A dynamic mixed strategy for an adversarial model based on OWA weights

Pablo J. Villacorta, David A. Pelta and Maria Teresa Lamata

Abstract— **Adversarial decision making is aimed at determining optimal decision strategies to deal with an adversarial and adaptive opponent. One defense against this adversary is to make decisions that are intended to confuse him, although our rewards can be diminished. In this contribution, we propose time varying decision strategies for a simple adversarial model. These strategies have been obtained by using linear OWA weights that have influence on the decision made. Several time-varying patterns have been used to build such strategies. We have compared their performance against static strategies. The new strategies tested consistently outperform the optimal static strategy in a variety of situations. This is an encouraging result that confirms these strategies deserve further investigation.**

I. Introduction

Adversarial decision is largely about understanding the minds and actions of ones 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 [1]. The field is also known as decision making in the presence of adversaries or adversarial reasoning.

In its most basic form, adversarial decision making involves two participants, S and T, each of which chooses an action to respond to a given 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 problem becomes complex as the participants have the possibility to learn the others strategy. Examples of this type 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. [1]

Adversarial decision making is aimed at determining optimal strategies (for S) against an adversarial and adaptive opponent (T) that is watching S's decisions. One defense against this adversary is to make decisions that are intended to confuse him, although S's rewards can be diminished. 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.

In previous work [2], a model to study the balance between the level of confusion induced and the payoff obtained was proposed. The main conclusion of such study was that one way to produce uncertainty in situations of repeated encounters is through decision strategies for S that contain certain amount of randomness. Villacorta and Pelta [3] present a study on automatic design of such randomized strategies. Here we focus on strategies for S that are not constant along the time, but change at certain time steps in the iterated process. A similar study was carried out in [4] but with a slightly different approach that did not make use of OWA weights.

The aim of this work is to use OWA weights [5] to randomize over eligible actions. With these OWAs, we intend to design time-varying decision strategies for agent S, and compare them with strategies that do not change over time. We call the former *dynamic* strategies, and the latter, *static* strategies. More specifically, in a dynamic strategy S uses initially a set of weights that are not OWAs directly but have been computed using OWAs. S uses such weights to make randomized decisions for a certain time, and then switches to a different set of weights (that were calculated in the same way using OWAs) at a certain moment, and so on. It is expected that these changes benefit his ability to confuse the adversary and increase S's reward. Through our experiments, we try to answer the following questions regarding such dynamic strategies:

1) *When must S change his strategy?*
2) *Which strategy must S use after each change?*

The contribution is organized as follows. Some basic concepts on adversarial reasoning are outlined in Section I-A. Section II describes the main characteristics and components of the model used. Section III deals with the need of randomized strategies and defines static and dynamic, time-changing decision strategies for agent S. Section IV provides a review of OWA (Ordered Weight Averaging) weights and their utility in our problem. In Section V we describe the computational experiments performed and the results obtained. Finally, Section VI is devoted to discussions and further work.

A. Adversarial Reasoning

As stated before, adversarial decision making is largely about understanding the minds and actions of one's opponent. A typical example is the threat of terrorism and other applications in 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. The development of intelligent training systems is also an interesting field.

Pablo J. Villacorta, David A. Pelta and Maria Teresa Lamata are with the Models of Decision and Optimization Research Group, Department of Computer Science and AI, University of Granada, Spain (email: {pjvi, dpelta, mtl}@decsai.ugr.es).

This work was supported in part by projects TIN2008-01948 and TIN2008-06872-C04-04 from the Spanish Ministry of Science and Innovation and P07-TIC-02970 from the Andalusian Government

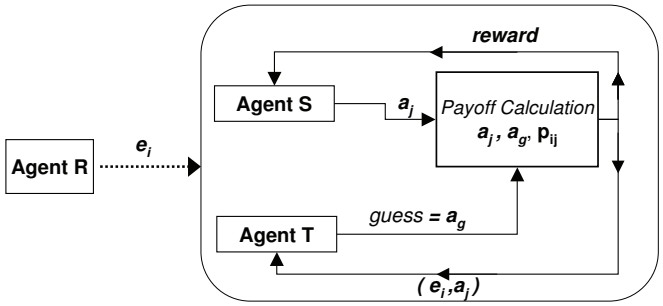


Fig. 1. Graphical representation of the model. inputs i_j are issued by agent R while response or actions a_k are taken by agent S.

