constantinos, Goldberg, Paul W, Papadimitriou, Christos H, 2009. The complexity of computing a Nash equilibrium. SIAM J. Comput. 39 (1), 195–259.
Elidrisi, Mohamed, Johnson, Nicholas, Gini, Maria, Crandall, Jacob, 2014. 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, pp. 1141–1148.
Fudenberg, Drew, Tirole, Jean, 1991. Game Theory. MIT press Cambridge, MA.
Gilboa, Itzhak, Zemel, Eitan, 1989. Nash and correlated equilibria: Some complexity considerations. Games Econom. Behav. 1 (1), 80–93.
Hanlen, LW, Miniutti, Dino, Smith, David, Rodda, David, Gilbert, Ben, 2010. Co-channel interference in body area networks with indoor measurements at 2.4 GHz: Distance-to-interferer is a poor estimate of received interference power. Int. J. Wirel. Inf. Netw. 17 (3–4), 113–125.
Hernandez-Leal, Pablo, Munoz de Cote, Enrique, Sucar, L Enrique, 2014. A framework for learning and planning against switching strategies in repeated games. Connect. Sci. 26 (2), 103–122.
Hernandez-Leal, Pablo, Kaisers, Michael, Baarslag, Tim, de Cote, Enrique Munoz, 2017a. A survey of learning in multiagent environments: Dealing with non-stationarity. ArXiv preprint arXiv:1707.09183.
Hernandez-Leal, Pablo, Zhan, Yusen, Taylor, Matthew E, Sucar, L Enrique, de Cote, Enrique Munoz, 2017b. An exploration strategy for non-stationary opponents. Auton. Agents Multi-Agent Syst. 31 (5), 971–1002.
Hernandez-Leal, Pablo, Zhan, Yusen, Taylor, Matthew E, Sucar, L Enrique, de Cote, Enrique Munoz, 2017c. Efficiently detecting switches against non-stationary opponents. Auton. Agents Multi-Agent Syst. 31 (4), 767–789.
Hoang, Dinh Thai, Lu, Xiao, Niyato, Dusit, Wang, Ping, Kim, Dong In, Han, Zhu, 2015. Applications of repeated games in wireless networks: A survey. IEEE Commun. Surv. Tutor. 17 (4), 2102–2135.
Jaramillo, Juan José, Srikant, Rayadurgam, 2007. DARWIN: distributed and adaptive reputation mechanism for wireless ad-hoc networks. In: Proceedings of the 13th Annual ACM International Conference on Mobile Computing and Networking, pp. 87–98.
Kaisers, Michael, Tuyls, Karl, 2010. Frequency adjusted multi-agent Q-learning. In: International Conference on Autonomous Agents and Multiagent Systems, pp. 309–316.

Kang, Kyungwon, Rakha, Hesham A., 2020. A repeated game freeway lane changing model. Sensors 20 (6), 1554.

Konorski, Jerzy, 2006. A game-theoretic study of CSMA/CA under a backoff attack. IEEE/ACM Trans. Netw. 14 (6), 1167–1178.

La, Richard J., Anantharam, Venkat, 2002. Optimal routing control: Repeated game approach. IEEE Trans. Automat. Control 47 (3), 437–450.

Le Treust, Mael, Lasaulce, Samson, 2010. A repeated game formulation of energy-efficient decentralized power control. IEEE Trans. Wireless Commun. 9 (9), 2860–2869.

Ma, Wenmin, Zhang, Haijun, Zheng, Wei, Wen, Xiangming, 2012. Differentiated-pricing based power allocation in dense femtocell networks. In: The 15th International Symposium on Wireless Personal Multimedia Communications. IEEE, pp. 599–603.

Mailath, George J., Samuelson, Larry, 2006. Repeated Games and Reputations: Long-Run Relationships. Oxford University Press.

Mertens, Jean-François, Sorin, Sylvain, Zamir, Shmuel, 2015. Repeated Games. Cambridge University Press.

