Evolutionary design and statistical assessment of strategies in an adversarial domain

Pablo J. Villacorta and David A. Pelta

***Abstract*— Adversarial decision making is aimed at finding strategies for dealing with an adversary who observes our decisions and tries to learn our behaviour pattern. Departing from a simple mathematical model, the present contribution extends it with strategies that vary along time, and motivates the use of heuristic search procedures to address the problem of finding good strategies within this new search space. Evaluation of this new class of strategies requires running a stochastic simulation so the comparison of strategies should be properly addressed. A new statistics-based technique for comparison of strategies is also proposed and tested in this context when coupled with a Genetic Algorithm. Computational experiments showed that the new strategies are better than previous ones, and that the results obtained with this new comparison technique are encouraging.**

I. Introduction

Adversarial decision making is aimed at determining optimal strategies against an adversarial enemy who observes our actions and learns from them. This situation arises in many areas of real life, with particular (but not the only one) interest in counter-terrorist combat and crime prevention [1], [2].

The field is also known as decision making in the presence of adversaries and we may talk about problems within an adversarial domain where an "adversary" exists. Essentially, the focus is on technologies for opponent strategy prediction, plan recognition, deception discovery and planning, and strategy formulation that not only applies to security issues but also to game industry, business, transactions, etc. [3]. For example, patrolling strategies can be viewed as another application of adversarial decision making: the aim is to design routes for patrolling trying to minimize the chance that an enemy enters a security border.

Given two agents or entities S and T (the adversary), both want to maximize their rewards that are inversely related. One defense for S is to make decisions that are intended to confuse T, although this will affect the ability of getting a more optimal reward. The question for S is how to define his decision strategies. Manually designed strategies could be good, but the designer may omit interesting alternatives due to its inherent limited ability to search in the space of strategies.

In the last years, automatic design by means of evolutionary techniques is gaining increasing attention. For example, in the design of self-assembly systems [4], certain kind of neural networks [5], controllers for collective robotics [6], oriented-tree networks [7], just to cite a few.

In order to automatically design such decision strategies, three important features should be taken into account: how to represent a strategy, how two strategies should be compared, and how the space of the strategies should be searched.

In this contribution we mainly focus on the second feature and we slightly discuss the first and third ones. In this context, our aims are: (a) to propose a new way of systematically describing strategies for an adversarial model, (b) to propose a statistics-based methodology to the pairwise comparison of strategies, and (c) to analyze the behaviour of an evolutionary algorithm when coupled with such pairwise comparison technique.

This contribution is organized as follows: in section II, the model explained in [8] is briefly summarized. Section III explains a new way of describing strategies. In section IV, the use of a search algorithm in this new context of automatic design of tactics is introduced, and the need of a fast, yet reliable comparison technique for tactics is motivated. The feasibility of a statistical approach is also discussed here. Section V is devoted to computational experiments and analysis of results. Finally, conclusions and further work are discussed in Section VI.

II. Adversarial Reasoning Model

The model we are dealing with is based on two agents S and T (the adversary), a set of possible inputs or events $I = \{i_1, i_2, \ldots, i_n\}$ issued by a third agent R, and a set of potential responses or actions $A_i = \{a_1, a_2, \ldots, a_m\}$ associated with every event. 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 $p_{ij} \in [0, 1]$ is the reward or profit associated with action j to respond to the event i.

Agent S must decide which action to take given a particular input i_k 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.

Pablo J. Villacorta and David A. Pelta are with the Models of Decision and Optimization Research Group, Department of Computer Science and AI, University of Granada, Spain (phone: +34 958242376; email: dpelta@decsai.ugr.es, olbapjo@correo.ugr.es).

