Automatic Design of Deterministic Sequences of Decisions for a Repeated Imitation Game with Action-State Dependency

Pablo J. Villacorta, Luis Quesada, and David Pelta

***Abstract*—A repeated conflicting situation between two agents is presented in the context of adversarial decision making. The agents simultaneously choose an action as a response to an external event, and accumulate some payoff for their decisions. The next event statistically depends on the last choices of the agents. The objective of the first agent, called the imitator, is to imitate the behaviour of the other. The second agent tries not to be properly predicted while, at the same time, choosing actions that report a high payoff. When the situation is repeated through time, the imitator has the opportunity to learn the adversary's behaviour. In this work, we present a way to automatically design a sequence of deterministic decisions for one of the agents maximizing the expected payoff while keeping his choices difficult to predict. Determinism provides some practical advantages over partially randomized strategies investigated in previous works, mainly the reduction of the variance of the payoff when using the strategy.**

I. Introduction

Adversarial decision is largely about understanding the minds and actions of one's opponent [1], [2]. 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. Clear examples are terrorism and other applications in Defense [3], along with 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. The development of intelligent training systems and the business field (competing firms) represent interesting potential applications as well, in which the agents are required to construct a model of the opponent that includes the opponent's model of the agent.

A brief survey of techniques including the combination of game theory with other approaches, such as planning, probabilistic risk analysis and stochastic games is presented in [2]. Other direct examples that demonstrate in which sense adversarial reasoning (and game theory in particular) can be fully used in real problems are the so-called security games. A particular case of security games is represented by the development of patrolling models for autonomous robots. The aim is to design routes for patrolling trying to minimize the chance that an enemy enters a security border. A lot of research is being done in this area, and several abstract models with their respective algorithmic solutions have been proposed so far [4], [5], [6]. Although this topic is not the focus of our particular work, such models bear a clear resemblance to the adversarial reasoning model analyzed in the next section

A special case of adversarial situation is the model presented in [7] and reviewed here. It involves two participants, S and T, each of which chooses an action as a response to an external input, 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 times T imitates S correctly, the smaller the reward of S.

Imitation games have been recently studied from a formal perspective in works by McLennan and Tourky [8], [9], [10]. They have proven as complex as a general game and some computational problems, and game-theoretic insights can be shown in them in a simple way, specially those related with time complexity of equilibrium computation, and with the equivalence of some mathematical problems with the problem of computing the Nash equilibrium of a two-personal game [9]. Nevertheless, these works focus on theoretical complexity issues and analogies with other formal problems. Further, none of them explicitly considers repeated imitation 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. 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 his behaviour, he should adopt some counter-measure to avoid this fact. A defense against T is to make decisions that are intended to confuse him, by forcing the presence of uncertainty while its payoff is as less affected as possible, using for instance randomized strategies [7], [11], [12], [13], [4]. However, it is also possible to use deterministic sequences of actions along the simulation for the same purpose.

The aim of this contribution is to present deterministic sequences of actions for S as an alternative to previously investigated randomized strategies. An automatic technique to design such strategies will be explained as well, and their performance will be compared with other strategies presented in previous works. The motivation behind this proposal is

P. J. Villacorta, L. Quesada and D. Pelta are with the Department of Computer Science and Artificial Intelligence, CITIC-UGR, University of Granada, Spain. Emails: {pjvi,lquesada,dpelta}@decsai.ugr.es

Algorithm 1 Sequence of steps in the model.

for $t = 1$ to L **do**
  A new input e_i arises.
  Agent T "guesses" an action a_g
  Agent S determines an action a_j
  Calculate payoff for S
  Agent T records the pair e_i, a_j
end for

the hypothesis that deterministic sequences of actions will decrease the variability of the results on particular runs in relation to usual randomized strategies, which can be highly beneficial, while keeping the agent difficult to predict.

The remainder of the work is structured as follows. Section II reviews the model and summarizes existing research. Section III explains the motivation of deterministic strategies and gives an analytic solution to find the optimal strategy under certain assumptions. The problem is then presented as a combinatorial optimization task in Section IV, and the need for heuristic techniques to solve it is introduced. Section V deals with the experiments and discusses the results. Finally, Section VI is devoted to conclusions and further work.

II. Model Description

The model we are dealing with was first presented in [7] and consists of two agents S and T (the adversary), a set of possible inputs or events $E = \{e_1, e_2, \ldots, e_n\}$ issued 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. There is a payoff function $P : E \times A \to \mathbb{R}$ such that we will note $P(e_i, a_j) = p_{ij}$ as the payoff associated with action a_j as a response to the event e_i. These payoffs can be collected into a payoff matrix P with dimensions $n \times m$.

Agent S must decide which action to take, given a particular input e_i and with a perfect knowledge of the payoff function P. His aim is to maximize the sum of the profits or rewards given a sequence of inputs. These are issued one at a time and they come from an external environment, represented by agent R. The inputs of the sequence are generated randomly as explained in section II-B.

