```
## Loading required package: bitops
```

The Uncertainty Quandary: A Study in the Context of the Evolutionary Optimization in Games and other Uncertain Environments

Juan J. Merelo[1], Federico Liberatore[1], Antonio Fernández Ares[1], Rubén García[2], Zeineb Chelly[3], Carlos Cotta[4], Nuria Rico[5], Antonio M. Mora[1], Pablo García-Sánchez[1], Alberto Tonda[6], Paloma de las Cuevas[1], and Pedro A. Castillo[1]

[1] Depto. ATC, University of Granada, Spain
jmerelo@geneura.ugr.es
[2] Escuela de Doctorado, University of Granada, Spain
[3] Lab. LARODEC, Institut Supérieur de Gestion, Tunisia
[4] Depto. LCC, University of Málaga, Spain
[5] Depto. EIO, University of Granada, Spain
[6] UMR 782 GMPA, INRA, Thiverval-Grignon, France

Abstract. In many optimization processes, the fitness or the considered measure of goodness for the candidate solutions presents *uncertainty*, that is, it yields different values when repeatedly measured, due to the nature of the evaluation process or the solution itself. This happens quite often in the context of computational intelligence in games, when either bots behave stochastically, or the target game possesses intrinsic random elements, but it shows up also in other problems as long as there is some random component. Thus, it is important to examine the statistical behavior of repeated measurements of performance and, more specifically, the statistical distribution that better fits them. This work analyzes four different problems related to computational intelligence in videogames, where Evolutionary Computation methods have been applied, and the evaluation of each individual is performed by playing the game, and compare them to other problem, neural network optimization, where performance is also a statistical variable. In order to find possible patterns in the statistical behavior of the variables, we track the main features of its distributions, *skewness* and *kurtosis*. Contrary to the usual assumption in this kind of problems, we prove that, in general, the values of two features imply that fitness values do not follow a normal distribution; they do present a certain common behavior that changes as evolution proceeds, getting in some cases closer to the standard distribution and in others drifting apart from it. A clear behavior in this case cannot be concluded, other than the fact that the statistical distribution that fitness variables follow is affected by selection in different directions, that parameters vary in a single generation across them, and that, in general, this kind of behavior will have to be taken into account to adequately address uncertainty in fitness in evolutionary algorithms.

1 Introduction

Optimization methods usually need a single-valued and reliable feedback on the quality of possible solutions to work correctly. This value, usually called *cost* or *fitness*, informs

the algorithm on the goodness of the solution and, when facing different alternatives, it is used to select a particular solution over others. This does not imply the necessity of a single floating point number as feedback; since these methodologies are based on comparisons, it is usually enough if the values can be partially ordered. In multiobjective optimization [11], for instance, two solutions can even be considered *non-comparable*, based on the set of fitness values they possess. In either case, the answer to the question "Is this solution better than the other?" needs to be either a 'Yes', or 'No', or 'Not decidable' in most optimization algorithms.

For many problems, however, the fitness or cost of a solution cannot be described by a single value or vector, because there is *uncertainty* when measuring it. Such uncertainty is inherent to most real-world physical systems, such as the one described in [9], where a control system is optimized through a stochastic procedure, but it also shows in machine learning and, in general, when the optimization algorithm uses a lower-level method which is, itself, stochastic. In these cases, the best way to describe the quality of a solution will be a random variable, not a single value or a vector of deterministic values. In our research, we routinely find this phenomenon in different optimization problems, such as:

- Optimizing the layout of a web-page using Simulated Annealing (SA) [48]. Since SA is a stochastic procedure, the fitness of an obtained solution will be a random variable.
- Training any kind of neural network [9,40]; in the first case, also mentioned above, we dealt with a physical installation, introducing another kind of randomness. Since training a neural network is usually a stochastic procedure, the error rate obtained after every training run will also follow a statistical distribution.
- Evolving game bots (autonomous agents) [43]. In this case, the uncertainty arises from the problem itself; in games, several factors such as the initial positions of the players or the opponent's behavior add further stochastic components, so that the final score will also be *uncertain* or *noisy*. In some cases, too, the bot itself will rely on probabilities to generate its behavior [15], in which case two different runs with exactly the same initial conditions and opponent will also yield different scores.
- In coevolutionary algorithms [46,15,47], individuals are evaluated by randomly choosing opponents from a pool, thus resulting in a fitness that is variable in a single generation and across generations [42].

In all these examples, it cannot be said that there is actually *noise* added to a *real* fitness. Instead, the fitness itself can be represented with a statistical variable, whose value arises from a stochastic process, evaluation, or training. In this sense, we are not concerned with the origin of this uncertainty. It could be noise in the measuring process, uncertainty in the fitness itself, or incomplete information like, for instance, what appears when surrogate models are used. It might be the case that the statistical nature of the fitness as a random variable might be different, distribution-wise, but we think that, except in the case that uncertainty is created by adding noise to fitness, the results obtained here will hold. In the problems used here, the randomness arises from the inherent stochasticity of the methods used to measure fitness, which, themselves, have a random element.

Despite a considerable amount of literature on problems with stochastic fitness values, there is a distinct lack of exhaustive research on the behavior of fitness functions, seen as random variables. That is why, after an initial study of noise in a specific game in [38], we dug into experimental data discovering that, even if the distribution in that particular case was always a Gamma, the parameters of the distribution were different. In that study, we proposed a solution to the noise issue, based on using the Wilcoxon comparison [57] as a selection operator. This meant that the random variable behaved in different ways depending on the particular individual, the state of evolution and, of course, the specific problem.

But, more importantly, this initial conclusion disagrees with the usual assumptions of optimization in uncertain environments, where it is usual to take a normal distribution of noise with fixed σ as the initial hypothesis [2]. For instance, in the functions of the Black Box Optimization Benchmarks [26], uncertainty was simulated by adding a Cauchy noise function centered in 0, that is, a centered, sharp bell-shaped distribution, with different widths. Either multiplicative or additive noise has been used in different occasions. However, our initial work hints that this is not the case in real-world optimization problems, ultimately invalidating the generality of the conclusions on different optimization methods obtained through the usual benchmarks. Besides, we also prove in [38] that, depending on the shape of the statistical distribution of the fitness, different methods could yield the best results. While methods that use the median or average would work well in centered distributions, other methods such as our technique, based on the Wilcoxon test, are better in more uncertain environments or, of course, in the case the noise distribution is not centered.

In this paper, we collect data from several different case studies, which will be presented later on, to find a stochastic model for the fitness using statistical tools. Our final objective is to eventually build a model as general as possible, able to account for most sources of uncertainty; failing that, to devise selection operators that are able to work with random fitness in a natural way.

This second part, if needed, will be the focus of future research.