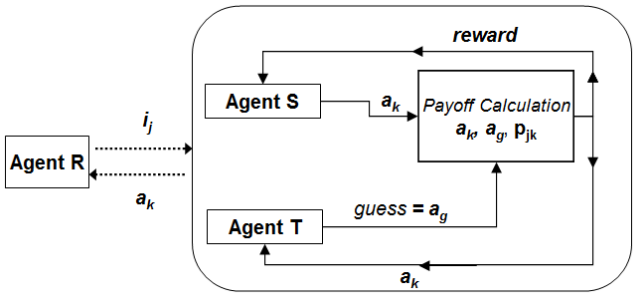


Fig. 1. Graphical representation of the model. Events i_j are issued by agent R while response or actions a_k are taken by agent W.

Algorithm 1 Sequence of steps in the model.

for $j = 1$ to E **do**
  A new input i_j arises.
  Agent T "guesses" an action a_g
  Agent S determines an action a_k
  Calculate payoff for S
  Agent T records the pair i_j, a_k
end for

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.

Given a new input i_j, 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 $M \times N$. O_{ij} stores the number of times that, in the past, agent S decided to take action i when the input was j.

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

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

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 at all when agent T has predicted his response properly.

The behaviour pattern of both agents can vary from a totally deterministic way to a totally random one. Agent T can use his observation matrix to make predictions in several manners, such as always choosing the most frequently observed action in the past, or a random action with a probability that is proportional to the number of times that action was observed in the past, etc. Agent S can always choose the action with the highest reward, but this behaviour is very easy to learn for an observer when repeated along time. He can also choose randomly or with a probability that is proportional to the payoff, or randomly among some of the actions with higher payoff. There exist many other patterns that try to balance payoff and confusion.

In this contribution, agent T applies a simple frequency-based decision strategy: the probability of selecting an action a_i for responding to event e_j is proportional to O_{ij}. Despite its simplicity, this strategy is one of the hardest for agent S as it was shown in [8].

III. A NEW SCHEME FOR STRATEGIES REPRESENTATION

In this section we describe one of the proposals of this work. We provide a novel and general way to define strategies, that also allows to model those strategies presented in [8]. The basic idea is to define a *tactic* or *plan* as a sequence of simpler decision strategies.

A strategy is described as a process with two stages. In the first stage, a number of different candidate actions are chosen into a "candidate set" according to certain criterion. The cardinal of the candidate set is a specific integer parameter $k \in [1, k_{max}]$. In the second stage, one single action is finally selected from the candidate set according to another different criterion. Therefore a strategy can be fully described in terms of three components: $< k >, < criterion >, < criterion >$.

The values available for $criterion$ are:

- BEST: picks the k best actions as a candidate set, according to their payoff indicated in the payoff matrix.
- RANDOM: picks k different actions in a totally random way.
- PROP-PAYOFF: picks k different actions randomly with a probability proportional to the payoff indicated in the payoff matrix for each action.

An example of strategy can be 4 | RANDOM | PROP-PAYOFF. It means that 4 different candidate actions are chosen randomly, and afterwards, one of them is chosen with a probability that is proportional to the payoff. Assuming, for instance, that our model has 5 different states, an extreme strategy would be 1- | BEST | BEST, which always selects the best action in response to any stimulus. The opposite extreme would be 5 | RANDOM | RANDOM which always selects one action totally randomly. In this case, the first stage of the strategy is useless because all the 5 available actions are chosen as candidate ones. In the middle of these extremes we have a great variety of strategies.

It should be noticed that this strategy's representation has one disadvantage: several strategies that seem to be different are in fact the same. Consider, for instance, all the strategies in the form K_{max} | ANY | RANDOM, with K_{max} being the number of states of our model, and ANY representing any of the three criteria. All these strategies select one action totally randomly among a candidate set that contains all the available actions, no matter which the first criterion is. The same occurs for 1 | RANDOM | ANY, and some others.

The number of truly different strategies in a model with K_{max} different states can be calculated as

$$\sharp STRATEGIES = 8 \times (K_{max} - 2) + 3 \tag{3}$$

The constant value 8 is the number of different combinations of the three explained criteria, considering that

the combination BEST | BEST is not considered because it leads to many equivalent strategies that do not even depend on k. Factor (K_{max} - 2) means that any k value can be used except the extremes 1 and K_{max} because they lead to extreme strategies that also represent several equivalent strategies, such as 1|BEST|RANDOM, 1|BEST|BEST, etc. These extreme strategies represent in fact just three different strategies - that is why we have added the value 3.

