Auton Agent Multi-Agent Syst (2013) 27:329–354
DOI 10.1007/s10458-012-9205-x

Forgetting as a way to avoid deception in a repeated imitation game

Pablo J. Villacorta · David A. Pelta · Maria T. Lamata

Published online: 11 August 2012

Abstract Adversarial decision making is aimed at determining optimal decision strategies to deal with an adaptive opponent. A clear example of such situation is the repeated imitation game presented here. Two agents compete in an adversarial model where one agent wants to learn how to imitate the actions taken by the other agent by means of the observation and memorization of the past actions. One defense against this adversary is to make decisions that are intended to confuse him. To achieve this, randomized strategies that change along time for one of the agents are proposed and their performance is analysed from both a theoretical and empirical point of view. We also study the ability of the imitator to avoid deception and adapt to a new behaviour by forgetting the oldest observations. The results confirm that wrong assumptions about the imitator's behaviour lead to dramatic losses due to a failure in causing deception.

Keywords Adversarial reasoning · Strategies · Decision making · Imitation · Repeated games

1 Introduction

Adversarial decision is largely about understanding the mind and actions of one's opponent. It is relevant to a broad range of problems where the actors are aware of each other, and they know they are contesting at least some of the other's objectives. The study of many of these strategic situations is carried with game theoretic tools, although other fields such as planning

P. J. Villacorta (✉) · D. A. Pelta · M. T. Lamata
Models of Decision and Optimization Research Group, CITIC-UGR, Department of Computer Science and AI University of Granada, C/Periodista Daniel Saucedo, 18071 Granada, Spain
e-mail: pjvi@decsai.ugr.es

D. A. Pelta
e-mail: dpelta@decsai.ugr.es

M. T. Lamata
e-mail: mtl@decsai.ugr.es

or probabilistic reasoning have also been used, as pointed out in [7]. The goal of this kind of analysis is to find optimal strategies taking into account not only one's preferences but also the beliefs and preferences of the opponents as perceived by oneself, which does not always match the true adversarial preferences.

A special case of adversarial situation is the model presented in [15] and reviewed here. It involves two participants, S and T, each of which chooses an action without knowing the choice of the other. As a result of these choices, a payoff is assigned to the participants. When this scenario is repeated many times, i.e. situations of repeated conflicting encounters arise, then the situation becomes complex as the participants have the possibility to learn the other's strategy. We can see this as a repeated *imitation* game, where the imitator T learns from the actions taken by S in the past. The more frequently T imitates S correctly, the smaller is the reward of S.

Imitation games have been recently studied from a formal perspective in works by McLennan and Tourky [10–12]. They are relevant for several reasons. Despite being a seemingly simple version of a non-cooperative two-personal game, they have proven as complex as a general game. It is shown in [11] that many problems related to the computation of an equilibrium in an imitation game are NP-complete, although McLennan and Tourky [10] proves that a mixed-strategy Nash equilibrium always exists in one-shot imitation games. Recall that this computation requires that a player knows the payoff matrices of both players.

However, all of these works focus on theoretical complexity issues and the analogy of Nash equilibrium problem with other, apparently non-related ones such as proving Kakutani's fixed-point theorem [10]. Further, none of these works explicitly considers repeated games. Here we expose a strategic situation in which, differently from many other studies, the agents do not have perfect knowledge of the game being played. To be precise, we present a finitely repeated imitation game where the imitator has no explicit access to the preferences of the other agent, as will be explained later. This makes our situation more realistic than traditional equilibrium-based approaches since we are not looking for the most rational strategy in a perfect-knowledge situation but in one that is based on repeated empirical observations which are likely to be deceptive.

When an agent S knows he is being observed by another agent T trying to learn from his behaviour, he should adopt some counter-measure to avoid this intrusion in his cognitive process. In that case, S should not choose his actions based just on his own preferences but must take into account the presence of the adversary and the fact that his behaviour should not be invariant and clear. A defense against T is to make decisions that are intended to confuse him, although S's reward can be diminished. Agent S wants to force the presence of uncertainty in order to confuse the adversary while its payoff is as less affected as possible, using randomized strategies [14,15,18] that minimize the payoff losses, either these are due to correct guesses or to non-optimal responses.

The aim of this work is twofold. Firstly, we will propose and analyze decision strategies for agent S that are not constant along the time, but change at certain time steps in the iterated process. We tackle the strategy design as a constrained non-linear optimization problem whose solution gives both the exact moment at which agent S must change and the new strategy he must use. Secondly, we will evaluate such strategies in a different scenario in which agent T forgets the oldest observations, in order to test if this is beneficial or not for T. This will test if the strategies presented in previous work are severely affected by wrong assumptions about the adversary or not.

The paper is organized as follows. Section 2 describes the main characteristics and components of the model used and discusses the suitability of game theoretic equilibrium concepts. Section 3 is a review of the material presented in [17], and deals with the need of randomized

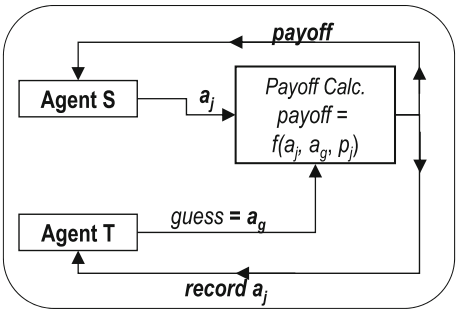

Fig. 1 Graphical representation of the model

Algorithm 1 Sequence of steps in the model.

for $l = 1$ to L **do**
  Agent T "guesses" an action a_g according to his strategy
  Agent S determines an action a_j according to his (randomized) strategy
  Calculate payoff for S as a function of p_j, a_g, a_j
  Agent T increases in one the value o_j
end for

strategies, explains a static mixed strategy for agent S, and also introduces the concept of dynamic, time-changing decision strategies as opposite to the former static strategy. The analytical expression of the expected payoff attained by S when using both kinds of strategies is explained step by step. An optimization process is also introduced here to obtain the best dynamic strategy under certain assumptions. Section 4 modifies the assumptions made in the preceding section to deal with an adversary with a limited observation memory, which yields a generalized expression of the expected payoff. In Sect. 5 we describe the computational experiments performed and the results obtained. They are aimed at checking the validity of the theoretical expressions with empirical results and comparing the performance of static and dynamic strategies with both unlimited and limited observation memory. Finally, Sect. 6 is devoted to discussions and further work.

2 Adversarial model

The model is a simplified version of the one presented in [15]. It is based on two agents S and T (the adversary or *imitator*) and a set of actions $A = \{a_1, a_2, \ldots, a_m\}$ available to both agents, with a set of payoffs $P = \{p_1, p_2, \ldots, p_m\}$ associated as follows. The agents play repeatedly a simultaneous imitation game. At each encounter[1], they must select an action at the same time and without knowing what the other agent will do. If S selects action a_j and T matches that choice (i.e. T was successful in predicting S's action), then T gets a payoff of one and S gets zero. Otherwise, S gets p_j and T gets none. Figure 1 shows a depiction of the model.

Algorithm 1 describes the steps of the model, with L being the length of the sequence of encounters that is known in advance by both agents. Different strategies for the agents are explained in Sect. 3. The reward calculation for S is formalised in expression (1).

$$f(a_j, a_g, p_j) = \begin{cases} 0 & \text{if } a_g = a_j \\ p_j & \text{otherwise} \end{cases} \tag{1}$$

[1] From now on, the terms *encounter* and *step* will be used interchangeably within the text.

Table 1 Payoff matrix of the simultaneous game played at each step

		T				
		a_1	a_2	a_3	...	a_m
S	a_1	(0,1)	(p_1,0)	(p_1,0)	...	(p_1,0)
	a_2	(p_2,0)	(0,1)	(p_2,0)	...	(p_2,0)
	a_3	(p_3,0)	(p_3,0)	(0,1)	...	(p_3,0)
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
	a_m	(p_m,0)	(p_m,0)	(p_m,0)	...	(0,1)

Showing payoffs for S and T for each possible outcome

This means that agent S gets no reward when agent T matched his response. The game outcomes and payoffs are summarized in the payoff matrix of the simultaneous game played at each stage, shown in Table 1. Notice that T is equally interested in guessing any action because all the correct guesses report the same payoff to him, while S does have an incentive for choosing some actions instead of some others because the payoff he may attain when not guessed is higher.

A key aspect of this game is that agent T does not know the payoffs S gets for each action when he is not correctly guessed, i.e. the vector $P = \{p_1, \ldots, p_m\}$ is unknown to T. However, after each encounter, both agents are informed of what the adversary has chosen in the last round. Agent T has his own memory as an observation vector $O = \{o_1, \ldots, o_m\}$ that he updates after each encounter. Element o_j stands for the number of times that, in the past, agent S has chosen action a_j. Agent T may take into account this information for future decisions.

Since agent T tries to learn S's behaviour, such behaviour should be partially randomized to avoid predictability. Although it is impossible to predict, a completely random behaviour might not be the best option because actions with low payoff may get selected too often, resulting in excessive loss with respect to the actions with the highest payoff.

2.1 On the suitability of equilibrium concepts

A note on the use of game theory tools should be done here. Apparently the model could be seen as a leader–follower game. In this kind of games, one of the players (the leader) has a strategic advantage over the other player (the follower) so he is able to *commit* to a strategy that he explicitly reveals to the follower in a credible way that guarantees he will not change the strategy he has committed to [5]. Committing to a strategy does not imply to reveal which action will be played by the leader (this is only the case if the commitment is to a pure strategy), since the strategy may be itself a randomization over the available actions (which is known as a *mixed* strategy [13]). In that case, the leader just reveals the probability distribution he is using to select an action. There are many works on security with autonomous agents that make use of this idea [1,3] and assume that the leader using a randomized strategy implicitly reveals his strategy because the follower observes the leader's behaviour for long enough before engaging in conflict. However, note that the situation presented in this work section requires that the follower, agent T, gives a response at each turn, i.e. at the same time that he is recording observations. The reason for which one may consider that this kind of game fits our problem is the fact that we are emphasizing agent T is able to record

observations of the behaviour of S in the past. To some extent this can be considered as if agent S is revealing his strategy.

