HALL-OF-FAME COMPETITIVE COEVOLUTIONARY ALGORITHMS FOR OPTIMIZING OPPONENT STRATEGIES IN A NEW GAME

Mariela Nogueira[1], Juan M. Gálvez, Carlos Cotta, Antonio J. Fernández-Leiva
[1]University of Computers Science in Cuba, University of Málaga in Spain
Email: [1]mnogueira@uci.cu, {ccottap|afdez}@lcc.uma.es

KEYWORDS
Coevolution, RTS game, Genetic Algorithm, Artificial Intelligence

ABSTRACT

This paper describes the application of competitive coevolution as a mechanism of self learning in a two-player real time strategy (RTS) game. The paper presents this (war) RTS game, developed by the authors as an open-source tool, and describes its (built-in) coevolutionary engine developed to find winning strategies. This engine applies a competitive coevolutionary algorithm that uses the concept of *Hall-of-Fame* to establish a long-term memory that is employed in the evaluation process. An empirical analysis of the performance of two different versions of this coevolutionary algorithm is conducted in the context of the RTS game. Moreover, the paper also shows, by an example, the potential of this coevolutionary engine as a prediction tool by inferring the initial conditions (i.e. army configuration) under which a battle has been executed when we know the final result.

INTRODUCTION

Artificial Intelligence (AI) implementation represents a challenge in game development, and a deficient game AI surely decreases the satisfaction of the player. Game AI has been traditionally coded manually which causes well-known problems such as loss of reality, sensation of artificial stupidity, and predictable behaviors, among others; to overcome them a number of advanced AI techniques have recently been proposed in the literature, and coevolution, a biologically inspired technique based on the interaction between different species, represents one of the most interesting techniques.

Coevolution has been shown to be successful on a number of applications but it also has a number of drawbacks Ficici and Bucci (2007), Watson and Pollack (2001). In consequence several alternatives have already been proposed to alleviate the inherent problems of the process; for instance, the integrative techniques such as the so-called Pareto Coevolution De Jong and Pollack (2004), Jong (2007), de Jong (2004), Ficici and Pollack (2000) that proffers the integration between coevolution with Evolutionary Multi-Objective Optimization; also the application of evolutionary game theory to the study of coevolutionary algorithms Ficici and Pollack (2000), Ficici (2004), Wiegand et al. (2002). However the primary remedy to cope with the pathologies of coevolution consists of proposing new forms of evaluating individuals during coevolution Rosin and Belew (1995), and memorization of a number of successful solutions to guide the search is one of the most employed. Following this idea, Rosin and Belew (1997) already proposed the use of a *Hall-of-Fame* based mechanism as an archive method and, since then, there have been similar proposals such as those described in de Jong (2004), Jaskowski and Krawiec (2010), Jong (2007), Yang et al. (2009).

Coevolutionary systems are usually based on two kinds of interactions: one in which different species interact symbiotically (i.e. the cooperative approach) and other in which species compete among them (i.e. the competitive approach). In the cooperation, an individual is decomposed in different components that evolve simultaneously and the fitness depends on the interaction between these components; in the competition, an individual competes with other individuals for the fitness value and, if necessary, will increase its fitness at the expense of its counterparts, decreasing their fitness, as happens in a competitive game Johnson et al. (2004). This latter approach resembles an arms race in which the improvement of some individuals causes the improvement in others, and viceversa. Moreover, coevolutionary arms races can in principle lead to the discovery of complex, original behaviors that can make the game dramatically more interesting and challenging Dziuk and Miikkulainen (2011). Several experiments have shown significants results in the application of coevolutionary models in competitive environments to study the emergence of strategies in simple and complex games Angeline and Pollack (1993), Sims (1994), Johnson et al. (2004), Avery et al. (2008a), Smith et al. (2010).

This paper deals with the application of competitive coevolution (CC) as a self learning mechanism in RTS games. As a first contribution, the paper presents *robotWars*, a RTS war game available as open-source, that has been developed by the authors of this paper to be used as a tool to do research on CC methods. As a second contribution, the paper describes the evolutionary engine built-in inside *robotWars*; this engine consists of a competitive coevolutionary algorithm that uses in the evaluation process a Hall-of-Fame that acts as a long-term memory mechanism; from this engine, we devise two different variants that are used to automatically produce game strategies to govern the behavior of the army in the game that can also beat its opponent counterpart. Finally, we show (through an example) that the coevolutionary en-

gine built-in into *robotWars* can also be used as a predictive model to guess what has happened in already finished battles and whose conditions were not known at the time.