In the original model, the strategy is fixed during all the simulation so the same strategy is used in every decision stage. A new approach is now proposed, consisting in varying the strategy every time we have to make a decision.

One way to do this is to have a predefined succession of strategies that can be used in a cyclic way. We call this succession of strategies a *tactic*. For instance, consider the tactic 3 | PROP-PAYOFF | RANDOM, 4 | BEST | PROP-PAYOFF, 2 | RANDOM | RANDOM, 2 | PROP-PAYOFF | PROP-PAYOFF.

We define the *length* L of a tactic as the number of strategies it contains. The previous example has length 4 as it is composed by 4 different strategies. When using the tactic in a simulation that consists in a succession of several inputs (most likely more than 4), the tactic is considered to be cyclic: agent S will response to the n-th input by using the $(n \; mod \; L)$ strategy.

Considering expression (3) and given a tactic with length L, it is clear that the number of possible tactics is

$$\sharp TACTICS_L = (8 \times (K_{max} - 2) + 3)^L \qquad (4)$$

In a simple model instance with just 5 different states (K_{max} = 5) and taking into account only tactics with $L = 5$, the above formula yields to 531.441 possible tactics.

In the experiments section we will show that different results are obtained when $L > 1$, thus verifying the usefulness of this proposal.

IV. On the comparison of tactics

Whatever search method is going to be used, it is necessary to have a fast, yet reliable method to compare two tactics.

Let's consider a, b as the tactics to be compared. We evaluate each one h times to obtain two vectors or populations of samples $V_a = \{o_1, o_2, \ldots, o_h\}$ and $V_b = \{o_1, o_2, \ldots, o_h\}$

The most simple way to compare a and b consists on just comparing the means $\frac{\sum V_a[i]}{h}$ vs. $\frac{\sum V_b[i]}{h}$ and choosing the one with the lower value as the best one. Another alternative is very similar but the comparison is based on the median of the two vectors. In this way, the comparison could be less affected by extremal values.

The problem of comparing two sets of samples is very common in the field of statistics, which provides powerful methods that do not require a high number of samples to work properly. Once the challenge of reliably comparing two tactics has been solved, we can consider that the non-determinism in the objective function is properly addressed.

In this work, we hypothesize that parametric tests can be applied to the problem of comparing two tactics, which in statistical terms can be viewed as that of comparing the means of two different populations. The most widely employed parametric test is the t-Student test for means comparison, but it requires some previous conditions to be verified. As it is a parametric test, data must be normally distributed and must have similar variance. These well known conditions are called normality and homoscedasticity, and there also exist specific tests to check both of them, namely the Lilliefors and the Levene test, respectively.

If these two conditions hold, then the comparison of tactics during the execution of any metaheuristic can be done on the basis of a parametric t-Student test.

Later on, we will prove that both conditions are met so we describe here how the comparison can be made. The idea is to build a confidence interval for the difference of the means of the two populations (two sets of samples) being compared. If the interval does not contain zero, then the result of the comparison is obvious. If the interval does contain zero, then more samples are needed to try to make a certain decision about which one is better. More simulations are run, their results are added to those already available and the test is repeated with the now larger populations. As can be seen, the number of simulations employed can vary from one comparison to another, so this is really a "redistribution" of the available simulations. All the extra samples obtained in a comparison are stored with the existing ones, attached to the individual, so in a comparison where more samples are needed, only the individual with fewer samples is re-sampled.

We start collecting 30 samples for each tactic. If the null hypothesis could not be rejected, another 30 simulations can be added, and even another 30 if necessary, until a maximum of 90 simulations are reached. If the confidence interval contains zero after 90 simulations, we decide that the tactic with lower mean is the better one. Of course, this decision has no statistical basis but, after doing 90 simulations, it is far more convincing than a decision taken using just 30 simulations.