In a leader–follower situation, the solution concept is the leader–follower equilibrium which maximizes the leader's expected payoff while at the same time the follower maximizes his own payoff, i.e. the follower's strategy is the best one for him, given the strategy imposed by the leader. This equilibrium is usually computed using bilinear programming tools and states that the best strategy for the follower is always pure [5]. Further, in order for this equilibrium to be computed, it is not necessary that the follower knows the leader's payoffs, and one can think this is just what happens in our problem: agent S knows both players' payoffs, but T does not have access to S's payoffs.

Having stated the reasons for which a leader–follower game seems initially to be the model to be fitted here, now we will discuss why, in our opinion, such choice is not the appropriate one. In our adversarial model, the only way to implicitly reveal a mixed strategy is through the repeated observations recorded by agent T. The best strategy for the follower is to choose the action that the leader selects with highest probability. If agent T takes the observed relative frequencies as probabilities, then his best response is to choose the action that he has observed the most in the past (which we call most frequent in the next section), since it reports the highest expected payoff for the follower.

However, the fact that the only clue about S's strategy are empirical observations motivates that agent S, anticipating the best-responser he may be facing, can use a deterministic, alternating strategy that systematically avoids selecting an action when it is currently the most frequently used. For this reason, the leader–follower equilibrium is not feasible since it always prescribes a pure strategy for the follower. Recall that our game is repeated along time and S could learn to avoid the predictions when they are made on a deterministic basis. Thus T's strategy should also be mixed. The explanation for this behaviour is that observing S's past behaviour is not equivalent to considering S is implicitly revealing his actual behaviour which, for instance, could be based on some other complicated, non-randomized behaviour rules that are difficult to perceive but do not actually constitute a mixed strategy. In other words, S does not make a true commitment and, as a result, T has no reason to consider S's past behaviour to be credible for the future and thus, to trust it.

Finally, note that conventional Nash equilibrium to mixed strategies is unfeasible since T does not know S's payoff and therefore T cannot compute such equilibrium[2].

For these reasons, we assume that agent T employs a different mixed strategy called Proportional to Frequency instead of Most Frequent, as described below.

3 Behaviour of the agents

3.1 Strategies for agent T

In [15], the authors presented and evaluated several strategies for agent S and T. A first, simple deterministic strategy for agent T is called *most frequent* (MF), which deterministically chooses the action that was observed most times in the past. But in that case, agent S could easily exploit this strategy by always selecting alternatively the best and the second-best actions, avoiding all the time a correct guess. For this reason, it seems reasonable that T uses a randomized strategy in his predictions. Another simple strategy is called *proportional*

[2] The use of uncoupled dynamics that do not need any knowledge of the adversary's payoff to eventually converge to mixed equilibrium under some conditions is proposed as future work in Sect. 6

to frequency (PF), with good performance according to empirical results [15]. It means that the probability of selecting an action as a prediction is proportional to the number of times (in percentage) such action was observed in the past (this information is stored in the observation vector O). As can be seen, it is a randomized strategy and because of that, it is safer than MF. In what follows, PF will be the strategy assigned to agent T.

We must point out that constructing a model for T is an aspect that deserves further investigation (see for instance [19] and [9]), possibly involving learning techniques.

In Sects. 3.2 and 3.3 we review the approaches published in [17] for agent S to emphasize the assumptions that allowed to compute such decision strategies. Sections 3.4, 4 and 5 present a completely novel study on how a change on certain adversarial conditions can affect the performance of decision strategies that were supposed to be optimal under original assumptions that do not hold anymore.

3.2 Static mixed strategy for agent S.

Agent S could use a totally deterministic strategy that always select the action with the highest payoff. However, this would be very easy to learn for agent T so he would quickly predict this behavior correctly after a short number of repetitions. S could also employ a totally random strategy that would select an action in a totally random way. This behaviour would be very hard to learn from observations but, on the other hand, the payoff attained would be low because bad actions (i.e. those with low payoff) may be selected with the same probability than best actions.

The need exists here to get to a good balance between confusion and payoff, as concluded in [15]. But instead of running computational simulations that test new proposals of strategies, our objective now is to calculate the expected payoff of new strategies without running a computational experiment.

As stated before, a mixed strategy is a set of weights representing a probability distribution over the actions. When a player has to do a movement, he uses this probability distribution to choose his action. In our model, we are interested in the *best randomization* or, in other words, the set of weights that lead to the highest payoff when playing against agent T. From now we will use the expression *set of weights* to refer to a probability distribution over the actions of the model. Thus such weights are in [0, 1] and their sum is one.

With these weights, it is possible to calculate the expected payoff for a given player, which is the sum of all the possible outcomes of the game weighted by the probability that each outcome eventually arises and by the payoff that outcome reports to the player. In the adversarial model we are dealing with, this means that we can weight each payoff in vector P by the probability that agent S eventually gets that payoff. This probability can be computed as the product of the probabilities of two independent events happening simultaneously. Agent S will attain payoff p_j if two conditions hold:

(i) Agent S must select action a_j as a response. This probability is noted α_j, although we will refer to it as the *weight* used by S to select action a_j.
(ii) In addition, S will only get the payoff p_j if agent T does not successfully predict his response. Actually this probability is independent of (i) if we consider it as a conditional probability that is conditioned to the fact that S has already selected action a_j.

This (conditional) probability can be computed as follows. In case agent S is using a non-variant weight α_j (a *static* mixed strategy) during a sequence of L repetitions of the game, then the probability that agent T does not guess his actions if T uses PF strategy is

(1 - α_j). The explanation is as follows. After L repetitions, since agent S uses α_j, then action a_j would have been selected $L \cdot \alpha_j$ times, and this is what agent T sees in O_j. The probability that T selects action a_j as a prediction is

$$P_{guess}(j) = \frac{O_j}{\sum_{j=1}^{m} O_j} = \frac{L \cdot \alpha_j}{L} = \alpha_j \tag{2}$$

with m being the number of actions available. The probability of not being guessed correctly is then $1 - P_{guess}(j) = 1 - \alpha_j$.

Taking into account the probabilities of the two conditions described above yields the following expression of the expected payoff for agent S after a sequence of L encounters when he uses weights α_j to select his actions:

$$EP_{\text{static}} = L \sum_{j=1}^{m} \alpha_j (1 - \alpha_j) p_j \tag{3}$$

If we want to maximize the expected payoff, we have to maximize expression (3) by computing the values of the optimal weights α_j. This can be achieved using numerical methods for constrained optimization. The optimization problem can be formalized as follows.

$$\max_{\{\alpha_j\}} \sum_{j=1}^{m} \alpha_j \cdot (1 - \alpha_j) \cdot p_j \tag{4}$$

subject to:

$$\sum_{j=1}^{n} \alpha_j = 1$$
$$\alpha_j \geq 0 \tag{5}$$

3.3 Dynamic mixed strategy for agent S

In the previous section we described a static strategy for agent S. It was static in the sense that the same set of weights is used all the time to make encounters for which S will use the same weights to choose his action. The static mixed strategy described above can be viewed as one single period, because the weights computed by S do not change along time. Now the idea is to define several periods and calculate the optimal mixed strategy for every period. The *length* of a period is the duration of the period, i.e. the number of consecutive encounters of the game during which agent S will use the same mixed strategy.

For a given period, the set of optimal weights is different from that of other periods. Let N_h be the length of the h-th period. The next example illustrates this concept. Suppose that we have a sequence of length $L = 100$ encounters. Then, we can define for instance four periods of length $N_1 = 30$, $N_2 = 10$, $N_3 = 20$ and $N_4 = 40$. Figure 2 shows a depiction of a dynamic mixed strategy.

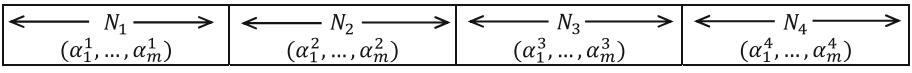

Fig. 2 Example of a dynamic strategy with four different periods. The *letters* inside each rectangle represent the length of that period and the mixed strategy to be used in that period

In order to calculate the best randomization under this scenario, we need to apply constrained optimization methods to compute the values of the optimal weights for each period, along with the optimal length of every period. In principle, we should also compute the optimal number of periods but this, as will be seen later, requires solving independent optimization problems since the number of periods affects the total number of variables of the problem. For this reason, the number of periods is assumed to be known. See Sect. 5.2.1 for a study on the influence of this parameter.