Agent T does not know the payoff function P but is watching agent S in order to learn from his actions. His aim is to reduce agent S payoff by guessing which action he will take as a response to each input of the sequence. The payoff for T can be assumed as 1 when he was successful in predicting S's action and 0 otherwise. Algorithm 1 describes the steps of the model, being L the length of the sequence of inputs. We assume L is known by the agents. Note that the repetition of the conflicting encounter is what motivates both agents to avoid easily learnable behaviour patterns.

Given a new input e_i, S and T issue responses a_j and a_g respectively. At the moment of the response, neither of the agents knows which action the adversary will choose. The payoff for S is computed as a function of both responses and the value p_{ij}. After the payoff has been calculated, agent T is informed of what S had chosen, and then T "records" the pair (e_i, a_j) in his own memory. This information can be used in the future by T to make his predictions. The memory in which agent T keeps records of the actions taken by S is modeled as an observation matrix O, with dimensions $n \times m$. O_{ij} stores the number of times that, in the past, agent S took action a_j when the input was e_i. The payoff attained by S at every step is given by:

$$p(e_i, a_j, a_g) = \begin{cases} p_{ij} & \text{if } a_j \neq a_g \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

This means that S gets no payoff each time T matches his response.

A. Behaviour of the Agents

The behaviour pattern of the agents can be diverse. Focusing on S as we will do in the remainder of the work, he could behave in a totally deterministic way that always chooses the action with the highest payoff for the given event. This would be very easy to learn for T after a few observations. On the other hand, he could behave in a totally random way, which would be completely unpredictable but may cause actions with low payoff to be chosen too frequently, resulting in a lowering in the payoff. In between, many different strategies are possible. Randomized strategies have been traditionally employed in adversarial situations to keep the adversary guessing. The most common approach is to compute the optimal probability distribution over the available actions, so that the average payoff taking into account such distribution is maximized. In two-player strategic situations, Nash equilibrium to mixed strategies computes a probability distribution such that a player cannot do better than using the randomization over his actions prescribed by his Nash equilibrium strategy. Unfortunately, this computation requires knowing the payoff function of the adversary, which is not the case here.

Agent T can use the information of the observation matrix in many ways. He could always predict using the action most frequently (MF) observed in the past. This is a deterministic behaviour that S could easily detect and avoid. For that reason, T is also expected to behave in a randomized way. Although the use of learning techniques will be investigated in future works, in what follows we will consider that T employs a so-called Proportional-to-Frequency (PF) strategy [7]. In this strategy, the probability that T selects action a_j as a prediction to event e_i is $O_{ij} / \sum_{k=1}^{m} O_{ik}$, with O being the observation matrix at the moment of the prediction.

B. Dependency Between the Current Action and the Next Event

The events coming from an external environment arise in a random way. The original model [7] considered a uniform distribution that was not influenced by the decisions of the agents, while a more recent work [14] presented an extension of the model in which the decisions of agent S do have an influence over the next event to arise. In the present

contribution we will deal with the latter. The dependency is modeled by a *dependency matrix* C that states the conditional probability distributions of the next event to arise, given the action taken by agent S in the present step:

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

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 $i = 1, ..., M$. In this expression, X is a discrete random variable representing the next event arising, and Y is a discrete random variable representing the current action taken by agent S that, as will be explained, is based on a randomized behaviour rule. Finally, let $(\pi_1, ..., \pi_n)$ 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 probabilites as uniform, $\pi_i = 1/n$ for $i = 1, ..., n$. With dependency in mind, now a *good* action for S is not one that just provides a high immediate payoff but that also causes events with high payoffs (on average) to be likely to arise in the next turn.

III. Deterministic Sequences of Decisions

In a repeated situation like this, the only information an agent has about the strategy of the adversary is his own intepretation of the sequences of decisions observed in the past. This opens the door to deception-oriented decisions. In particular, such sequence does not have to necessarily follow a true probability distribution. An alternating strategy for S may be perceived over the time as a randomized one since S does not always take the same action for a given event, although the behaviour rule used by S may be deterministic.

In general, the approach followed in previous works regarding novel strategies had the following steps: (i) provide a theoretical function (if possible) describing the expected payoff for S with a proposal of a novel strategy; (ii) apply an optimization process to that function in order to find the optimal strategy under the given assumptions, and (iii) evaluate the results empirically in order to test the variability of the results with respect to the expected ones, and the improvement with respect to other, simpler strategies.

In such works [12], [11], the strategies analyzed always contained some degree of randomization. Therefore, the payoff they provide can be computed theoretically only by considering the probability that each possible outcome of the game (i.e. each combination of actions from both players in response to each event) eventually occurs. The value obtained is called the *expected payoff* and it represents the average payoff that can be attained by S after several independent runs in the same conditions when applying the strategy being evaluated. Nevertheless, the payoff in one particular run is subject to fluctuations that sometimes are very severe.