Almost twenty years ago, P. Thagard [6] stated that in adversarial problem solving, one must anticipate, understand and counteract the actions of an opponent. Military strategy, business, and game playing all require an agent to construct a model of an opponent that includes the opponent's model of the agent.

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 in [1]. Other direct examples that demonstrate in which sense adversarial reasoning (and game theory in particular) can be fully used in real problems are patrolling models for autonomous robots. For further details, please refer to [7], [8] and [9].

II. Adversarial model

The model we are dealing with was first presented in [2] and it consists of on two agents S and T (the adversary), a set of possible inputs $E = \{e_1, e_2, \ldots, e_i, \ldots, e_n\}$ issued by a third agent R, and a set of potential responses or actions $A = \{a_1, a_2, \ldots, a_j, \ldots, a_m\}$ associated with every input. We have a payoff or rewards matrix P :

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

where p_{ij} is the reward or profit associated with action a_j to respond to the input e_i.

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. For the experiments, the inputs of the sequence are independent and generated randomly.

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. Algorithm 1 describes the steps of the model, being E the length of the sequence of inputs.

Algorithm 1 Sequence of steps in the model.

for $l = 1$ to L **do**
  A new random 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

Given a new input e_i, S and T issue responses a_k and a_g respectively. Agent T keeps records of the actions taken by S using an observation matrix, O, with dimensions $n \times m$. O_{ij} stores the number of times that, in the past, agent S decided to take action a_j when the input was e_i.

The reward calculation for S at every stage is defined as:

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

where F is:

$$F(a, b) = \begin{cases} 0 & \text{if } a = b \\ 1 & \text{otherwise} \end{cases} \tag{2}$$

This means that agent S gets no reward when agent T matched his response. In other words, S wants to avoid being imitated but at the same time he wants to minimize payoff loss due to sub-optimal decisions.

III. Behaviour of the agents

In this section, we provide alternatives for modeling the behavior of both agents. For simplicity, we assume that the inputs issued by agent R are equiprobable and that the number of inputs equals the number of actions (i.e, the payoff matrix is square).

A. Strategies for Agent T

Agent T applies a very simple frequency-based decision strategy. Given an input e_i , T uses a strategy called Proportional to the Frequency (PF): the probability of selecting an action j as a prediction to input e_i is proportional to O_{ij} (the observed frequency from agent S) [2].

B. Static mixed strategy for Agent S.

Agent S could use a totally deterministic strategy that always select the action with the highest payoff as a response to current input e_i. However, this would be very easy to learn for agent T so he would quickly predict this behavior correctly after a short number of inputs. S could also employ a totally random strategy that would select an action in a totally random way. This behaviour would be very hard to learn from observations but, on the other hand, the payoff attained would be low because bad actions (i.e. those with low payoff) may be selected with the same probability than best actions. It is clear that the need exists here to get to a good balance between confusion and payoff, as concluded in [2].

In classic game theory, a randomization over the existing actions (responses) is called a *mixed strategy*. A mixed strategy 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 do a movement, he uses this probability distribution to choose his action. In our model, we are interested in the *best randomization*, or in other words, the set of weights that lead to the highest payoff when playing against agent T.

With these weights, it is possible to calculate the so-called expected payoff for a given player, which is the sum of all the possible outcomes of the game weighted by the probability that each outcome inputually arises. In our adversarial model, this means that we can weight each payoff by the probability that agent S eventually gets that payoff. This probability is computed as the product of the probabilities of several independent facts happening simultaneously. Agent S will attain payoff p_{ij} if three conditions hold at the same time:

1) Input e_i must arise. We will refer to this probability as $P[E = e_i]$.
2) Agent S must select action a_j as a response. This probability is noted α_{ij}.
3) Finally, S will only get the score p_{ij} if agent T does not sucessfully predict his response. This probability can be computed as follows.