A. Use of Genetic Algorithms as search heuristic

As we stated in the introduction, evolutionary algorithms have been shown as a good alternative for doing "automatic design".

In this work, we will use a basic Generational Genetic Algorithm (GGA) to explore the space of cyclic tactics and evaluate the role of statistics-based comparison. The main characteristics of the GGA are described next.

Representation scheme: an individual models a tactic that is represented as a vector (succession) of strategies. The length of the vector equals the length of the tactic. A strategy can be described in terms of three components, i.e. the k-value, the first selection criterion and the second selection criterion. This can be represented as an integer with three digits where we encode each criterion with a fixed digit. For example, strategy 3-RANDOM-BEST can be encoded as 312 where the first 3 means K value is 3, 1 means first criterion is RANDOM and 2 means second criterion is BEST. Thus, a tactic is represented by a vector of 3-digit integers (discrete

representation scheme). The problem of having two strategies that are really the same has been overcame by using "special" integers.

Fitness function: the evaluation of an individual (a tactic) involves running a simulation several times. Given a set of random input sequences, the simulation is run and the gap values are recorded. As a consequence, a vector of values is returned. In order to evaluate all tactics in the same conditions, the set of input sequences is randomly generated before the metaheuristic starts.

Crossover operator: traditional one-point crossover is used. There are no extra restrictions after the crossover.

Mutation operator: it randomly selects one gene (strategy) and replaces it with a new randomly generated strategy.

Parents selection scheme: binary tournament. Notice that any p-ary tournament is also possible because it just requires sorting the p candidate individuals to tell which is the winner, and this operation can be described in terms of binary comparisons of individuals.

Population initialization: Given a specified length L for the individuals, each one of the single strategies composing the tactic is randomly selected.

Replacement scheme: as the GA is generational, the whole population is replaced in each generation. Elitism was also used to keep the best solution of each generation replacing the worst solution of the next generation.

Unless otherwise stated all the experiments were performed with the following parameters: (a) population size: 50 individuals, (b) crossover probability: 0.8, (c) mutation probability: 0.2, (c) stopping criterion: 150.000 simulations (provided the evaluation of an individual requires, at least, 30 simulations, and the comparison of individuals may spend extra simulations), (d) length of the tactics explored: $L = 4$.

The reader should note that no specific study was conducted to determine a set of "best" parameters.

V. Computational Experiments and Results

Experiments are divided in three parts. First one is devoted to show that using cyclic strategies is beneficial. Second part shows that that parametric tests can be applied to the problem of comparing tactics. The third part analyze the performance of the GGA when using this comparison methodology during its execution.

The adversarial model parameters used in each simulation were fixed as follows: length of the input sequences: 500, number of states: 5, payoff matrix: the same matrix was used in all simulations. The payoffs were randomly generated in the real interval $[0, 1]$ using the function $f(x) = log10(x)$ with x being a random integer in $[1, 10]$.

A. Suitability of cyclic strategies

In this part, we will show that using tactics where $L > 1$ leads to different results than those obtained with $L = 1$.

The experiment is quite simple: we did an exhaustive enumeration of all the tactics with $L = \{1, 2, 3\}$ and the results obtained are shown in table I.

ht

TABLE I
Performance (Gap) of tactics with $L = 1, 2, 3$

	$L = 1$	$L = 2$	$L = 3$
N. Tactics	27	27^2	27^3
Mean	48,733	45,09	43,891
Std. dev.	15,240	7,710	5,396
Best	39,658	39,55	39,520
Worst	99,205	99,20	99,202