GAME DESCRIPTION

This section is devoted to *robotWars*[1], our arena for cooperative evolution; *robotWars* is a test environment that allows two virtual players (i.e. game AIs) to compete in a 3 dimensional scenario of a RTS war game, and thus it is not a standard game itself in the sense that no human players intervene interactively during the game; however it is a perfect scenario to test the goodness and efficacy of (possibly hand-coded) strategies to control the game AI and where the human player can influence the game by setting its initial conditions. To provide a good 3D simulation, the game was programmed with Ogre3D.

In *robotWars*, two different armies (each of them controlled by a virtual player - i.e. a game AI) fight in a map (i.e. the scenario) that contains multiple obstacles and has limited dimensions. Each army consists of a number of different units (including one general), in the initial configuration both armies have identical number of units, and the army that first wipes out the rival general is considered the winner. The game is executed in turns and in each turn one army is allowed to send, separately, one order to each of its constituent units (i.e. each unit can received a different order to be executed).

Virtual players are internally coded as a 4 dimension matrix where the first dimension has 7 different values corresponding to the *type of unit*(i.e. general, infantry, chariot, air force, antiaircraft, artillery, cavalry), the second dimension to *objective closeness* (i.e. a binary value: 0 if the unit is closer to the enemy general than to its mate general, and 1 otherwise), the third dimension to *numeric advantage* (i.e. are there, in a nearby space[2], more mate units than rival units? A binary answer:yes/no), and the fourth dimension to *health* (i.e. an amount that indicates the health level of the army as high, medium or low, respect to the numbers of living units). Each position of the matrix acts as a gen and stores one of the following 5 actions: *attack*[3], *advance*, *recede*, *crowd together*, or *no operation*.

The whole matrix represents a strategy that controls, deterministically, the behavior of an army during the game. For a specific type of unit there are 12 possible different states (i.e. $2 \times 2 \times 3$, all the possible value combinations considering the last three dimensions of the matrix), and basically, in a specific turn of the game each unit of the army will execute the action stored in the state in which the unit perceives that it is. Note that all the units (irrespective of their type) are managed by the same controller, and in any instant of the game, the team behavior will be the result of summing up all the action taken by each of its constituent units. Note however that this does not mean all the units execute the same action because the action to be executed by a unit will depend on its particular situation in the match and its specific category.

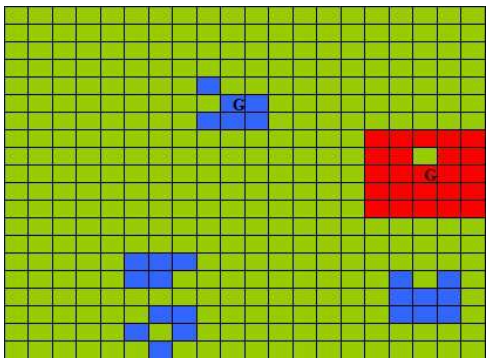


Figure 1: A simple map without obstacles

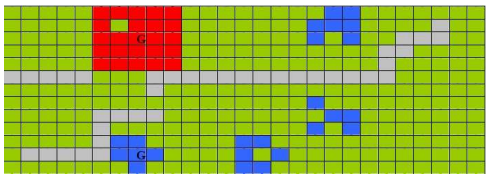


Figure 2: A map with obstacles

robotWars includes two interesting tools: the *Battle generator*, that allows a human to generate different scenarios for the game by changing the topologies and structures of the armies; two types of army formation are initially allowed: *testudo or tortoise* formation, in which the units are placed as a homogeneous block and the general is positioned in its center, and *band or guerrilla* formation, in which the units are dispersed in the battle scenario; Figure 1 shows a very simple scenario with an army initially positioned in tortoise formation (red) and the other in guerrilla formation (blue); also the green cell represents a walkable area, and the letter G points out the general position. Figure 2 shows a map with obstacles which are symbolized by gray cells. The other tool, the *Battle simulator*, receives as entry two different game AIs and a map (that includes not only the orographic features of the terrain but also the topologies and structure of the two armies) over which a game will be run and provides a deterministic simulation of a war between the two armies. There are a number of configurable parameters such as maximum number of iterations or initial army formation for example. Figure 3 shows a screenshot of a battle in the game.

HALL-OF-FAME BASED COMPETITIVE COEVOLUTIONARY ALGORITHM (HofCC)

Our objective is to apply competitive coevolution techniques to automate the generation of victorious strategies for the game. According to the strategies encoding explain in the

[1] `http://www.lcc.uma.es/~afdez/robotWars`

[2] Each unit u placed in a position (x_u, y_u, z_u) in the map has its own *visual range* that embraces any position (x, y, z) that is placed to a maximum distance τ of its own position; this means that each unit only receives information that is perceived inside its visual range.

[3] This action starts the combat against the nearest enemy.

Figure 3: Screenshot of *robotWars* game

previous section, the search space is $5^{7\times2\times2\times3} = 5^{84}$, which is really huge, and precisely this forms the basis for bases the employment of metaheuristic techniques.