The rest of the paper is organized as follows. In Section 2 we present the state of the art for evolutionary algorithms in uncertain environments, to be followed by a short presentation of the four problems with uncertainty whose measures will be used in this paper in Section 3. Results will be presented in Section 4, followed by our conclusions.

2 State of the Art

The most comprehensive, although not recent, review of the state of the art for evolutionary algorithms in *uncertain* environments is presented by Jin and Branke in [29], while more novel papers such as [50,3] and [49] include brief updates. Goh and Tan [23] performed a similar survey, focused on multiobjective optimization.

In their survey, Jin and Branke state that uncertainty is categorized into noise, robustness issues, fitness approximation, and time-varying fitness functions. In addition, different options for dealing with the uncertainty are discussed. In principle, the approach presented in this paper is designed to model the first kind of uncertainty, namely, noise or uncertainty in fitness evaluation. It can be argued that there is uncertainty in the

true fitness as stated in the third category. However, we think that, in general, the third issue refers to the case in which expensive fitness functions are replaced by surrogate functions which carry a certain amount of error, and whose value varies as the surrogate models are updated. Independently from the origin of uncertainty, Jin and Branke suggest several methods to tackle it, based either on *implicit* / explicit averaging over fitness measures [27,13] or on a threshold imposed during the selection phase. Papers such as Stroud's [56], Esteban-Díaz's [13] or Di Mario's [12] use this kind of approach to deal with noise. Other authors [18] propose to use new rank-based selection and mutation operators in order to evolve a neural network topology used as a controller for a robot. Results show that those operators are suitable for problems where the fitness landscape is noisy, but it is still using a central value for the fitness that might not be always valid.

Since then, several other solutions for uncertainty have been proposed. A usual approach for scientists more focused on obtaining a straightforward solution to the optimization problem without modifying existing tools and methodologies, is just to disregard the noise in the fitness and take whatever value is returned by a single evaluation, often after re-evaluating all individuals at each generation. This option seems to work especially well if the population is large [27], since the selective pressure is lower and solutions have the chance to be evaluated several times before being selected or discarded; this leads, if the population is large enough, to an *implicit averaging* as mentioned in [29]; in fact, Rattray and Shapiro [53] in their theoretical model compute by how much the *crisp* population must be enlarged to overcome the *problem* of noise. This solution is exploited in our previous research in games, although one evaluation in some of these works consists, in fact, of an average of several evaluations, on different maps or considering different opponents, see for instance [43,42,34], or in the evolution of neural networks [4,40].

The key to the efficiency of this approach stems from the fact that selection used in evolutionary algorithms is usually stochastic, so uncertainty in fitness evaluation could have the same effect as randomness in selection or a higher mutation rate, which might make the evolutionary process easier in some particular cases [50]. In fact, Miller and Goldberg proved that an infinite population would not be affected by noise [41] and Jun-Hua and Ming studied the effect of noise in convergence rates [31], proving that an elitist genetic algorithm finds at least one solution in noisy environments with probability one, although with a lowered convergence rate. This possible positive effect of uncertainty in evaluation leads to some authors calling it "a blessing and the curse" in the context of surrogate models [44], which, as we have seen before, carry with them a degree of uncertainty and randomness.

In real-world problems, however, populations are finite: in fact, using large populations decreases the algorithm's efficiency and can be time consuming, so the usual approach for dealing with fitness with a degree of randomness is to enlarge the population to a value bigger than would normally be needed in a non-noisy environment, while keeping it to a manageable size. Furthermore, it has been proved recently that using two parents to generate offspring, that is, crossover, is able to successfully deal with noise [20], while an evolutionary algorithm based mainly on mutation, such as the μ+1 EA, or evolutionary programming [19], would suffer a considerable degradation of performance. However, crossover is part of the standard kit of evolutionary algorithms,

so using it and increasing the population size has the advantage that no special provision or change in the implementation has to be made. There is no big decreasing in efficiency as long as oversized populations are not used. Using oversized populations, however, might have a good effect on the algorithm in general, if appropriate computational resources are available [33].

Another way to deal with uncertainty which is more theoretically sound is using *real* averaging, that is, a statistical central tendency indicator, which usually is the *average*; average happens to be equal to the median in the case of the random variable following the normal distribution. In this case, resampling is used to acquire a statistically significant amount of measures and then the average is computed over them. This strategy has been called *explicit averaging* by Jin and Branke, and it is used, for instance, in [31]. Explicit averaging decreases the fitness variance, thus reducing uncertainty, but defining the appropriate sample size for the averaging process is not straightforward [1]; besides, this central tendency might not be representative if the noise is not centrally distributed, as proved in [39]. Our research group uses this approach in some cases, with an important difference: individuals are not re-evaluated every additional generation, but their fitness value is the average of several evaluations, performed immediately [42]. Most authors use several measures of fitness for each new individual [10], although other averaging strategies have also been proposed, for example averaging over the neighbourhood of the individual or using *resampling*, that is, heuristically requiring more fitness measurements [35]. This assumes that there is, effectively, a real average of the fitness values, which is true for Gaussian random noise and other distributions (such as Gamma's or Cauchy's), but it does not necessarily hold for all distributions. In this paper, we are going to model these distributions in order to verify whether this assumption is indeed correct.

To the best of our knowledge, other central tendency measures such as the median, which might be more adequate for certain noise models, have not been tested; the median always exists, while the average might not exist for non-centrally distributed variables. Besides, most models keep the number of evaluations fixed and independent of its value, which might result in bad individuals being evaluated multiple times before finally being discarded; some authors have proposed *resampling*, [51,52], which will effectively increase the number of evaluations and thus slow down the search. In any case, using explicit averaging usually requires just a small change to the algorithm framework, by using the average of several evaluations as the new fitness function. Thus, it is usually the method preferred by researchers and practitioners using off-the-shelf libraries such as ECJ [37].

In order to improve the efficiency of the algorithm, or the running time, these two averaging approaches that are focused on the evaluation process might be complemented with changes to the selection process. For instance, a threshold [54,52] that is related to the noise characteristics to avoid making comparisons of individuals that might, in fact, be very similar or statistically the same; this is usually called *threshold selection* and can be applied either to explicit or implicit averaging fitness functions. Uncertainty can also be used to compare different algorithms, with some authors proposing, instead of taking more measures, testing different solvers [8], some of which might be more affected by noise than others. However, recent papers have proved that sampling might

be ineffective [49] in some types of evolutionary algorithms, adding running time without an additional benefit in terms of performance. This is one lead we will try to follow in the current paper, by modeling noise in order to eventually design an algorithm that behaves correctly in that environment.

All the aforementioned approaches still face the issue of the statistical representation of the *true* fitness, even more so if there are instead several measures that represent, *as a set* the fitness of an individual, such as the case study described in [39]. This is what we have been using in many of our papers: a method that uses resampling via a memory attached to every individual that stores all fitness measures and uses either explicit averaging or statistical tests like the non-parametric Wilcoxon test. In order to test this approach on benchmark problems more realistic that the ones adopted so far, we need to characterize the noise that actually appears in games and other real-world case studies for optimization.

