DE GRUYTER OPEN

Int. J. Appl. Math. Comput. Sci., 2015, Vol. 25, No. 3, 617–630
DOI: 10.1515/amcs-2015-0045

amcs

A REPEATED IMITATION MODEL WITH DEPENDENCE BETWEEN STAGES: DECISION STRATEGIES AND REWARDS

PABLO J. VILLACORTA [a,*], DAVID A. PELTA [a]

[a]Models of Decision and Optimization (MODO) Research Group
CITIC-UGR, Department of Computer Science and Artificial Intelligence
University of Granada, ETSIIT, C/Periodista Daniel Saucedo Aranda s/n, 18071 Granada, Spain
e-mail: {pjvi,dpelta}@decsai.ugr.es

Adversarial decision making is aimed at determining strategies to anticipate the behavior of an opponent trying to learn from our actions. One defense is to make decisions intended to confuse the opponent, although our rewards can be diminished. This idea has already been captured in an adversarial model introduced in a previous work, in which two agents separately issue responses to an unknown sequence of external inputs. Each agent's reward depends on the current input and the responses of both agents. In this contribution, (a) we extend the original model by establishing stochastic dependence between an agent's responses and the next input of the sequence, and (b) we study the design of time varying decision strategies for the extended model. The strategies obtained are compared against static strategies from theoretical and empirical points of view. The results show that time varying strategies outperform static ones.

Keywords: adversarial decision making, imitation, strategies, state dependence, reward.

1. Introduction

Adversarial reasoning is largely about understanding the minds and actions of one's opponent. It is relevant to a broad range of problems where the actors are actively and consciously contesting at least some of each others' objectives and actions (Kott and McEneany, 2007). The field is also known as decision making in the presence of adversaries or adversarial reasoning.

The threat of terrorism has fueled the investments and interest in the development of computational tools and techniques for adversarial reasoning, mostly oriented to homeland defense, but it is possible to envisage less dramatic applications in computer games, where the user is the adversary and the computer characters are provided with adversarial reasoning features in order to enhance the quality, difficulty and adaptivity of the game, which together improve the gaming experience. The development of intelligent training systems is also an interesting field.

More than twenty years ago, Thagard (1992) stated that in adversarial problem solving, one must anticipate, understand and counteract the actions of an opponent. Military strategies, business, and game playing all require an agent to construct a model of an opponent that includes the opponent's model of the agent.

1.1. Related work.

In the context of this contribution, adversarial decision making is modeled as an imitation game played by two agents, S and T, each of which chooses an action to respond to an external event, without knowing the choice of the other. Each of the agents receives a payoff that depends on the choices of both of them. When the same conflicting situation arises many times (i.e., repeated encounters), the decision making process becomes more difficult as the participants have the possibility to learn the other's strategy. The more frequently T guesses S's response, the smaller the payoff S receives. The goal of agent S is thus to maximize the total payoff attained after a sequence of encounters. Examples of this type of imitation games can be found in the military field, but also in problems of real-time strategy games, government vs. government conflicts, economic adversarial domains, team sports (e.g., RoboCup), competitions (e.g., poker), etc. (Kott and McEneany, 2007). Teaching and learning by imitation has been recently applied to develop automatic car driving

*Corresponding author

controllers in video-games (Cichosz and Pawełczak, 2014).

A key aspect of the aforementioned situation is the fact that agent T, who is trying to learn what action S will select at each moment given each of the external events, does not know S's motivations to choose each of the actions available. In other words, T does not have access to the payoff attained by S for each possible outcome. Hence, the only information T can use to predict S's response consists of the observations collected about S's behavior in the past. Contrarily, S is aware of the existence of an adversary aimed at learning its behaviour, so it knows T will receive a payoff only when its prediction matches S's action. Therefore, the information is clearly asymmetric in this problem.

Existing works on imitation games (McLennan and Tourky, 2006; 2010a; 2010b) usually focus on theoretical aspects of these games with complete information, such as complexity issues (imitation games are proved to be as complex as any general bi-personal game) and the equivalence of the computation of optimal solutions to the game, i.e., Nash equilibria, with other problems such as proving Kakutani's fixed-point theorem (McLennan and Tourky, 2006). However, none of these works take into account incomplete information based merely on observations, like we will do in our model.

1.2. Game-theoretical considerations. In the model outlined above, we analyze decision making strategies for S in a context of repeated encounters against T. One defense is to make decisions that are intended to confuse T, although S's rewards can be diminished. However, this situation is of upmost interest in the case of adversarial reasoning as what agent S wants is to make its behavior as uncertain or unpredictable as possible. In other words, S wants to force the presence of uncertainty in order to confuse the adversary while its payoff is as less affected as possible. Essentially, this is how S can defend against an opponent who is trying to learn S's decision rules.

Game theory is generally perceived as a natural choice to deal with adversarial reasoning problems, and is particularly appealing to model direct competitive interactions between two agents, like the one described above. Randomized strategies play a fundamental role in game theory and have been extensively studied. A brief survey of techniques where the combination of game theory with other approaches is highlighted, jointly with probabilistic risk analysis and stochastic games, is presented by Kott and McEneany (2007). Other direct examples that demonstrate in which sense adversarial reasoning in general and game theory in particular have been successfully used in real problems are the so-called security games (Tambe, 2012; Paruchuri *et al.*, 2008; Amigoni *et al.*, 2009). These models are aimed at designing randomized strategies to defend an area (sometimes an airport, a perimeter and the like) against an adversary who first studies and learns the defender's strategy. Such models bear a clear resemblance to the adversarial reasoning model described in this contribution.

The type of interaction we are modeling has some characteristics that make it different from usual games. First, one of the agents does not know the payoff attained by the adversary for each possible outcome. This makes the traditional equilibrium to mixed strategies (Osborne and Rubinstein, 1994) unfeasible since agent T cannot compute its equilibrium strategy. Second, since the interaction is repeated, both agents are expected to use a randomized strategy to keep the adversary guessing, and avoid easy learning of its actions. One might be tempted to analyze the situation as a leader–follower game, where S would play the role of the leader due to the strategic advantage of having more information about the payoffs than the opponent, and T would be the follower. However, this approach is not appropriate because S does not explicitly announce its strategy to T, and therefore there is no explicit commitment like in true leader–follower situations (Conitzer and Sandholm, 2006). The only information T has about S's strategy consists of empirical observations collected in the past, and they may be (intentionally) deceptive. Assuming T is aware of this, it would be unsafe for it to interpret them as if they were reflecting S's strategy. In other words, the commitment of S is not reliable, and thus it is not a true leader–follower situation because the follower T will not trust the leader's pseudo-commitment (Conitzer and Sandholm, 2006). More details on game-theoretic considerations can be found in the work of Villacorta *et al.* (2013).

1.3. Aims and contribution. In a previous work (Pelta and Yager, 2009), a model to study the balance between the level of confusion induced and the payoff obtained was proposed. The main conclusion of such a study was that one way to produce uncertainty is through decision strategies for S that contain a certain amount of randomness. Here we focus on how to design decision strategies that can be used by S in situations of repeated conflicting encounters. Villacorta and Pelta (2012) show how theoretical expressions can predict the expected payoff attained by one of the agents when using certain simple strategies, while Villacorta *et al.* (2013) analyze a situation that is simpler than the one presented here. The main difference with the model we will develop here is that, in the study of Villacorta *et al.* (2013), the outcome of an encounter does not affect the payoff attainable at the next encounter, thus the model did not consider the existence of different kinds of events.

