An Analysis of Hall-of-Fame Strategies in Competitive Coevolutionary Algorithms for Self-Learning in RTS Games

Mariela Nogueira, Carlos Cotta, and Antonio J. Fernández-Leiva

University of Computers Science, Havana, Cuba
University of Málaga, Málaga, Spain
mnogueira@uci.cu,{ccottap, afdez}@lcc.uma.es

Abstract. This paper explores the use of Hall-of-Fame (HoF) in the application of competitive coevolution for finding winning strategies in *RobotWars*, a two-player real time strategy (RTS) game developed in the University of Malaga for research purposes. The main goal is testing different approaches in order to implement the concept of HoF as part of the self learning mechanism in competitive coevolutionary algorithms. Five approaches were designed and tested, the difference between them being based on the implementation of HoF as a long or short-term memory mechanism. Specifically they differ on the police followed to keep the members in the champions' memory during an updating process which deletes the weakest individuals, in order to consider only the robust members in the evaluation phase. It is shown how strategies based on periodical update of the HoF set on the basis of quality and diversity provide globally better results.

Keywords: coevolution, RTS game, game's strategy, evaluation process, memory mechanism

1 Introduction

Artificial Intelligence (AI) implementation represents a challenge in game development: a deficient game AI surely decreases the satisfaction of the player. Game AI has been traditionally coded manually, causing 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. Coevolution, a biologically inspired technique based on the interaction between different species, represents one of the most interesting techniques to this end.

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 cooperation-based approaches, an individual is typically decomposed in different components that evolve simultaneously and the fitness depends on the interaction between these components; in competition-based approaches, an individual

competes with other individuals for the fitness value and, if appropriate, will increase its fitness at the expense of its counterparts, whose fitnesses decrease. This latter approach resembles an army race in which the improvement of some individuals causes the improvement in others, and vice versa.

This paper deals with the application of competitive coevolution (CC) as a self learning mechanism in RTS games. As a first contribution, the paper analyzes the performance of different approaches in order to apply the concept of Hall-of-Fame (HoF) defined by Rosin and Belew in [1] as a long-term memory in competitive coevolutionary algorithms; the analysis is conducted in the context of the real-time strategy (RTS) game *RobotWars*. The goal is to produce automatically game strategies to govern the behavior of an army in the game that can also beat its opponent counterpart. As a second contribution this work proposes alternatives for optimizing two key aspects in the implementation of the HoF, which are, the diversity of the solutions, and the growth of the champions' memory.

This paper is organized as follows. Next, –and given that we focus in this work on exploring different variants of HoF as an evaluation and memory mechanism in competitive coevolutionary settings– we present an overview of competitive coevolution in games. In Section 3 we explain the game which is our arena for competitive coevolution. Section 4 describes ours variants for implementing a Hall-of-Fame based competitive coevolutionary algorithm. In Section 5 we analyze the results obtained by each variants in many experiments. And finally in Section 6 we closure this investigation.

2 Background on Competitive Coevolution in Games

Coevolution has been shown to be successful on a number of applications but it also has a number of drawbacks [2]. The primary remedy to cope with the inherent pathologies of coevolution consists of proposing new forms of evaluating individuals during coevolution [1], and memorization of a number of successful solutions to guide the search is one of the most employed. Following this idea, [3] already proposed the use of a *Hall-of-Fame* (HoF) based mechanism as an archive method and, since then, there have been similar proposals such as those described in [4], [5] and [6]. According to [7] the question of how to actually use the memory in the coevolution tends to fall into two areas: inserting individuals from memory into the coevolution, or evaluating individuals from the populations against the memory. Precisely, our investigation fits to this latter area, we have implemented different variants of Hall-of-Fame for controlling evaluation process in a CC algorithm, which tries to find wining strategies for the game *RobotWars*.

Several experiments have showed significants results in the application of coevolutionary models in competitive environments; for example the study described in [8] on competitive fitness functions in the Tic Tac Toe game, the application of simple competitive models for evolving strategies in a pursuit-evasion game [9], or the evolution of both morphology and behaviors of artificial creatures through competition in a predator-prey environment [10]. Competi-

tive coevolution continues to be useful nowadays, and has been used heavily in complex scenarios like those that emerge in strategy games; so, [11] coevolved artificial intelligent opponents with the objective of training human players in the context of a game of type capture-the-flag. Also, [12] analyzed the employment of coevolution for creating a tactical controller for small groups of game entities in a real-time capture-the-flag game; a representation for generating adaptive tactics using coevolved Influence Maps was proposed, and the result was the attainment of an autonomous entity that plays in coordination with the rest of the team to achieve the team objectives. More recently, [13] explores several methods for automatically shaping the coevolutionary process, and this is done by modifying the fitness function as well as the environment during evolution.