3 Case studies used in this paper

The fitness analyzed in the four different case studies, all related to computational intelligence in games, are described in this paper: generation of character backstories in a MAssive Drama Engine for non-player characters (MADE), described in subsection 3.1, optimization of bots for playing the real time strategy game (RTS) Planet Wars in 3.2, optimization of the ghost team in Ms. Pac-Man, which will be described in subsection 3.3, automatic generation of autonomous players for the famous RTS StarCraft, explained in subsection 3.4 and an artificial neural network optimization problem using an EA 3.5.

These five problems have been chosen for two main reasons: the origin of uncertainty is different for each of the case studies; and data for the experiments is readily available, with the possibility of running further experimental trials, if needed. In the case of MADE, fitness is computed through a simulation; in the case of Planet Wars, the bot themselves have a random component, with its representation including probabilities of different courses of action; in Ms. Pac-Man, uncertainty lies in the nature of the game itself; and the huge amount of possibilities in StarCraft, with a considerable number of units behaving independently, creates an extremely high source of uncertainty. These scenarios are not a complete representation of all possible causes of uncertainty in optimization, but we think that the sample is big and varied enough to generalize the obtained results, which will be presented in the next section.

In all cases, three generations were chosen, and they are different depending on the problem. We feature the first generation (except in the case of MADE), a intermediate generation and one close to the end of the evolution, containing individuals close to the solution. These were chosen to check the progress of the two statistical parameters in different situations: close to random in the case of the first generations, and close to the *real* value, in the case of the last ones. The particular number of generations is not really important, the importance is how close they are to the end of the evolution, which is different in each case.

We will next examine the creation of character backstories in the problem called MADE.

3.1 Creation of character backstories

MADE [21] is a framework for the automatic generation of virtual worlds that allow the emergence of backstories for secondary characters that can later on be included in videogames. In this context, an *archetype* is a well-known behaviour present in the imaginary collective (for example, a "hero" or a "villain"). Given a fitness that takes into account the existence of different N_a archetypes for a virtual world, MADE uses a Genetic Algorithm (GA) [24] to optimize the parameter values of a Finite State Machine (FSM) that models the agents of that world. For the evaluation, a world is simulated using this parameter set, and the log is analyzed to detect behaviours of the world agents that match the desired archetypes.

As the evolved parameters are the probabilities to jump from one state to another in the FSM, each fitness evaluation is performed executing the virtual world five times per individual, obtaining the average fitness. Selection is, therefore, performed comparing this average fitness, with a binary tournament. Fitness values range from 0 to N_a, and are calculated taking into account the rate of occurrence of the archetypes in the execution log.

3.2 A 'simple' real-time strategy game: Planet Wars

Planet Wars [16] is a simple Real-Time strategy (RTS) game. In RTS games, the objective is to defeat the enemy using resources available in the map to build and manage units and structures: differently from turn-based strategy games, in RTS all choices have to be performed in real time.

Planet Wars provides a simplification of the usual elements in RTS games: one kind of unit (spaceships) and one kind of resources and structures (planets). Spaceships are automatically generated on the planets controlled by a player, and they are used to conquer enemy planets, the main way to defeat the enemy.

In this paper we are using the results obtained from the Genebot algorithm [17]. This algorithm optimizes seven parameters of a hand-coded FSM, two of which are probabilities. These values are used in expressions used by the bot to take decisions, such as the selection of the target planet to attack or reinforce; this implies that the actions of the bot will be different every time the bot acts, that is, some state transitions are based on probabilities. Fitness is calculated confronting the bot obtained from the parameter set of the FSM five times against a competitive hand-coded bot. The result of each match takes into account the 'slope' of the number of player spaceships during the time of the match. Positive results mean that the bot won, as the slope will be positive, and vice versa. Theoretical values are in the range $[-1,1]$, although these extremes are impossible to attain in the game. A value of -1 would indicate that the player lost all their ships at startup, while 1 would mean the contrary: it generated all the spaceships and won in the initial time. The fitness of an individual is the sum of all five results, and therefore is in the range $[-5,5]$.

3.3 Ghost team optimization

Ms. Pac-Man is a variant of the famous Pac-Man game that extends its mechanics with several extra features, the most interesting being the inclusion of a random event that

reverses the direction of the ghosts. This game is used in the Ms. Pac-Man vs Ghosts competition [36], where participants can submit controllers for both Ms. Pac-Man and the Ghost Team, the first trying to maximize its score, the second trying to minimize Ms. Pac-Man's. The framework used to test the methodology analyzed defines the following restrictions for the Ghost Team:

- A ghost can never stop and if it is in a corridor it must move forward.
- A ghost can choose its direction only at a junction.
- Every time a ghost is at a junction the controller has to provide a direction from a set of feasible directions.
- After 4000 game ticks, a level is considered completed and the game moves on to the next one.

In the methodology applied to this case study, published in [34], the fitness of each individual is computed as the maximum score obtained by eight different Ms. Pac-Man controllers. Some of these controllers are the champions of past editions of the international competition, so they are very tough rivals for the ghost team.

3.4 A complex real-time strategy game: StarCraft

StarCraft has become a *de facto* testbed for AI research in complex RTS games [45]. In fact, given the high variety of game features, such as configuration options, game modes, units, maps, etc; along with the existence of several frameworks and tools related with it; researchers have exploited the game for a great variety of topics: micro and macro management of units, temporal and spatial reasoning, battle planning, combat results prediction, optimal paths and dealing with problems such as the one that is the topic of this paper, uncertainty in the evaluation of the fitness, among others.

The individuals described in this subsection have been generated using StarCraftGP [22], a Genetic Programming (GP) [32] framework that automatically generates the source code of high-level strategies of bots. In this case, Linear GP [55] was used to generate the building order of the units to create, and also the rules to activate during the game: for example, when to attack the enemy or when to collect more materials. Each individual is a source code file in C++ that is compiled during the evaluation. A population of 32 individuals was evolved during 30 generations. The rest of the parameters used are presented in [22].

As in some of the games described above, the fitness of one individual is computed pitting the bot against different enemies, each one following a different strategy. More specifically, in this case every individual faces three *divisions* of enemies (considered as weak, medium and strong rivals), each division containing four different enemies.

The original fitness function assigned a higher weight to a victory against the stronger enemies, i.e. it used a lexicographical fitness, so one victory in a higher tier was considered better than more victories in the immediately lower one. For example, one individual that wins 1 time against one enemy of every tier was considered better than one individual that beats all individuals from the medium and weak tier, but none in the strong one.