The aim of this paper is twofold. First, we present an extension to the original model in which statistical

dependence is introduced between the action taken by agent S and the next event to arise. This is an important issue the agents should take into account before making a decision, because the action selected will have influence over the next event to arise at the next encounter, and such an event determines the maximum payoff attainable when the agents choose their actions. Second, decision strategies for agent S that are not constant along time but change at certain time steps in the iterated process are proposed for this extended model. More specifically, we tackle strategy design as a constrained non-linear optimization problem whose solution gives both the exact moment at which agent S must switch its strategy, and which strategy it must use. The work departs from the results shown by Villacorta and Pelta (2011) as well as Villacorta *et al.* (2013), who developed such an idea on a model with no statistical dependence between the events and the actions.

In connection with the aforementioned objectives, the experiments we will conduct are aimed at answering the following questions:

1. Is there any substantial difference between the theoretical expected payoff and the average payoff attained by empirical simulations in the extended model?
2. Do dynamic mixed strategies outperform a static mixed strategy in terms of the expected payoff in the extended model?
3. How are dynamic strategies affected by the number of different periods employed?

The remainder of the work is organized as follows. Section 2 describes the main characteristics and components of the model used, including the novel mechanism of statistical dependence between an action and the next input. Section 3 deals with the need for randomized strategies, both static and dynamic. The analytical expression of the expected payoff attained by S when using both the kinds of strategies is given and explained in detail, and the need for an optimization process for determining the best parameters in this expression is motivated. In Section 4 we describe the computational experiments performed and the results obtained. Finally, conclusions and further work are discussed in Section 5.

2. Adversarial model

We will first introduce the original adversarial model, and after that we will extend it by adding a statistical action–event dependence. The model we are dealing with was presented by Pelta and Yager (2009) and consists of two agents S and T (the adversary or imitator), a set of possible inputs or events $E = \{e_1, e_2, \ldots, e_n\}$ issued

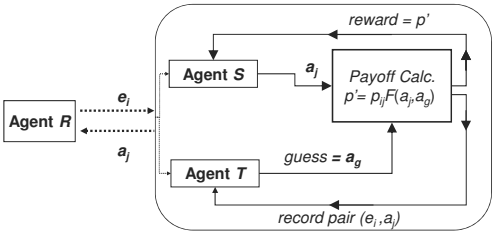


Fig. 1. Graphical representation of the model.

by the external environment (represented as a third agent R), and a set of potential responses or actions $A = \{a_1, a_2, \ldots, a_m\}$ that can be chosen as a response to an event (see Fig. 1). The payoff function for agent S can be expressed as a matrix P:

$$P(n \times m) = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1m} \\ p_{21} & p_{22} & \cdots & p_{2m} \\ p_{31} & p_{32} & \cdots & p_{3m} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n1} & p_{n2} & \cdots & p_{nm} \end{pmatrix},$$

where p_{ij} is the reward attained by S when choosing action a_j to respond to event e_i. This matrix is only known by S.

The agents repeatedly engage in a conflict situation. At each encounter, the agents issue responses to an external event e_i which is stochastically generated as explained in Section 2.1. Agent S chooses an action a_j as a response to this event, taking into account the payoff matrix P. Its aim is to maximize the accumulated reward after a sequence of encounters. At the same time, agent T tries to guess the action S will choose, possibly using information about S's past responses to the same event. As a result of the agents' decisions, a payoff is assigned to them, as follows. If T's guess a_g matches S's action, then T receives 1 and S receives 0, i.e., S gets no reward each time T successfully guesses its choice. Otherwise, T gets 0 and S gets p_{ij}. This can be expressed as

$$p' = p_{ij} \cdot F(a_g, a_j), \tag{1}$$

where

$$F(a, b) = \begin{cases} 0 & \text{if } a = b, \\ 1 & \text{otherwise.} \end{cases}$$

Note that the above payoff scheme entails that no partial similarity between the actions is allowed: T's prediction either matches S's action (and T gets 1 point) or not (and T gets 0 points), but no partial matching is possible.

After each encounter, agent T is informed of what S had chosen. In our model, we assume T stores this information in an observation matrix $O_{n \times m}$, where o_{ij} stands for the number of times that, in the past, agent S has selected action a_j as a response to event e_i. T records

Algorithm 1. Sequence of steps in the model.

for $l = 1$ to L **do**
  A new input e_i arises.
  Agent T "guesses" an action a_g, applying some decision strategy
  Agent S determines an action a_j, applying its own decision strategy
  Calculate the payoff for S as indicated in Eqn. (1)
  Agent T records the pair e_i, a_j
end for

a new pair (e_i, a_j) by increasing the entry o_{ij} by 1. As mentioned before, this information can be used by T for future predictions. Algorithm 1 describes the steps of the model, L being the length of the sequence of inputs, which we assume known in advance by both agents.

2.1. Statistical dependence between events. In the original model (Pelta and Yager, 2009) the events coming from the environment were *independent* and *randomly generated* following a *uniform* probability distribution. Independence means there is no relation between the current event, the current response and the next event. Uniformly generated events means that, every time an event is going to be generated, all of them are equally probable. Notice that considering other probability distributions different from the uniform one would not change the methodology of the existing works about this model (Villacorta and Pelta, 2012; 2011; Villacorta *et al.*, 2013). It just would require to repeat the experiments and adjust the results, but the conclusions of such works do not depend on the probability distribution employed because, in the model considered so far, the agents' decisions have no influence on the forthcoming events, and therefore the agents do not need to include any information about the event probabilities in their decision strategies. Up to now, the probability distribution of the events has been considered an external, inalterable constraint.

However, extending the model by introducing statistical dependence between the events of consecutive stages is a different matter. We can think of several kinds of dependence. The first one is the dependence between the event of one stage and that of the next stage. The question is, if this information is useful for an agent in any way. In other words: before giving a response to the current event, is it relevant for the decision maker to know (in a probabilistic sense) which event will arise next? The answer is no, because every time an event arises the agent should try its best, disregarding the next event since it will not be affected by the current decision. With this kind of dependence, we would be in the same setting mentioned above, i.e., it would be equivalent to considering no dependence at all from the agents' point of view.

The second type of dependence is that between the action taken in the current stage by one or both agents and the next event to arise. That is why an arrow labeled a_j (action taken by S) goes from the agents back into the environment (agent R) in Fig. 1. This setting will be analyzed in the remainder of our study as it has interesting implications. The most important is that, before making a decision for the current event, an agent should consider that its choice will affect not only the immediate payoff of the current stage but also the maximum payoff attainable in the next stage, since not all the events offer the same range of payoffs. Such a consideration captures the fact that some events may be rare or *critical*, so they provide very high payoffs when the response is chosen properly, while some others are less important, so they provide very low payoffs regardless of the action chosen. With this in mind, now a *good* action is not just the one that provides a high payoff for the current event (immediate payoff) but also causes events with very high rewards[1] to be more likely to arise at the next stage.

In our model, we introduce dependence only between the action chosen by agent S and the next event to arise. We assume that the information of such stochastic dependence is available only to agent S, in the form of a *conditional probability matrix* C with dimensions $m \times n$ shown below:

$$C(m \times n) = \begin{pmatrix} P[X = e_1|Y = a_1] & \dots & P[X = e_n|Y = a_1] \\ P[X = e_1|Y = a_2] & \dots & P[X = e_n|Y = a_2] \\ \vdots & \ddots & \vdots \\ P[X = e_1|Y = a_m] & \dots & P[X = e_n|Y = a_m] \end{pmatrix}.$$

The value C_{ij} stands for the conditional probability $P[X = e_j|Y = a_i]$, so $\sum_j P[X = e_j|Y = a_i] = 1$ for every row $i = 1, \dots, m$. In this expression, X is the discrete random variable representing the next event arising, and Y is the discrete random variable representing the current action taken by agent S that, as will be explained, is based on a randomized behavior rule. Finally, let $(\pi_1, \dots, \pi_n)$ be the probabilities of each event to arise at the first step of the simulation. We cannot give conditional probabilities in this case as there is no previous action. For our experiments, we will take such probabilities as uniform, $\pi_i = 1/n$ for $i = 1, \dots, n$.