Monsef, Ehsan, Saniie, Jafar, 2015. A repeated game approach for interference mitigation of priority-based body area networks. In: 2015 IEEE International Conference on Electro/Information Technology. EIT. IEEE, pp. 324–329.

Moura, José, Hutchison, David, 2018. Game theory for multi-access edge computing: Survey, use cases, and future trends. IEEE Commun. Surv. Tutor. 21 (1), 260–288.

Nachbar, John H., Zame, William R., 1996. Non-computable strategies and discounted repeated games. Econom. Theory 8 (1), 103–122.

Nash, John, 1951. Non-cooperative games. Ann. of Math. 286–295.

Niyato, Dusit, Hossain, Ekram, 2008. Competitive pricing for spectrum sharing in cognitive radio networks: Dynamic game, inefficiency of nash equilibrium, and collusion. IEEE J. Sel. Areas Commun. 26 (1), 192–202.

Parras, Juan, Zazo, Santiago, 2019. Repeated game analysis of a CSMA/CA network under a backoff attack. Sensors 19 (24), 5393.

Parras, Juan, Zazo, Santiago, 2020. A distributed algorithm to obtain repeated games equilibria with discounting. Appl. Math. Comput. 367, 124785.

Peski, Marcin, 2014. Repeated games with incomplete information and discounting. Theor. Econ. 9 (3), 651–694.

Powers, Rob, Shoham, Yoav, Vu, Thuc, 2007. A general criterion and an algorithmic framework for learning in multi-agent systems. Mach. Learn. 67 (1–2), 45–76.

Qu, Zhengyang, Chen, Di, Sun, Gaofei, Wang, Xinbing, Tian, Xiaohua, Liu, Jing, 2012. Efficient wireless sensor networks scheduling scheme: Game theoretic analysis and algorithm. In: 2012 IEEE International Conference on Communications. ICC. IEEE, pp. 356–360.

Roy, Sankardas, Ellis, Charles, Shiva, Sajjan, Dasgupta, Dipankar, Shandilya, Vivek, Wu, Qishi, 2010. A survey of game theory as applied to network security. In: 2010 43rd Hawaii International Conference on System Sciences. IEEE, pp. 1–10.

Semasinghe, Prabodini, Hossain, Ekram, 2016. A cheat-proof power control policy for self-organizing full-duplex small cells. In: 2016 IEEE International Conference on Communications. ICC. IEEE, pp. 1–6.

Shahid, Adnan, Aslam, Saleem, Kim, Hyung Seok, Lee, Kyung-Geun, 2014. Distributed joint resource and power allocation in self-organized femtocell networks: A potential game approach. J. Netw. Comput. Appl. 46, 280–292.

Shoham, Yoav, Powers, Rob, Grenager, Trond, 2007. If multi-agent learning is the answer, what is the question? Artificial Intelligence 171 (7), 365–377.

Stimpson, Jeff L., Goodrich, Michael A., Walters, Lawrence C., 2001. Satisficing and learning cooperation in the prisoner's dilemma. In: International Joint Conference on Artificial Intelligence, pp. 535–540.

Stone, Peter, 2007. Multiagent learning is not the answer. It is the question. Artificial Intelligence 171 (7), 402–405.

Wunder, Michael, Yaros, John Robert, Kaisers, Michael, Littman, Michael, 2012. A framework for modeling population strategies by depth of reasoning. In: International Conference on Autonomous Agents and Multiagent Systems, pp. 947–954.

Xiao, Yuanzhang, Park, Jaeok, Van Der Schaar, Mihaela, 2012. Repeated games with intervention: Theory and applications in communications. IEEE Trans. Commun. 60 (10), 3123–3132.

Zhou, Zhengyuan, Glynn, Peter, Bambos, Nicholas, 2016. Repeated games for power control in wireless communications: Equilibrium and regret. In: 2016 IEEE 55th Conference on Decision and Control. CDC. IEEE, pp. 3603–3610.