Conversely, to ease comparisons among noise in the present study, we have calculated an aggregated fitness function that still respects this decision, that is, prioritizing

victories of harder divisions, by giving different weights to each one. The following equation describes the fitness function:

$$F_{StarCraft} = 21 \times A + 5 \times B + C + R \tag{1}$$

where A is the number of victories against enemies in the strongest tier, B is the number of victories against the middle tier, and C is the number of victories against the weakest enemies. Thus, for example, one victory in the middle tier is worth more points (5 points) than 4 victories in the weak tier (4 points). Also, a coefficient of the aggregated score at the end of all the games, R has been added, in order to deal with ties in number of victories. This is in fact an internal score computed by the game, that takes into account all the aspects of a match, ranging from the number of kills to the type and quality of units built.

Moreover, the evaluation process is quite time-consuming, so in order to save execution time, at least one enemy of the weak tiers must be defeated before allowing the individual to proceed to fight the next one. To this end, if a bot does not win against weaker rivals, we consider it cannot defeat the stronger ones: so the evaluation terminates at that point, with the current score.

3.5 Artificial Neural Networks Optimization Using an EA: GProp

The design of an Artificial Neural Network (ANN) [28] requires to set both, the structure of its set of hidden layers, along with the parameters it uses, weights and learning constants).

G-Prop ("genetic backpropagation") [6,5,7] aims to solve the problem of finding appropriate initial weights, number of neurons in the hidden layer and learning parameters for a Multilayer Perceptron (MLP) with a single hidden layer. It does so by combining an EA and the QuickProp method [14] for training MLPs. The EA selects the MLP's initial weights, picks its learning rate, and changes the number of neurons in the hidden layer through the application of specific genetic operators. Since this representation of the neural network is then trained using QuickProp, which is an stochastic gradient-descent algorithm, the results will have a certain variability, resulting in the uncertainty in the fitness that will be studied here. In this problem, fitness is the classification accuracy or success rate; this fitness is obtained after training the MLP, which sets its weights, and then testing it on a test set. After classification, weights will be different, so the test result will also be, making fitness *noisy* and thus amenable to analysis in the paper.

4 Experiments and Results

With the case studies presented above, data on fitness values is collected by selecting a few random individuals in every generation of the considered EAs, and measuring their fitness 100 times, intentionally using much more repetitions than a normal optimization method would. Thus, every individual is represented by a random variable sampled 100 times. According to the usual assumptions, this random variable should follow a normal

distribution, with a certain σ and centered on the *true* fitness value. In order to verify this hypothesis, we plotted the distribution's *skewness*, that is, its asymmetry, and its *kurtosis*, which is a parameter related to its shape [25] A symmetrical distribution, like the normal distribution, has skewness and kurtosis equal to 0; asymmetric distributions, such as the Gamma that we had found in previous papers [38], have non-zero skewness and kurtosis, related to their θ and κ parameters, for instance. These parameters are what defines the statistical distribution; κ is the shape parameter and skewness is $2/\sqrt{\kappa}$, which means that it is only 0, corresponding to normality, if κ grows to infinity. Kurtosis is $6/\kappa$, implying the same. A random variable can have skewness and kurtosis fixed at any value: thus, we present these values in the following figures, with skewness plotted as the *x* axis against kurtosis on the *y* axis.

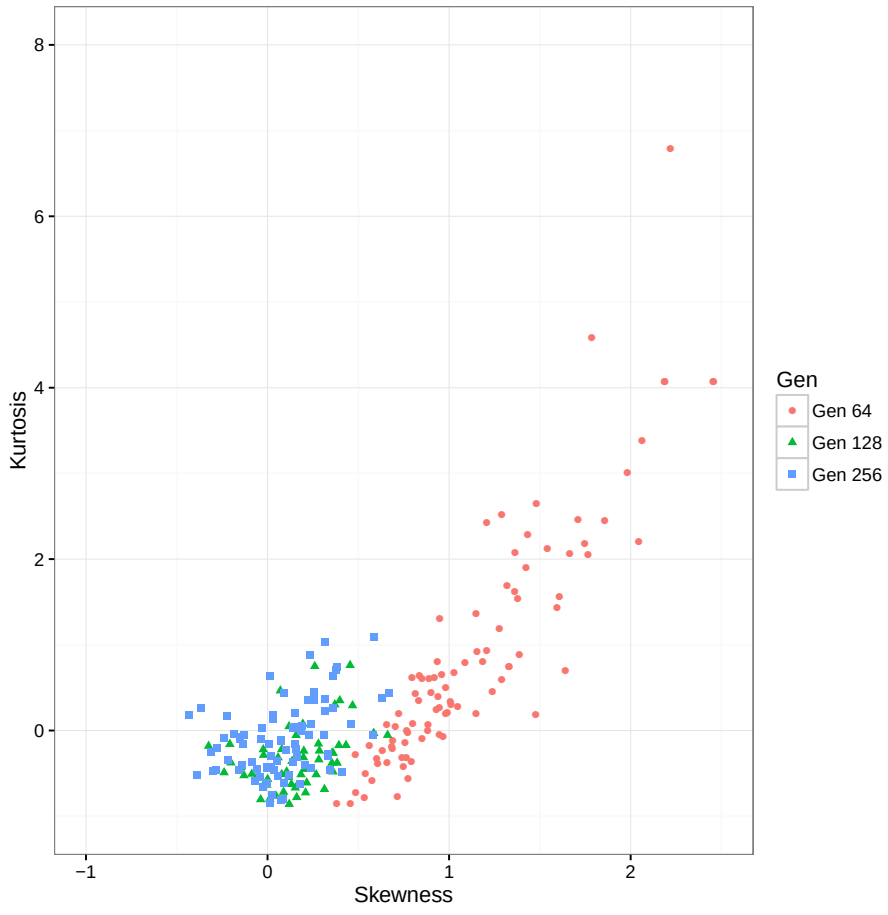

Fig. 1. Skewness and kurtosis for fitness in several generations of the MADE problem. Different colors represent different generations.

Figure 1 represents skewness and kurtosis in the MADE case study, for which we took measures of a variable amount of individuals every generation, from 100 in generation 64 to around 50 in the latest generation. The number was variable because some of them stopped before finishing. Anyway, the number of measurements is enough for the statistical analysis. You can already see that the distribution is not normal, since almost no individual has a kurtosis and skewness equal to zero; some of them, however, are close. This will be the case for the rest of the experiments, too; in some very limited cases fitness distribution will be almost normal in the first or the last generations, but that will never be the case for all individuals or even a significant fraction, nullifying the hypothesis of fitness behaving like a crisp fitness with gaussian added noise. As generations proceed, a curious convergence towards the normal distribution is observed; in the first generations, values of skewness and kurtosis are quite high and correspond to an arbitrary distribution (Beta or uniform): however, as the simulation proceeds, the two values approach zero. It must be noted, however, that they do not converge exactly to 0, meaning that, even if uncertainty in this case can be approached by a normal distribution, such an approximation would only be correct for the latest generations of the simulation. In general, individual fitness in MADE will follow an arbitrary distribution with a general shape and asymmetry.