3. Behavior of the agents

In the next subsections, we provide alternatives for modeling the behavior of both agents.

3.1. Strategies for agent T. Different strategies for T were proposed by Pelta and Yager (2009). One possible

[1] Those for which the values of the corresponding row of the payoff matrix are all large.

strategy is the so-called most frequent (MF), where agent T selects the action a_g that has been most frequently used by S in the past when the event was e_i. This strategy is clearly unsafe because S could exploit it by avoiding consecutively selecting the same action twice.

Another strategy for T performing reasonably well is called proportional to frequency (PF): the probability of selecting an action a_j as a prediction to event e_i is proportional to O_{ij} (the observed frequency from agent S) (Pelta and Yager, 2009). This strategy is randomized so it keeps S guessing during all the iterations. Although more sophisticated strategies for T may be investigated in the future, we will assume in the remainder of the work that T uses this strategy.

From now on, we will focus our discussion on the behavior of agent S.

3.2. Static mixed strategy for agent S. Agent S could use a totally deterministic strategy that always selects the action with the highest payoff as a response to the current stimulus e_i. However, this would be very easy to learn for agent T, who would quickly predict this behavior correctly after a low number of games. 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 a low the payoff) may be selected with the same probability that best actions.

A need exists here to get for a good balance between confusion and payoff, as concluded by Pelta and Yager (2009). Our objective now is to calculate the expected payoff of new strategies.

We now borrow some concepts from game theory to explain how the agents behave. A randomization over the existing actions (responses) is called a *mixed strategy*. It is a set of weights representing a probability distribution over the actions (i.e., the sum of the weights is 1). When a player has to choose an action, it uses this probability distribution to make its decision. 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. To be precise, we will consider that an agent uses a specific mixed strategy $(\alpha_{i1}, \ldots, \alpha_{im})$ for each event e_i, so it maintains n different strategies and uses one of them according to the event e_i for which it must issue a response.

With these weights, it is possible to calculate the so-called expected payoff for agent S, which is the sum of all the possible outcomes of the game weighed by the probability that each outcome eventually occurs. In the adversarial model we are dealing with, this means that we can weigh each payoff of the payoff matrix P by the probability that agent S eventually gets that payoff. This probability can be computed as the product of the probabilities of several independent events happening simultaneously. Agent S will attain payoff p_{ij} if three conditions hold:

(i) Event e_i must arise. We will refer to this probability as $P[I = e_i]$. In this case, I is the discrete random variable representing the occurrence of each event at the *current* stage.

(ii) Assuming that event e_i has arisen, agent S must select action a_j as a response. We will note this probability α_{ij} but can be formally written as $P[Y = a_j | I = e_i]$.

(iii) Finally, S will only get the score p_{ij} if agent T does not successfully predict its response.

The probabilities involved in condition (ii) constitute the mixed strategy we are searching for. The calculation of the probability of conditions (i) and (iii) is described below.

Condition (i): Marginal probability of an event. first of all, recall that we had previously defined another random variable X that represents the occurrence of each event at the *next* stage. When the simulation advances one step, $P[I = e_i]$ adopts the value of $P[X = e_i]$ computed in the preceding step, for $i = 1, \ldots, n$. Notice that we are not given the probability distribution of I, so the values $P[I = e_i]$ have to be computed using the known conditional probabilites of matrix C and the mixed strategy of S mentioned in (ii).

Actually the only data we have about the marginal probabilities $P[I = e_i]$ of the occurrence of the external events are those of the *first* step of the game, $P_1[I = e_i] = \pi_i$. Recall that the decisions of agent S influence the marginal probabilites, and such decisions are modeled by the mixed strategy $(\alpha_{ij}, j = 1, \ldots, m)$ it employs for each event e_i. If we apply the *theorem of total probability* to the first step, we have

$$\begin{aligned} P_1[Y = a_j] &= \sum_i P[Y = a_j | I = e_i] P_1[I = e_i] \\ &= \sum_i \alpha_{ij} \pi_i, \quad j = 1, \ldots, m. \end{aligned} \tag{2}$$

The subscript of P indicates the step to which it is referred. Once those values are known, they can be substituted in the following expression, which results from applying again the theorem of total probability since $[Y = a_j], j = 1, \ldots, m$ constitute another partition of the sample space:

$$\begin{aligned} P_1[X = e_i] &= \sum_j P[X = e_i | Y = a_j] P_1[Y = a_j] \\ &= \sum_j C_{ji} P_1[Y = a_j], \quad i = 1, \ldots, n. \end{aligned} \tag{3}$$

Now, the values $P_1[X = e_i], \quad i = 1, \ldots, n$, can be taken as the values $P_2[I = e_i]$. The probability that an event arises at the next step when the *current* step is number 1 is equivalent to the probability that an event arises at the current step if the current step is number 2. Then, since the values $P_2[I = e_i], \quad i = 1, \ldots, n$, are known, they can be used to compute the same probabilities at Step 2, applying the same expressions indicated above. In general, the following equations can be applied recursively for $k \geq 2$:

$$\begin{aligned} P_k[Y = a_j] &= \sum_i P[Y = a_j | I = e_i] P_k[I = e_i] \qquad (4) \\ &= \sum_i \alpha_{ij} P_{k-1}[X = e_i], \quad j = 1, \ldots, m, \\ P_k[X = e_i] &= \sum_j P[X = e_i | Y = a_j] P_k[Y = a_j] \\ &= \sum_j C_{ji} P_k[Y = a_j], \quad i = 1, \ldots, n, \quad (5) \\ P_k[I = e_i] &= P_{k-1}[X = e_i]. \end{aligned}$$

Condition (iii): Probability of not being guessed. Assume agent S is using a mixed strategy so that it employs α_{ij} to select the action a_j with payoff p_{ij}. Assume that agent T uses strategy PF, and let us suppose that a certain event e_i has arisen L_i times during the repeated game. Then action a_j will have been selected $L_i \cdot \alpha_{ij}$ times, and this number is what agent T has recorded in O_{ij}. The probability that T selects action a_j as a prediction using PF is then

$$P_{\text{guess}_{ij}} = \frac{O_{ij}}{\sum_{j=1}^{m} O_{ij}} = \frac{L_i \cdot \alpha_{ij}}{L_i} = \alpha_{ij}, \qquad (6)$$

with m being the number of actions available. Actually the value L_i is unknown as it depends on the sequence of decisions made by S, but in the last expression it is simplified and eventually disappears. Therefore the probability of not being guessed correctly is $1 - P_{\text{guess}} = 1 - \alpha_{ij}$.

Expected payoff with static mixed strategies. Using the probabilities of the three conditions described above, the expected payoff for agent S after a sequence of L inputs when it uses weights $\alpha = (\alpha_{ij})$ to select its actions has the following expression:

$$EP_{\text{static}}(\alpha) = \sum_{k=1}^{L} \sum_{i=1}^{n} P_k[I = e_i] \sum_{j=1}^{m} \alpha_{ij}(1 - \alpha_{ij}) p_{ij}. \qquad (7)$$

Notice that the probability that each event arises evolves along time because it also depends on the choices made by agent S.