Our RTS game incorporates a competitive coevolutionary algorithm that uses a long-term memory to keep the winning strategies found in each coevolutionary step; more specifically, each army (i.e. player) maintains its own Hall-of-Fame (HoF) Rosin and Belew (1995) in which its own winning strategies (with respect to the set of winning strategies of its opponent) found in each coevolutionay step will be saved. More recent studies continue exploring in the use of HoF Lichocki et al. (2008), Avery et al. (2008b) as a long-term memory mechanism, and although each author can propose their own implementation, the original idea of the method remains. Algorithm 1 shows schema of the said algorithm with our implementation of HoF. A strategy is considered *winning* if it achieves a certain score (see below) when it deals with each of the strategies belonging to the set of winning strategies of its opponent (i.e. the rival Hall-of-Fame). The initial objective is to find a winning strategy of player 1 with respect to player 2 (i.e. the initial opponent) so that the HoF of player 2 is initially loaded with some strategies (randomly or manually initialized: line 2). Then a standard evolutionary process tries to find a strategy for player 1 that can be considered as victorious (lines 7-13). A strategy is considered winning if its fitness value is above a certain threshold value ϕ (line 14) that enables the tuning of the selective pressure of the search process by considering higher/lower quality strategies; in case of success (line 14), this strategy is added to the HoF of player 1 (line 16) and the process is initiated again but with the players' roles interchanged (line 17); otherwise (i.e. no winning strategy is found) the search process is restarted again. If after a number of coevolutionary steps no winning strategy is found the search is considered to be stagnated and the coevolution finishes (see while condition in line 4). At the end of the whole process we obtain as a result two sets of winning strategies associated respectively to each of the players.

With respect to the evaluation of candidates for a specific player p (where $p \in \{player\ 1, player\ 2\}$), the fitness of a specific strategy is computed by addressing it with each of the (winning) strategies in the Hall-of-Fame of its opponent player (note that the Hall-of-Fame acts as a long-term memory by keeping all the winners found previously and all of them are also used in the evaluation process). Given a specific strategy s its fitness is computed as follows:

$$fitness(s) = \frac{\sum_{j=1}^{k}(p_j^s + extras_s(j)}{k} \quad (1)$$

where $k \in \mathbb{N}$ is the cardinality of the opponent HoF (i.e. number of opponent winning strategies found so far), $p_j^s \in \Re$ returns ϕ points if strategy s beats strategy h_j belonging to the opponent HoF (i.e. victorious case), $\frac{\phi}{2}$ in case of a draw, and 0 if h_j wins to strategy s; Also:

$$extras_s(j) = c - nturn_{sj} + c * \Delta health_{sj} + c * \Delta Alive_{sj} \quad (2)$$

where $c \in \mathbb{N}$ is a constant, $nTurn_{sj} \in \mathbb{N}$ is the number of turns spent on the game to achieve a victory of s over its opponent h_j (0 in case of draw or defeat), $\Delta health_{sj} \in \mathbb{N}$ is the difference between the final health of the army (i.e. sum of the health of all its living units) controlled by strategy s at the end of the match and the corresponding health of the enemy army, and $\Delta Alive_{sj}$ is its equivalent with respect to the number of living units at the end of the combat. This fitness definition was formulated based on our game experience, and it values the victory above any other result.

Algorithm 1: HofCC()

```
1  nCoev ← 0; A ← player_1; B ← player_2; φ ← thresholdvalue;
2  HoF_A ← ∅; HoF_B ← InitialOpponent();
3  pop ← Evaluate(HOF_B); // Evaluate initial population
4  while nCoev < MaxCoevolutions ∧ NOT(timeout) do
5      pop ←RandomSolutions(); // pop randomly initialized
6      i ← 0;
7      while (i < MaxGenerations) ∧ (fitness(best(pop)) < φ) do
8          parents ←Select (pop);
9          childs ← Recombine (parents, p_X);
10         childs ← Mutate (childs, p_M);
11         pop ← Replace(childs);
12         pop ← Evaluate(HOF_B);
13     end while
14     if fitness(best(pop)) ≥ φ then //winner found!
15         nCoevolutions ← 0; // start new search
16         HoF_A ← HoF_A ∪ {best(pop)}
17         temp ← A; A ← B; B ← temp; // interchange players'
               roles
18     else
19         nCoev ← nCoev + 1; // continue search
20     end if
21 end while
```

EXPERIMENTS AND ANALYSIS

For the experiments we have considered two maps (with and without obstacles respectively), 4 possible combinations of the two types of formation (i.e. tortoise vs. guerrilla, tortoise vs. tortoise, guerrilla vs. tortoise, and guerrilla vs. guerrilla), and two possible initial predefined strategies, *offensive* or *random*, to be charged initially in the HoF of player 2