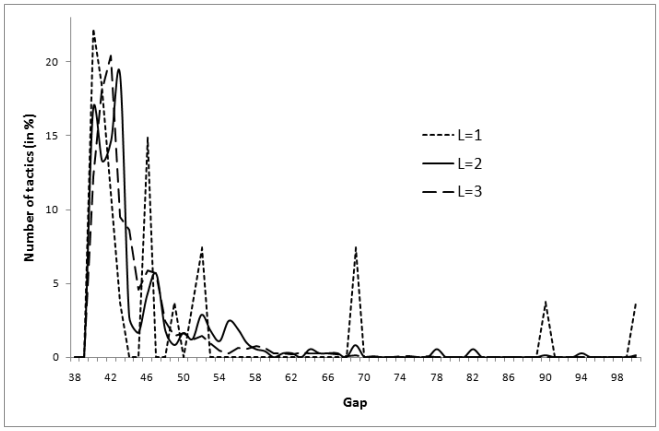


Fig. 2. Number of tactics of each length per gap value. Results are shown as percentages the total number of tactics available per length.

One can clearly observe that using different values for L lead to different results. On average, as L increases, the mean gap decreases.

The "best" existing tactic of length 2 is better that the best one with period 1. Also the whole set of tactics with period 2 is better, on average, than the set of tactics with period 1. The same holds for the comparison of periods 3 and 2. Although numeric differences seem small, pairwise non-parametric U Mann-Whitney tests indicate that the differences are significant from a statistical point of view.

The "Worst" tactics, having an almost 100% of gap, correspond to behaviours where agent S always take the best action in terms of payoff. As a consequence, the observation matrix of agent T has just one non-zero value per row, that is the one selected.

The distribution of the tactics per gap value is shown in Fig. 2. The curves represent the corresponding histograms for each L value and percentages are used for better visualization. The reader should be aware that the number of available strategies for each L is different so the highest peak at the beginning for $L = 1$ does not mean that there were more strategies for $L = 1$ than for $L = 3$.

B. Feasibility study on the use of parametric tests

In order to check that parametric tests can be applied for tactics comparison, the Kolmogorov-Smirnov test with Lilliefors correction was used to test normality and the Levene test was used to test variance similarity. The tests were run with SPSS statistical software.

The sample of tactics used to evaluate both factors were not generated randomly, but during the execution of the genetic algorithm itself. The genetic algorithm was run once, and every new individual generated, either in the initial population or as a result of crossover or mutation, was annotated in a separate file and evaluated using exactly 100 simulations. Hence, 100 samples were obtained as a result of the evaluation of each tactic. The experiment was designed this way, and not just with independent, randomly generated tactics, because this environment reproduces the conditions where the statistical comparisons will be done. At the end, we collected 1034 tactics/individuals.

In order to verify if the samples follow a normal distribution, we run 1034 Kolmogorov-Smirnov tests with Lilliefors correction. The results are clear: up to 95% of the tactics follow a normal distribution (see table II).

In order to test homoscedasticity, we randomly selected 42 individuals from this original 1034-element set. Then each individual was compared with all the others running a Levene test for assessing the equality of variance. A total of 861 comparisons were done. The results are shown in table III. Again, in a very high percentage of cases (85%) the equality of variances holds.

So, the results allows to conclude that the two conditions required for a parametric t-test are met by our set of strategies in a very high percentage of the cases. This means the next experiments, which try to evaluate the benefits of using a t-test during the genetic algorithm, make sense and are correct from a theoretical point of view.

TABLE II
RESULTS OF THE NORMALITY TEST FOR 1034 TACTICS. SIGNIFICANCE THRESHOLD: 0.05

	Quantity	Percentage
Non-normal	57	5,51 %
Normal	977	94,49 %
Total	1034	100 %

TABLE III
RESULTS OF THE LEVENE TEST FOR 861 COMPARISONS. SIGNIFICANCE THRESHOLD: 0.05

Variance	Quantity	Percentage
Different	125	14,52 %
Equal	736	85,48 %
Total	861	100 %

C. On the behaviour of GGA with statistics-based comparison

In the previous section, we verified the theoretical feasibility of using parametric tests for tactics comparison.