In case agent S is using a non-variant weight (or probability) α_{ij} during an input sequence of length L to select payoff p_{ij} , then the probability that agent T does not guess his actions if T uses PF strategy is (1-α_{ij}), as explained in the following reasoning. After L_i inputs of a certain kind e_i, since agent S uses α_{ij} , then action a_j will have been selected $L_i \cdot \alpha_{ij}$ times, and this is what agent T sees in O_{ij}. The probability that T selects action a_j as a prediction is then

$$P_{guess} = \frac{O_{ij}}{\sum_{j=1}^{m} O_{ij}} = \frac{L_i \cdot \alpha_{ij}}{L_i} = \alpha_{ij} \tag{3}$$

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

Taking into account the probabilities of the three conditions described above yields to the following expression of the expected payoff for agent S after a sequence of L inputs when he uses weights α_{ij} to select his actions:

$$EP_{static} = L \cdot \sum_{i=1}^{n} P[E = e_i] \cdot \sum_{j=1}^{m} \alpha_{ij} \cdot (1 - \alpha_{ij}) \cdot p_{ij} \tag{4}$$

If we want to maximize the expected payoff, we have to maximize expression 4 by computing the values of the optimal weights α_{ij} . This can be achieved using numerical optimization methods, such as a gradient descent, subject to two restrictions: (a) $\alpha_{ij} \geq 0$ and (b) $\sum_{j=1}^{m} \alpha_{ij} = 1$. When the inputs are independent, as we are considering in our model, then the set of optimal weights for each input e_i can be computed separately, because a set of weights does not interact with the rest. This yields to n independent optimization problems (one per input), each of them having m free variables (one per different action).

The optimization problem for the input e_i described above can be formalized as follows.

$$\max_{\{\alpha_{ij}\}} \sum_{j=1}^{m} \alpha_{ij} \cdot (1 - \alpha_{ij}) \cdot p_{ij} \tag{5}$$

subject to:

$$\sum_{j=1}^{n} \alpha_{ij} = 1$$
$$\alpha_{ij} \geq 0 \tag{6}$$

The problem above refers only to weights of row i. There are thus n independent optimization problem like (5), one for each row.

C. 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 he used all the time the same probability distribution (set of weights) to make a probabilistic decision. We now propose changing these weights along time or, more precisely, at certain moments. Two questions arise at this point:

1) *When must S change his strategy?*
2) *Which strategy must S use after each change?*

None of these questions have an easy answer. In the first case, S must take into account what he knows about the state of the problem, what the adversary is supposed to know, how the adversary is supposed to behave, and maybe also a history of S's responses to the most recent inputs. At this stage of the research, we do not use information about the actual state of the problem nor mantain such history. Agent S just switches from one strategy to another, based on a *change frequency* that we have to define in advance.

Regarding the second question, notice that the information agent T has observed in the past is a basic element when predicting S's responses. This suggests that changing the strategy of S (i.e. changing the weights he uses for making decisions along time) can be a good way to introduce confusion while still getting high rewards. To be precise, we want to study

1) How the change frequency affects the payoff attained by S, and
2) How OWAs can be useful to define different sets of weights so agent S can switch from using one set to using another at certain moments.

In a dynamic strategy, a *period* is 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 apply a different mixed strategy for every period. The *length* of a period is the duration of the period, i.e. the

number of inputs during which agent S will use the same mixed strategy.

The next example illustrates this concept. Suppose that we have an input sequence of length L = 1000 inputs. Then, we can define for instance 4 periods of lengths N^1 = 300, N^2 = 100, N^3 = 200 and N^4 = 400. For a given period, the set of optimal weights can be very different from that of other periods. Since the probability of every input to arise is independent from that of the other inputs, once again we can solve the problem independently for each input. For that reason, we can define a different number of periods and with different length for each input e_i, and thus design a dyamic strategy to be used when that concrete input arises. Such dynamic strategy will be different from other dynamic strategies that are to be used when other inputs arise.

In spite of the fact that the length of the periods and the weights used in each period can vary for each possible input e_i, the strategies tested in this work have the same length in all periods, and the set of weights used in one period is used again after a certain number of periods. This is due to the way we have defined the particular dynamic strategies of this paper, but the design methodology explained above is a general one and allows any number of periods with different lengths and weights.