-see Line 2 in Algorithm 1). The *offensive* strategy was preprogrammed and basically executes the action of attacking when the unit perceives that it is in an advantageous state; otherwise, it executes the action of going back (i.e. retreating). And *random* strategy selects a random action (from the five predefined actions) for each cell of the action matrix. In summary, 16 distinct battle scenarios were considered.
In addition, two variants of the HofCC algorithm were used: the first version of the algorithm (version A) uses the maximum size for the matrix of strategies as described in Section and thus reserves memory for storing the behavior (i.e. the action to be done) of all types of military units under all types of possible situations (i.e. states), even though not all types of units form part of the army; the consequence is that genetic operators can be acting on genes that will not have an influence on the fitness evaluation. The second version (version B) is an optimized version that considers, for the genetic search, only those units that are actually in the army, so the genetic operators act on the 100% of the useful genes that might influence the results, and surely expedites the search process (this is important to alleviate in certain form the computation cost associated to both coevolution and the Hall-of-Fame based evaluation process).
We only show the graphs for four battles, the other results are collected and analyzed in a table at the end of this section. Note that we use the colors red and blue to distinguish the armies.

CONFIGURATION OF THE EXPERIMENTS

For the experiments we have used a steady-sate genetic algorithm (GA - see Lines 7-13 in Algorithm 1) with the aim of finding a winning strategy with respect to a set of strategies (stored in the HoF of the opponent) that were considered winning in previous stages of the coevolutionary algorithm; this GA employed binary tournament for selection, Bernoulli crossover, it-flip mutation, elitist replacement, $MaxCoevolution = 15$, $Maxgenerations = 300$, population size was set to 100, and standard values for crossover and mutation probabilities were used ($p_X = .9$ and $p_M = 1/nb$ respectively where nb is the number of genes); and $\phi = 2000$ so that a strategy that defeats all the strategies in the HoF of its opponent is surely considered as victorious, although others can also be. The choice of these values is due to a previous analysis of the mean fitness reached by individuals in the competitions. We also set the constant c in Equation (2) to 200, a representative value of the range of scores that were obtained from the battle simulator after executing a number of games.
Note that a strategy is wining only within the proposed battle scenario and that we do not expect to find a global optimal strategy for a generic scenario. Our analysis has been guided by the following indicators:

- *Best fitness in each coevolution*: displays the best strategies and possible stagnation produced in the coevolutionary steps.
- *Number of generations used in each coevolution*: represents the number of iterations that are required to find a winning strategy. It helps to identify whether the problem's difficulty increases as best solutions are obtained, or if it remains stable.
- *Equality percentage of genes*: it allows us to know whether the new strategies adopted by the search algorithm suffer a continuous readjustment and massive variation of their genes, or if instead they are the result of changes produced on small sets of genes in the strategies found in previous coevolutionary steps.

In what follows, due to limitations of space, we only show the results obtained in 4 scenarios and will focus mainly on the first indicator mentioned. In any case, at the end of the section a summary of the results is provided.

RESULTS OF VERSION A

Figures 4 and 5 show the performance of the best fitness achieved by each army during the coevolution, using as initial enemy strategy the *offensive* (Fig. 4) and the *random* (Fig. 5). The battles were fought on the map shown in Figure 1. The red bar represents the best solutions found by an army (red army) and blue bars identifies the other army. The black line located on the 2000 fitness points represents the threshold at which the solutions are classified as winning strategies. When a solution exceeds this line, the next bar has a different color, which means that this army found a winning strategy, and the other army now tries to find, in the next coevolutionary step, another strategy that can beat the winners (including the current champion) encountered by the rival in the previous steps of the coevolutionary algorithm. A sequence of candidates represented by bars of equal color (where none of them exceeds the threshold that identifies a winning solution) means that the search does not progress as no winning strategy can be found.

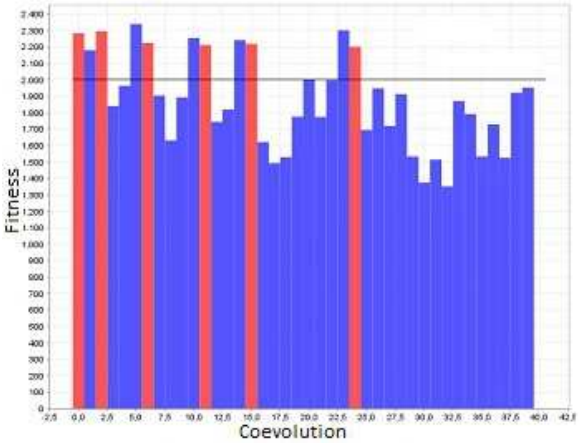


Figure 4: Version A: Offensive intial strategy; Best fitness

In these scenarios, the red army has less difficulties for finding winning strategies whereas the blue army, on the other