Optimal strategy maximizing the expected payoff. If we want to maximize the expected payoff, we have to maximize expression (7) by computing the values of the optimal weights α_{ij}. This leads to an optimization problem with $m \times n$ variables since agent S uses a different mixed strategy for each event (each strategy having m variables) and there are n different events. It can be expressed as

$$\max_{\{\alpha_{ij}\}} \left\{ \sum_{k=1}^{L} \sum_{i=1}^{n} \left(P_k[I = e_i] \sum_{j=1}^{m} \alpha_{ij}(1 - \alpha_{ij}) p_{ij} \right) \right\} \qquad (8)$$

subject to

$$\begin{aligned} \sum_{j=1}^{m} \alpha_{ij} &= 1, \quad i = 1, \ldots, n, \\ \alpha_{ij} \geq 0, \quad i &= 1, \ldots, n, \quad j = 1, \ldots, m. \qquad (9) \end{aligned}$$

Recall that agent S, which is interested in solving the problem to get the maximum reward, must know in advance the number of repetitions or steps L that the game will have, and should be aware that T will use PF as well. If the value of L is unknown, the agent will have to estimate it somehow. The optimization problem should include all the terms of expression (7) that depend on the values α_{ij}. Since the probabilites P_k also depend on them, they must be part of the target function to be maximized.

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 was used during the game. We now propose changing these weights along time. Although this concept was introduced by Villacorta and Pelta (2011) as well as Villacorta *et al.* (2013), the application to the extended model requires the calculation of new expressions to predict the payoff attained by S.

First, we define a *period* as a series of consecutive inputs for which S will use the same weights to answer. The static mixed strategy described above can be viewed as a 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 events during which agent S will use the same mixed strategy. In order to gain flexibility and achieve a higher reward, we can define a different number of periods of different length for each event. We will call N_i^h the length of the h-th period of event e_i.

The next example illustrates this concept. Suppose that we have an input sequence of length L and that a given event e_i is expected to arise $L_i = 100$ times along the sequence. Then, we can define, for instance, 4 periods of lengths $N_i^1 = 30$, $N_i^2 = 10$, $N_i^3 = 20$ and $N_i^4 = 40$.

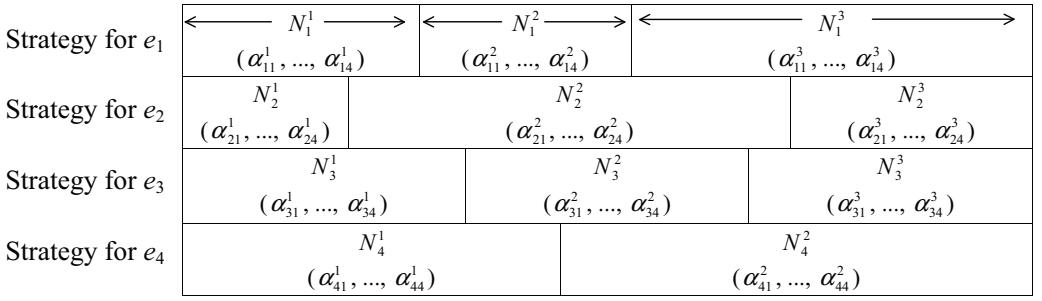


Fig. 2. Example of different periods for each event in a model instance with 4 different events. The letters inside each rectangle represent the length of that period and the mixed strategy to be used in it.

For a given period, the set of optimal weights is different from that of other periods because the distribution of the payoffs in a row of the payoff matrix may differ a lot from other rows. Figure 2 shows another example of different dynamic mixed strategies for each event, with a different number of periods and/or different moments of change. The length of the whole input sequence is $L = 1000$. Suppose there exist $n = 4$ different kinds of inputs in our model and $m = 4$ different actions, so a mixed strategy is a vector of 4 weights. If the inputs are uniformly distributed, then each event is expected to arise about 250 times, so the sum of the lengths of the periods for any event should be 250.

In order to calculate the best randomization under this scenario, we need to obtain the expression of the expected payoff for a dynamic strategy. The basics to obtain the expected payoff are again premises (i), (ii) and (iii) stated in Section 3.2, as explained next. Some new issues have to be taken into account to address the dynamic situation with statistical dependence.

Estimation of the number of occurrences of an event. Before computing the probabilities of conditions (i) and (iii), recall that in the dynamic case the number L_i of occurrences of an event is unknown and depends on the sequence of decisions made by S. We are able to compute only the marginal probability $P_k[I = e_i]$ of each event at step k. However, that information, together with the total number of steps L of the game, is enough to estimate L_i, since $L_i = \sum_{k=1}^{L} P_k[I = e_i]$. In general, let L_i^k be the estimated number of times event e_i has arisen after k steps of the game, which can be computed as

$$L_i^k = \sum_{t=1}^{k} P_t[I = e_i]. \tag{10}$$

Since this is an estimation based on probabilities, it is expected to be a real (not integer) value. However, this number will be used as a summation limit in the next section, so in that case it has to be rounded to the nearest integer.

Condition (i): Marginal probability of an event. From now, we will focus only on one single event e_i. Let $(\alpha_{ij}^h)_{j=1,...,m}$ be the set of weights agent S uses to choose an action as a response to an input of type e_i during the h-th period. Then, within a given period, α_{ij}^h represents the probability that S selects action a_j when event e_i has already arisen, also expressed as $P^h[Y = a_j | I = e_i]$. The novel part is that the values $P^h[Y = a_j | I = e_i]$ are different from one period h_1 to another h_2, but the calculation method is the same as described in Eqns. (4) and (5). In this case, we must be careful to substitute the adequate values of $P^h[Y = a_j | I = e_i]$ according to the period h to which step k (of the $P_k[Y = a_j]$ being computed) belongs. In a more formal way, Eqn. (4) can be rewritten to consider a distinct mixed strategy for each period as follows:

$$\begin{aligned} P_k[Y = a_j] &= \sum_i P^{H_i(k)}[Y = a_j | I = e_i] P_k[I = e_i] \\ &= \sum_i \alpha_{ij}^{H_i(k)} P_{k-1}[X = e_i], \quad j = 1, \ldots, m. \end{aligned} \tag{11}$$

Function H_i maps an absolute step $k : 1 \leq k \leq L$ to the period h whose set of weights $(\alpha_{ij}^h)_{j=1,...,m}$ must be used to issue a response within the dynamic strategy for event e_i. It is based on the estimation of the number of times that e_i has arisen after $k-1$ steps as follows:

$$H_i(k) = \begin{cases} 1 \text{ if } L_i^{k-1} < N_i^1, \\ 2 \text{ if } N_i^1 \leq L_i^{k-1} < N_i^1 + N_i^2, \\ 3 \text{ if } N_i^1 + N_i^2 \leq L_i^{k-1} < N_i^1 + N_i^2 + N_i^3, \\ \vdots \end{cases}$$

The above changes only affect the computation of the estimated number of times each event arises along the simulation, see (10).