Other research that addresses the use of the HoF concept as an evaluation mechanism in a competitive-coevolutive environment was given by Pawel Lichocki in [14]. He implemented the HoF with three useful extensions included: uniqueness, manual teachers, and Competitive Fitness Sharing [3]. The results of this work showed that HoF works better than SET (Single Elimination Tournament) [8], but this method was not sufficient to prevent the lack of diversity in the population. In our variants of HoF the probability of a repeated member being inserted in the memory is minimal, because we do coevolutions by turns of two independent populations, and each coevolutive turn begins with a new population which evolves until finding an unique champion, or reaching the maximum number of continuous coevolution without success; and this champion must defeat all the members of the opponents' HoF. The use of manual teachers is also possible in our system: the first coevolutive iteration may be started with *random* or *offensive* –manually defined– strategies..

Another interesting perspective was presented in [15] where the authors using the game of Tempo as a test space, tried to ease the selection of optimal strategies by clustering the solutions in the population of a coevolutionary system through the concept of similarity. This cluster system integrated a long-term memory that valued the changes produced in the environment to trigger appropriate coevolution. The game of Tempo has also been used with the aim of improving the creation of smart agents in [16] and [17].

3 Game Description

This section is devoted to *RobotWars*[1], our arena for competitive coevolution which will be presented for first time in [18]. *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.

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

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) and the army that first wipes out the enemy general is considered the winner. 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 friendly general, and 1 otherwise), the third dimension to *numeric advantage* (i.e. are there, in a nearby space, more friendly units than enemy units? A binary answer:yes/no), and the fourth dimension to *health* (i.e. an amount that indicates the health level as high, medium or low). Each position of the matrix acts as a gen and stores one of the following 5 actions: *attack*, *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.

4 Hall-of-Fame based Competitive Coevolutionary Algorithm and Variants

Our objective is to apply competitive coevolution techniques to automate the generation of victorious strategies for the game described above. According to the strategies encoding shown in the previous section, the search space is $5^{7\times2\times2\times3} = 5^{84}$, which is really huge, and cannot be efficiently explored using implicit enumeration techniques due to the inherently complex and non-linear behavior of game simulations. Thus, the use of metaheuristic techniques is approached.

Using our RTS game we test five variants of a competitive coevolutionary algorithm that uses the HoF as a memory mechanism to keep the winning strategies found in each coevolutionary step, to this end the best individual from each generation is retained for future testing. In our approach, each army (i.e. player) maintains its own HoF, in which its own winning strategies (with respect to the set of winning strategies of its opponent) found in each coevolutionary step will be saved.

Regarding the use and implementation of HoF some aspects must be defined. The first is the criteria for inserting a new member in the memory. Also we have

considered different policies for maintaining the champions in the set, regarding this issue one has to take into account the contribution of the individual (i.e., the champion) to the search process as, for instance, it might be the case that some opponents that belong to very old generations do not show a valuable performance in comparison with opponents generated in recent generations and thus they might be easily beaten; it is therefore crucial to remove those champions not contributing to the solution what, in other words, represents a mechanism to control the size of the champions' memory. Another relevant aspect concerns to the selection of those strategies from HoF that will be employed in the evaluation process; considering all the champions might produce more consistent solutions at the expense of a very high computational cost (note that a simulation of the match must be executed for each champion involved in the evaluation; we will provide more details on this further on). Next we present our HoF-based competitive coevolutionary algorithm (HofCC) and five variants that precisely differ in the policy of establishing the aspects mentioned previously.

4.1 Basic HofCC

Algorithm 1 shows the schema of our basic algorithm HofCC. A specific 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 enemy 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 have 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.