The shape of the graph for the Planet Wars case study, shown in Figure 2 for two different generations, is different but shares some similarities. The dispersion also decreases as evolution proceeds, with the shape of the distribution becoming closer to the normal distribution in generation 50. Nevertheless, the initial kurtosis is quite high and values above 2 and below 0 are found even later in the evolution. Noise is, thus, *noisy* and does not conform to a single shape, even less a normal one; this implies that using a single statistical model to represent noise will never be too close to reality, since the shapes of the statistical distribution are, in general, quite different from the normal distribution and then different among themselves even for a single problem, that is, the shape of the statistical distribution of fitness values in uncertain environments is, itself, uncertain or *noisy*.

The graph for the third case study, ghosts in Ms. Pac-Man, is different in several aspects, and is shown in Figure 3. First we have to take into account, as explained in 3.3, that differently from the previous cases, the fitness for a ghost team is the maximum, not an average of several values. This causes a curious behavior of fitness: in the first generation, several individuals have *crisp* values; however, this is less and less true, becoming more "random" as generations proceed, that is, the set of values the fitness has got begins to have many different values while in the first generations it had one or a few. To put it in another words, in the first generation the set of fitness measures could look like $\{\texttt{x x x y x x x}\}$. As evolution proceeds, the measures in the set tend to be all different That is why the behavior shown in the graph is completely different: distributions get increasingly asymmetric and their shape grows further away from a normal distribution and closer to a Beta distribution. Even if the trend is different from the other two problems, the overall aspect is the same: there is no single distribution that is able to describe the shape of fitness with an uncertainty component, for all considered generations.

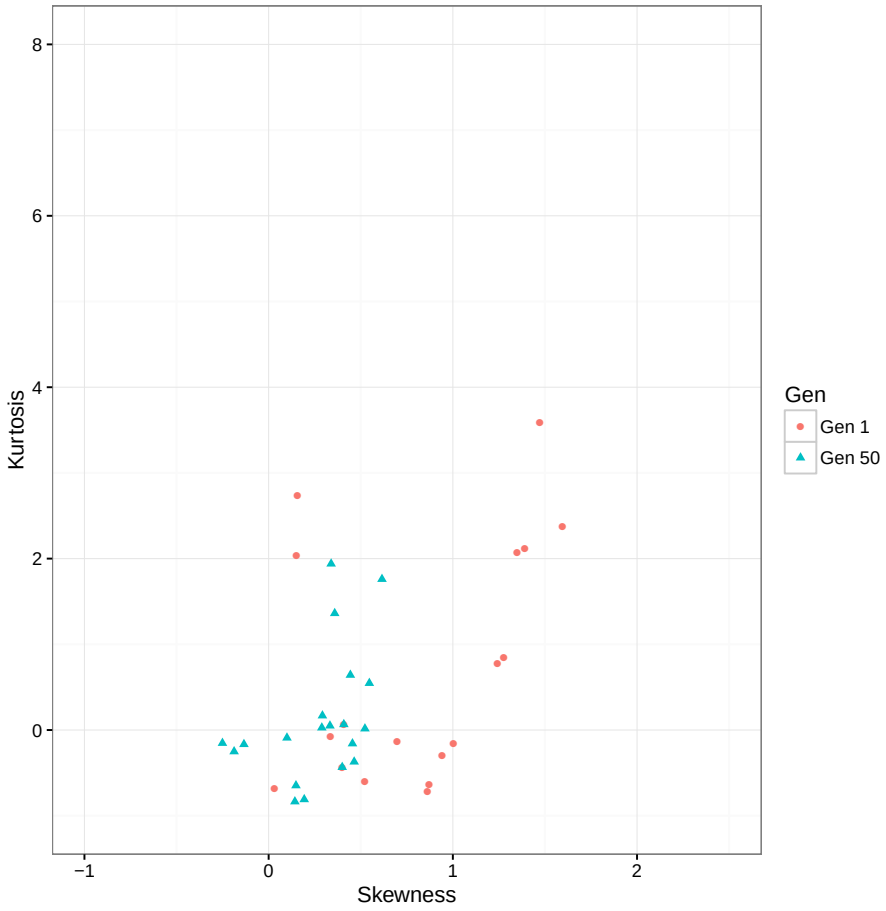

Fig. 2. Skewness and kurtosis for fitness in several generations of the Planet Wars problem. Different colors represent different generations.

The last game we have evaluated is StarCraft, with kurtosis and skewness shown in Figure 4. In this case evaluation takes a very long time, that is why only a few samples were available. That might be the reason it is not quite clear if there is a trend. The latest generation seems to be a bit closer to normal distribution, but intermediate generations tend to have a high value. However, even if values seem to be closer in generation 30, they are in some cases positive and in other negative, indicating a distribution that is flatter than the Gaussian and with the *bump* more loaded to the right of the center. Once again, this proves that using non-parametric methods like Wilcoxon are a better approach than using central measures such as the average.

For the sake of completeness, we have also included in this paper a problem that comes from a different area: genetic optimization of neural networks. The skewness/kurtosis graph is included in Figure 5. Since the problem is completely different,