IV. OWA WEIGHTS FOR MIXED STRATEGIES

The weights used by S to make random decisions in each period can be computed in multiple ways. One possibility is to use Ordered Weight Averaging weights (OWAs), since they allow to obtain several sets of weights by changing a generating parameter.

The OWA operator has received very much attention since its appearance. In 1988 Yager [5] introduced the OWA operator to provide a method for aggregating multiple inputs that lie between the max and min operators. Many extensions and practical applications have been proposed for the past two decades, showing that they are potentially useful in any field where aggregation of information is required.

An OWA [5], [10] of dimension m is a function $F : \mathbb{R}^m \rightarrow \mathbb{R}$ that has associated with it a weighting vector $W = [w_1, ..., w_m]^T$ such that (i) $w_j \in [0, 1]$, and (ii) $\sum_j w_j = 1$. Furthermore,

$$F(a_1, a_2, ..., a_m) = \sum_j w_j b_j$$

where b_j is the j-th largest element of the a_j. Notice that a weight w_i is not associated with a specific argument but with an ordered position of the aggregate. This ordering operation essentially provides a non-linear aspect to this aggregation operation.

A number of properties can be associated with these operators. It is first noted that the OWA aggregation is *commutative*, that is, the aggregation is indifferent to the initial indexing of the argument. A second property is *monotony*, i.e., if $\hat{a_j} \geq a_j$ for all j, then $F(\hat{a_1}, \hat{a_2}, ..., \hat{a_m}) \geq F(a_1, a_2, ..., a_m)$. The third property is *idempotency*: in particular, if $a_j = a$ for all j, then $F(a_1, a_2, ..., a_m) = a$. The satisfaction of these three properties assures that these operators belong to a class called *mean operators*. Min and Max are particular cases of OWA operators. The Max operator is recovered when $W = W_{max} = [1, 0, ..., 0]^T$. The Min operator is recovered when $W = W_{min} = [0, 0, ..., 1]^T$. Both operators are bounds of the OWA aggregation:

$$\mathrm{Min_j}\{a_j\} \leq F(a_1, a_2, ..., a_m) \leq \mathrm{Max_j}\{a_j\}$$

We can refer to the weights with the low indices as weights at the top, and those with higher indices as weights at the bottom. Using this convention, we see that if most of the weights are at the top, then the aggregation is emphasizing the higher-valued arguments in the calculation of the aggregated value. If most of the weights are at the bottom, then the aggregation is emphasizing the smaller-valued argument in the aggregation.

A. Linear OWA weights

It is possible to generate OWA weights by using only linear functions. This approach was suggested in Cables and Lamata [11]. The first step is to identify a set of linear functions that satisfy the properties of the OWAs operators. Summarizing, [11] proposes the following linear function to generate OWAs:

$$f(x) = ax + \frac{1}{m} - a\left(\frac{1+m}{2}\right) \tag{7}$$

where x is the index j of the weight w_j and m is the number of elements we want to aggregate, which are equivalent to the number of available actions (responses) in our model. Parameter a is the slope of the line and controls how the weights are distributed, i.e. which elements will be emphasized most. As stated in [11], $a \in \left(-\frac{2}{m(m-1)}, \frac{2}{m(m-1)}\right)$.

B. OWA weights in dynamic strategies

The main reason for which we employ OWAs is that several sets of weights can be generated by adjusting only one parameter that affects the relation between the weights. Our idea is that agent S switches between different sets of weights along the time during the repeated decision process, in order to increase the confusion induced but also trying to get high rewards. The switching frequency and the weights we alternate are issues to be studied, and they will be described in further detail in Section V.

Let us focus only on one input e_i. We first compute, say, r different sets of OWA weights for input e_i, each of them with m weights (as many as the number of actions in our model). Let $W_i^k = \{w_1, ..., w_m\}, k = 1, ..., r$ be each of these sets (also called *weighting vectors* in OWA terminology). Although we calculate the OWA weights w_j, they are not the final weights that S uses to make decisions in a mixed strategy. Every weight w_j is then multiplied by one of the payoffs p_{ij} of row i of the payoff matrix P. After this, the result of the multiplication is normalized (again into [0, 1]) to obtain r sets of weights $V_i^k = \{v_1, ..., v_m\}, k = 1, ..., r$. The normalized elements $v_j \in [0, 1]$ will be the weights of

the mixed strategy of S, i.e. the probability that S chooses action j as a response to input e_i. Finally, a dynamic mixed strategy that makes use of the r sets of weights is applied by changing the current set of weights V_i^k every certain number of inputs. In other words, every set of weights V_i^k is used during a period (recall Section III-C). Of course, the same set of weights can be used in more than one period if necessary. More details are given in the following section.