Regarding to the evaluation of candidates for a specific player p (where $p \in \{player\ 1, player\ 2\}$), the fitness of an specific strategy is computed by facing it against a selected subset of the (winning) strategies in the Hall-of-Fame of its opponent player (that we call the *selected opponent set*). Given a specific strategy s its fitness is computed as follows:

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

where $k \in \mathbb{N}$ is the cardinality of the selected opponent set, $p_j^s \in \Re$ returns ϕ points if strategy s beats strategy h_j belonging to the selected opponent set (i.e.

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 |  |  i ← i + 1;
14 |  end while
15 |  if fitness(best(pop)) ≥ φ then //winner found!
16 |  |  nCoev ← 0; // start new search
17 |  |  HoF_A ← HoF_A ∪ {best(pop)}
18 |  |  temp ← A; A ← B; B ← temp; // interchange players' roles
19 |  else
20 |  |  nCoev ← nCoev + 1; // continue search
21 |  end if
22 end while
```

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 $\Delta Alive_{sj} \in \mathbb{N}$ 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.

4.2 HofCC variants

This section is devoted to describe five variants of our HofCC algorithms; basically these variants differ in the nature (i.e., in this case, size) of the HoF when it is used as a memory mechanism, ranging from short-term memory versions to long-term memory instances.

4.2.1 HofCC-Complete: in this variant the Hall-of-Fame acts as a long-term memory by keeping all the winners found in previous coevolutions, and all of them are also used in the evaluation process. So, in the coevolutionary step n each possible solution of army A fight again each solution in $\{B_1, B_2, ...B_{n-1}\}$, where B_i is the champion found by army B in the i (for $1 \leq i \leq n-1$) coevolutionary step.

Note that the cardinality of the selected opponent set and the cardinality of the HoF in the coevolutionary step n are equal (i.e., $k = n$ in Equation 1).

4.2.2 HofCC-Reduced: here the HoF acts as the minimum short-term memory mechanism by minimizing the number of battles required for evaluating an individual; that is to say, in the n co-evolutionary step, each individual in army A only faces the latest champion inserted in the HoF of army B (i.e., B_{n-1}). Note that this means $k = 1$ in Equation 1.

4.2.3 HofCC-Diversity: in this proposal the HoF acts as a long-term memory mechanism, but the content of the HoF is updated by removing those members that provide less diversity. The value of *diversity* that an individual in the HoF provides is calculated by the genotypic distance as follows: we manage the memory of champions as a matrix in which each row represents a solution and each column a gen (i.e., an action in the strategy). Then, we compute the entropy value for a specific column j as follows:

$$H_j = -\sum_{i=1}^{k}(p_{ij} \log p_{ij}) \tag{3}$$

Where p_{ij} is the probability of action i in column j, and k is the length of the memory. Finally the entropy of the whole set is defined by the following formula:

$$H = \sum_{j=1}^{n} H_j \tag{4}$$

The higher the value of H the greater the diversity of the set. For determining the diversity's contribution of a specific solution, we calculate the value of entropy with this solution inside the set, and the corresponding value with this solution out of the set, and finally, the difference of these two values represents the contribution of diversity.

The number of individuals to be deleted from the memory should be set by the programmer as a percentage value (α) representing the portion of the HoF to be removed; in other words, the HoF (with cardinality $\#HoF$) is ordered according to the diversity value in a decreased order and the last $\ulcorner \frac{\#HoF \times n}{\alpha} \urcorner$ individuals in this ordered sequence are removed. The frequency of updating (λ) is also a parameter of this version (i.e., the HoF is updated every λ coevolutions)

The motivation of this proposal is to maintain certain diversity among the members of the HoF, and at the same time to reduce (or maintain an acceptable value for) the size of the memory. With this idea, we assume that the deleted individuals will not affect the quality of the found solutions.