Now, we will analyze three different versions of GGA that differs on how the individuals are compared

- GGA_{mean}: two individuals are compared using the mean of 30 simulation results
- GGA_{median}: two individuals are compared using the median of 30 simulation results

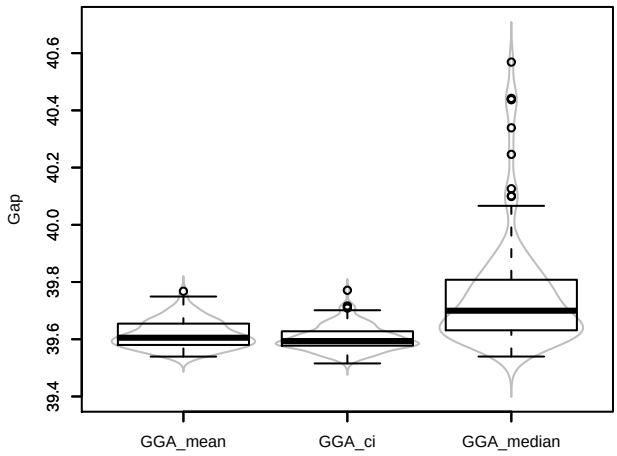


Fig. 3. Distribution of the best 100 gap values of every GGA. Together with the boxplots, an estimation of the distribution density is plotted behind, in light-gray color

- GGA_{ci}: two individuals are compared using statistical testing from 30,60 or 90 simulation results as needed.

Each algorithm was allowed 100 independent runs. As the third algorithm uses the evaluations available in a different way, a generation based analysis is not possible.

In order to compare the algorithms' performance, the best solutions obtained by all the executions must be evaluated in the same conditions. So, we took again these solutions and we evaluated them once again with 8000 simulations. After this, we had three sets of 100 samples each, corresponding to the three different algorithms.

The obtained results are shown first in Figure 3, where a boxplot for each algorithm is shown together with an estimation of the values' density distribution. One can observe that GGA_{ci} box and density are slightly compact with no noticeable outliers. In the contrary, GGA_{median} is clearly the worst alternative while GGA_{mean} looks quite similar to GGA_{ci} (although it has a higher number of outliers).

Table IV shows a summary of relevant values where results from the exhaustive enumeration of tactics with $L = 3$ are repeated for comparison purposes. The first point to notice is that any version of GGA can consistently find good solutions in a larger search space (with $L = 4$) and this is also a confirmation of the role of parameter L in the quality of the tactics.

Also, one can observe that GGA_{mean} and GGA_{ci} are able to obtain (on average) better tactics than GGA_{median}. An U Mann-Whitney test for the comparison of GGA_{mean} and GGA_{ci} states that they are different with a confidence level of 90% (p-value=0.736).

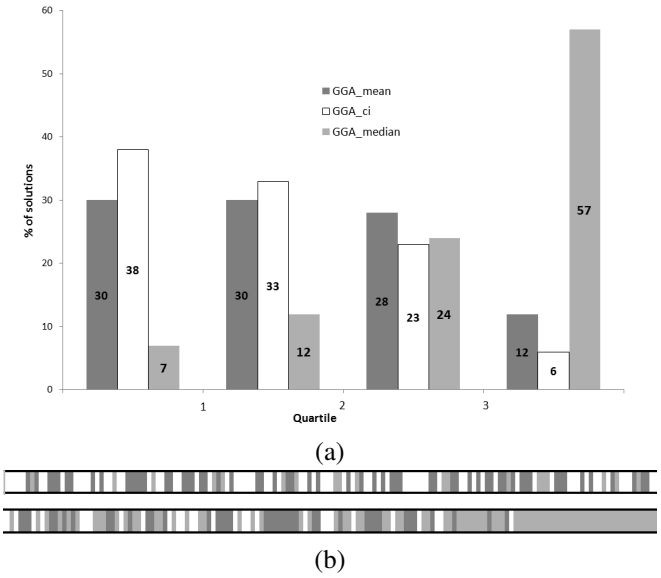
Results can also be seen from another point of view. We took those 300 best solutions (every method provide 100 solutions) and we sort them in increasing order of gap. Next,

ht

TABLE IV

VALUES OF GAP FOR THE TACTICS OBTAINED BY EVERY ALGORITHM. $L = 3$ CORRESPONDS TO VALUES COMING FROM THE EXHAUSTIVE ENUMERATION.