The objective function of such constrained optimization problem is the expression of S's expected payoff for a dynamic strategy, which can be computed as follows. Let $(\alpha_1^h, \ldots, \alpha_m^h)$ be the set of weights that agent S uses to choose an action during the h-th period. Then, within a given period, α_j^h represents the probability that S selects action a_j. The difficult part of the expression we need is computing the probability of not being guessed, which is the same within a period but different from one period to another. After the first period of length, say, N_1, the observation vector O has the following values (absolute frequencies of the responses given in the past by S):

$$O = \left(\begin{array}{cccc} N_1 \cdot \alpha_1^1 & \cdots & \cdots & N_1 \cdot \alpha_m^1 \end{array} \right)$$

Clearly, the probability of not being guessed P_{NG} during the first period is $(1\text{-}\alpha_j^1)$, according to the same explanation given in Sect. 3.2. This reasoning becomes more complicated when considering the observation vector at the end of the second period, whose length is N_2:

$$O = \left(\begin{array}{cccc} N_1 \cdot \alpha_1^1 + N_2 \cdot \alpha_1^2 & \cdots & \cdots & N_1 \cdot \alpha_m^1 + N_2 \cdot \alpha_m^2 \end{array} \right)$$

According to these values, the probability *at the end of the second period* that agent T selects action a_j as a prediction is

$$P_{guess}(j) = \frac{O_j}{\sum_{j=1}^m O_j} = \frac{N_1 \cdot \alpha_j^1 + N_2 \cdot \alpha_j^2}{N_1 + N_2}$$

so the probability of not being guessed *at the end* of the second period is

$$P_{NG}(j) = 1 - P_{guess}(j) = \frac{N_1(1 - \alpha_j^1) + N_2 \cdot (1 - \alpha_j^2)}{N_1 + N_2}$$

What happens in the middle, i.e. *during* the second period? The probability of not being guessed changes at every step because the number of times each response has been observed by T varies along time. This variation can be modeled as follows. At a certain step s of the second period (s is measured from the beginning of the period, so $0 \leq s \leq N_2$), the probability that T correctly predicts response a_j is

$$P_{guess}(j, s) = \frac{O_j}{\sum_{j=1}^m O_j} = \frac{N_1 \cdot \alpha_j^1 + s \cdot \alpha_j^2}{N_1 + s} \tag{6}$$

Notice we have added a second argument to this probability to emphasize that it also depends on the step s of the simulation. The probability of not being guessed is then

$$P_{NG}(j, s) = 1 - P_{guess}(j, s) = \frac{N_1(1 - \alpha_j^1) + s(1 - \alpha_j^2)}{N_1 + s}$$

As stated before, notice that this probability changes at every step within a period. Now, it is possible to generalize this reasoning to obtain the probability of not being guessed at step s of the h-th period ($0 \leq s \leq N_h$):

$$P_{NG}(j,h,s) = 1 - P_{guess}(j,h,s) = \frac{\sum_{k=1}^{h-1} N_k(1-\alpha_j^k) + s(1-\alpha_j^h)}{\sum_{k=1}^{h-1} N_k + s}$$

Argument h points out the period to which s belongs, although this is just a formalism that will be eliminated below. The expression of the total expected payoff with H periods of length N_h is a generalization of (3), using (7) as probability of not being guessed:

$$EP_{\text{dynamic}} = \sum_{h=1}^{H}\sum_{s=1}^{N_h}\sum_{j=1}^{m} \alpha_j^h \cdot \frac{\sum_{k=1}^{h-1} N_k(1-\alpha_j^k) + s(1-\alpha_j^h)}{\sum_{k=1}^{h-1} N_k + s} p_j \tag{7}$$

The next expression should be also verified: $\sum_{k=1}^{H} N_h = L$. Recall that L is known in advance by both agents.

Once again, the reader should note that this expression is valid only when T uses the strategy Proportional to Frequency. Further, S must know in advance the length of the sequence (the exact number of steps of the complete simulation) to add the last constraint to the optimization process.

With this approach, the number of unknown parameters is greater than that of static mixed strategies. Recall that the number of periods H is given by the user in advance, but the optimal length of each period N_h is unknown. In addition, instead of computing only m weights as in the static mixed strategy, we have to compute $m \cdot H$ weights (H sets of weights) plus the H lengths of the periods N_h for $h = 1, \ldots, H$ which are integer values. The optimization problem in this case can be formulated as follows:

$$\max_{\{\alpha_{ij}^h\}\cup\{N_i^h\}} \sum_{h=1}^{H}\sum_{s=1}^{N_h}\sum_{j=1}^{m} \alpha_j^h \cdot \frac{\sum_{k=1}^{h-1} N_k(1-\alpha_j^k) + s(1-\alpha_j^h)}{\sum_{k=1}^{h-1} N_k + s} p_j \tag{8}$$

subject to:

$$\begin{aligned} \sum_{j=1}^{m} \alpha_j^h = 1, \quad & h = 1, \ldots, H \\ \alpha_j^h \geq 0, \quad j = 1, \ldots, m \quad & h = 1, \ldots, H \\ \sum_{h=1}^{H} N_h = L & \end{aligned} \tag{9}$$

As explained before, the number of unknown variables to be optimized when we are computing a dynamic strategy with H periods is $m \cdot H + H = H(m+1)$. Therefore, a different optimization problem with a different number of variables is generated for each H value (see Fig. 2). The way to find the best choice for H is to solve the optimization problem for different H values and accept the strategy that provides the highest payoff for S.

3.4 A generalized notation for the expected payoff

In this part we will rewrite expression (7) with a slightly different notation that simplifies and generalizes the previous one and that will be employed in Sect. 4.1.

For a given step d of the simulation, let $PCD(d) : N \rightarrow N$ be the index (starting in one) of the period that precedes the one to which d belongs, or zero if d belongs to period one. The step d is measured from the beginning of the simulation, so in this case $1 \leq d \leq L$. The

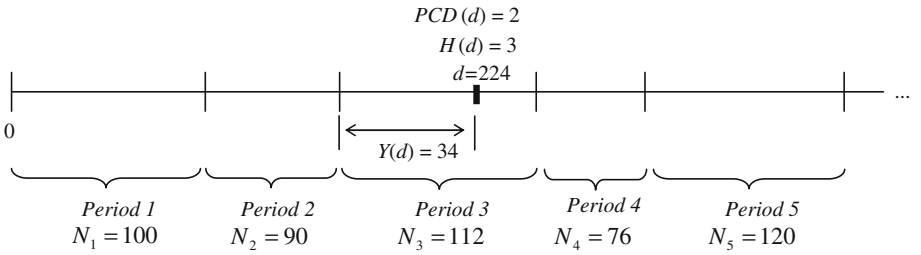

Fig. 3 Example of dynamic strategy and the notation explained in this section

name *PCD* stands for *preceding*. Let $H(d)$ be the period to which step d belongs. $H(d)$ and $PCD(d)$ can be defined in terms of the lengths of the periods as follows:

$$PCD(d) = \begin{cases} max\{k \in N : \sum_{h=1}^{k} N_h < d\} & \text{if } d > N_1 \\ 0 & \text{otherwise} \end{cases} \qquad (10)$$

$$H(d) = PCD(d) + 1$$

Now let $Y(d)$ be the number of steps measured from the beginning of period $H(d)$:

$$Y(d) = d - \sum_{k=1}^{PCD(d)} N_k \qquad (11)$$

Figure 3 depicts the meaning of these functions when applied for instance to step $d = 224$ belonging to the third period of a strategy whose five first periods have length 100, 90, 112, 76 and 120.

Finally, let $OBS(d, j)$ be the value of cell O_j after d steps. It can be expressed in terms of the weights used by S in his decisions, as follows. The *otherwise* part was written to make this function also valid for the case of limited memory, as explained later.

$$OBS(d, j) = \begin{cases} \left(\sum_{k=1}^{PCD(d)} N_k \alpha_j^k\right) + Y(d)\alpha_j^{H(d)} & \text{if } d > 0 \\ 0 & \text{otherwise} \end{cases}$$

As can be seen, (12) is the numerator of expression (6), generalized for an arbitrary number of steps d. The total number of observations in the observation vector after d steps is exactly d:

$$\sum_{j=1}^{m} OBS(d, j) = \left(\sum_{j=1}^{m} \sum_{k=1}^{PCD(d)} N_k \alpha_j^k\right) + \sum_{j=1}^{m} Y(d)\alpha_j^{H(d)} =$$

$$= \left(\sum_{k=1}^{PCD(d)} N_k \sum_{j=1}^{m} \alpha_j^k\right) + Y(d) \sum_{j=1}^{m} \alpha_j^{H(d)} =$$

$$= \left(\sum_{k=1}^{PCD(d)} N_k\right) + d - \left(\sum_{k=1}^{PCD(d)} N_k\right) = d$$

so the probability of being guessed after d steps when choosing action a_j can be rewritten as $\frac{OBS(d,j)}{d}$ if $d > 0$. In the first encounter ($d = 0$), nothing has been observed yet by agent

T so we assume he makes a prediction in a totally random way. The probability of being guessed is thus $1/m$ in this case. With these functions, expression (7) can be rewritten as

$$EP_{\text{dynamic}}(d) = \sum_{j=1}^{m} \alpha_j^1 (1 - 1/m) p_j + \sum_{s=2}^{d} \sum_{j=1}^{m} \alpha_j^{H(s)} \left(1 - \frac{OBS(s-1, j)}{s-1}\right) p_j \quad (12)$$

where d is the number of steps of the simulation, provided that we have defined enough periods to cover, at least, that number of steps.

4 Forgetting as a way to avoid deception

The key point of the optimized mixed strategies presented in [17] is that they intend to manipulate the observations made by T so that S has long periods to safely choose very good actions without being guessed. In other words, they were exploiting the fact that T considers all the observations done in the past and gives all of them the same importance. This means that, if T observed a very frequent action many times in the past, and uses Proportional to Frequency, then it is difficult for him to predict something different in the future unless he observes a different action approximately as many times as the first one. We can see this phenomenon as a sort of "inertia" which makes T difficult to recover after a series of observations of one very frequent action. For that reason, S manipulates with its actions T's observation vector to his convenience, and in fact the optimization process tells S the best way to do this at every step.