Here, the cardinality of the selected opponent set k in the evaluation phase (see Equation 1) is the cardinality of the opponent HoF after executing the updating of the memory (i.e., after removing the individuals).

4.2.4 HofCC-Quality: in this version, we follow a similar approach to that applied in HofCC-Diversity but now the HoF is ordered with respect to a measure of quality that is defined as the number of defeats that an individual obtained

in the previous coevolutionary step; in other words, a simple counter variable associated with each member of the HoF stores the number of defeats that were computed for the corresponding member during the evaluation process of the opponent army in the previous coevolutive turn.

Based on our game experience, we assume that this metric is representative of the strength of a solution, and the aim is to keep only the robust individuals in the champions' memory by removing the weak strategies.

As in the HofCC-Diversity, the parameters α and λ have to be set, and the cardinality of the selected opponent set k is exactly the same.

4.2.5 HofCC-U: this variant of HofCC follows the idea of optimizing the memory of champions, but in this case we propose a multiobjective approach where each solution has a diversity value and also a quality value as described previously associated with it. Then, a percentage value (α) from the set of dominated solutions according to the multiobjective values is removed; if the set of dominate solutions is empty then HoF is ordered according to the measure of quality and the solutions with worst quality will be removed.

As in the previous algorithms (HofCC-Quality and HofCC-Diversity) the frequency of updating the HoF is an important parameter that must be defined.

This proposal uses a different fitness function (to that shown in Equation 1) whose definition was inspired by the Competitive Fitness Sharing(CFS) [3]. The main idea is that a defeat against opponent X has more importance if there are other individuals that defeated X. So, a penalization value N for each individual i (for $1 \leq i \leq k$) in the population is then calculated as follows:

$$N_i = 1 - \frac{\sum_{j=1}^{k} \frac{v_{ij}}{V(j)}}{k} \tag{5}$$

where $v_{ij} = 1$ is the individual i of the population defeats the strategy (or champion) j in the HoF (whose cardinality is k) and 0 otherwise; and

$$V(j) = \sum_{i=1}^{n} v_{ij}$$

is the number of individuals in the population which defeat opponent j of the HoF. As a consequence, $N_i \approx 0$ if the candidate i defeats all opponents of HoF and the solution i itself is one of the few candidates to do so; $N_i = 1$ if it defeats no opponent; and $0 < N_i < 1$ depending on how many times it wins and how common is to beat certain opponents. The fitness of a candidate i is then computed as follows:

$$F_i = P_i - cN_i \tag{6}$$

where P_i is the result obtained in the battles by Equation 1, and $c \in \mathbb{N}$ is a coefficient that scales N_i in order to make it meaningful with respect to value P.

5 Experiments and Analysis

Due to the high computational cost to execute the experiments we have considered a unique battle scenario without obstacles and in which the formation (i.e., morphology) of armies is the type of *tortoise vs. tortoise*, and the initial predefined enemy strategy is *random*.

The five variants of the HofCC described in previous section has been considered for the experiments.

5.1 Configuration of the Experiments

Eleven instances of our algorithms were used: one for HofCC-Complete (HofC); one for HofCC-Reduce (HofR); and three for each of the HofCC-Diversity, HofCC-Quality, and HofCC-U varying according to the values of $\alpha = 10\%$ (i.e., *(HofDiv-10, HofQua-10, HofU-10)*, $\alpha = 30\%$ (*(HofDiv-30, HofQua-30, HofU-30)*), and $\alpha = 50\%$ (*(HofDiv-50, HofQua-50, HofU-50)*). Also we set $\lambda = 3$ for all the versions of HofCC-Diversity, HofCC-Quality, and HofCC-U.

Ten runs per each algorithm instance were executed, and in all of them we have used a steady-sate genetic algorithm (GA - note that this corresponds to 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, bit-flip mutation, elitist replacement, $MaxCoevolution = 15$ (it represents the numbers of continuous coevolutions in the same army without finding a champion solution), $Maxgenerations = 300$, population size was set to 100, the limit of evaluation was 230000 (i.e., the timeout value), 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 (6) to 200, a representative value of the range of scores that were obtained from the battle simulator after executing a number of games.