	$L = 3$	GGA_{mean}	GGA_{ci}	GGA_{median}
Mean	43,891	39,617	39,604	39,761
Std. dev.	5,396	0,049	0,046	0,207
Best	39,520	39,539	39,515	39,540
Worst	99,202	39,768	39,771	40,569



(a)

(b)

Fig. 4. Distribution of the best 300 tactics obtained by all the algorithms. In (a), each bar indicates the number of solutions in each quartile obtained by every method. In (b), the top strand shows the first 150 solutions while the bottom the remaining ones. The color represents the algorithm which obtained it and the color code is the same as in (a).

we count the number of solutions from each algorithm that appear in every inter-quartile segment and the result is shown in Figure 4(a). Every bar shows the number of solutions in each quartile provided by every method. One can observe that GGA_{ci} provided the highest number of solutions to the first and second segment. In the third one, most of the solutions were obtained by GGA_{mean} while the fourth one is plenty of solutions obtained by GGA_{median}. 4(b) shows two coloured strands representing the 300 best solutions (from 1 to 150 in the top strand, 150 to 300 in the bottom one) and their position when ordered. The color represents the algorithm who obtained such solution (GGA_{ci}: white, GGA_{mean}:dark grey, GGA_{median}: light gray). From this plot, one can observe the distribution of solutions within each segment. Clearly, there are more white blocks in the upper strand, specially at the beginning, thus indicating that the quality of such solutions is better than the other ones.

Now, we will try to provide insights on the behaviour of GGA_{ci}. One interesting aspect to analyze is how many comparisons had a clear "winner" after 30, 60, 90 or more simulations. A clear winner means that the gap of one tactic

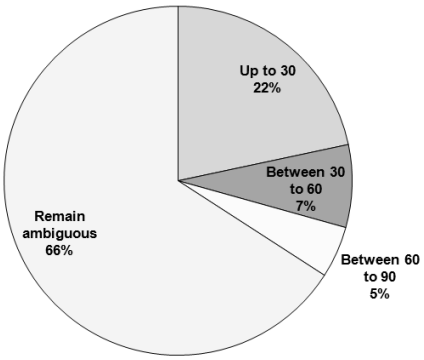


Fig. 5. Percentage of comparisons that required 30, 60, 90 or more simulations. Average results taken from 30 executions of the GGA.

is different from the one of another from a statistical point of view. If after sampling 90 simulations, no clear winner arise, then we took the one with lower gap but we consider such comparison as "ambiguous".

The results in Fig. 5 shows that 22% percent of the comparisons are reliable with just 30 simulations. Adding more samples, allows to increase the detection of the "winner" tactics slightly. It is interesting to note that the number of comparisons that remain ambiguous after 90 simulations is still very high (66%). However, comparing the means after 90 simulations in these doubtful cases seems to be more reliable than doing the same with just 30 simulations.

Given two tactics a and b, the difference between GGA_{mean} and GGA_{ci} will come from comparisons where a was better than b using 30 samples, but then, after more sampling, occurred that b was better than a (this would only be possible with GGA_{ci}). We analyze the number of times on which the "best" tactic changed after additional sampling. We have run GGA_{ci} and we recorded the number of comparisons per generation (in percentage) where the best tactic changed. The results averaging 100 independent runs, are presented in Fig. 6 and indicate that such value rapidly increases in the initial generations, while reaching an almost constant value around 25% in the rest of the run. In Fig. 5 we observed that just 22% of comparisons were significant with 30 simulations. In turn, 78% were not significant and it is from this percentage where the 25% previously mentioned should be considered. For example, in 1000 comparisons the additional sampling would change the initial result in around 200 cases. Without doubt, this is a pretty high value and reveals a weakness of mean-based comparisons. One can argue that this could be overcome if more samples are added from the beginning to the mean calculation, but this would potentially waste simulations when they are not needed.