Condition (iii): Probability of not being guessed. This is the only part from our previous results (Villacorta and Pelta, 2011; Villacorta *et al.*, 2013) that remains

unchanged so the explanations of this section can also be found in those works. The difficult part of the expression we need involves the computation of the probability of not being guessed. After the first period of length, say, N_i^1, the observation matrix O has the following values in row i (representing absolute frequencies of the responses given in the past by S to inputs of type e_i):

$$T(n \times m) = \begin{pmatrix} \cdots\cdots\cdots\cdots\cdots\cdots \\ N_i^1 \cdot \alpha_{i1}^1 \quad \ldots \quad N_i^1 \cdot \alpha_{im}^1 \\ \cdots\cdots\cdots\cdots\cdots\cdots \end{pmatrix}.$$

The probability of not being guessed P_{NG} during the first period is (1-α_{ij}^1), according to the same explanation as given in Section 3.2. This reasoning becomes more complicated when considering row i of the observation matrix at the end of the second period, whose length is N_i^2:

$$T(n \times m) = \begin{pmatrix} \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ N_i^1\alpha_{i1}^1 + N_i^2\alpha_{i1}^2 \quad \ldots \quad N_i^1\alpha_{im}^1 + N_i^2\alpha_{im}^2 \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \end{pmatrix}.$$

According to the values of the former matrix after 2 periods, the probability *at the end of the second period* that agent T selects action a_j as a prediction is

$$P_{\text{guess}_{ij}} = \frac{O_{ij}}{\sum_{j=1}^m O_{ij}} = \frac{N_i^1\alpha_{ij}^1 + N_i^2\alpha_{ij}^2}{N_i^1 + N_i^2},$$

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

$$P_{NG_{ij}} = 1 - P_{\text{guess}} = \frac{N_i^1(1-\alpha_{ij}^1) + N_i^2(1-\alpha_{ij}^2)}{N_i^1 + N_i^2}.$$

What happens in between, 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 $0 \leq s \leq N_i^2$, with N_i^2 being the length of the second period), the probability that T correctly predicts response j to event i is

$$P_{\text{guess}_{ij}} = \frac{O_{ij}}{\sum_{j=1}^m O_{ij}} = \frac{N_i^1\alpha_{ij}^1 + s\alpha_{ij}^2}{N_i^1 + s},$$

and the probability of not being guessed is then

$$P_{NG_{ij}} = 1 - P_{\text{guess}} = \frac{N_i^1(1-\alpha_{ij}^1) + s(1-\alpha_{ij}^2)}{N_i^1 + s}.$$

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 ($0 \leq s \leq N_i^h$):

$$P_{NG_{ij}} = \frac{\sum_{k=1}^{h-1} N_i^k(1-\alpha_{ij}^k) + s(1-\alpha_{ij}^h)}{\sum_{k=1}^{h-1} N_i^k + s}. \quad (12)$$

If we want to express such a probability as a function of the number t of times that event e_i has arisen, we can rewrite the above expression as follows:

$$P_{NG_{ij}}(t) = \frac{\sum_{k=1}^{U_i(t)-1} N_i^k(1-\alpha_{ij}^k) + Y_i(t)(1-\alpha_{ij}^{U_i(t)})}{t}, \quad (13)$$

$$U_i(t) = \begin{cases} 1 \text{ if } t < N_i^1, \\ 2 \text{ if } N_i^1 \leq t < N_i^1 + N_i^2, \\ 3 \text{ if } N_i^1 + N_i^2 \leq t < N_i^1 + N_i^2 + N_i^3, \\ \vdots \end{cases}$$

$$Y_i(t) = t - \sum_{k=1}^{U_i(t)-1} N_i^k.$$

Function U_i maps the number of occurrences of event e_i (including the present one) to the period whose weights should be used at the current stage, according to the dynamic strategy for event e_i. It is similar to the function H_i defined in the previous section, but its argument is not a step of the simulation but the number of times event e_i has arisen. It is basically a formalism, rather than a true mathematical function. Similarly, function Y_i computes the number of times that event e_i has arisen during the *current* period of the dynamic strategy, which is equivalent to the value of s in expression (12). In other words, it is the number of occurrences of e_i, but counted from the beginning of the current period.

Expected payoff with dynamic mixed strategies. The expression of the total expected payoff with dynamic strategies is shown below. It is a generalization of (7), using (13) as the probability of not being guessed. The marginal probabilities of each event $P_k[I = e_i]$ explained in the previous section do not appear explicitly in this expression, but recall that the estimates L_i depend on such probabilities:

$$EP_{dyn}((\alpha_{ij}^h), (N_i^h)) = \sum_{i=1}^{n} \sum_{k=1}^{\lfloor L_i \rfloor} \sum_{j=1}^{m} \alpha_{ij}^{U_i(k)} \cdot P_{NG_{ij}}(k) \cdot p_{ij}. \quad (14)$$

Optimal strategy maximizing the expected payoff. Recall that the values L_i ultimately depend on the values α_{ij}^h. If we want to maximize the expected payoff according to this expression, the optimization problem must contain all those unknown weights simultaneously. The number of periods H_i of a dynamic strategy is not part of the optimization process and must be set by the user. For simplicity, we will consider that number to be the same for all events, so $H_i = H$ for all $i = 1, \ldots, n$. The length of the periods for strategy i should equal the number of times that each event e_i is expected to arise, L_i: $\sum_{h=1}^{H_i} N_i^h = L_i, \quad i = 1, \ldots, n$. These should be additional constraints in the optimization process that will be carried out to determine the optimal values of the weights α_{ij}^h and the periods N_i^h that we are searching.

With this approach, the number of unknown parameters is greater than that of static mixed strategies. Instead of computing only $m \times n$ weights, we have to compute H sets of weights per event, and all these variables are related because they mutually influence the probabilities of the events that will arise, so the problem cannot be broken down into smaller optimization problems as proposed by Villacorta and Pelta (2011). In total, the problem being solved has $(n \times m \times H) + (n \times H)$ unknown variables. In this sum, the first product is the number of weights α_{ij}^h. A dynamic strategy has H periods, and there are m weights in each period. Since we allow more flexibility, S maintains a different strategy for each event so there are n independent dynamic strategies with $H \times m$ weights each. The second product represents the lengths of the periods, which are positive integers. Recall that the optimal length N_i^h of every period is being optimized, and there are H periods in each of the n dynamic strategies used by S. The problem is thus a non-linear mixed optimization one. The above summation yields 120 unknown real variables for a simple instance of our adversarial model with $m = 5$ actions, $n = 5$ events and dynamic strategies with $H = 4$ different periods, which means that the complexity of the problem makes it hard to solve using exact mathematical optimization methods. Formally, the optimization problem can be described as

$$\max_{\{\alpha_{ij}^h \cup N_i^h\}} \left\{ \sum_{i=1}^{n} \sum_{k=1}^{\lfloor L_i \rfloor} \sum_{j=1}^{m} \alpha_{ij}^{H_i(k)} \cdot P_{NG_{ij}}(k) \cdot p_{ij} \right\} \quad (15)$$

subject to

$$\sum_{j=1}^{m} \alpha_{ij}^h = 1, \quad i = 1, \ldots, n, \quad h = 1, \ldots, H,$$

$$\alpha_{ij}^h \geq 0, \quad i = 1, \ldots, n, \quad j = 1, \ldots, m, \quad h = 1, \ldots, H$$

$$\sum_{h=1}^{H} N_i^h = L_i, \quad i = 1, \ldots, n.$$

Recall that

$$L_i = \sum_{k=1}^{L} P_k[I = e_i], \quad i = 1, \ldots, n.$$

4. Experiments and results

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

1. Is there any substantial difference between the theoretical expected payoff and the average payoff attained by empirical simulations?
2. Do dynamic mixed strategies outperform a static mixed strategy in terms of the expected payoff?
3. How are dynamic strategies affected by the number of different periods employed?

In order to answer these questions, we follow the next steps.