Our analysis has been guided by the following indicators which are applied for all runs of each algorithm: *Best fitness*: shows the best value of fitness found by the search process; *Average fitness*: shows the average fitness value reached in the coevolutive cycle; *Number of evaluations*: indicates the total number of battles which are realized during the evaluation process; *Number of defeats*: indicates the total number of defeats obtained in an *All(vs)All* fighting among the best solutions found by each version of algorithm.

In what follows, we analyze the results obtained in ten independent executions for the eleven versions of HofCC, and focus on the mentioned indicators; we have used the well-known non-parametric statistical tests to compare ranks namely Kruskal-Wallis test [19] with a significance of 95%. When this test detects significant differences in the distributions, we have performed multiple tests

using the Dunn−Sidak method [20] in order to determine which pairs of means are significantly different, and which are not. Next, the results obtained in the experiments for each indicator are presented.

5.2 Results of average fitness

Figure 1 shows the behavior of the average fitness for each algorithm instance. In this figure the algorithms are sorted according to the median of the distribution (i.e., the vertical red line). The Kruskal-Wallis test confirms that the differences between values are statistically significant (see the first row in Table 1). The HofR algorithm reaches the worst results for this indicator, such results may be a sign that this algorithm does not exploit the search space sufficiently because in this version the HoF acts as a short memory mechanism, whereby it is easier to find a champion than for the rest of the algorithms. Moreover, note that the best results are obtained by algorithms which optimize the use of HoF (in terms of diversity, quality, or both), and at the same time do not reduce too significantly the memory size; note also that the average fitness value decreases in those cases where the HoF suffered a reduction of 50% during the updating process. In the results of multiple tests for the value of average fitness, the HofR distribution has significant differences respect to the majority of the algorithms (except HofU-50, and HofQua-50), and the rest of the versions have a similar behavior.

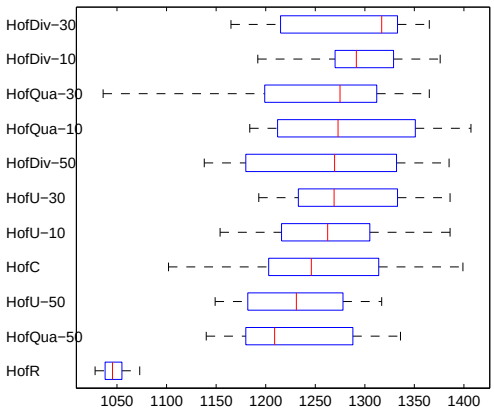

Fig. 1. Average Fitness obtained in each algorithm

Table 1. Results of Kruskal-Wallis test for all the indicators ($p$value< 0.05).

Indicator	$p$value
Average fitness	$2.6205E-004$
Best fitness	0.0175
Number of evaluations	$0,2214$
Numbers of defeats	$5.6909E-007$

5.3 Results of best fitness

Figure 2 shows the results of best fitness for each algorithm in the executions. Note that HofR finds the higher values. This situation can be generated by the fact that in HofR the individuals compete only against one opponent during the evaluation process, therefore it is easier to obtain higher scores. For the rest of the algorithms a high value in this indicator may be given by a further intensification in the search. As another interesting point, note that in many cases the algorithms with poor results in terms of average fitness, have good results here; except in the case of the HofDiv-10 and HofU-30 which maintain good results in both analysis.

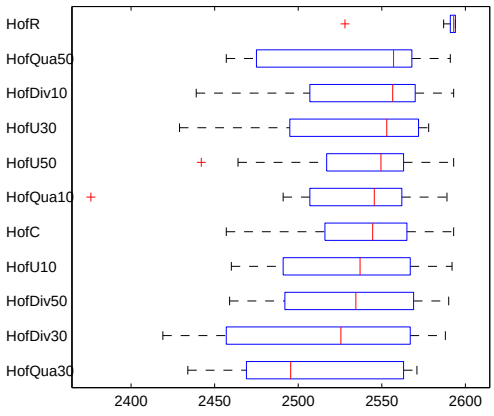

Fig. 2. Best fitness obtained by each algorithm

Table 1 (row 2) displays the results computed by Kruskal-Wallis' test confirming there are significative differences among the distributions. The results of multicompare test shows that the HofQua-30 and HofDvi-30 have relevant differences with respect to the HofR; the rest of the algorithms have similar values.

5.4 Results of number of evaluations

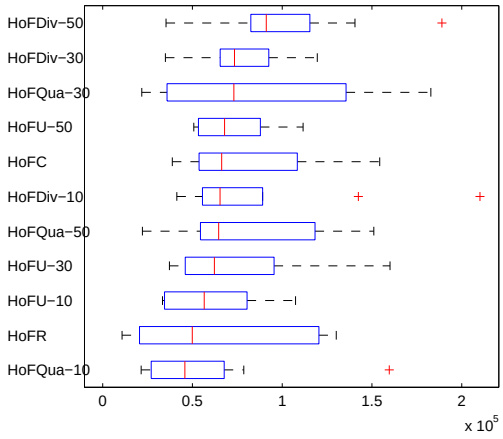

Fig. 3. Numbers of evaluations employed by each algorithm