As mentioned in Sects. 3.2 and 3.3, expressions (3) and (7) consider that T is playing proportional to frequency with an unlimited memory, i.e. T would be annotating every action a_j forever. We hypothesize that a limited memory for T would be beneficial for T (or harmful for S) because it would attenuate or even eliminate this inertial behaviour that makes T easy to manipulate. This way T can be thought of as a more intelligent adversary that only takes into account the most recent observations and simply forgets older ones.

4.1 Expected payoff with limited memory

The following reasoning can be applied to obtain the expression of the expected payoff when T uses PF but has limited memory. As stated before, limited memory means that the number of observations T can store in his memory is limited to a certain number. When the memory is full and a new action a_j is to be stored, the oldest observation is simply deleted. Suppose that the oldest observation was action a_t, then the deletion consists in decreasing the value O_t in one unit. This idea can be applied to obtain an expression of the expected payoff as follows. Suppose that we have observed $N_1 + s$ steps, and that agent S was using a mixed strategy of two periods of lengths N_1 and N_2. Then, recall that after $N_1 + s$ steps, the observation vector of T will look like

$$O = \left(\begin{array}{cccc} N_1 \cdot \alpha_1^1 + s \cdot \alpha_1^2 & \ldots & \ldots & N_1 \cdot \alpha_m^1 + s \cdot \alpha_m^2 \end{array} \right)$$

Suppose that now we add a new observation. We do not know what exact action S will do in the next step, but we can model the behaviour using the probability that the action is each of

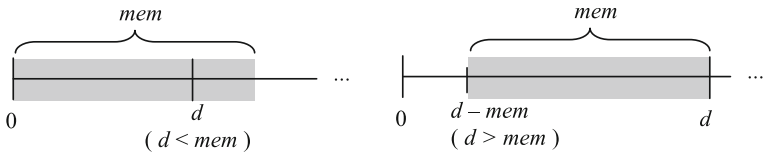

Fig. 4 Depiction of limited memory in a temporal line at step s of the sequence in two cases: when d is smaller (*left*) and greater (*right*) than the capacity of the memory. In the former case, agent T still remembers everything from the beginning of the simulation but in the latter, he only remembers the *mem* last responses of S

the possible ones. So after one additional step, the same row will now look like

$$O = \left(N_1 \cdot \alpha_1^1 + (s+1) \cdot \alpha_1^2 \ldots \quad \ldots N_1 \cdot \alpha_m^1 + (s+1) \cdot \alpha_m^2\right)$$

Notice that we have not added one unit to any component, but we have added α_j^2 to every component of the vector. Now let us see what happens if now T *forgets* the first observation he made, at stage, say, s_0. This means he will forget an observation that was made when S was in the first period of his strategy, i.e. S was using the set of weights $\{\alpha_1^1, \ldots, \alpha_m^1\}$ at that moment. Thus the probability that the action observed at step s_0 was a_1 is α_1^1, the probability that it was a_2 is α_2^1,…and the probability that it was a_m is α_m^1. Since we do not know what really happened at that step because we are modelling an expected payoff, we cannot subtract one to any concrete component of the observation vector. What we should do is *subtract the probability* that the action observed in that moment was each of the possible actions, just in an analog way that we added that probability to every component of the vector at the time when we considered a new observation. Thus the observation vector will look like

$$O = \left((N_1 - 1) \cdot \alpha_1^1 + (s+1) \cdot \alpha_1^2 \ldots \quad \ldots (N_1 - 1) \cdot \alpha_m^1 + (s+1) \cdot \alpha_m^2\right)$$

It is important to note that the weights that are subtracted correspond to those used by S at the moment of the observation that is being deleted. This means that it is necessary to compute which set of weights was being used by S at the moment of the observation. Function (12) was designed for this purpose, because it calculates what was in the observation vector at a certain step, no matter how many different periods were employed from the beginning to that step. If the capacity of the memory is limited, it is enough to delete everything that had been observed up to a certain step corresponding to the one before the oldest step that is still stored in memory. Thus the content of O_j after step d ($1 \leq d \leq L$) of the simulation with an observation memory limited to *mem* steps ($d > mem$) can be expressed as $OBS(d, j) - OBS(d - mem, j)$. The case where $d - mem \leq 0$ is nonsense but was added to the definition of function OBS to enable a general expression for the expected payoff. Obviously, $\forall j$, $OBS(d - mem, j) = 0$ when $d \leq mem$, meaning that T has observed nothing before the beginning of the simulation. Limited memory can be seen as a sliding window, as shown in Fig. 4.

Summarizing, expression (12) can be generalized to support limited observation memory of *mem* steps as follows.

$$EP_{\lim}(d, mem) = \sum_{j=1}^{m} \alpha_j^1 (1 - 1/m) p_j$$
$$+ \sum_{s=2}^{d} \sum_{j=1}^{m} \alpha_j^{H(s)} \left(1 - \frac{OBS(s-1, j) - OBS(s-1-mem, j)}{\min\{s-1, mem\}}\right) p_j \qquad (13)$$

5 Experiments and results

The experiments we conducted are aimed at answering the following questions:

1. Do the results obtained with the analytical expressions match those obtained by empirical simulations?
2. Do dynamic mixed strategies outperform a static mixed strategy in terms of expected payoff?
3. Does a limited memory have an impact over the expected payoff? Is it beneficial or detrimental for agent T to forget the oldest observations?

5.1 Experimental settings

The parameter configuration of the model instance that has been used in the empirical evaluation of strategies was the following:

– Number of different actions: $m = 5$.
– Number of encounters: $L = 500$.
– Payoff vectors: 15 different vectors were tested. The payoffs of every vector are summarized in Table 2.

Evaluation of a strategy To compare empirical and theoretical results for every payoff vector, we did the following. For each payoff vector, the numerical values p_j are substituted in the theoretical expression as well as the number of periods H that is required in case of dynamic mixed strategies (recall that we tested H varying from 1 to 4), and the optimization algorithm is then executed. The algorithm returns the values of the optimal strategy for that

Table 2 The 15 payoff vectors used in the experiments, separated in three groups according to the payoff structure, and the maximum total payoff attainable by S after 500 encounters

Vector	Payoffs					Maximum reward possible after 500 encounters
V_1	1	0.9	0.95	0.8	0.85	500
V_2	0.8	0.9	0.6	0.7	1	500
V_3	1	0.85	0.7	0.4	0.55	500
V_4	1	0.6	0.8	0.4	0.2	500
V_5	0.25	0.01	0.5	1	0.75	500
V_6	1.1	0.95	0.9	1.05	1	550
V_7	1.2	1	1.1	0.9	0.8	600
V_8	1.3	1	1.15	0.85	0.7	650
V_9	1.2	1.4	1	0.8	0.6	700
V_{10}	1.5	1	0.75	1.25	0.5	750
V_{11}	0.8	0.6	0.4	1.5	1	750
V_{12}	0.8	0.6	0.4	1.75	1	875
V_{13}	0.8	0.6	0.4	2	1	1000
V_{14}	0.8	0.6	0.4	2.25	1	1125
V_{15}	0.8	0.6	0.4	2.5	1	1250

payoff vector. Once we have such values defining the strategy, we evaluate it theoretically by substituting them in the theoretical expressions (3) (if it is a static strategy) or (7) and (13) (if it is dynamic), to obtain the expected payoff. In addition to this, we also use that strategy in 100 independent empirical simulations with the current payoff vector, and take the average of the executions to compare it with the theoretical expected payoff.

Optimization algorithm The *NMaximize* command of the *Mathematica* software package was used to solve the constrained optimization problems (4) and (8). The maximization it carries out is not analytical but employs approximate heuristic methods that depend on the initial solution. However, the same command was used for both the static and dynamic strategies so this phenomenon affects both families of strategies in the same way.

Recall that the number of periods H is not part of the optimization process but must be set by the user. In order to test the influence of such parameter, we tested H varying from 1 to 4 periods and compared the results. Insights on this are provided in Sect. 5.2.1. On the other hand, the optimal length N_h of every period is part of the set of variables to optimize.

5.2 Results

In order to answer the first question posed above, first note that the theoretical expressions have to be evaluated given the numerical values (period lengths and weights) that define the concrete strategy being tested. In our case, the strategies that we will evaluate are those obtained by the optimization process explained in previous sections.

Figure 5a shows a comparison of the expected and empirical payoff for an optimal dynamic mixed strategy with four periods with unlimited memory (see (7)). The plot confirms an almost perfect matching between the predicted and the actual payoff in all the payoff vectors. The payoff is presented as a percentage of the accumulated payoff over the *maximum*, which is the total payoff attainable if agent S always selects the action with the highest payoff and he is never guessed. This would be the ideal situation that only occurs when there is no adversary. Figure 5b shows again a perfect matching in case of limited memory for T, so expression (13) has been proven to be correct as well.

We now analyze the performance of both static and dynamic mixed strategies with unlimited memory to answer the second question posed above. The results are shown in Fig. 6. This figure proves a very important result. In all the payoff vectors tested, the three optimal dynamic mixed strategies outperformed the optimal static strategy. In addition, increasing the number of periods was always beneficial in terms of the payoff attained. Recall that these results do not come from a simulation but from a prediction made using the expressions, so they are not influenced by random factors. Notice that the greater gain in performance was achieved in vectors V_{11} to V_{15}, which are those where the highest payoff is much greater than the rest. This is a particularly encouraging result for problems in which it is especially important to do the best action as many times as possible.