The idea of having a deterministic strategy contrasts with such randomized strategies. In principle, yet not looking at their performance on average, they present some practical advantages. Firstly, they do not introduce more randomness in the model: the two only sources of uncertainty when applying a deterministic sequence of actions are the stochastic nature of the sequence of events, and the stochastic nature of agent T's strategy PF. There is no additional degree of randomness due to randomized strategies as happened in previous works, and this may reduce the variability of the results on particular runs. Secondly, having total certainty of which the action will be at a certain point of the sequence may be relevant for S, depending on the circumstances. If the actions represent, for instance, the movement of a patrolling robot belonging to a security enterprise, then it can be beneficial to know in advance which the position will be at every moment in order to plan on-line adjustments and reparations, or to know for sure which areas will be less secure in every period.

A. Dynamic Programming Approach

Under the assumption of PF strategy for agent T, dynamic programming gives us an exact solution to the problem of computing the optimal sequence of decisions, in terms of payoff for S, for a given matrix of conditional probabilities C and a payoff matrix P, and assuming the number of steps L of the game is known by both agents.

The key to build a solution is to consider that, when choosing an action $a_j^{(t)}$ at step t, we are giving a response to an unknown event. However, we know the probability that such event is e_1, or e_2,... or e_n, since those probabilities only depend on the previous action $a^{(t-1)}$. After choosing the action $a_j^{(t)}$, and since we are uncertain about the event to which we have just given a response, we cannot increase in one unit any particular cell of T's observations matrix. The solution is to increase *all* the cells of column j. The magnitude of the increase for each row $i = 1, ..., n$ is exactly the conditional probability of the current event e_i given the previous action $a^{(t-1)}$, $P[X = e_i|Y = a^{(t-1)}]$. Such updating operation over the observation matrix is carried by the updateColumn(O, $a^{(t-1)}$, a_j) function in the equations that follow. When the second argument is null, meaning there is no previous action, the prior probabilities are used instead of the conditional probability matrix.

Solving the problem for a given length L can be easily decomposed into problems of length $L - 1$ of the same nature for which an optimal solution is required in order to have an optimal solution of the bigger one. Following this idea, Bellman's optimality principle holds for this problem, as explained next. The problem has the key ingredients to be solved by dynamic programming:

a) Structure of the nested problems: The problem at any step can be described as finding the optimal sequence of actions for the remaining steps, given (i) the observation matrix at current step, (ii) the action taken in the previous step, and (iii) the number of steps remaining after current decision has been made. The optimal payoff should consider

the summation of the immediate payoff for choosing an action in the current event plus the optimal payoff that can be attained in the rest of the sequence, which is the solution to the subproblem constitued by the new state (i.e. modify the observation matrix according to the immediate action taken in the current step and decrease the remaining number of steps in one). The solution to the overall problem is given by the sequence of decisions that provide the optimal payoff with an empty observation matrix when no previous actions has been taken and the number of steps remaining after the first decision equals $L-1$.

b) Bellman's optimality principle: This principle is required in order to apply dynamic programming. It will be proven by contradiction. Let $(a^{(1)},...,a^{(L)})$ be an optimal sequence of L decisions that begins with an empty observation matrix, such that the observation matrix after the last decision is $O^(L)$. For any $1 < k < L$, let $O^{(k)}$ be the observation matrix before the k-th decision of that sequence. Then

(i) The subsequence $(a^{(1)},...,a^{(k-1)})$ must be the best subsequence of $k-1$ decisions that reaches step k with an observation matrix that is identical to $O^{(k)}$ when the initial observation matrix is empty, and

(ii) The subsequence $(a^{(k)},...,a^{(L)})$ must be the best subsequence of $L-k+1$ decisions when the initial observation matrix is $O^{(k)}$, no matter which the contents of the final observation matrix are.

(iii) If any of the above two statements does not hold, i.e. the corresponding subsequence is not optimal, then it could be substituted by the optimal one, and thus the new overall decision sequence $(a'^{(1)},...,a'^{(L)})$ would be better than the original. Since the original sequence was optimal by hypothesis, we reach a contradiction. Therefore Bellman's optimality principle holds.