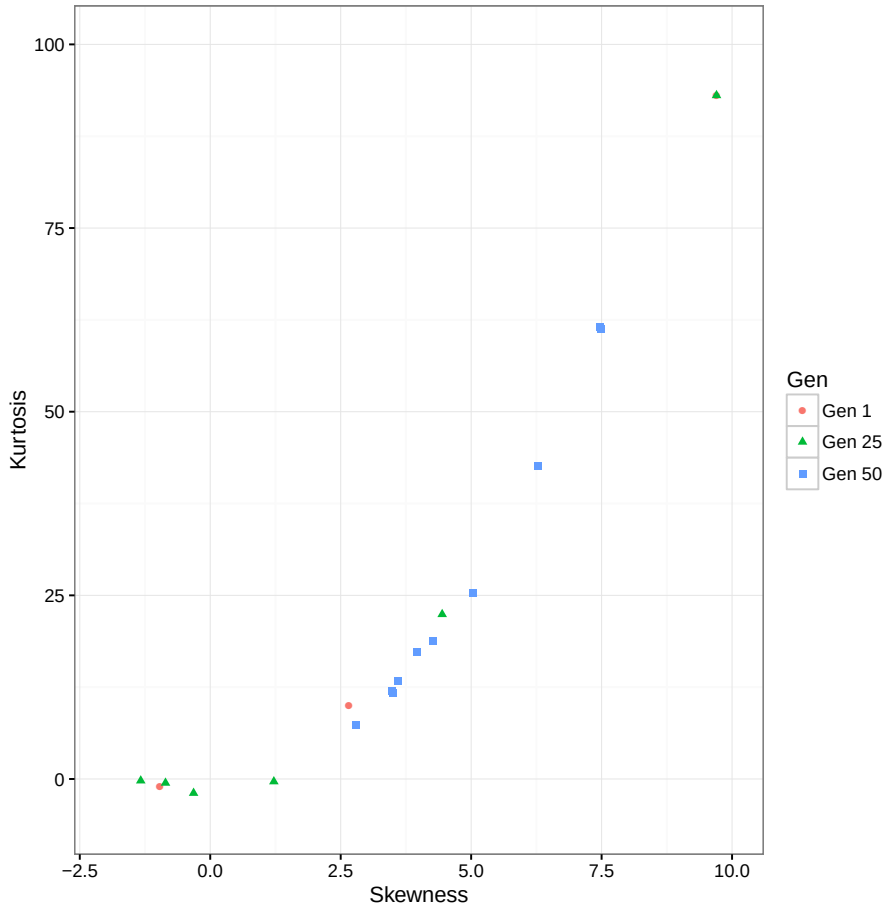

Fig. 3. Skewness and kurtosis for fitness in several generations of the Ms. Pac-Man problem. Different colors represent different generations.

the distribution of the values is also completely different. For starters, skewness tends to be negative, indicating distributions with a long tail to the right; that means that, even if the value is centered along a particular value, there are many values that are larger that this central value. Once again, resampling cannot change the fact that the average will not be an accurate description for the whole data. Besides, values tend to get closer to 0 although in every generation there are values quite far away from them; e.g in the last generation, a neural net whose fitness distribution has kurtosis of 15, indicating a very sharp bump, is present, but it also has a low kurtosis of almost -4 indicating a long tail to the right. The conclusion in this case is similar in the sense that values tend to change while they keep away from a single kurtosis and skewness; even having less values than the latter if both of them are set to 0.

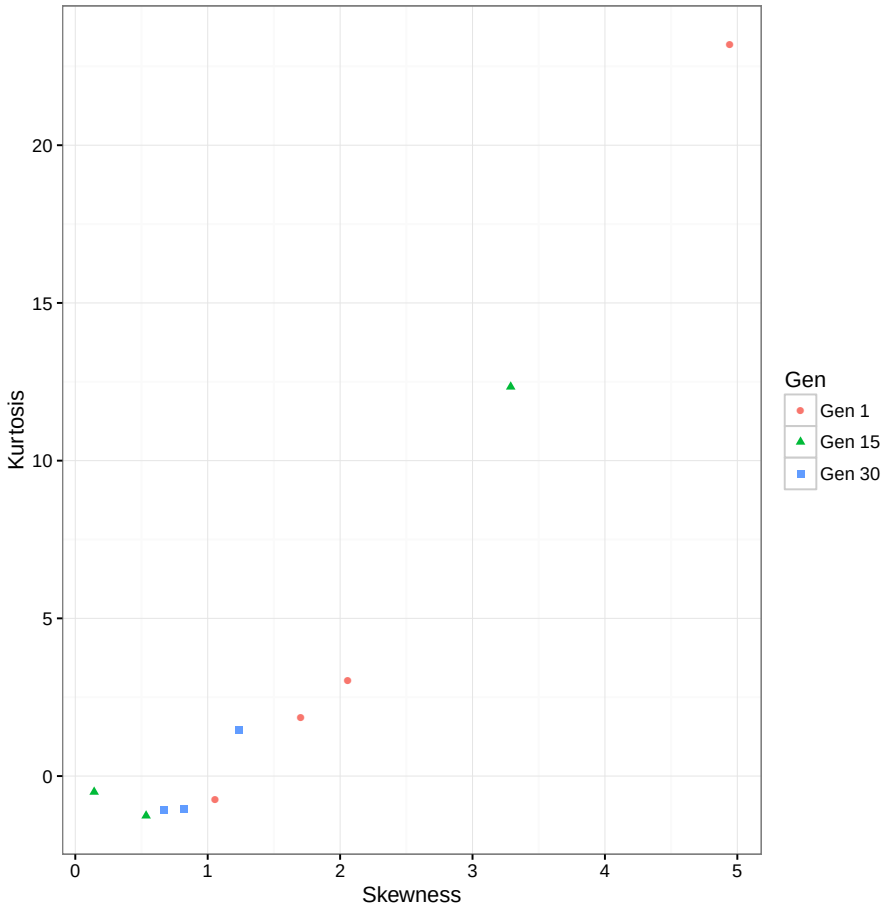

Fig. 4. Skewness and kurtosis for fitness in several generations of the StarCraft game. Different colors (or shades of gray) represent different generations.

5 CONCLUSIONS

In this paper, we set out to study the statistical distribution that best fits the stochastic fitness values of single individuals in several case studies in the area of games; we have also included a genetically optimized neural network for the sake of comparison. Stochastic optimization evolutionary algorithms applied to MADE, Planet Wars, Ms. Pac-Man, and StarCraft exploit different ways to compute the fitness values, but for all of them the fitness value is not a fixed number but a random variable. This is also the case in G-Prop, the genetically optimized multilayer perceptron. We prove the hypothesis that not only noise does not follow the normal, or Gaussian, distribution, or other centrally-distributed models such as Cauchy, which have been used repeatedly in literature for benchmarking selection methods in the presence of noise; but also it does not