In order to answer the third question, the impact of limited memory over the expected payoff is going to be studied. As explained in Sect. 4, limited memory is a violation of the assumptions made for the optimized dynamic mixed strategies. In other words, the optimization process carried out was assuming some facts that are not true anymore, so the results obtained are not optimal now. Figure 7 shows the expected payoff using the optimized dynamic strategies with four periods (which were the best-performing ones) both with unlimited and limited memory. Recall that all these strategies being tested now were originally designed assuming an unlimited-memory adversary so our aim now is to measure how the existing strategies are affected if the assumption is violated. According to Fig. 7,

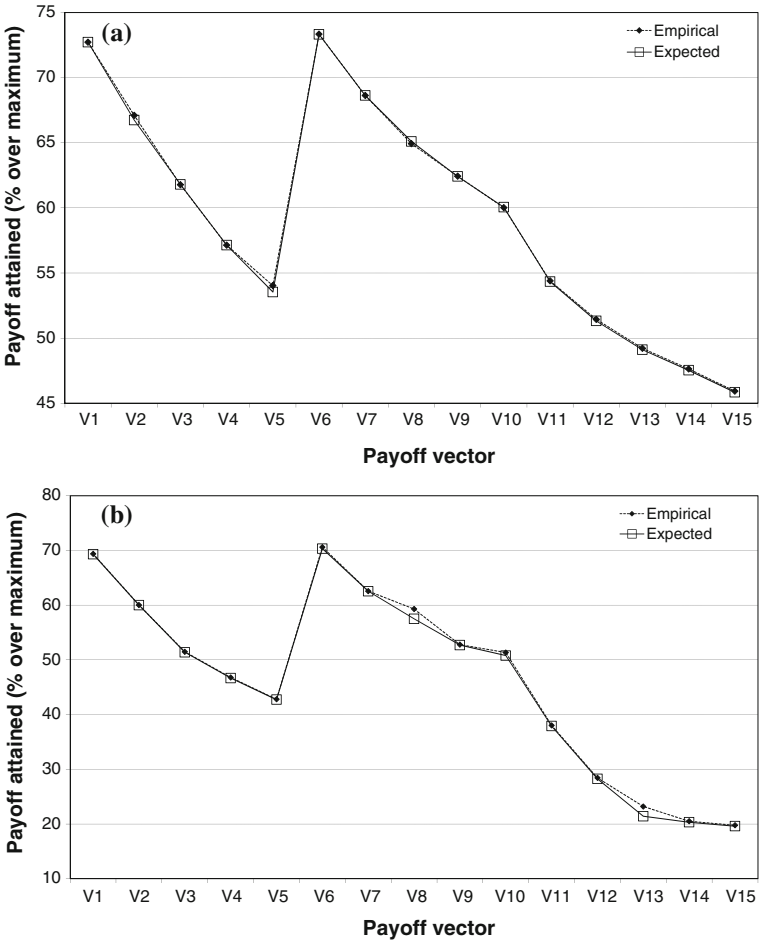

Fig. 5 Expected and empirical payoff for S of dynamic mixed strategies with four periods and unlimited memory (**a**) and with limited memory of 30 steps (**b**)

in all cases it was beneficial for T to have a limited memory as this allowed him to recover faster from deception and consequently the payoff of S was much lower. The case of payoff vectors V_{11} to V_{15} is dramatic. In these vectors, S achieved the greatest gain with respect to static strategies using four-period dynamic strategies. But now that T has limited memory, however, S only attains 20 % of the maximum payoff in V_{13}, V_{14} and V_{15}, 30 % in V_{12} and less than 40 % in V_{11}.

It is important to note that agent S could have designed better strategies if he knew in advance the length of the limited memory, i.e. if he had an accurate model of the adversary. In that case, he could exploit this knowledge: it would suffice to substitute the objective function of the optimization problem by the one with limited memory given in Sect. 4.1. As a result, S would not suffer such payoff loss, but we think that assuming such an accurate knowledge of T is too unrealistic as the information would be very asymmetric and favour S even more than in the current model.

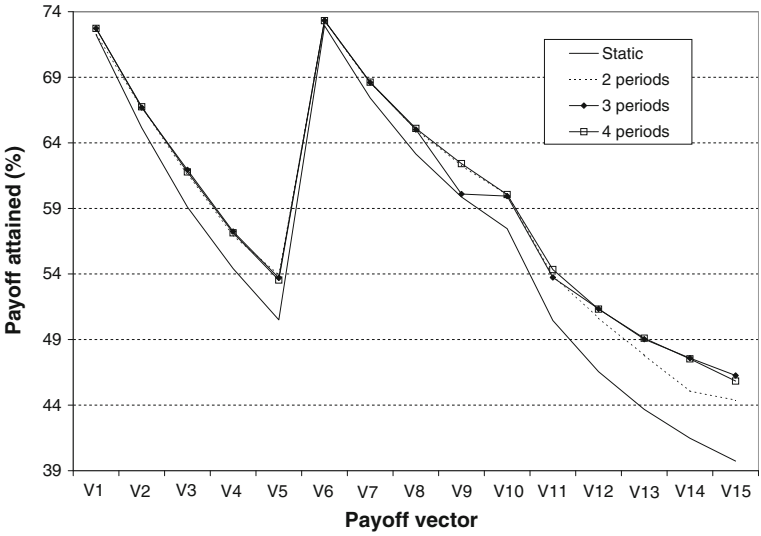

Fig. 6 Expected payoff for S with unlimited memory for every payoff vector

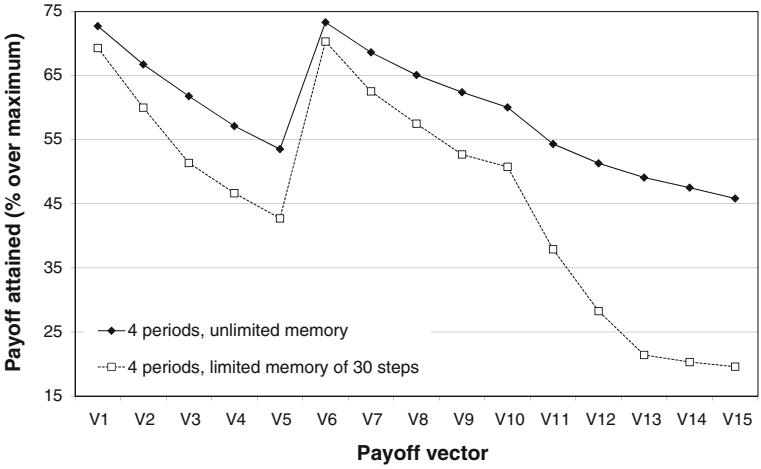

Fig. 7 Expected payoff for S of the four-period dynamic strategies originally designed under unlimited memory assumption, now being tested with both unlimited and limited memory for every payoff vector

Strategies obtained with four periods Examining the optimal dynamic strategies obtained is useful to understand how agent S tries to cause deception along time. Figure 8 shows four selected dynamic strategies with four periods. The mixed strategy used in each period is depicted in a 5-axis star-plot where the axes represent the $m = 5$ available actions of the model $a_1, \ldots, a_5$. The continuous and dashed polygons represent the probabilities of choosing each action during each of the periods. Since the dynamic strategies depicted have four periods, we have plotted the first two periods in one plot (on the left side) and the last two periods in another (on the right), to avoid excessive overlapping on one single plot that makes it difficult to read. Notice that in the last period, usually the best and second-best actions are the only ones with a probability greater than zero of being chosen. However these choices are

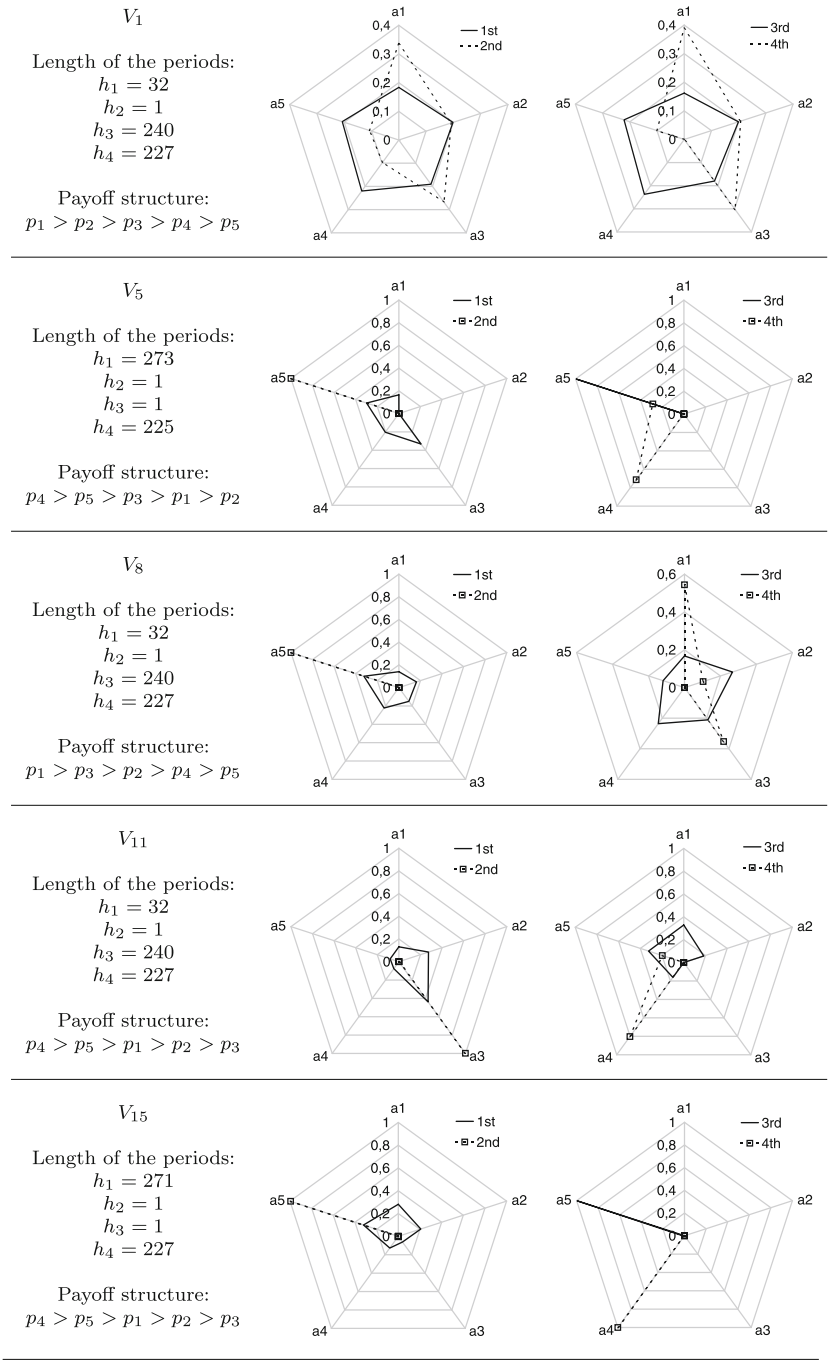

Fig. 8 Depiction of the optimal four-period strategies obtained for vectors V_1, V_5, V_8, V_{11} and V_{15} after applying optimization, assuming unlimited observation memory of T