(iv) Further, since the principle holds, the final contents of the observation matrix of the optimal subsequence $(a^{(k)},...,a^{(L)})$ should necessarily equal those of the final observation matrix after the optimal original sequence $(a^{(1)},...,a^{(L)})$.

c) Recurrence equation: A recurrence equation to describe the optimal payoff of the process when the observation matrix before solving step t is O, the previous action at step $t-1$ was a_k and there are still n steps remaining in addition to the current step, is the following:

$$g(O,a_k,n)=\max_{a_j\in A}\left\{\sum_{i=1}^{n}C_{ki}\left(1-\frac{O_{ij}}{\sum_{s=1}^{m}O_{is}}\right)p_{ij}+ g(\text{updateColumn}(O,a_k,a_j),a_j,n-1)\right\}, 0<n<L-1 \quad (2)$$

Base case:

$$g(O,a_k,0)=\max_{a_j\in A}\left\{\sum_{i=1}^{n}C_{ki}\left(1-\frac{O_{ij}}{\sum_{s=1}^{m}O_{is}}\right)p_{ij}\right\} \quad (3)$$

Starting equation for the whole sequence of length L (in this case O is assumed to be empty):

$$g(O,null,L-1)=\max_{a_j\in A}\left\{\sum_{i=1}^{n}\pi_i\,(1-1/m)\,p_{ij}+ g(\text{updateColumn}(O,\text{null},a_j),a_j,L-1)\right\} \quad (4)$$

In the above expressions, the payoff p_{ij} attained by S is weighted by the product of (i) the probability that event e_i arises, which depends on the previous action as expressed in the conditional probability matrix C, and (ii) the probability of not being guessed correctly assuming T uses strategy PF. Only when the two events occur, S attains payoff p_{ij}. Since they are both independent, the probability that they occur simultaneously is the product of them happening separately.

Finally, note a call to g generates m recursive calls. Because all the possible combinations of actions are thus exhaustively examined, and this number, m^L, is huge, this approach is computationally unfeasible. However, the same idea employed in this simplified method, i.e. increasing a column of the observations matrix by the conditional probabilities of each event given the previous action, can be applied to obtain an analytical expression of the expected payoff of a deterministic sequence of actions. This is explained in the next section.

IV. Design of Sequences as a Combinatorial Optimization Problem

Due to the unfeasibility of the dynamic programming approaches, we propose using a different technique to find a good (not necessarily optimal) sequence of decisions. The task can be tackled as a combinatorial optimization problem, in which finding a good sequence of decisions for S of a known length can be viewed as finding the combination of L integer values (each value representing an action from 1 to m) that maximizes S's expected payoff when using that sequence against T. As we will explain in the experiment section, L is usually a large value such as 500.

a) Objective function: The function to be maximized is the following. Given a sequence of L decisions or actions $(a^{(1)},...,a^{(L)})$ expressed as a sequence of integers $(d_1...,d_L)\in\{1,...,m\}^L$, the expected payoff for S if agent T uses strategy PF is given by function $f:\{1,...,m\}^L\to\mathbb{R}$:

$$f(d_1,...,d_L)=\sum_{i=1}^{n}\pi_i(1-1/m)p_{i,d_1}+ \sum_{t=2}^{L}\sum_{i=1}^{n}C_{d_{t-1},i}\left(1-\frac{O^{(t)}_{i,d_t}}{\sum_{k=1}^{m}O^{(t)}_{i,k}}\right)p_{i,d_t} \quad (5)$$

Here $O^{(t)}$ represents the observation matrix of agent T after the first $t-1$ decisions.

This function can be expressed in a procedural way as shown in Algorithm 2. It shows how to measure the expected payoff of a deterministic sequence of decisions, and will be employed as the objective function to be maximized to find the

Algorithm 2 Pseudo-code of the objective function.

poff $\leftarrow \sum_{i=1}^{n} \pi_i(1-1/m)p[i,d_1]$
$O[i,d_1] \leftarrow \pi_i \quad i = 1,...,n$
for $t = 2$ to L **do**
  for $i = 1$ to n **do**
    poff $\leftarrow$ poff + $C[d_{t-1},i]\left(1 - \frac{O[i,d_t]}{\sum_{k=1}^{m} O[i,k]}\right)p[i,d_t]$
    $O[i,d_t] \leftarrow O[i,d_t] + C[d_{t-1},i]$
  end for
end for
return poff

Algorithm 3 Steps of the model when S plays a deterministic sequence $(d_1,...,d_L)$ and T plays Proportional to Frequency

payoff $\leftarrow 0$; $O \leftarrow \mathbf{0}_{n\times m}$
for $t = 1$ to L **do**
  A new event e_i arises stochastically as follows:
  if $t > 1$ **then**
    e_i arises according to distribution $(C[d_{t-1},i])_{1\le i\le n}$
  else
    e_i arises according to distribution $(\pi_i)_{1\le i\le n}$
  end if
  T randomly selects a_g accord. to $(O_{ig}/\sum_j O_{ij})_{1\le g\le m}$
  S deterministically selects action a_{d_t}
  payoff $\leftarrow$ payoff + $p(e_i,a_g,a_{d_t})$ # see equation (1)
  $O[i,d_t] \leftarrow O[i,d_t] + 1$ # Agent T records pair (e_i,a_{d_t})
end for

best sequence. Among the several heuristics for combinatorial optimization, preliminary experiments with three of them were conducted: Ant Colony Optimization (MAX-MIN Ant System variant), a variant of Differential Evolution that discretizes before evaluating the fitness, and a Generational Genetic Algorithm (GGA) with integer coding. They all showed very similar results in the deterministic sequence design problem, so we chose the GGA simplest algorithm, which is also the fastest. More details on the setup of the algorithm can be found in the next section.