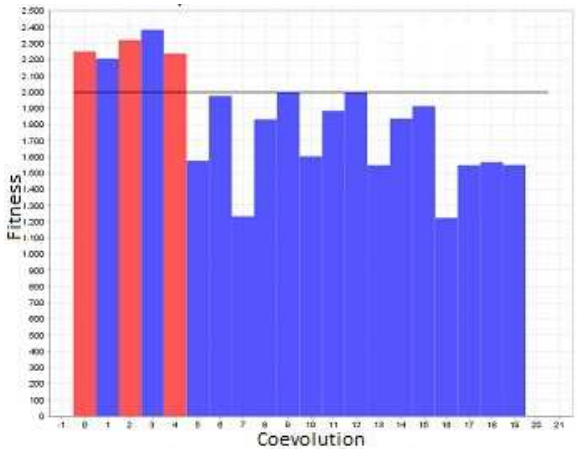


Figure 5: Version A: Random intial strategy; Best fitness

hand, has serious problems and requires an increasing number of coevolutionary steps to confront the new strategies proposed by the red army. In principle, observing the ease with which the red army finds a winning strategy, we can decide that the *tortoise* formation can provide advantages over the *guerrilla* formation, in this type of map (i.e. open and unobstructed).

Regarding the percentage of equality between the solutions, the analyzed data (graphs not shown for reasons of space) showed that solutions never exceed 30% of equality. In general, we observe that the solutions of the blue army made more drastic genetic modifications which clearly can be interpreted as an attempt to explore alternative regions in the search space, although finally this was a frustrated attempt.

Regarding the average number of game turns needed in each battle, an analysis was done for each scenario. As shown in Equation (2), the fitness function rewards those solutions which require fewer game turns for beating the opponent, and those which also generate fewer casualties in relation to the number of dead units of its rival army. The result of this analysis was that the new strategies found require fewer turns to defeat the opponent. In the two initial strategy options (*offensive* and *random*) both armies tend to minimize the number of turns used for each new strategy encountered.

We also analyzed the difference between the number of units which survive at the end of battles in each of the armies. We note that in the case of setting an *offensive* initial strategy all these differences were positive, which means that all winning solutions have more survivors than their corresponding opponent strategies. This is a logical result because all winning solutions tend to reach victory by means of the destruction of all units of the rival, although the reality is that the victory can be obtained without necessarily destroying all the opponent's units (for example, by simply destroying its general). In this way it is perfectly possible and correct to find solutions with negative differences favorable to the opponent. In fact, we have seen that despite all differences being positive, in general this difference tends to decrease in the new strategic solutions, which means that as strategies are refined it becomes more difficult to find solutions to overcome the rival without affecting the integrity of the units, forcing these solutions to sacrifice an increasingly larger number of units to achieve the desired military victory. This decrease in the difference between surviving units denotes some convergence towards more strategic results, where solutions are more balanced between them. This balance tends to make zero the difference between the number of casualties suffered by both armies during the battles.

On the other hand, for the case of setting a *random* initial strategy, we observed that both armies found strategic solutions which generated a greater number of casualties on the opponent in every coevolutionary step. And this continues being a positive outcome because it is interpreted as the finding of more and more effective strategies that achieve victory generating further damage to the opponent.

RESULTS OF VERSION B

For analyzing the results obtained with version B of the algorithm, Figures 6 and 7 show the behavior of the best fitness found in each coevolution for both types of initial enemy strategy (*offensive* and *random*), respectively. These battles took place in the same scenario that we saw previously for Figures 4 and 5.

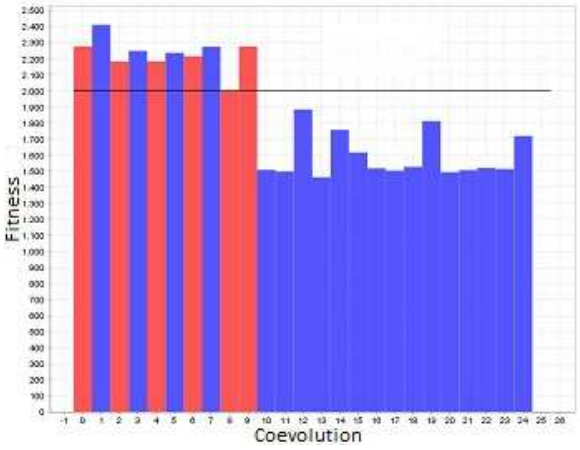


Figure 6: Version B: Offensive intial strategy; Best fitness

In conclusion we do not see any detail that shows an improvement in the succession of coevolution returned by this version of the algorithm. Specifically, there is a very similar sequence of coevolution between the results obtained by both versions (A and B), and they coincide even in the army which reached stagnation (the blue one) because it cannot overcome the opponent strategy.