supposed to be safe because the rest of the actions were chosen many times during the first periods. This behaviour is emphasized in vectors in which the best action has associated a very prominent payoff in relation with the other actions, as happens in V_5 and V_{15}. In the last period the strategy just chooses the best action all the time, but since we assume unlimited memory, agent S will not be able to recover from all the previous observations and as a result, it is still safe to deterministically choose the best action during all the fourth period.

5.2.1 Insights into the behaviour with limited memory

In order to understand why limited memory is beneficial for T, we propose to study the probability of making a correct guess, and how it changes along time. This study must be done for one specific action since the evolution of this probability is different from one action to another. Recall that dynamic strategies are aimed at inducing confusion at the beginning to safely choose the best action after the initial periods. For this reason, the most interesting action to track this evolution is the best action (i.e. that with highest payoff) of each vector. Figure 9 shows the evolution of the probability of being guessed correctly $(O_b / \sum_{j=1}^{m} O_j)$ when choosing the best action a_b of payoff vectors V_1, V_{11} and V_{15}. Agent T employs a four-period dynamic mixed strategy. The changes on the guess probability are more abrupt when the memory is limited because the observation vector starts to display the new behaviour very quickly after a change of S. As a consequence, T is able to adapt to the new situation faster and the deception that S tried does not work very well. The cases of vector V_{15} is specially dramatic. As it was very important to choose the best action as many times as possible because it has a remarkably high payoff, S had planned a first period with several different bad choices, and then switching to a second period with a single repeated best choice. Now T quickly notices this change and when S starts choosing always the best action in the second period, T learns this behaviour and starts being successful in his predictions from a very early stage of this second period.

Since limited memory for T seems to be very harmful for S in relation to unlimited memory, it may be interesting as well to find out what the exact impact of memory capacity over the expected payoff is. In the extreme case, with a memory of only one step of capacity, T would always make his election based on the last action observed. Of course, if S were aware of this, it would be very easy for him to exploit this behaviour to his own benefit, but for now we will assume S does not know. We will study the impact in relation to the payoff vector used in order to discover if limited memory affects some vectors more than others. Figure 10 shows the results. The lines represent the theoretically expected payoff for a given capacity value after 500 encounters. As can be seen in the plot, we have tested the capacity of the observation vector varying from 1 to 500. A capacity of 500 in a 500-step setting is equivalent to infinite memory.

First of all, the three plots of Fig. 10 confirm that the shorter the memory, the worse for S or equivalently the better for T. As expected, the payoff vector has a clear influence on the impact of limited memory. For instance in vectors V_1 to V_5 as well as V_6 to V_{10}, the slope of the curves 4 and 5 as well as 9 and 10 is greater than the others, because these vectors are "more difficult" than the rest. By *difficult* we mean that the payoffs of the vector are very different among them, so every time a good action is correctly guessed by T, it leads S to a great loss in payoff. In these cases it is specially important to choose good actions to avoid great losses. In vectors V_{11} to V_{15} one of the payoffs is much greater than the rest as explained in preceding sections. All the curves of that group have big slopes meaning that the impact of the capacity of the memory is greater. Secondly, all the curves tend to present big slopes at the beginning but turn horizontal after a certain value of capacity. This value

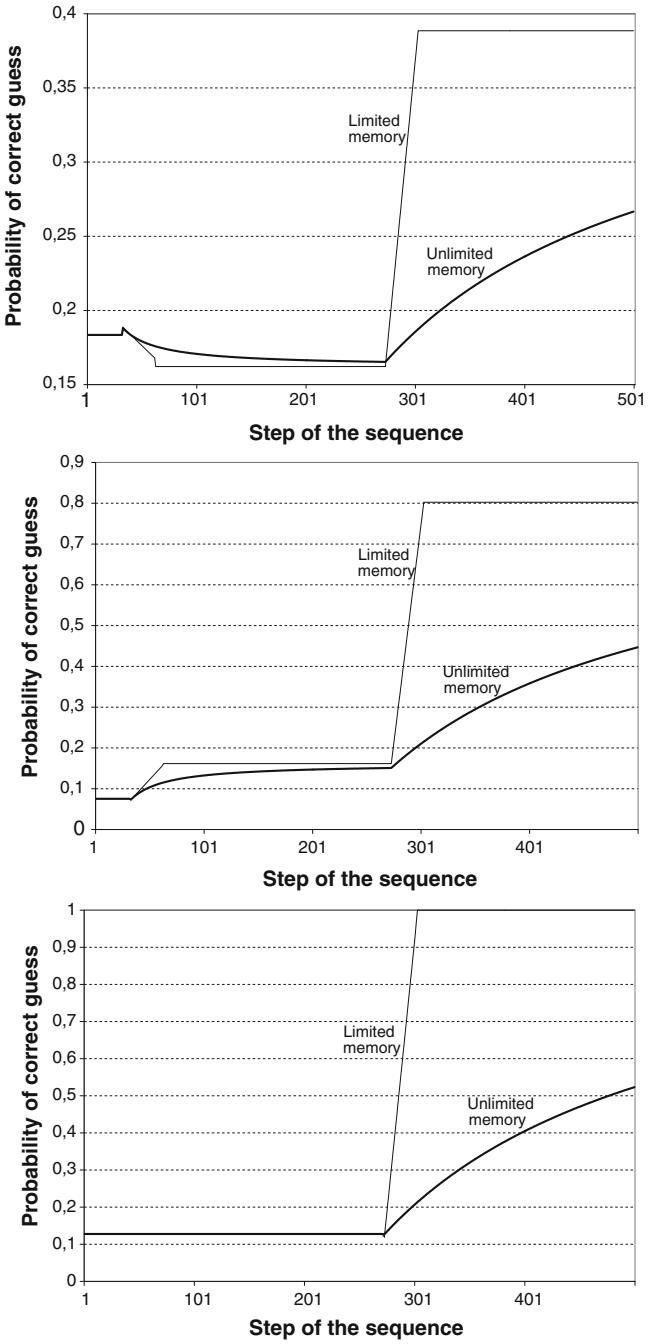

Fig. 9 Evolution of the probability that T successfully matches the action chosen by S if S chooses the action with highest payoff of each vector. S is using the optimal four-period dynamic strategy obtained for payoff vectors V_1 (*top*), V_{11} (*middle*), and V_{15} (*bottom*)

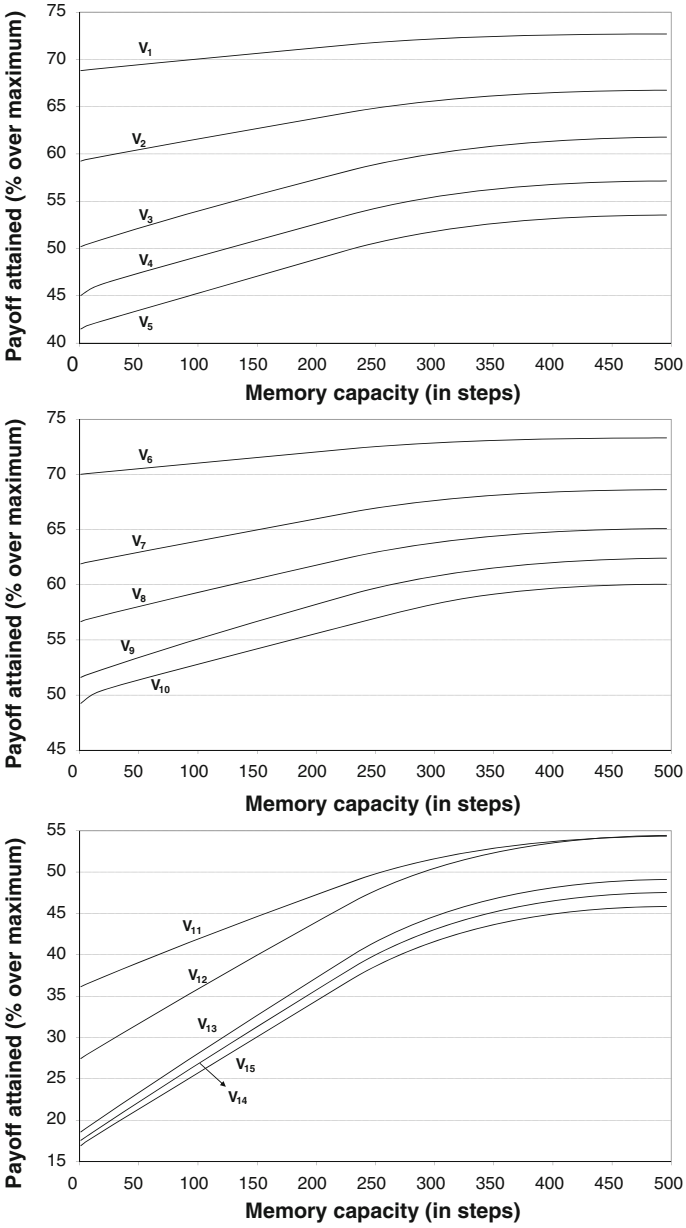

Fig. 10 Expected payoff for S as a function of the capacity of the observation memory of T for every payoff vector. Agent S is using the optimal four-period dynamic strategy originally designed under the assumption of unlimited adversarial memory

can be thought of as a *threshold*. Above this value, it is almost impossible for T to recover from the past observations and adapt to a new situation (strategy), which means that S has been successful in his deception.