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

- number of events and actions: $n = m = 5$,
- length of the input sequences: $L = 500$,
- matrix of conditional probabilities:

$$C = \begin{pmatrix} 0.2 & 0.5 & 0.15 & 0.1 & 0.05 \\ 0.4 & 0.1 & 0.25 & 0.05 & 0.2 \\ 0.15 & 0.2 & 0.4 & 0.1 & 0.15 \\ 0.1 & 0.1 & 0.2 & 0.5 & 0.1 \\ 0.3 & 0.4 & 0.3 & 0 & 0 \end{pmatrix}.$$

Payoff matrices. 15 different matrices were tested. For each matrix, a set of m payoffs is defined, and every row of the matrix has a permutation of the same set. The payoffs of every matrix are summarized in Table 1. The rest of the rows of each matrix are different permutations of the set displayed in the table. An extra column is shown, containing the maximum total payoff attainable by S after 500 events if it always chooses the action with the largest payoff and is never guessed. This is the ideal situation that would only occur when there is no adversary, so it is taken as a reference.

As can be seen in the table, the matrices are characterized by an increasing difference between the payoff attained when choosing the best action, the second-best action, and so on. When this difference is big, it becomes much more important to achieve a good balance between the payoff attained and the confusion caused, since choosing low payoff actions to induce confusion leads to a great loss in the payoff when compared with the greatest-payoff action.

Table 1. Set of payoffs associated with each payoff matrix.

Payoff matrix	First row					Max. reward after 500 ev.
M_1	1	0.9	0.95	0.8	0.85	500
M_2	0.8	0.9	0.6	0.7	1	500
M_3	1	0.85	0.7	0.4	0.55	500
M_4	1	0.6	0.8	0.4	0.2	500
M_5	0.25	0.01	0.5	1	0.75	500
M_6	1.1	0.95	0.9	1.05	1	550
M_7	1.2	1	1.1	0.9	0.8	600
M_8	1.3	1	1.15	0.85	0.7	650
M_9	1.2	1.4	1	0.8	0.6	700
M_{10}	1.5	1	0.75	1.25	0.5	750
M_{11}	0.8	0.6	0.4	1.5	1	750
M_{12}	0.8	0.6	0.4	1.75	1	875
M_{13}	0.8	0.6	0.4	2	1	1000
M_{14}	0.8	0.6	0.4	2.25	1	1125
M_{15}	0.8	0.6	0.4	2.5	1	1250

Evaluation of a strategy. When a strategy is evaluated empirically, Algorithm 1 is run 100 independent times and the payoff attained by S is annotated. This value is transformed into a percentage over the maximum payoff attainable in one 500-event execution (see Table 1). The average of such percentages is taken as the empirical payoff of the strategy.

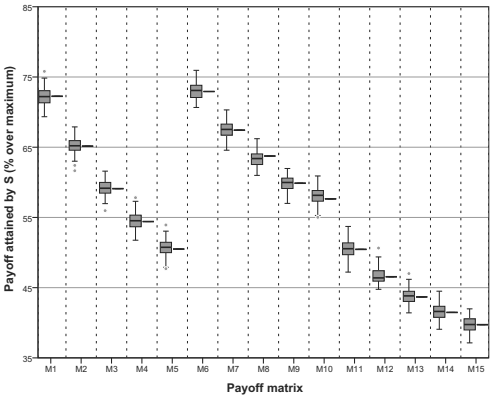
Optimization algorithm. An important point is the optimization method employed to solve the problems formalized in (8) and (15). A 4-period dynamic mixed strategy in an adversarial model with $n = 5$ events and $m = 5$ actions has 120 variables (100 real numbers representing the weights of the strategies at each period, and 20 integer values representing the lengths of the periods for each event) to be optimized. This represents a very hard optimization problem, so we have made use of a heuristic optimization method called *differential evolution* (DE) because it is particularly well suited for real optimization. A lot of variants of DE have been proposed since it was first introduced by Storn and Price (1997) as well as Price *et al.* (2005). We have employed an enhanced auto-adaptive variant called *self-adaptive DE* (SADE (Qin *et al.*, 2009)), which shows specially good performance in high-dimensionality problems. We used a Java implementation that is freely available[2] and has been already used in machine learning studies that ultimately require real optimization (Triguero *et al.*, 2011). No formal study has been conducted to tune the parameters of the algorithm but some preliminary experiments led to the following values:

- crossover operator: binomial crossover,
- crossover probability: 0.5 for static strategies and 0.9 for dynamic strategies,
- scaling factor: 0.5 for both static and dynamic strategies,
- population size: 50,
- number of iterations: 50000.

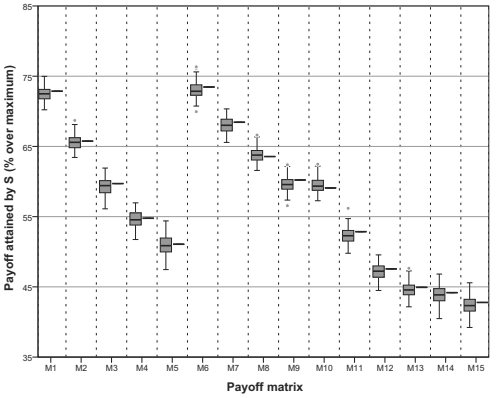
In order to answer the first question asked at the beginning of this section, it is necessary to evaluate empirically and theoretically one (or more) mixed strategy and check that both results match. We could pick any arbitrary strategy for this, but we decided to employ the strategies obtained after applying a basic DE optimization algorithm. The steps were the following:

1. For each payoff matrix, run a fast, basic DE twice (once to get an optimized static strategy and once more to get a 4-period dynamic strategy) with the above parameters. As a result, we get 15 optimized static strategies and 15 optimized dynamic strategies. Since the only goal here is to compare the expected payoff with empirical values, it is not important to do several independent runs of the optimization process.
2. Evaluate every strategy theoretically, applying expressions (7) or (14) as required if the strategy is static or dynamic. As a result, we get 15 expected payoff values corresponding to static strategies and 15 values corresponding to dynamic strategies.
3. Evaluate every strategy empirically by running Algorithm 1 100 independent times. The total payoff is annotated after each run, and the mean of the 100 resulting values is taken as the empirical payoff of the strategy. As a result, we get 15 empirical payoff values corresponding to the static strategies and another 15 corresponding to dynamic strategies.
4. Compare the expected and empirical payoff every strategy to check the differences and the variability. Recall that the expected payoff indicates the average behavior, and that is why several independent runs are required when empirically evaluating a strategy.

[2] `http://sci2s.ugr.es/EAMHCO/src2/advancesDEs.zip`.



(a)

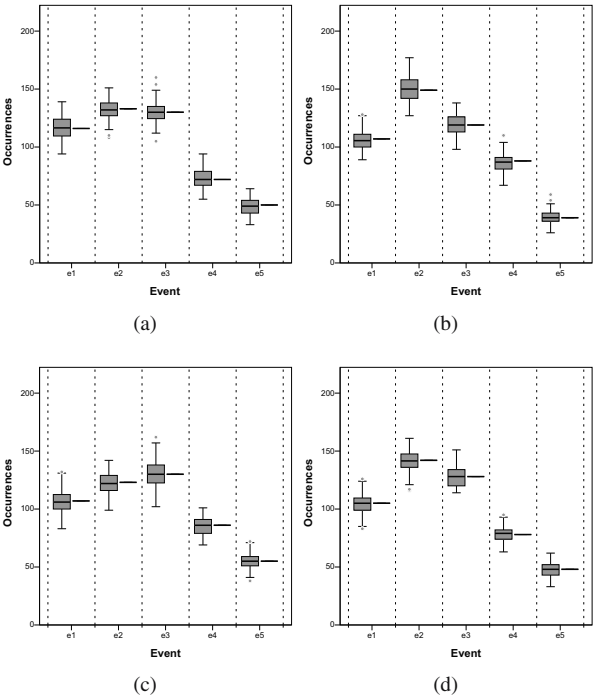


(b)

Fig. 3. Empirical (gray boxes) and expected payoff (lines outside the boxes) for every payoff matrix using the optimized static and dynamic strategies. Values expressed as percentages over the maximum possible. Data of the empirical payoff after 100 runs are depicted in gray boxes while the expected payoff is represented by lines outside the boxes. Static mixed strategies (a), 4-period dynamic mixed strategies (b).