In the case of equality percentage among the solutions nothing unusual was appreciated, although the percentages in the version B were lower due to the more drastic changes which were made in the genotype of the solutions found; this might even be interpreted as a greater difficulty for algorithm B to find winning solutions.

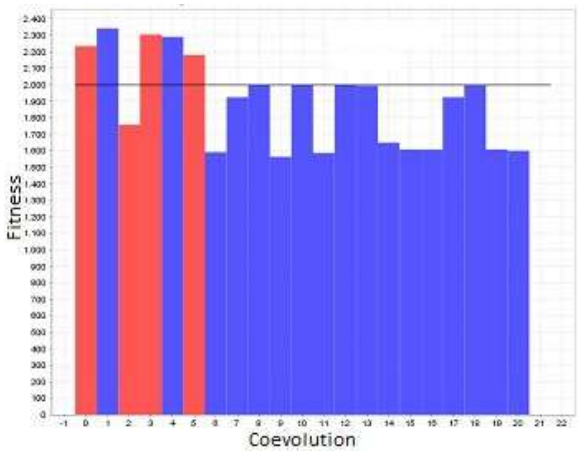


Figure 7: Version B: Random intial strategy; Best fitness

SUMMARY OF THE RESULTS FROM BOTH VERSIONS

In summary, using version B we note that the armies are able to find more refined solutions, and sometimes solutions that tip the scales of victory over the army which in version A was considered a loser are found. In general, version B produced a fitness evolution very similar to that of version A, although sometimes it adds an extra level of coevolution which provides a new solution that was not found by A. This shows that the way to model the solutions in version B where the genes are only destined to represent the behavior of military units which actually participate in the battle, makes genetic operators more accurate and effective, which likely results in a greater ability to explore solutions that version A could not reach. This conclusion was somewhat expected, because the adaptation of the strategy matrix to the context of the battle reduces the complexity of the problem, and thus reduces the number of possible solutions, discarding those solutions that really were useless, and this issue has a significant impact on the effectiveness of version B with respect to that shown by version A.

Tables 1 and 2 summarize the results obtained by both algorithms when they were executed on the 16 scenarios initially considered in the experiments (see Section). Each battle is identified by the initial opponent strategy adopted (first row), the armies formation (columns 1 and 2), the map where the battle was executed (i.e. Map 1 - without obstacles -, or Map 2 - with obstacles), and the winner army (red, or blue).

Observe that the blue army was affected the most because it stagnated (i.e. did not find better solutions) in 24 out of 32 cases. If we examine the results with respect to the initial enemy strategies, we observe that the blue army failed in 14 out of 16 cases when the initial strategy is *offensive*; however, when the initial strategy is *random* the results are more than compensated. This indicates that the selection of the initial opponent has a strong influence on the results.

From an orography point of view, and considering the two battles between *tortoise* and *guerrilla* formations, the *tortoise* formation tends to dominate in Map 1, which is completely free of obstacles, whereas in Map 2, which has a more complex topology, the *guerrilla* formation is the dominant; this result forms the idea that this type of formation are more suitable for fighting in areas with more obstacles.

Table 1: Algorithm HofCC (Version B): Summary of the results

Formation		Initial Offensive		Initial Random	
Blue army	Red army	Map 1	Map2	Map 1	Map2
Tortoise	Tortoise	Red	Red	Blue	Blue
Tortoise	Guerrilla	Red	Red	Red	Blue
Guerrilla	Tortoise	Blue	Red	Blue	Red
Guerrilla	Guerrilla	Red	Red	Red	Red

Table 2: Algorithm HofCC (Version B): Summary of the results

Formation		Initial Offensive		Initial Random	
Blue army	Red army	Map 1	Map2	Map 1	Map2
Tortoise	Tortoise	Red	Red	Blue	Red
Tortoise	Guerrilla	Red	Red	Red	Red
Guerrilla	Tortoise	Blue	Red	Blue	Red
Guerrilla	Guerrilla	Red	Red	Red	Red

SIMULATION OF THE BATTLE OF CARRHAE

This section is devoted to showing, by an example, how the HofCC engine can also be used to argue (or guess) what could have happened initially to obtain a certain final result. Here we consider the Battle of Carrhae, an historical battle that took place in the year 53 B.C. near the town of Carrhae; it was an important battle between the Parthian Empire and the Roman Republic. The Parthian Spahbod Surena decisively defeated a Roman invasion force led by Marcus Licinius Crassus. It was the first of the battles between the Roman and Persian empires, and one of the most crushing defeats in Roman history. Table 3 shows some data (i.e. statistics) associated with this battle[4].