Finally, in order to clarify how the model is applied when S plays a given deterministic sequence of L decisions $(d_1,...,d_L)$ and T plays Proportional to Frequency, we provide Algorithm 3 as a particular case of Algorithm 1 under such conditions. under such conditions.

V. Experiments and Results

The experiments are aimed at answering the following questions:

(i) Does the theoretical expected payoff of a deterministic sequence (Eq. (5)) match the average peformance when evaluating the sequence empirically, after several independent runs of algorithm 1?
(ii) Do deterministic sequences peform better (in any way) than static/dynamic mixed strategies investigated in previous work?

A. Model Configuration

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

- Number of different inputs and different actions: n = m = 5
- Length of the input sequences: $L = 500$.
- Matrix of conditional probabilities:

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

a) 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, summarized in Table I. The rest of the rows of each matrix are different permutations of the set displayed in the table. We also display the maximum total payoff attainable by S after 500 events if he always chooses the action with the largest payoff and is never gussed correctly. This is the ideal situation that would only occur when there is no adversary. This table is the same used in [14] since we will compare the results with those in that work. The same happens with the conditional probability matrix shown above.

TABLE I
Set of payoffs associated to 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

B. Optimization Settings

a) Optimization algorithm: After the preliminary experiments already mentioned, we selected a GGA with elitism [15], [16] for finding good sequences of decisions. The implementation is part of the Eva2 free software for evolutionary optimization [17]. No formal study has been carried at this step of the research to tune the parameters, although some experiments showed small differences in the performance when changing the parameters. The specific parameters and operators employed were the following:

- Representation scheme: a solution is represented as a vector of length L whose values at each position are positive integers between 0 and 4 (integer coding).

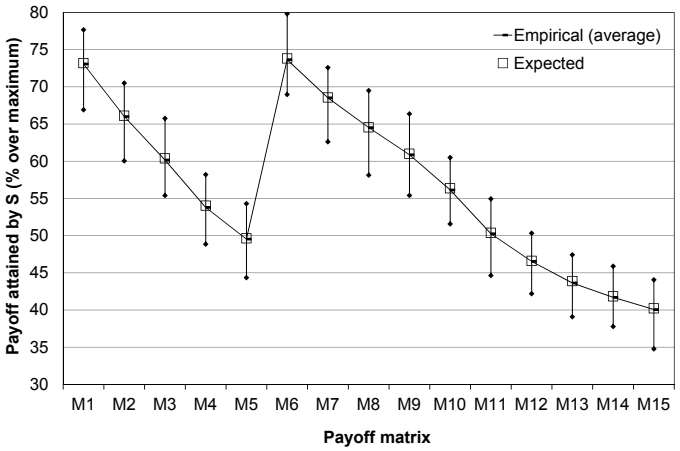


Fig. 1. Expected payoff and average, minimum and maximum empirical payoff over 1000 independent runs of the best solutions found after 5 runs of the GGA for each payoff matrix

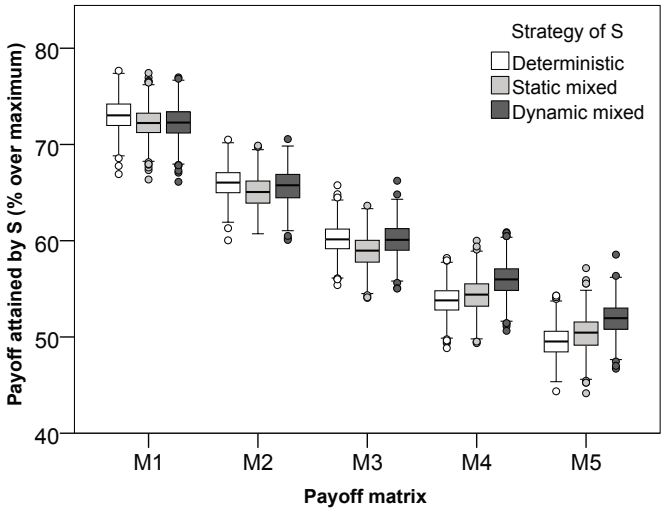
- Fitness function: algorithm 2.
- Crossover operator: three-point crossover.
- Crossover probability: 0.85
- Mutation operator: nominal mutation (a value in a position is replaced by other integer value randomly chosen in $\{0, 1, 2, 3, 4\}$).
- Mutation probability: 0.25
- Population size: 100 individuals.
- Stopping criterion: performing 500.000 fitness evaluations.
- Parent selection operator: perform ten 4-individual tournaments.

b) Empirical evaluation of a strategy: After the optimization algorithm ends, the best solution found is post-evaluated empirically. Algorithm 1 is run 1000 independent times and the payoff attained by S at each run (a "sample") is annotated. This value is transformed into a percentage over the maximum payoff attainable in a 500-event execution (see Table I). The average of such percentages is taken as the empirical payoff of the strategy, but all the 1000 samples are also stored in order to display box-plots to compare their variability, as explained below.