Figures 3(a) and 3(b) show a comparison of the expected and empirical payoff of the static and dynamic mixed strategies found in Step 1. The comparison was done as indicated in Step 4. The plots confirm an almost perfect matching between the expected and the empirical average payoff attained.

In the case of dynamic strategies, one of the most difficult parts is the correct estimation of the number of occurrences of each event. These values were also annotated in the simulations of the optimized 4-period dynamic strategies, in order to contrast them with the theoretical estimation. The results are displayed in Fig. 4, which only shows 4 of the 15 payoff matrices (M_1, M_5, M_{10} and M_{15}) because the results on the others are analogous. An almost perfect matching was achieved between the expected and the empirical average number of events of each type.



(a) (b) (c) (d)

Fig. 4. Number of occurrences of each event during a 500-step simulation with 4-period dynamic strategies for 4 of the payoff matrices. Empirical (gray boxes) and expected values (lines outside the boxes) after 100 independent runs with each payoff matrix. M_1 (a), M_5 (b), M_{10} (c), M_{15} (d).

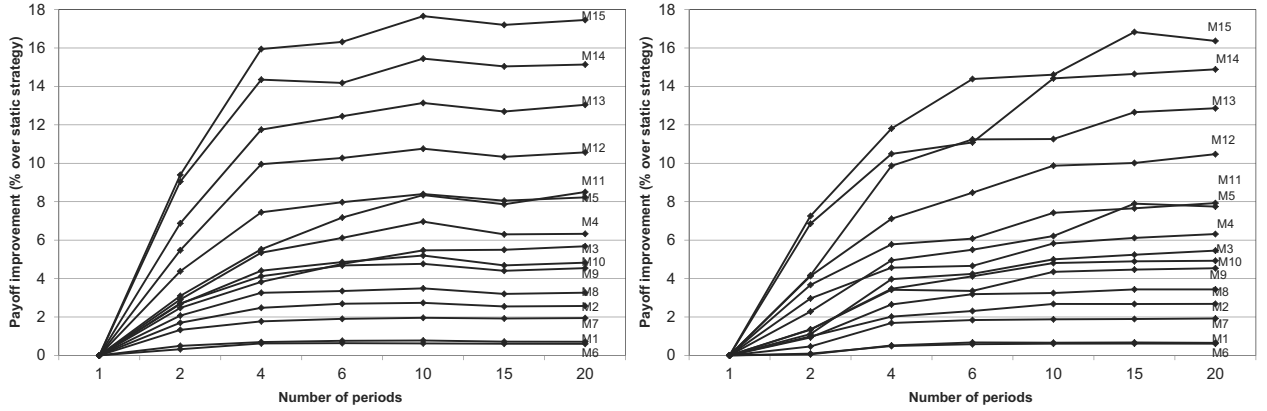
4.1. Static vs. dynamic strategies: Performance comparison. We now analyze the performance of both static and dynamic mixed strategies by comparing their expected payoff to answer the second question asked above. For this experiment, the strategies tested are the best ones obtained for each payoff matrix after 5 independent runs of the SADE optimization algorithm. The results are shown in Table 2 and Fig. 5.

Table 2. Payoff attained by static and dynamic strategies found by SADE, as a percentage over the maximum.

Strategy:	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
Static	72.5	65.4	59.3	54.6	50.7	73.2	67.6	63.4
Dyn $H=4$	73.0	67.0	62.0	57.6	53.5	73.7	68.9	65.5

Strategy:	M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}
Static	60.1	55.9	50.6	46.7	43.8	41.6	39.9
Dyn $H=4$	62.6	58.2	54.4	51.4	49.0	47.6	46.2

The most important conclusion from this figure is the following. In all the payoff matrices tested, the optimal 4-period dynamic mixed strategy consistently outperformed the optimal static strategy. The differences are statistically significant ($p = 6.1 \times 10^{-5}$) at any significance level according to a paired Wilcoxon signed rank test with the two samples of Table 2 (the samples are



(a) (b)

Fig. 6. Improvement achieved with dynamic mixed strategies for different numbers of periods. Average results over 5 independent runs of the SADE optimization algorithm for each payoff matrix. L = 500 steps (a), L = 1000 steps (b).

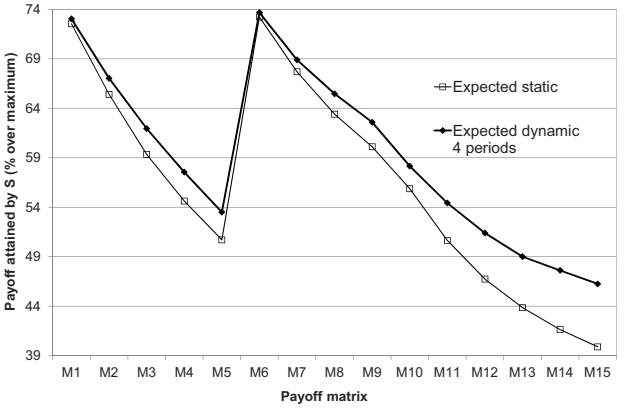


Fig. 5. Expected payoff of the best static and 4-period dynamic strategies found by SADE in 5 independent runs. Values expressed as percentages over the maximum possible.

paired due to the payoff matrices they share). Recall that each value of the table does not come from a simulation but from a prediction made using the expressions, so it is not influenced by random factors apart from the payoff matrix generated for each scenario. Notice that the greater gain in performance was achieved in matrices M_{11} to M_{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 very important to do the best action as many times as possible because its payoff is much greater than that of the rest of the alternatives.

4.2. On the influence of the number of periods. In order to provide insights into the impact the number of periods of a dynamic mixed strategy has over the performance, an experimental sampling with different number of periods was done. Again, the SADE optimization algorithm was run 5 times independently to find a dynamic strategy with a fixed number of periods, and the average of the best solutions found in the 5 runs was annotated. This was repeated for $H \in \{1, 2, 4, 6, 10\}$ in two different contexts: one with $L = 500$ steps and one with $L = 1000$ steps, in order to assess how the number of steps affects the conclusions. The improvements for each payoff matrix with respect to the optimal static mixed strategy ($H = 1$) are displayed in Fig. 6.

Figure 6(a) shows that increasing the number of periods causes a severe improvement at the beginning, but becomes less pronounced after 4 periods, reaching its maximum at 10 periods. This means that using more than 4 periods hardly enhances performance. Therefore our choice of $H = 4$ was a good one to show the benefits of the dynamic strategies approach (although some additional gains are also observed for up to 10 periods). Further, it is confirmed once again that dynamic mixed strategies provide a higher improvement when applied to more difficult payoff matrices. For instance, M_{15} is the most difficult one and where the improvement is the largest, followed by M_{14} and M_{13}. The same happens for matrices M_5, M_4 and M_3, and also for M_{10}, M_9 and M_8.

In Fig. 6(b) the trend is similar, but the improvement is gradual and continues growing until reaching 20 periods (although the gain when moving from 15 to 20 is quite small). Since 1000 steps are being considered here, we can expect that there is more room for improvement using more periods because there is more time available to switch to a new strategy. Intuitively, this should be done when T has learnt S's strategy and can no longer be deceived, except by switching to a new set of weights.