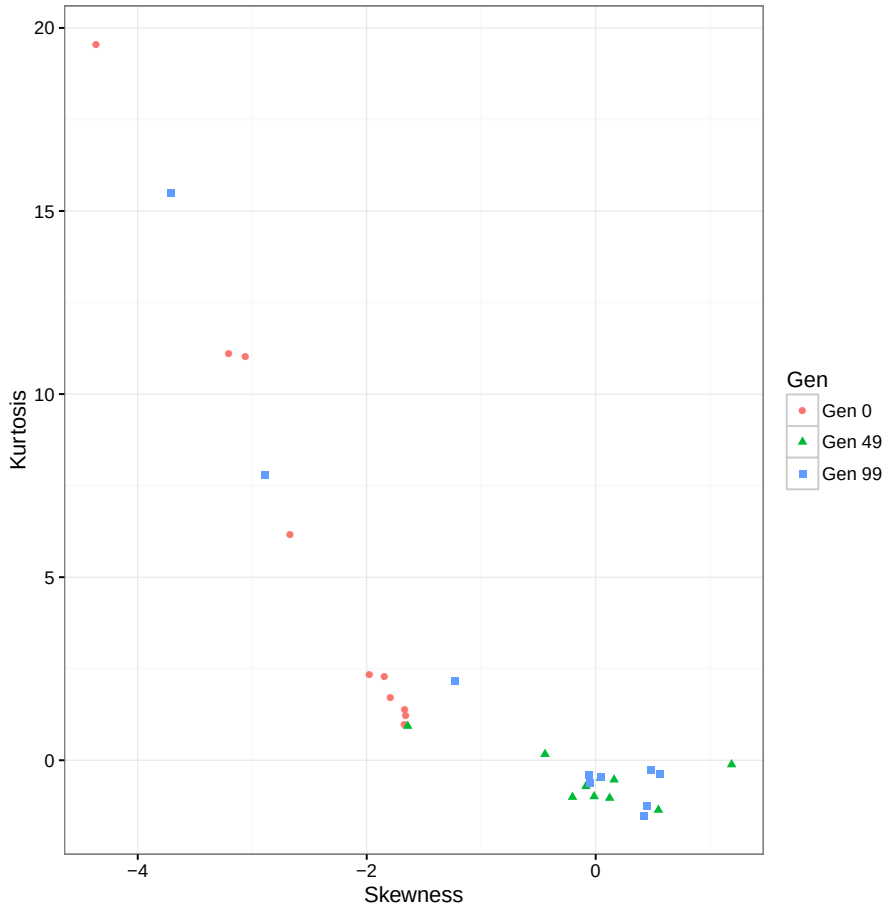

Fig. 5. Skewness and kurtosis for fitness in several generations of the MLP training problem. Different colors (or shades of gray) represent different generations.

follow a single, particular distribution even when considering a single case study or a single generation.

This conclusion follows from our study of the parameters of the statistical variables that describe fitness. The best way to describe them is using two parameters: kurtosis and skewness. These two parameters have been computed and plotted for candidate solutions extracted from each of the case studies, proving that not only distributions are asymmetrical and not bell-shaped, but that their shape changes within a single problem and in different stages of the computation; this is in accordance with the conclusions reached by Rattray and Shapiro in [53] for evolution of finite populations in the presence of noise. In some cases, like MADE, it seems clear that due to the fact that averages are used as a representative for selection, individuals whose fitness is closer to a central shape are oversampled and thus selected preferably, with almost-central individuals

in the latest stages being a consequence of this fact. In other cases, when fitness is computed in a different way or selection takes another form, the effect is exactly the opposite. Using averages or other central measures like the median does not seem to be supported by the results of this paper since in many cases and almost always in the early stages of the evolution, fitness, being a random variable, does not pass a centrality test and it might not even possess a reliable, that is, statistically significant, average. A better way of comparing any fitness with uncertainty would be, as proposed by the authors, using non-parametric tests such as the Wilcoxon test that impose a partial order on the individuals [38]. This partial order can be used, in several different ways, for selection.

The fact that there is no single model representing the distribution of fitness also implies that it is an error to use centrally distributed random variables added to an actual fitness to test operators and algorithms that operate in uncertainty. Either real values should be used, such as the ones proposed above, or a distribution with varying shape and symmetry such as Beta. However, in this case we should take into account that "true" or "crisp" fitness *does not really exist*, so any modelization of uncertain values that uses noise added to a fitness value is, in the more general case, wrong, although it might still return correct results in some cases. If the fitness evaluation is expensive and tests have to be performed for new selection operators, the best way to model uncertainty would be to use *different* statistical models applied to every individual, with different skewness and kurtosis. However, this would be only a first-order approximation and it might still favor methods that use averages. Following the model proposed by Jin [30] for surrogate models, assuming normality in fitness will make selectable some individuals that should not be. Assessing this error and its impact on selection, and comparing how different methods, such as the one based in statistical techniques and proposed previously, reduce that error is also left as future work.

What remains to be done is to effectively apply Wilcoxon-based comparisons to the case studies above. Since real-world case studies are computationally expensive to evaluate, we plan to create a benchmark for problems with uncertainty which reflects in the best possible way how fitness is organized in a wide array of problems. In order to attain this goal, we will examine as many uncertain problems as possible, in the attempt to deduce a model of noise what as general as possible.

6 ACKNOWLEDGEMENTS

This work has been supported in part by projects TIN2014-56494-C4-3-P (Spanish Ministry of Economy and Competitiveness), SPIP2014-01437 (Dirección General de Tráfico), PRY142/14 (Fundación Pública Andaluza Centro de Estudios Andaluces en la IX Convocatoria de Proyectos de Investigación), PROY-PP2015-06 (Plan Propio 2015 UGR), and project CEI2015-MP-V17 of the Microprojects program 2015 from CEI BioTIC Granada. We would like also to thank the anonymous reviewers for this paper, for suggesting new readings and avenues of research.

References

1. Aizawa, A.N., Wah, B.W.: Scheduling of genetic algorithms in a noisy environment. Evolutionary Computation 2(2), 97–122 (1994)