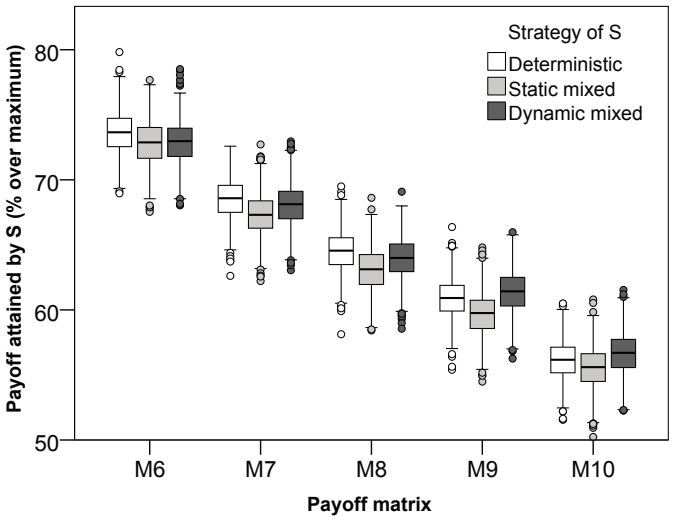
C. Results

The procedure to answer the first question was the following. Five independent runs of the GGA were done for each payoff matrix, and the best solution of the 5 executions was annotated. The expected payoff of those solutions is exactly the fitness value as given by the GGA because it is using the expected payoff expression (5) as fitness function. The empirical payoff is computed as explained above. The results are depicted in Fig. 1 and show an almost perfect matching. The maximum and minimum performance over the 1000 empirical evaluations has been depicted too, in order to illustrate the variability to which a deterministic sequence is subject. This will be analysed in detail later.

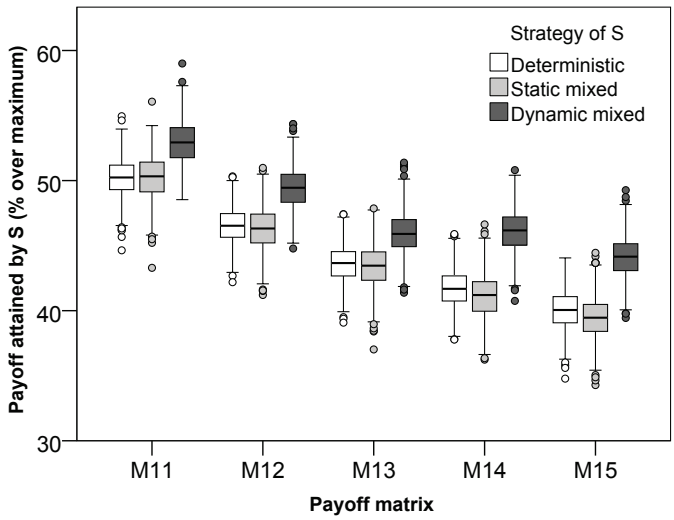
The answer to the second question can be found in Fig. 2.



(a)



(b)



(c)

Fig. 2. Boxplots of the 1000 empirical evaluations of deterministic sequences obtained by the GA and static and dynamic mixed strategies obtained by SADE as explained in [14]. All differences are statistically significative except in M3 between deterministic and dynamic mixed, and M12 between deterministic and static mixed, according to 1-way ANOVA.

Here, the performance of the deterministic sequences has been compared with that of static mixed strategies and dynamic mixed strategies automatically designed using a different evolutionary optimization technique called SADE (Self-Adaptive Differential Evolution [18]). Such strategies were proposed in a previous work [14] in which further details on the design technique employed can be found. Basically, it consists in the application of a heuristic for continuous optimization to find a probability distribution (set of weights) over the actions available, using the average of 100 empirical simulations of the model as the fitness function. Agent S uses this distribution to select an action.

Static mixed strategies consist on a separate probability distribution for each event. This distribution does not change for a given event during all the simulation. On the other hand, dynamic mixed strategies consist on several different probability distributions for a given event. The probability distribution used for that event changes after the event has arisen for a number of times, with this (integer) number also being a parameter of the strategy. The best performing strategies used 4 different probability distributions along the simulation for a given event.

The nature of the strategies is quite different from that of a deterministic sequence of actions, specially in the case of dynamic mixed strategies since they take into account the number of times each specific event has arisen in the past to search for a sequence of probability distributions (usually 3 or 4). For each event, agent S applies a set of weights until that event has arisen a certain number of times, and then switches to a different one. This kind of temporal information cannot be stored in a deterministic sequence of actions unless it is parameterized in any way, which is not being done here. In addition, static mixed strategies, despite being simpler than dynamic mixed strategies, also store a different set of weights for each event of the model (although not taking into account the number of times each event has arisen). This also represents an extra degree of flexibility that is lacking in the deterministic sequences of actions proposed here.