V. Experiments and results

We have considered 5 different values for parameter a of the expression of the linear OWA: $a \in \{-0.1, -0.05, 0, 0.05, 0.1\}$. The OWAs obtained are summarized in Table I.The difference between OWA weights and the final weights used in each period of a dynamic mixed strategy has been depicted in Fig. 3.

TABLE I
LINEAR OWA WEIGHTS FOR 5 ALTERNATIVES

Slope a	w_1	w_2	w_3	w_4	w_5
-0.1	0.4	0.3	0.2	0.1	0
-0.05	0.3	0.25	0.2	0.15	0.1
0	0.2	0.2	0.2	0.2	0.2
0.05	0.1	0.15	0.2	0.25	0.3
0.1	0	0.1	0.2	0.3	0.4

TABLE II
THE 15 PAYOFF MATRICES TESTED

Matrix	Values				
M_1	0.8	0.85	0.9	0.95	1
M_2	0.6	0.7	0.8	0.9	1
M_3	0.4	0.55	0.7	0.85	1
M_4	0.2	0.4	0.6	0.8	1
M_5	0.01	0.25	0.5	0.75	1
M_6	0.9	0.95	1	1.05	1.1
M_7	0.8	0.9	1	1.1	1.2
M_8	0.7	0.85	1	1.15	1.3
M_9	0.6	0.8	1	1.2	1.4
M_{10}	0.5	0.75	1	1.25	1.5
M_{11}	0.4	0.6	0.8	1	1.5
M_{12}	0.4	0.6	0.8	1	1.75
M_{13}	0.4	0.6	0.8	1	2
M_{14}	0.4	0.6	0.8	1	2.25
M_{15}	0.4	0.6	0.8	1	2.5

We want to test, in terms of the payoff attained by agent S, these normalized weights when used in the periods of a dynamic mixed strategy. Two factors are going to be tested:

1) The *change frequency*, measured as the duration of the periods of a dynamic strategy (all the periods having the same duration), and
2) The *change pattern* according to which the set of weights of the next period is selected (from all the sets available) when we switch from one period to another.