On the influence of the number of periods We have done all the experiments so far with four-period dynamic strategies. The value of the number of periods originates a different optimization problem in each case, increasing the number of variables in $m + 1$ as the number of periods increases in one unit. For this reason, this value should not be too high to keep the size of the optimization problem in reasonable limits. Further, it should be in accordance with the number of encounters L that are going to take place, since too short periods have a low impact on the performance of the overall strategy, and the corresponding set of weights cannot modulate properly the overall behaviour due to lack of rounds.

We have sampled H from 2 to 30 by running the optimization process for each H value, for adversaries with memory capacity of 30, 100, 200 and 500 (unlimited memory) steps. The results are shown in Fig. 11 and prove that the impact of dynamic strategies is different depending on the adaptive capacity of the adversary, with a shorter memory representing a more adaptive opponent. In case of unlimited memory, the best performance is achieved with four periods, and increasing the number of periods does not give S a higher payoff (the corresponding line in the plots becomes horizontal very quickly). This is because unlimited memory allows for greater space for deception and thus it is enough to have less periods. In case the observation memory is limited, then it becomes more important to employ a strategy with a high number of periods, because the deception achieved when S switches from one period to another is quickly attenuated when oldest observations are forgotten. In other words, agent T can quickly adapt and learn the new behaviour. As a result, the effect of switching to a different set of weights becomes useless after a very short time, and therefore another change in the weights used by S is needed soon after the previous one. It can be seen in the plots that in all cases (except for 30-step memory capacity), a threshold on the number of periods exists that gives the maximum performance, and increasing the number of periods beyond the threshold does not clearly improve the payoff. This behaviour mirrors the results of unlimited memory, although the threshold value depends on the payoff vector and the memory capacity of the adversary. If the number of periods is high enough, then it is possible to exploit the limited memory to achieve a greater payoff than with unlimited memory, as shown in payoff vector V_{10}, but this also depends on the payoff vector of the problem.

5.2.2 Study of the competitive ratio against a clairvoyant adversary

An interest kind of analysis consists in comparing the payoff attained by S using our strategies with that attained in the mixed equilibrium situation, even though such equilibrium to mixed strategies cannot be played since T does not know S's payoff. The aim is to compute the *competitive ratio*, which was defined originally for online tasks, i.e. those consisting in satisfying a sequence of requests that arise one at a time, without knowledge of future requests [8,16]. It can be computed as the performance ratio of the algorithm that does not know in advance which demands will be received, divided by the payoff attained with an optimal, clairvoyant algorithm which knows in advance all the requests and their ordering.

Our imitation problem is not exactly an on-line task because there is nothing new at each repeated encounter that is unknown for any of the agents before that turn. However, it is true that agent T does not have full knowledge of the situation because he does not have access to S's payoffs. Therefore, we can compute the competitive ratio by considering the situation in which agent T knows S's payoffs, and thus he is able to compute and play the mixed strategy prescribed by mixed Nash equilibrium.

By definition of Nash equilibrium [13], in case T plays mixed Nash equilibrium, the best S can do is play his own Nash equilibrium too. Any other strategy would be worse for S.

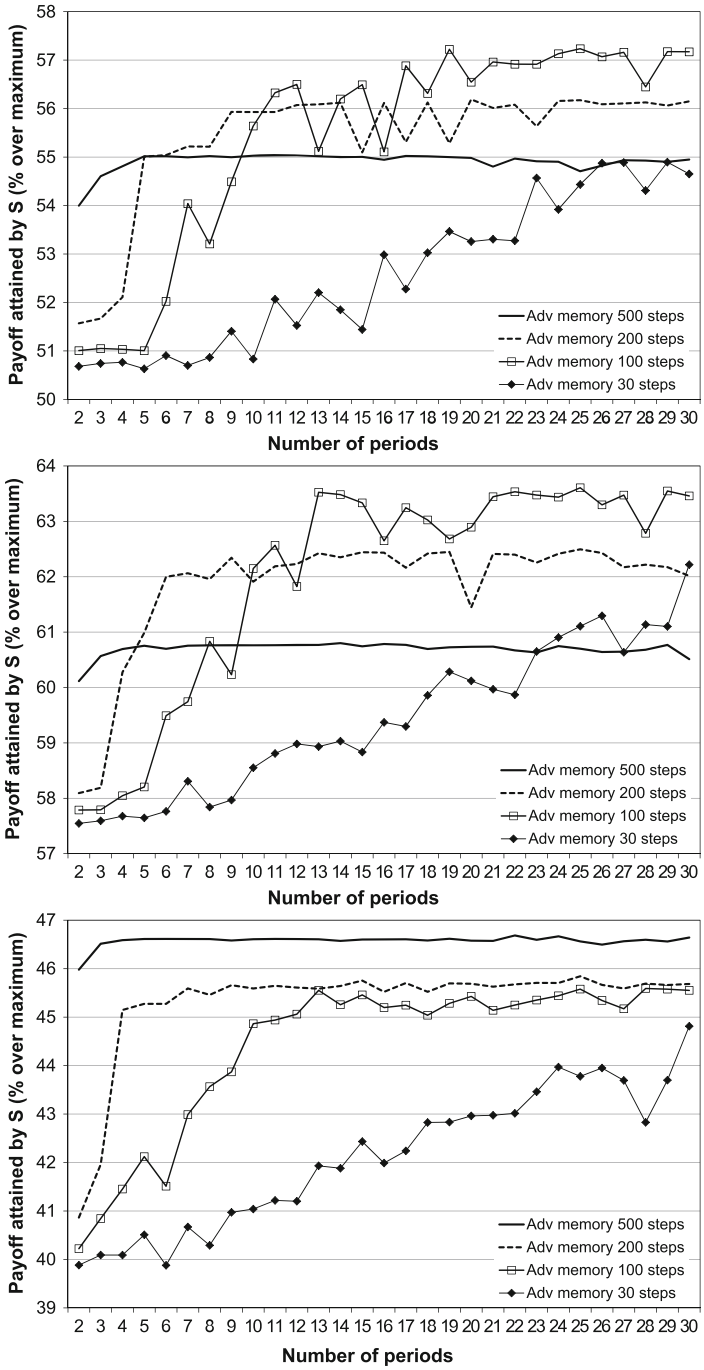

Fig. 11 Expected payoff for S for payoff vectors V_5 (*top*), V_{10} (*middle*) and V_{15} (*bottom*), as a function of the number of periods of the optimized dynamic strategy he employs. Strategies were found assuming the memory capacity of T was known before the optimization process starts.

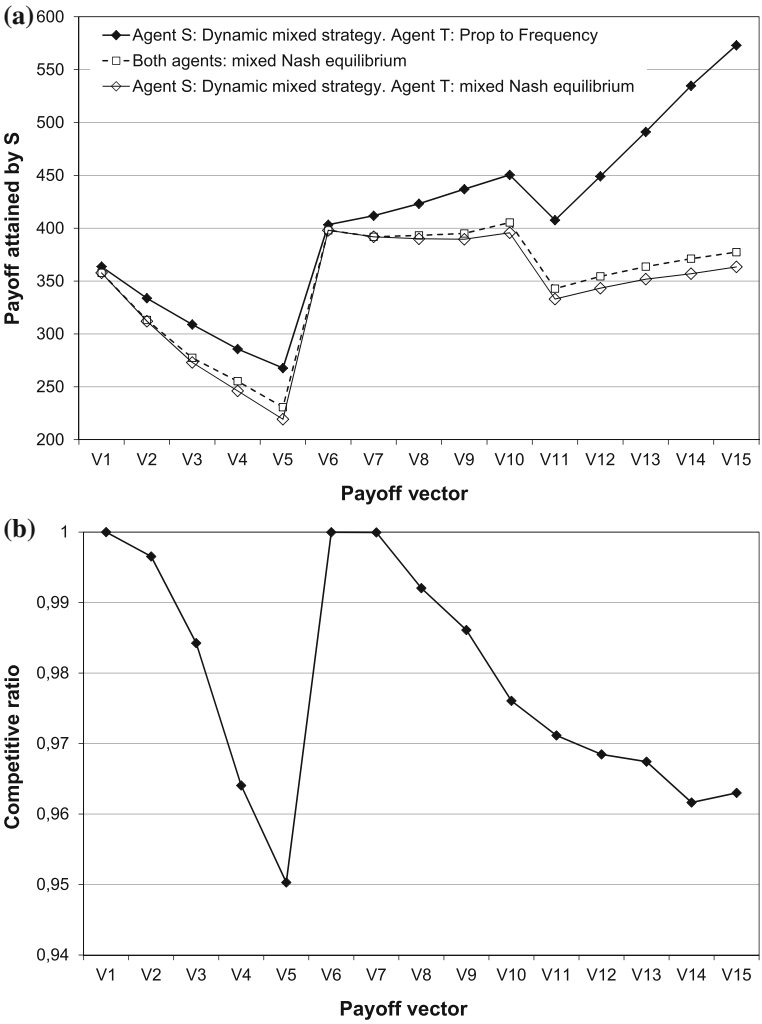

Fig. 12 Payoff attained by S when playing against different kind of adversaries (**a**) and competitive ratio in the case that S plays four-period dynamic strategies designed originally for an unlimited-memory adversary playing PF, and T is assumed clairvoyant and plays mixed Nash equilibrium (**b**). Note that **b** is the quotient of the two series at the bottom of **a**