Notice that the optimal value of H also depends on the length of the simulation, since it is necessary to play a given strategy for long enough (before switching to a different one) in order to cause the intended manipulation, reflected in how agent T's observation matrix changes. For this reason, the improvement is small after $H = 4$

and stops at $H = 10$ when the number of steps is 500, but keeps growing gradually until $H = 20$ when 1000 steps are played. The trend of Fig. 6(b) also confirms that SADE still works well when increasing the number of unknowns, as additional gains are achieved with $H = 15$ and $H = 20$ in several cases, provided that the simulation is long enough. Therefore, the little and subsequently no improvement at all after $H = 10$ in Fig. 6(a) is due to the number of steps of the simulation (too short for using strategies with more than 10 periods), and not to a failure of the optimization process.

5. Conclusion and further work

An extension to an existing adversarial model was proposed consisting in introducing statistical dependence between actions and the next event. Static and dynamic mixed strategies for an agent in this extended model were successfully designed using heuristic optimization methods. Analytical expressions of the expected payoff for both strategies were provided and validated also from an empirical point of view. The design of good strategies was tackled as a non-linear mixed optimization problem and solved using heuristic techniques. Furthermore, dynamic mixed strategies found by the SADE optimization algorithm showed to outperform the best static mixed strategies found by the same algorithm in all the scenarios tested, especially when the difference between the payoff of the best action and the payoff of the rest of actions becomes greater. All of these results are encouraging.

Further work on this topic will include investigating expressions that do not depend on the prior knowledge of an external parameter about the game that is to be played (in this case, the length of the input sequence), and also more complex time varying strategies that take into account some on-line conditions of the current state of the game. Other interesting aspects of the model concerning practical issues of the applicability of these strategies, such as establishing minimum thresholds for the payoff or studying to what extent such a variance in the results can be accepted in a real situation, will be addressed too. On-line learning mechanisms for agent T might also be investigated.

Acknowledgment

This work was supported in part by the projects TIN2011-27696-C02-01 from the Spanish Ministry of Science and Innovation, P11-TIC-8001 from the Andalusian government, GENIL-PYR-2014-9 from CEI-Biotic Granada, and FEDER funds. P.J. Villacorta also acknowledges an FPU scholarship from the Spanish Ministry of Education.

References

Amigoni, F., Basilico, N. and Gatti, N. (2009). Finding the optimal strategies for robotic patrolling with adversaries in topologically-represented environments, *Proceedings of the 26th International Conference on Robotics and Automation (ICRA'09), Kobe, Japan*, pp. 819–824.

Cichosz, P. and Pawełczak, Ł. (2014). Imitation learning of car driving skills with decision trees and random forests, *International Journal of Applied Mathematics and Computer Science* **24**(3): 579–597, DOI: 10.2478/amcs-2014-0042.

Conitzer, V. and Sandholm, T. (2006). Computing the optimal strategy to commit to, *Proceedings of the 7th ACM Conference on Electronic Commerce, EC'06, Ann Arbor, MI, USA*, pp. 82–90.

Kott, A. and McEneany, W.M. (2007). *Adversarial Reasoning: Computational Approaches to Reading the Opponents Mind*, Chapman and Hall/CRC, Boca Raton, FL.

McLennan, A. and Tourky, R. (2006). From imitation games to Kakutani, `http://cupid.economics.uq.edu.au/mclennan/Papers/kakutani60.pdf`, (unpublished).

McLennan, A. and Tourky, R. (2010a). Imitation games and computation, *Games and Economic Behavior* **70**(1): 4–11.

McLennan, A. and Tourky, R. (2010b). Simple complexity from imitation games, *Games and Economic Behavior* **68**(2): 683–688.

Osborne, M. and Rubinstein, A. (1994). *A Course in Game Theory*, MIT Press, Cambridge, MA.

Paruchuri, P., Pearce, J.P. and Kraus, S. (2008). Playing games for security: An efficient exact algorithm for solving Bayesian Stackelberg games, *Proceedings of the 7th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'08), Estoril, Portugal*, pp. 895–902.

Pelta, D. and Yager, R. (2009). On the conflict between inducing confusion and attaining payoff in adversarial decision making, *Information Sciences* **179**(1–2): 33–40.

Price, K., Storn, R. and Lampinen, J. (2005). *Differential Evolution: A Practical Approach to Global Optimization*, Natural Computing Series, Springer-Verlag New York, Inc., Syracuse, NJ.

Qin, A.K., Huang, V.L. and Suganthan, P.N. (2009). Differential evolution: Algorithm with strategy adaptation for global numerical optimization, *IEEE Transactions on Evolutionary Computation* **13**(2): 398–417.

Storn, R. and Price, K. (1997). Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces, *Journal of Global Optimization* **11**(10): 341–359.

Tambe, M. (2012). *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned*, Cambridge University Press, New York, NY.

Thagard, P. (1992). Adversarial problem solving: Modeling an opponent using explanatory coherence, *Cognitive Science* **16**(1): 123–149.

Triguero, I., Garcia, S. and Herrera, F. (2011). Differential evolution for optimizing the positioning of prototypes in nearest neighbor classification, *Pattern Recognition* **44**(4): 901–916.

Villacorta, P.J. and Pelta, D.A. (2012). Theoretical analysis of expected payoff in an adversarial domain, *Information Sciences* **186**(4): 93–104.

Villacorta, P.J., Pelta, D.A. and Lamata, M.T. (2013). Forgetting as a way to avoid deception in a repeated imitation game, *Autonomous Agents and Multi-Agent Systems* **27**(3): 329–354.

Villacorta, P. and Pelta, D. (2011). Expected payoff analysis of dynamic mixed strategies in an adversarial domain, *Proceedings of the 2011 IEEE Symposium on Intelligent Agents (IA 2011), Paris, France*, pp. 116–122.

Pablo J. Villacorta received his M.Sc. degree in computer engineering in 2009, his Master's in soft computing and intelligent systems in 2010, and his M.Sc. (Hons.) in statistics in 2012 from the University of Granada, Spain. He is currently pursuing his Ph.D. degree within the Models of Decision and Optimization Research Group, in the Department of Computer Science and Artificial Intelligence, University of Granada. He has published over 20 papers in international journals, national and international conferences. His Ph.D. thesis proposal was awarded the GENIL Prize for the most promising work at the *2011 Conference of the Spanish Association for Artificial Intelligence (CAEPIA)*. His research interests include mathematical and statistical software, soft computing, technology foresight, adversarial decision making, and game-theoretic security models. Please refer to his webpage at: `http://decsai.ugr.es/~pjvi` for more details.

David A. Pelta is an associate professor at the Department of Computer Science and AI, University of Granada, Spain. He holds a degree in computer science (1998) from the National University of La Plata, Argentina, and a Ph.D. (2002) from the University of Granada. He is a member of the Models of Decision and Optimization Research Group, where he conducts research on soft computing techniques, cooperative strategies for optimization, adversarial reasoning and interdisciplinary applications. He is actively involved in research projects funded by various organizations (Spanish government, European Community, Andalusian government, etc.). He has published more than 30 journal papers, co-edited 7 books and 5 special issues of relevant journals. He has co-advised five Ph.D. and three Master's theses. He founded (jointly with Natalio Krasnogor) the workshop series on *Nature Inspired Cooperative Strategies for Optimization (NICSO)*, serves on the editorial board of the *Memetic Computing* journal and acts as a reviewer for many other journals, like *Soft Computing, Applied Soft Computing, Swarm Intelligence*, etc. More details are available on his webpage at `http://decsai.ugr.es/~dpelta`

Received: 1 April 2014
Revised: 16 January 2015