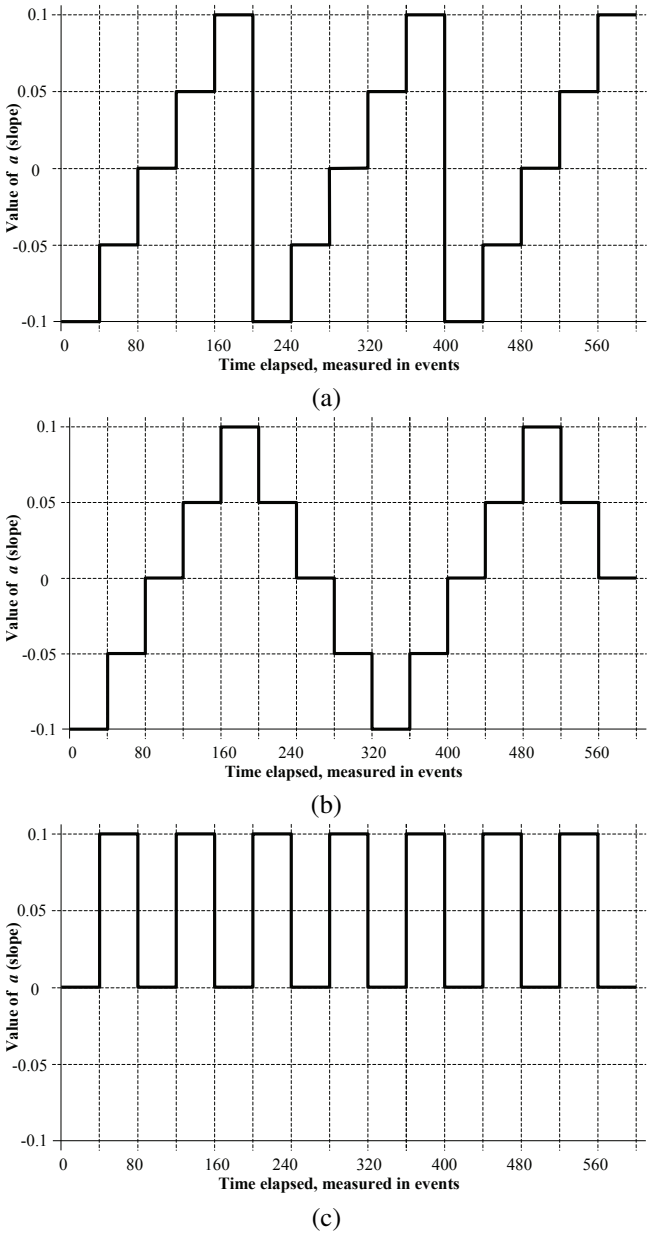
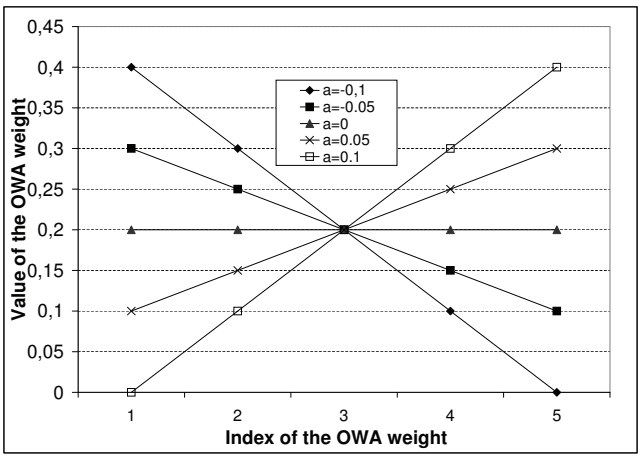


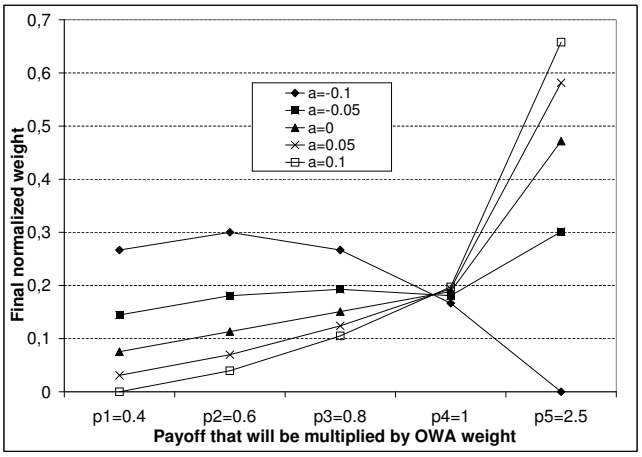
Fig. 2. Change patterns of linear OWA weights, with periods of 40 inputs. (a) Serrated. (b) Periodic. (c) Alternating

We have defined three different change patterns (serrated, periodic and alternating) that are depicted in Fig. 2. Each pattern changes the generating parameter a (slope) of the OWA weights following a different sequence. We have also evaluated three different durations for every period: 40, 60 and 120 inputs. Fig 2 shows the three patterns when the duration of a period is 40 (see X axis of the plots). Similar patterns were applied for durations 60 and 120.

In order to evaluate the change patterns and change frequencies, the input sequence must be long enough so every input arises enough times to study the real behavior of the pattern applied to that input. For this reason, we took a model with 5 different actions but only one input. This means that the same input e_0 arises at every step. As can be seen in the figures, the length of the input sequence in all



(a)



(b)

Fig. 3. OWAs weights and their influence in the final normalized weight for a mixed strategy with payoff matrix M_{15}. (a) Linear OWAs. (b) Normalized weights after applying OWAs of (a) to M_{15}.

our experiments is always 600 inputs.