D. On the Relation of Dynamic Strategies with Deterministic Sequences

Nevertheless, it can be seen in the figure that deterministic sequences of actions clearly outperform static mixed strategies in all the payoff matrices tested. Further, they present a slightly smaller variability, specially in the most difficult payoff matrices M_{11} to M_{15}, in which the difference between the payoff of the best action and that of the rest of the actions is large. Dynamic mixed strategies still provide a better performance in almost all cases. Results were validated with a 1-way ANOVA since each batch of 1000 independent simulations are normally distributed, with the factor being the kind of strategy (deterministic, static mixed, dynamic mixed).

In order to check if this phenomenon is due to the ineffectiveness of the GGA that searchs for a good deterministic sequence or to the nature of the strategies itself, an additional set of experiments has been conducted. The experiment is aimed at uncovering if the sequence of decisions is good "per se", or if the reason of the good performance is the more flexibility of the philosophy of a dynamic strategy. We hypothesize that changing the behaviour depending on the number of times each event has arisen leads to a better performance, together with using different strategies for each event. The sequences of actions generated by the randomized dynamic strategy are only a result of this philosophy, because they are generated just in the appropiate way considering what has happened in the past with each kind of event.

The experiment designed was the following. Among the 1000 samples collected for each payoff matrix when empirically evaluating the dynamic mixed strategy found by SADE for that matrix, the best sample was annotated apart, together with the 500-decision sequence made by S within that concrete sample. Then, such sequence of actions was empirically evaluated with 1000 independent runs of Algorithm 1. This process was repeated for each payoff matrix. The resulting samples are depicted in Fig. 3 together with the samples coming from the evaluation of the deterministic sequences found by the GGA mentioned in the previous section. As can be seen, the deterministic sequences found by the GGA clearly outperform those coming from the best evaluation of a dynamic strategy found by SADE. This confirms our hypothesis as those sequences, which provided a very good performance (the best among 1000 evaluations indeed), are not good per se, but only in relation of what had happened in the past in that simulation. In other words, the sequence was generated as a result of a more intelligent analysis of the situation at each step, namely which is the current event and how many times it has arisen in the past. It is this analysis that makes them perform well, and not the sequence of actions itself and independently of the situation.

The concluding remark is that better performance with deterministic sequences of decisions can only be achieved by allowing more flexible strategies and by searching in a different search space that allows independent sequence of decisions for each event, rather than employing more sophisticated combinatorial optimization algorithms over the current m^L search space. However, the fitness function for this process possibly becomes more complicated than expression (5), so a careful theoretical study should be done first.

VI. Conclusions and Further Work

A repeated imitation game between two agents has been presented. The game to be played at each turn is different and depends on the choice of one of the agents. Deterministic sequences of actions for agent S have been succesfully designed by tackling the problem as a combinatorial optimization problem. The obtained strategies show improvement only with respect to static mixed strategies proposed in previous work. The reasons that make dynamic mixed strategies perform better have been analysed, and insights have been provided on the nature of both kinds of strategies.

Further research on this model will focus on the design of more flexible deterministic sequences of strategies. A very