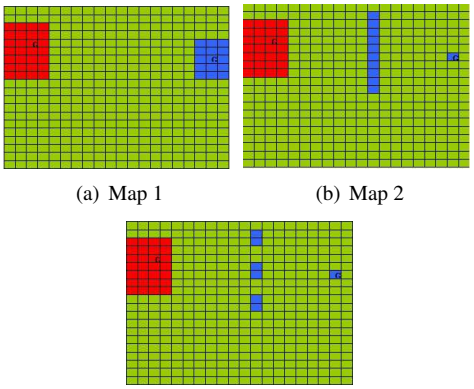
We have modeled three virtual scenarios for the battle where we considered a map free of obstacles as we assume that these should have weight in the strategic decisions. The red army represents the Roman army whereas the blue army represents the Parthian army (less numerous than the Roman infantry but more powerful). The real types of soldiers employed in the Battle (i.e. legionaries, cavalry, horse archers, etc) were adapted to our game by an intuitive mapping of the

[4] `en.wikipedia.org/wiki/Battle_of_Carrhae`, April 2012.

Table 3: Dates of The Battle of Carrhae

Roman Empire	**Parthian Empire**
Strength	**Strength**
35,000 legionaries 4,000 cavalry, 4,000 light infantry	9,000 horse archers, 1,000 cataphracts
Casualties and losses	**Casualties and losses**
20,000 dead, 10,000 captured, 10,000 escaped	$\sim$ 100

type of units built-in inside *robotWars*, and for the simulation we respected the ratio between the number of units belonging to both armies. Version B of HofCC algorithm was applied in three different scenarios for the battle (shown in Figure 8).



(a) Map 1 (b) Map 2

(c) Map 3

Figure 8: Battle of Carrhae: Scenarios for the simulation

In the first simulated battle (Map 1 in Figure 8) the romans have infantry units with displacement velocity and shot scope set by default, whereas as the Parthian army makes mainly use of cavalry units. Figures 9 and 10 show the results, in terms of best fitness value found in each evaluation, found by considering first an *offensive* initial opponent strategy, and second a *random* one, respectively. Observe in Figure 9 that the fitness evolution resembles the historical reality.
In the case of *random* strategy, the Romans find a victory in the second turn of coevolution, perhaps as a consequence of an imperfection in the strategic defensive formation of the Parthian cavalry that has its general completely unprotected. After this, the blue army takes total control of the battle and the Romans are not able to find a winning strategy, and here, the fitness evolution resembles the historical reality.
In the second map we reduced the number of Parthian units (from 14 to 10) given as result a ratio of 3 to 1 (in favor to the Romans); even in these circumstances the Parthians won the battle as the fitness evolution was quite similar to the results shown previously. In the simulation of this game, the

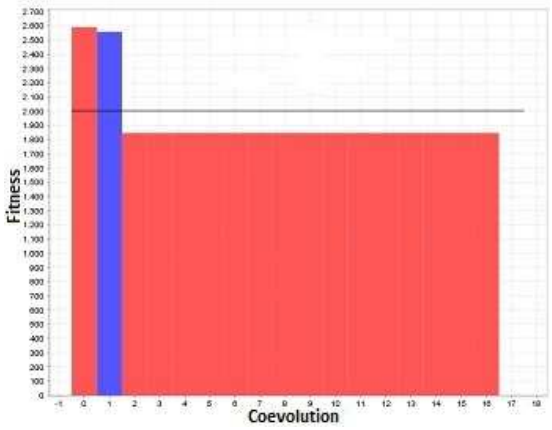


Figure 9: Best fitness with *ofensive* initial strategy

Parthian army executed approaches towards the Roman units followed by quick attacks carried out from a certain distance to avoid a direct confrontation with the Roman infantry units. This scenario represents with certain accuracy the historical events that really happened.
In the third scenario, we imposed a ratio of 5 to 1 by reducing by even more the number of Parthian units. However, in this case, the Parthian cavalry was insufficient for beating the Roman army that easily got the victory.

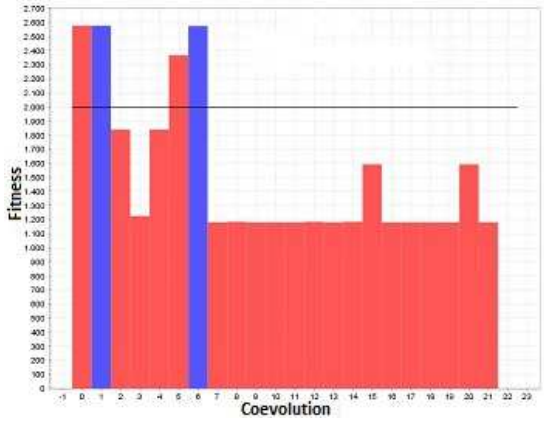


Figure 10: Best fitness with *random* initial strategy

CONCLUSIONS AND FUTURE WORK

This paper has dealt with the employment of competitive coevolution (CC) in RTS games. We have first presented a new RTS game called *robotWars* that can be used as a platform for researching with CC. We have also described a coevolutionary algorithm that has been implemented in the game described and that has experimentally shown promising results for searching winning strategies in this game.
We have also shown, through an example, the ability of the coevolutionary algorithm to reason about historical events according to known results. In the future we plan to investigate if this ability can be extrapolated to show a predic-

tive capacity, even in other areas outside the game arena that consider complex scenarios (with emergent properties for instance).