For the case of the number of evaluations the results are shown in Figure 3 and according to the statistical tests performed (see Table 1), there is no significant statistical difference in the distribution data. In this indicator we noted that the increasing in the number of evaluations is in consonance to the length of the coevolutive cycle; except in the case of HofR which presents a very long cycle and has no influence because during the evaluation process of this algorithm the individuals face a single opponent, and this decreases the number of evaluations significantly. Consider that the coevolutive cycle's length is determined by the number of coevolutions that use the algorithm to find an undefeated champion (i.e a member of the HoF which can not be defeated by the opponent side), and it helps to identify whether the problem difficulty increases as best solutions are obtained, or if it remains stable. In all algorithms (except HofR) the rigor of the competition increases until it reaches the point at which the algorithm can not exceed the level of specialization achieved. However, in HofR the cycles were very long, because the quality of the solutions was stagnated; and it was necessary to limit the length of cycles up to 500 iterations.

5.5 Results of number of defeats

For this test, the last champions (i.e. the last member added to the HoF) found by each algorithm instance (in each execution) fought in an *All versus All* tournament. The results with respect to the amount of defeats are shown in Figure 4; and Table 1 (row 4). The main differences are in the values of HofR, and HofC which still maintain the same poor results as the previous indicators. On the

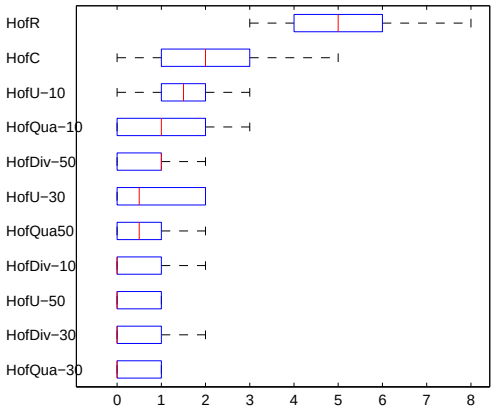

Fig. 4. Numbers of defeats obtained by each algorithm in an *All (vs) All* tournament

other hand, HofDiv-10 and HofDiv-30 again obtains the best values. Curiously, the HofQua-30 which had the worst results in the analysis of best fitness has a low ranking of defeats here, this is certainly an indicator that the fitness measure used is insufficient. The instances of HofQua-10 and HofU-10 have a similar behavior with high numbers of defeats. Another detail that attracts attention is that variants that reduce the HoF by 50% in the previous indicators have not shown encouraging results, except for the HofDiv-50, however in this analysis we can see that they are in the middle top of the ranking.

5.6 Summary of the results

We can conclude that, for all the experiments, the versions that incorporate updating the HoF were more efficient than HofR and HofC. In the case of the best fitness analysis the HofR shows higher values, however in the fighting tournaments the strategies generated by this algorithm were the weakest, so we can say that the fitness function is not sufficiently representative of the individuals' strength, and undoubtedly the loss of transitivity in this algorithm causes a total disengagement of the search.

Regarding to the numbers of evaluations there are not significant differences, and here let us underline that the game used does not allow very long evolutionary cycles due to performance limitations. This means the HoF obtained is not large, so it is not possible to demonstrate the benefits of those algorithms which optimize the size of the HoF and in turn reduce the number of necessary evaluations.

By analyzing the families of algorithms we can conclude that the HofDiv's variants shown the best performance for all the indicators. In the fitness analysis HofDiv shown the best results, while the HofU and HofQua reached similar be-

havior. And as general rule, for these indicators, the variants which update HoF by 10% and 30% obtain a better performance; however in the tournaments, the algorithms which use a high percentage showed some slight advantages, although the differences between the versions that update the HoF are not statistically significant for any indicator.

We have also shown that the fitness values are not related with the solution robustness by executing fighting tournaments; this is a particular result which can be interpreted as follows. During the coevolutive cycle the search space is explored, but the self-learning mechanism falls into a local optimum and gets trapped there, so that the solutions found improve their fitness values without a global improvement in a more general context.

6 Conclusions and Future Work