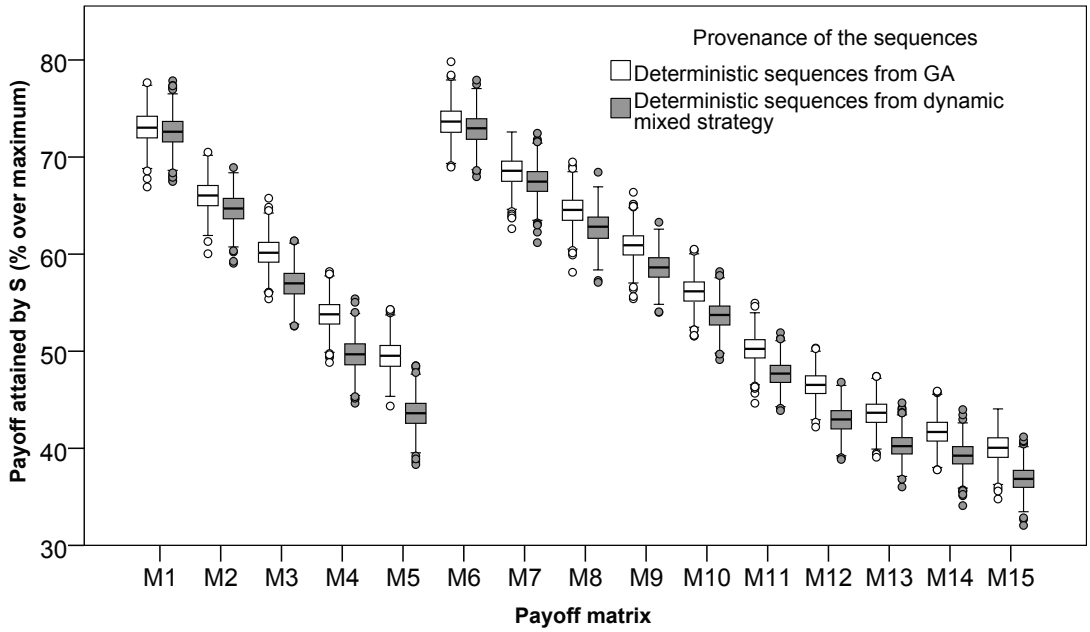


Fig. 3. Comparison of the empirical payoff of deterministic sequences designed by the GGA and sequences arisen during the best evaluation of a dynamic mixed strategy obtained by SADE

interesting work can also consider a finite horizon analysis, in which the players do not know exactly the number of steps of the simulation. The application of learning techniques for agent T is also a natural extension of this study, as well as strategies that take into account the real state of the game and thus, they cannot be totally designed before the game but have to be designed or tuned on-line. Finally, the application of some of these ideas to real domains, specially those related to autonomous robotic patrolling, are to be investigated as well.

ACKNOWLEDGMENTS

This work has been partially funded by projects TIN2008 - 06872-C04-04 and TIN2011-27696-C02-01 from the Spanish Ministry of Economy and Competitiveness, and P07-TIC-02970 and P11-TIC-8001 from the Andalusian Government. The first and second authors acknowledge two FPU scholarships from the Spanish Ministry of Education.

REFERENCES

[1] P. Thagard, "Adversarial problem solving: Modeling an opponent using explanatory coherence," *Cognitive Science*, vol. 16, no. 1, pp. 123 – 149, 1992.

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

[3] J. P. Dickerson, G. I. Simari, V. S. Subrahmanian, and S. Kraus, "A graph-theoretic approach to protect static and moving targets from adversaries," in *Proceedings of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, 2010, pp. 299–306.

[4] F. Amigoni, N. Basilico, and N. Gatti, "Finding the optimal strategies for robotic patrolling with adversaries in topologically-represented environments," in *Proceedings of the 26th Int. Conf. on Robotics and Automation (ICRA 2009)*, 2009, pp. 819–824.

[5] N. Agmon, S. Kraus, and G. Kaminka, "Multi-robot perimeter patrol in adversarial settings," in *Proceedings of the 2008 IEEE Int. Conf. on Robotics and Automation (ICRA 2008)*, 2008, pp. 2339–2345.

[6] M. Jain, J. Tsai, J. Pita, C. Kiekintveld, S. Rathi, M. Tambe, and F. Ordónez, "Software assistants for randomized patrol planning for the LAX Airport Police and the Federal Air Marshal Service," *Interfaces*, vol. 40, pp. 267–290, 2010.

[7] D. Pelta and R. Yager, "On the conflict between inducing confusion and attaining payoff in adversarial decision making," *Information Sciences*, vol. 179, pp. 33–40, 2009.

[8] A. McLennan and R. Tourky, "From Imitation Games to Kakutani," 2006, unpublished.

[9] ——, "Imitation games and computation," *Games and Economic Behavior*, vol. 70, no. 1, pp. 4 – 11, 2010.

[10] ——, "Simple complexity from imitation games," *Games and Economic Behavior*, vol. 68, no. 2, pp. 683 – 688, 2010.

[11] P. Villacorta and D. Pelta, "Theoretical analysis of expected payoff in an adversarial domain," *Information Sciences*, vol. 186, no. 1, pp. 93–104, 2012.

[12] ——, "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*, 2011, pp. 116 – 122.

[13] P. Paruchuri, J. P. Pearce, and S. Kraus, "Playing games for security: An efficient exact algorithm for solving bayesian stackelberg games," in *Proceedings of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, 2008, pp. 895–902.

[14] P. J. Villacorta and D. A. Pelta, "Design of strategies by means of Differential Evolution in an iterated imitation game with dependency between turns," in *Proceedings of the VIII Congreso Español sobre Metaheuristicas, Algoritmos Evolutivos y Bioinspirados (MAEB 2012)*, 2012, in press (in Spanish).

[15] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Kluwer Academic Publishers, Boston, MA, 1989.

[16] J. H. Holland, *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, 1975.

[17] M. Kronfeld, H. Planatscher, and A. Zell, "The EvA2 optimization framework," in *Learning and Intelligent Optimization Conference, Special Session on Software for Optimization (LION-SWOP)*, ser. Lecture Notes in Computer Science, LNCS, C. Blum and R. Battiti, Eds., no. 6073, 2010, pp. 247–250.

[18] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Trans. on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, 2009.