Table II shows the payoff matrices tested in the experiments. Every row of the table is indeed a whole payoff matrix, since in the model used in our experiments we consider only one input, so the payoff matrices have only one row. Our goal is, first, to compare the performance of dynamic strategies that follow each of the change patterns and frequencies explained above and, second, to compare their performance with that of the optimal static mixed strategy (described in section III-B) for every payoff matrix. The optimization process involving the static strategy was carried out using the Microsoft Excel Solver optimization tool, which makes use of the Generalized Reduced Gradient (GRG2) algorithm [12].

The behaviour is shown in Fig. 4. Although it is not shown in the figure, several of these combinations perform better than the optimal static strategy. The greatest improvement was achieved when the dynamic strategy switches in periods of 120-input length, following a serrated change pattern. These results are depicted in Fig. 5. The payoff of agent S has been calculated as a percentage over the total payoff

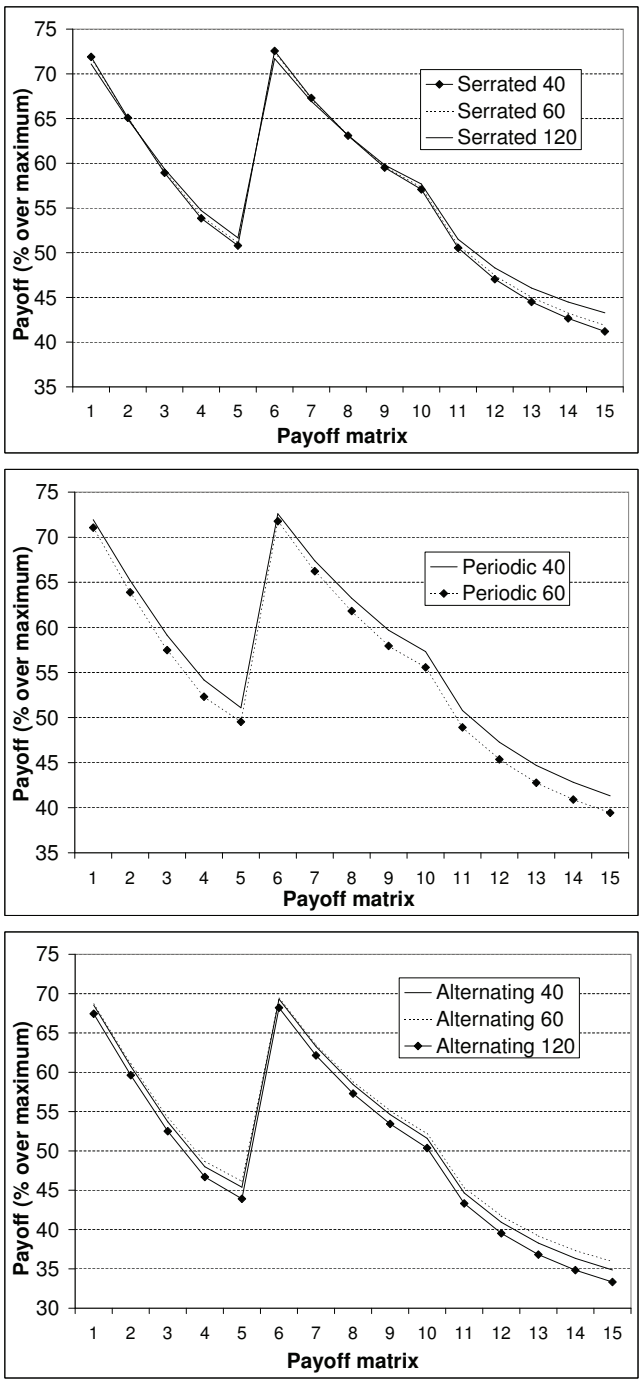


Fig. 4. Payoff attained for every combination of change frequency and change pattern with linear OWAs

that could have been attained if we always choose the action with highest payoff and suppose there is no adversary who diminishes the reward. This value is computed for every payoff matrix by multiplying the lenght of the input sequence (in our case, 600) by the maximum payoff of that matrix. Notice that the greatest improvement is achieved in matrices M_{11} to M_{15} in which there exists one action whose payoff is much greater than all others (see Table II). This configuration seems specially realistic because it models scenarios where a clear objective (or action) must be accomplished, so any

other goal different from that one has little interest.