Therefore, the interesting situation is to assess (a) the payoff attained by S when he uses a dynamic mixed strategy that was designed assuming PF for T and T plays the mixed Nash equilibrium strategy, with respect to (b) the payoff attained by S when both agents play Nash equilibrium (which is the most clairvoyant situation because both agents know the adversary's payoffs and also know that the adversary knows their own payoff, so they both have a motivation to play Nash equilibrium). As stated before, if T plays Nash equilibrium, we know for sure that S's payoff when playing any strategy different than the Nash equilibrium (in particular, when playing a dynamic mixed strategy) is smaller than the payoff when playing S's own Nash equilibrium. The aim is to assess how big this difference can be, in ratio.

Such experiment has been conducted assuming a four-period dynamic mixed strategy for S that was found by the optimization algorithm under the assumption that T was using PF. The strategy found was then played by S against an adversary using the mixed Nash equilibrium. The one-shot mixed Nash equilibrium strategy for T was computed using the Gambit software tool [3]. The results are depicted in Fig. 12. It can be seen in the figure that the competitive ratio, as defined in Fig. 12, is very close to one in all cases. This means that a dynamic mixed strategy designed under the assumption of and unlimited-memory PF adversary is not seriously damaged if the adversary actually plays the one-shot mixed Nash equilibrium at every encounter, instead of PF as he was assumed to play. Here, *not seriously damaged* refers to S's payoff in relation to the payoff S could have attained if he also plays Nash equilibrium instead of a four-period dynamic mixed strategy. Of course, any of these payoffs are notably lower than the payoff S could attain if T actually played PF (see the series at the top of Fig. 12a), but this situation is not the one considered for the competitive ratio.

6 Conclusions and further work

Static and dynamic mixed strategies for an agent in an adversarial model have been successfully designed using numerical optimization methods. Analytical expressions of the expected payoff for both strategies have been provided and validated also from an empirical point of view. Furthermore, optimal dynamic mixed strategies have shown to outperform optimal static mixed strategies in all the scenarios tested, specially when the difference between the payoff of the best action and the payoff of the rest of actions becomes greater.

However, such conclusions are valid only when we accept some assumptions about the behaviour of the adversary, agent T. These assumptions include that (a) the adversary has an unlimited observation memory (so an agent that knows he is being watched can manipulate the observer agent with his own behaviour), and (b) T uses a strategy proportional to the observed frequency to make his predictions, which allows S to previously take this into account when designing his strategies. If these conditions are not met, i.e. our model of the opponent is not accurate, then the strategies previously designed are not optimal any more, i.e. if T has a limited observation memory, then the manipulation that S had planned has no effect, thus leading to important (sometimes dramatic) losses in the payoff attained by S. Again, theoretical expressions of the expected payoff were provided and successfully contrasted with empirical simulations for this new situation. When T simply guesses with the last action observed (one-step memory), the results were very bad for S because in a mixed strategy a player often repeats probabilistically the same action in consecutive turns. Notice that one easy way to avoid this behaviour is to use the so-called *Random among K best actions* strategy, which was one of the very first strategies presented in [15,18].

Since game theory in general usually relies on assumptions such as perfect rationality of all the players (all are supposed to be *best-responsers*) or perfect knowledge of the utility functions (which does not hold here because T does not know S's payoffs, which is the key aspect of our model), it is clear that new robust approaches are required to solve such adversarial scenarios. Therefore, further work may include using game-theory related models to study what happens when the assumptions about the adversary are violated, in a similar way as the present work. Such studies represent the starting point to enhance the application of these models in real-world contexts [2,4,6]. In addition, extensions of the model can be

[3] It is freely available at www.gambit-project.org.

introduced, such as a generalization in which the payoffs of the game change at every step with a correlation between an action chosen by S and the next game to be played (this is formally called a *stochastic game*). This means that the decisions not only should take into account the immediate payoff of current round but also if the payoffs of the next game to be played are high, as a function of the current choice. Intuitively, one can think of each game to be played as an external stimulus that both agents can perceive, and which requires a response (be it a movement, a complete plan, or anything else) from them. Experimentation with more sophisticated (learning) techniques for T to make accurate predictions could be interesting as well, specially those concerning convergence to equilibrium without knowing the adversary's payoff (uncoupled dynamics). Finally, the application of the model to real situations is of great interest and is currently under research.

Acknowledgments This work was supported in part by the projects TIN2008-01948, TIN2008-06872-C04-04 and TIN2011-27696-C02-01 from the Spanish Ministry of Science and Innovation, P07-TIC-02970 and P11-TIC-8001 from the Andalusian Government. The first author also acknowledges the funding of a FPU scholarship from the Spanish Ministry of Education. Finally we want to thank the anonymous referees for their helpful comments and suggestions that have contributed to the improvement of the paper.

References

1. Amigoni, F., Basilico, N., & Gatti, N. (2009). Finding the optimal strategies for robotic patrolling with adversaries in topologically-represented environments. In *Proceedings of the 26th international conference on robotics and automation (ICRA 2009)* (pp. 819–824). Kobe: IEEE Press.
2. Amigoni, F., Basilico, N., Gatti, N., Saporiti, A., & Troiani, S. (2010). Moving game theoretical patrolling strategies from theory to practice: An usarsim simulation. In *Proceedings of the 27th international conference on robotics and automation (ICRA 2010)* (pp. 426–431). Barcelona: ICRA.
3. Amigoni, F., Gatti, N., & Ippedico, A. (2008). A game-theoretic approach to determining efficient patrolling strategies for mobile robots. In *Proceedings of the international conference on web intelligence and intelligent agent technology (IAT 2008)* (pp. 500–503). Sydney: IAT.
4. Basilico, N., Gatti, N., Rossi, T., Ceppi, S., & Amigoni, F. (2009). Extending algorithms for mobile robot patrolling in the presence of adversaries to more realistic settings. In *Proceedings of the international conference on web intelligence and intelligent agent technology (IAT 2009)* (pp. 557–564). Milan: IAT.
5. Conitzer, V., & Sandholm, T. (2006) Computing the optimal strategy to commit to. In *Proceedings of the 7th ACM conference on electronic commerce, EC '06* (pp. 82–90). Ann Arbor: IEEE Computer Society.
6. Korzhyk, D., Conitzer, V., & Parr, R. (2011). Solving stackelberg games with uncertain observability. In *Proceedings of the 10th international conference on autonomous agents and multiagent systems (AAMAS 2011)* (pp. 1013–1020). Taipei: AAMAS.
7. Kott, A., & McEneany, W. M. (2007). *Adversarial reasoning: Computational approaches to reading the opponents mind*. Boca Raton: Chapman and Hall/CRC.
8. Manasse, M. S., McGeoch, L. A., & Sleator, D. D. (1988). Competitive algorithms for on-line problems. In *Proceedings ACM symposium on theory of computing* (pp. 322–333). Chicago: ACM.
9. Markovitch, S., & Reger, R. (2005). Learning and exploiting relative weaknesses of opponent agents. *Autonomous Agents and Multi-Agent Systems, 10*, 103–130.
10. McLennan, A., & Tourky, R. (2006) From imitation games to kakutani. mimeo: Yale University.
11. McLennan, A., & Tourky, R. (2010). Imitation games and computation. *Games and Economic Behavior, 70*(1), 4–11.
12. McLennan, A., & Tourky, R. (2010). Simple complexity from imitation games. *Games and Economic Behavior, 68*(2), 683–688.
13. Osborne, M., & Rubinstein, A. (1994). *A course in game theory*. Cambridge: MIT Press.
14. Paruchuri, P., Pearce, J. P., & Kraus, S. (2008). Playing games for security: An efficient exact algorithm for solving bayesian stackelberg games. In *Proceedings of 7th international conference on autonomous agents and multiagent systems (AAMAS 2008)* (pp. 895–902). Estoril: AAMAS.
15. Pelta, D., & Yager, R. (2009). On the conflict between inducing confusion and attaining payoff in adversarial decision making. *Information Sciences, 179*, 33–40.

16. Sleator, D., & Tarjan, R. (1985). Amortized efficiency of list update and paging rules. *Communications of the ACM, 28*(2), 202–208.
17. Villacorta, P., & Pelta, D. (2011). Expected payoff analysis of dynamic mixed strategies in an adversarial domain. In *Proceedings of the 2011 IEEE symposium on intelligent agents (IA 2011). IEEE symposium series on computational intelligence* (pp. 116 – 122). Paris: SSCI.
18. Villacorta, P., & Pelta, D. (2012). Theoretical analysis of expected payoff in an adversarial domain. *Information Sciences, 186*(1), 93–104.
19. Zuckerman, I., Kraus, S., & Rosenschein, J. S. (2012). The adversarial activity model for bounded rational agents. *Journal of Autonomous Agents and Multi-Agent Systems, 24*(3), 374–409.