We have also identified some weaknesses on which we must work to improve our coevolutionary model. For this, let us begin by exploring the use of the Hall-of-Fame, which has the great advantage that it very easy to implement, also ensures an evaluation mechanism that takes into account previous champions (i.e. a memory mechanism), and favors the transitivity between the strategies of the hall. However, these benefits can be dulled by the efficiency of the algorithm which is negatively affected when the size of the hall grows.

ACKNOWLEDGEMENTS

This work is partially supported by Spanish MICINN under project ANYSELF (TIN2011-28627-C04-01), and by Junta de Andalucía under project P10-TIC-6083 (DNEMESIS).

REFERENCES

Angeline P. and Pollack J., 1993. *Competitive environments evolve better solutions for complex tasks*. In *Proceedings of the Fifth International Conference on Genetic Algorithms*. San Mateo, California, 264–270.

Avery P.; Greenwood G.; and Michalewicz Z., 2008a. *Coevolving strategic intelligence*. In *Evolutionary Computation. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*. IEEE, 3523–3530.

Avery P. et al., 2008b. *Coevolving a computer player for resource allocation games: using the game of Tempo as a test space.* Ph.D. thesis, School of Computer Science University of Adelaide.

de Jong E., 2004. *Towards a bounded Pareto-Coevolution archive*. In *Evolutionary Computation, 2004. CEC2004. Congress on*. IEEE, vol. 2, 2341–2348.

De Jong E.D. and Pollack J.B., 2004. *Ideal Evaluation from Coevolution*. *Evol Comput*, 12, no. 2, 159–192. ISSN 1063-6560.

Dziuk A. and Miikkulainen R., 2011. *Creating intelligent agents through shaping of coevolution*. In *Evolutionary Computation (CEC), 2011 IEEE Congress on*. IEEE, 1077–1083.

Ficici S., 2004. *Solution concepts in coevolutionary algorithms*. Ph.D. thesis, Brandeis University.

Ficici S. and Pollack J., 2000. *A game-theoretic approach to the simple coevolutionary algorithm*. In *Parallel Problem Solving from Nature PPSN VI*. 467–476.

Ficici S.G. and Bucci A., 2007. *Advanced tutorial on coevolution*. In *Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation*. ACM, New York, NY, USA. ISBN 978-1-59593-698-1, 3172–3204.

Jaskowski W. and Krawiec K., 2010. *Coordinate System Archive for Coevolution*. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*. IEEE, 1–10.

Johnson R.; Melich M.; Michalewicz Z.; and Schmidt M., 2004. *Coevolutionary tempo game*. In *Evolutionary Computation. CEC'04. Congress on*. vol. 2, 1610–1617.

Jong E., 2007. *A monotonic archive for pareto-coevolution*. *Evolutionary Computation*, 15, no. 1, 61–93.

Lichocki P. et al., 2008. *Evolving players for a real-time strategy game using gene expression programming*. Master's thesis, Poznan Universtity of Technology.

Rosin C. and Belew R., 1995. *Methods for competitive coevolution: Finding opponents worth beating*. In *Proceedings of the Sixth International Conference on Genetic Algorithms*. San Francisco, CA, 373–380.

Rosin C. and Belew R., 1997. *New methods for competitive coevolution*. *Evolutionary Computation*, 5, no. 1, 1–29.

Sims K., 1994. *Evolving 3D morphology and behavior by competition*. *Artificial life*, 1, no. 4, 353–372.

Smith G.; Avery P.; Houmanfar R.; and Louis S., 2010. *Using co-evolved RTS opponents to teach spatial tactics*. In *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on*. Citeseer, 146–153.

Watson R. and Pollack J., 2001. *Coevolutionary dynamics in a minimal substrate*. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001)*. Morgan Kaufmann, 702–709.

Wiegand R.; Liles W.; and De Jong K., 2002. *Analyzing cooperative coevolution with evolutionary game theory*. In *Evolutionary Computation. CEC'02. Proceedings of the 2002 Congress on*. vol. 2, 1600–1605.

Yang L.; Huang H.; and Yang X., 2009. *A Simple Coevolution Archive Based on Bidirectional Dimension Extraction*. In *Artificial Intelligence and Computational Intelligence, 2009. AICI'09. International Conference on*. IEEE, vol. 1, 596–600.