The analyzed results allow us to conclude that whenever we update the memory of champions using a selection criteria the search process provides better solutions. In our experiments the updating of HoF using the *diversity* as the selection criteria showed the best performance. We also detected that affecting the transitivity among the solutions has a direct influence on the quality of the search result, and therefore, the removing of members in the champions' memory should be done carefully. However, it is important to decrease to avoid performance problems; in our experiments we did not suffer this, but there are cases in which the exhaustive exploration of the search space requires to significantly enlarging the size of the champions' memory what can affect the performance of the algorithm considerably.

As for future work we propose testing these algorithms in other games. Trying at that time to achieve a more complete representation of individuals by genetic encoding, and a better adjustment of the evaluation function. The games that we will choose should allow us to experiment with long coevolutionary cycles, to carry out a performance analysis of the algorithms.

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

1. Rosin, C.D., Belew, R.K.: Methods for Competitive Co-Evolution: Finding Opponents Worth Beating. In: ICGA. (1995) 373–381
2. Ficici, S.G., Bucci, A.: Advanced tutorial on coevolution. In: Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation, New York, NY, USA, ACM (2007) 3172–3204

3. Rosin, C., Belew, R.: New methods for competitive coevolution. Evolutionary Computation **5**(1) (1997) 1–29
4. de Jong, E.: Towards a bounded Pareto-Coevolution archive. In: Congress on Evolutionary Computation, CEC2004. Volume 2., IEEE (june 2004) 2341–2348
5. Jaskowski, W., Krawiec, K.: Coordinate System Archive for coevolution. [21] 1–10
6. Yang, L., Huang, H., Yang, X.: A simple coevolution archive based on bidirectional dimension extraction. In: Artificial Intelligence and Computational Intelligence, 2009. AICI'09. International Conference on. Volume 1., IEEE (2009) 596–600
7. Avery, P.M., Greenwood, G.W., Michalewicz, Z.: Coevolving strategic intelligence. In: IEEE Congress on Evolutionary Computation, IEEE (2008) 3523–3530
8. Angeline, P.J., Pollack, J.B.: Competitive Environments Evolve Better Solutions for Complex Tasks. In: ICGA. (1993) 264–270
9. Reynolds, C.: Competition, coevolution and the game of tag. In Brooks, Maes, P., eds.: Proceedings of Artificial Life IV, Cambridge, Massachusetts, MIT Press (1994) 59–69
10. Sims, K.: Evolving 3D Morphology and Behavior by Competition. Artificial Life **1**(4) (1994) 353–372
11. Smith, G., Avery, P., Houmanfar, R., Louis, S.J.: Using co-evolved RTS opponents to teach spatial tactics. In Yannakakis, G.N., Togelius, J., eds.: CIG, IEEE (2010) 146–153
12. Avery, P., Louis, S.J.: Coevolving team tactics for a real-time strategy game. [21] 1–8
13. Dziuk, A., Miikkulainen, R.: Creating intelligent agents through shaping of coevolution. In: IEEE Congress on Evolutionary Computation. (2011) 1077–1083
14. Lichocki, P.: Evolving players for a real-time strategy game using gene expression programming (2008) Master thesis, Poznan Universtity of Technology.
15. Avery, P.M., Michalewicz, Z.: Static experts and dynamic enemies in coevolutionary games. In: IEEE Congress on Evolutionary Computation. (2007) 4035–4042
16. Johnson, R., Melich, M., Michalewicz, Z., Schmidt, M.: Coevolutionary Tempo game. In: Evolutionary Computation. CEC'04. Congress on. Volume 2. (2004) 1610–1617
17. Avery, P., et al.: Coevolving a computer player for resource allocation games: using the game of Tempo as a test space. PhD thesis, School of Computer Science University of Adelaide (2008)
18. Nogueira, M., Gálvez, J., Cotta, C., Fernández-Leiva, A.: Hall of Fame based competitive coevolutionary algorithms for optimizing opponent strategies in a new RTS game. In Fernández-Leiva et al., A., ed.: 13th annual European conference on simulation and AI in computer games (GAMEON'2012), Málaga, Spain, Eurosis (November 2012) 71–78
19. Kruskal, W., Wallis, W.: Use of ranks in one-criterion variance analysis. Journal of the American Statistical Association **47**(260) (1952) 583–621
20. Sokal Robert, R., Rohlf James, F.: Biometry: the principles and practice of statistics in biological reseach. W.H. Freeman and Company, New York (1995)
21. Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2010, Barcelona, Spain, 18-23 July 2010. In: IEEE Congress on Evolutionary Computation, IEEE (2010)