VI. CONCLUSIONS

A new kind of strategies for an adversarial model has been proposed and successfully tested. The performance of cyclic strategies that vary along time showed to be better than that of simpler strategies. To be precise, cyclic tactics with length 2 and 3 performed better, on average, than the

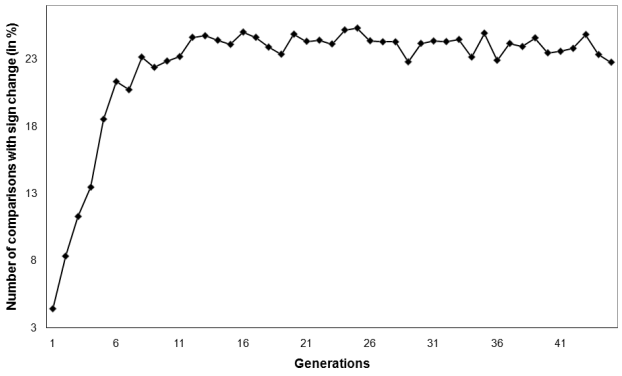


Fig. 6. Comparisons where the corresponding best solution changed after additional sampling (percentage over the number of comparisons per generation). Average results from 100 executions of the GGA_{ci}.

original static strategies. Also, the feasibility of a parametric, t-test-based comparison technique for this class of tactics has been successfully demonstrated from a theoretical point of view in our adversarial domain. Finally, this new comparison technique has been studied when coupled with a Generational Genetic Algorithm (GGA) exploring the space of tactics with length 4. A comparison with other two simple techniques like mean-based and median-based has been done as well. Results show that the proposal is beneficial: GGA_{ci} obtained better quality solutions than the other alternatives.

An important finding was that using resampling if needed in our method, make that the results of comparisons between solutions changed in more than 20% of the cases. This confirms that the t-test-based technique deserves further investigation because it can help metaheuristics to guide the search in a more reliable, yet efficient way. We envisage that this comparison technique could be useful in dealing with noisy fitness functions. Moreover, if t-test could not be applied due to absence of normality in the data samples, non-parametric tests could be applied.

ACKNOWLEDGMENT

This work was supported in part by the project TIN2008-01948 from the Spanish Ministry of Science and Innovation and P07-TIC-02970 from the Andalusian Government.

REFERENCES

[1] A. Kott and M. Ownby, "Tools for real-time anticipation of enemy actions in tactical ground operations," in *Proceedings of the 10th International Command and Control Research and Technology Symposium*, 2005.

[2] R. Popp and J. Yen, *Emergent Information Technologies and Enabling Policies for Counter-Terrorism*. John Wiley and Sons Hoboken, NJ, 2006.

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

[4] N. Krasnogor, G. Terrazas, D. Pelta, and G. Ochoa, "A critical view of evolutionary design of self-assembly system," in *Proceedings of the Conference on Artificial Evolution (EA'05)*, ser. Lecture Notes in Computer Science, vol. 3871. Springer, 2005, pp. 179–188.

[5] H.-S. Park, W. Pedrycz, and S.-K. Oh, "Evolutionary design of hybrid self-organizing fuzzy polynomial neural networks with the aid of information granulation," *Expert Systems with Applications*, vol. 33, no. 4, pp. 830–846, 2007.

[6] G. Baldassarre and S. Nolfi, "Strengths and synergies of evolved and designed controllers: A study within collective robotics," *Artificial Intelligence*, vol. 173, no. 7-8, pp. 857–875, 2009.

[7] S. Salcedo-Sanz, M. Naldi, A. M. Perez-Bellido, A. Portilla-Figueras, and E. G. Ortiz-Garcia, "Evolutionary design of oriented-tree networks using cayley-type encodings," *Information Sciences*, vol. 179, no. 20, pp. 3461–3472, 2009.

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