An additional remark should be done on dynamic strategies. They do not require an optimization problem to be solved for every payoff matrix. On the contrary, the optimal static strategy does require solving an independent non-linear constrained optimization problem for every row of every payoff matrix.

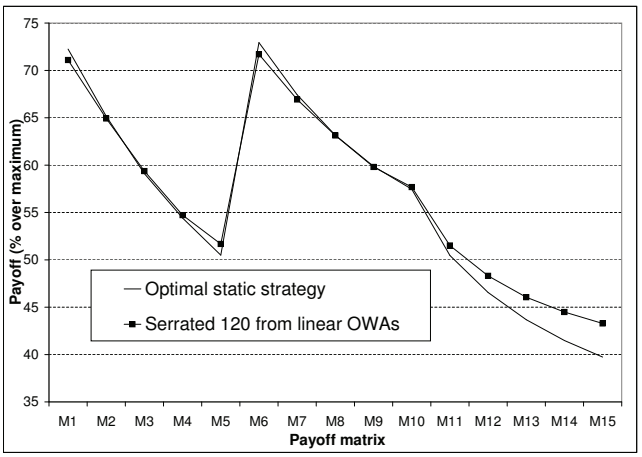


Fig. 5. Payoff attained by the optimal static strategy and the best performing dynamic strategy

VI. Conclusions and further work

A new kind of strategies that vary along time have been proposed and tested in an adversarial model. These strategies consist of several sets of weights that constitute a probability distribution over the eligible actions. The agent uses a set of weights for a period of time, and then switches to another one. OWA (Ordered Weight Averaging) weights have been employed to compute the weights to be used in each period. Several possibilities of change patterns and change frequencies have been investigated and succesfully tested, proving that they outperform the optimal static strategy computed by numerical optimization methods. The serrated change pattern, which does not depend on the payoff matrix, has shown the best results, since it consistently outperforms the optimal static strategy for all payoff matrix except two or three (in which the loss of performance was very small). This is an encouraging result regarding the investigation of more complex time-varying patterns in the future.

Further work on this topic may include adapting the strategy to the particular circumstances of the decision maker at every decision step (i.e. the current state of the problem), instead of using a fixed precomputed strategy like we are doing now. This may also include using a history of most recents responses and the resulting payoffs. Special emphasis should be put on the adaptation mechanism that modifies the probability distribution used by the agents along the time, and also in the conditions that should be evaluated when testing if a change in the weights is really desirable.

References

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

[2] 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.

[3] P. Villacorta and D. Pelta, "Evolutionary design and statistical assessment of strategies in an adversarial domain," in *Proceedings of the IEEE Conference on Evolutionary Computation (CEC'10)*, 2010, pp. 2250–2256.

[4] D. Pelta and R. R. Yager, "Dynamic vs. static decision strategies in adversarial reasoning," in *Proceedings of the Joint 2009 IFSA/EUSFLAT Conference*, 2009, pp. 472–477.

[5] R. R. Yager, "On ordered weighted averaging aggregation operators in multicriteria decision making," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 18, no. 1, pp. 183–190, 1988.

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

[7] 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 International Conference on Robotics and Automation (ICRA'09)*, 2009, pp. 819–824.

[8] 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'08)*, 2008, pp. 895–902.

[9] F. Amigoni, N. Gatti, and A. Ippedico, "A game-theoretic approach to determining efficient patrolling strategies for mobile robots," in *Proceedings of the International Conference on Web Intelligence and Intelligent Agent Technology (IAT'08)*, 2008, pp. 500–503.

[10] R. R. Yager, "Families of OWA operators," *Fuzzy Sets and Systems*, vol. 59, no. 2, pp. 125–148, 1993.

[11] E. Cables-Perez and M. T. Lamata, "OWA weights determination by means of linear functions," *Mathware & Soft Computing*, vol. 16, pp. 107–122, 2009.

[12] D. Fylstra, L. Lasdon, J. Watson, and A. Waren, "Design and Use of the Microsoft Excel Solver," *Interfaces*, vol. 28, no. 5, pp. 29–55, 1998.