2. Arnold, D.: Evolution strategies in noisy environments-a survey of existing work. In: Theoretical aspects of evolutionary computing. pp. 239–250. Springer-Verlag (2001)
3. Bhattacharya, M., Islam, R., Mahmood, A.: Uncertainty and evolutionary optimization: A novel approach. In: Industrial Electronics and Applications (ICIEA), 2014 IEEE 9th Conference on. pp. 988–993 (June 2014)
4. Castillo, P.A., González, J., Merelo-Guervós, J.J., Prieto, A., Rivas, V., Romero, G.: G-Prop-III: Global optimization of multilayer perceptrons using an evolutionary algorithm. In: GECCO-99: Proceedings Of The Genetic And Evolutionary Computation Conference. p. 942 (1999)
5. Castillo, P.A., Merelo-Guervós, J.J., Prieto, A., Rivas, V., Romero, G.: G-Prop: Global optimization of multilayer perceptrons using GAs. Neurocomputing 35, 149–163 (November 2000), http://dx.doi.org/10.1016/S0925-2312(00)00302-7, available from http://geneura.ugr.es/pub/papers/castilloNC.ps.gz
6. Castillo, P., Carpio, J., Merelo-Guervós, J.J., Rivas, V., Romero, G., Prieto, A.: Evolving multilayer perceptrons. Neural Processing Letters 12, 115–127 (October 2000), http://dx.doi.org/10.1023/A:1009684907680
7. Castillo, P., Merelo-Guervós, J.J., Prieto, A., Rojas, I., Romero, G.: Statistical analysis of the parameters of a neuro-genetic algorithm. IEEE Transactions on Neural Networks 13(6), 1374–1394 (November 2002), available from http://ieeexplore.ieee.org/iel5/72/22620/01058074.pdf
8. Cauwet, M.L., Liu, J., Teytaud, O., et al.: Algorithm portfolios for noisy optimization: Compare solvers early. In: Learning and Intelligent Optimization Conference (2014)
9. Chiaberge, M., Merelo, J.J., Reyneri, L.M., Prieto, A., Zocca, L.: A comparison of neural networks, linear controllers, genetic algorithms and simulated annealing for real time control. In: Proceedings of the European Symposium on Artificial Neural Networks. Index available from http://www.dice.ucl.ac.be/esann/proceedings/esann1994/content.htm. pp. 205–210. D facto (Brussels) (1994), available from http://polimage.polito.it/~marcello/articoli/esann.94.jj.pdf and a scanned version from http://www.dice.ucl.ac.be/Proceedings/esann/esannpdf/es1994-533-S.pdf
10. Costa, A., Vargas, P., Tinós, R.: Using explicit averaging fitness for studying the behaviour of rats in a maze. In: Advances in Artificial Life, ECAL. vol. 12, pp. 940–946 (2013)
11. Deb, K.: Multi-objective optimization using evolutionary algorithms, vol. 16. John Wiley & Sons (2001)
12. Di Mario, E., Navarro, I., Martinoli, A.: A distributed noise-resistant particle swarm optimization algorithm for high-dimensional multi-robot learning. In: Robotics and Automation (ICRA), 2015 IEEE International Conference on. pp. 5970–5976 (May 2015)
13. Esteban-Diaz, J., Handl, J.: Implicit and explicit averaging strategies for simulation-based optimization of a real-world production planning problem. Informatica (03505596) 39(2) (2015)
14. Fahlman, S.: Faster-Learning Variations on Back-Propagation: An Empirical Study. Proceedings of the 1988 Connectionist Models Summer School, Morgan Kaufmann (1988)
15. Fernández-Ares, A., Mora, A.M., Arenas, M.G., Guervós, J.J.M., García-Sánchez, P., Valdivieso, P.A.C.: Co-evolutionary optimization of autonomous agents in a real-time strategy game. In: Esparcia-Alcázar, A.I., Mora, A.M. (eds.) Applications of Evolutionary Computation - 17th European Conference, EvoApplications 2014, Granada, Spain, April 23-25, 2014, Revised Selected Papers. Lecture Notes in Computer Science, vol. 8602, pp. 374–385. Springer (2014)
16. Fernández-Ares, A., Mora, A.M., Guervós, J.J.M., García-Sánchez, P., Fernandes, C.: Optimizing player behavior in a real-time strategy game using evolutionary algorithms. In: IEEE Congress on Evolutionary Computation. pp. 2017–2024. IEEE (2011)

17. Fernández-Ares, A., Mora, A.M., Guervós, J.J.M., García-Sánchez, P., Fernandes, C.M.: Optimizing strategy parameters in a game bot. In: Cabestany, J., Rojas, I., Caparrós, G.J. (eds.) IWANN (2). Lecture Notes in Computer Science, vol. 6692, pp. 325–332. Springer (2011)
18. Flores, D.: Rank based evolution of real parameters on noisy fitness functions: Evolving a robot neurocontroller. In: 10th Mexican International Conference on Artificial Intelligence (MICAI). pp. 72–76. IEEE (2011)
19. Fogel, L.J., Owens, A.J., Walsh, M.J.: Artificial intelligence through simulated evolution. John Wiley (1966)
20. Friedrich, T., Kötzing, T., Krejca, M., Sutton, A.M.: The Benefit of Sex in Noisy Evolutionary Search. ArXiv e-prints (Feb 2015)
21. García-Ortega, R.H., García-Sánchez, P., Mora, A.M., Merelo, J.: My life as a sim: evolving unique and engaging life stories using virtual worlds. In: ALIFE 14: The Fourteenth Conference on the Synthesis and Simulation of Living Systems. vol. 14, pp. 580–587 (2014)
22. García-Sánchez, P., Tonda, A.P., Mora, A.M., Squillero, G., Guervós, J.J.M.: Towards automatic starcraft strategy generation using genetic programming. In: 2015 IEEE Conference on Computational Intelligence and Games, CIG 2015, Tainan, Taiwan, August 31 - September 2, 2015. pp. 284–291. IEEE (2015)
23. Goh, C.K., Tan, K.C.: An investigation on noisy environments in evolutionary multiobjective optimization. Evolutionary Computation, IEEE Transactions on 11(3), 354–381 (2007)
24. Goldberg, D.E.: Genetic Algorithms in search, optimization and machine learning. Addison Wesley (1989)
25. Groeneveld, R.A., Meeden, G.: Measuring skewness and kurtosis. The Statistician pp. 391–399 (1984)
26. Hansen, N., Finck, S., Ros, R., Auger, A.: Real-parameter black-box optimization benchmarking 2009: Noisy functions definitions (2009)
27. Hansen, N., Niederberger, A.S., Guzzella, L., Koumoutsakos, P.: A method for handling uncertainty in evolutionary optimization with an application to feedback control of combustion. Evolutionary Computation, IEEE Transactions on 13(1), 180–197 (2009)
28. Haykin, S.: Neural Networks: A Comprehensive Foundation. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edn. (1998)
29. Jin, Y., Branke, J.: Evolutionary optimization in uncertain environments - a survey. IEEE Transactions on Evolutionary Computation 9(3), 303–317 (2005), cited By (since 1996)576
30. Jin, Y.: Surrogate-assisted evolutionary computation: Recent advances and future challenges. Swarm and Evolutionary Computation 1(2), 61–70 (2011)
31. Jun-hua, L., Ming, L.: An analysis on convergence and convergence rate estimate of elitist genetic algorithms in noisy environments. Optik - International Journal for Light and Electron Optics 124(24), 6780 – 6785 (2013), http://www.sciencedirect.com/science/article/pii/S0030402613007730
32. Koza, J.R.: Genetic programming - on the programming of computers by means of natural selection. Complex adaptive systems, MIT Press (1993)
33. Laredo, J.L.J., Dorronsoro-Díaz, B., Fernandes, C., Merelo-Guervós, J.J., Bouvry, P.: Oversized populations and cooperative selection: Dealing with massive resources in parallel infrastructures. In: Nicosia, G., Pardalos, P.M. (eds.) LION. Lecture Notes in Computer Science, vol. 7997, pp. 444–449. Springer (2013)
34. Liberatore, F., Mora, A., Castillo, P., Merelo, J.: Comparing heterogeneous and homogeneous flocking strategies for the ghost team in the game of ms. pac-man. Computational Intelligence and AI in Games, IEEE Transactions on PP(99), 1–1 (2015)
35. Liu, J., St-Pierre, D.L., Teytaud, O.: A mathematically derived number of resamplings for noisy optimization. In: Proceedings of the 2014 Conference Companion on Genetic and Evolutionary Computation Companion. pp. 61–62. GECCO Comp '14, ACM, New York, NY, USA (2014), http://doi.acm.org/10.1145/2598394.2598458

36. Lucas, S.M.: Ms pac-man versus ghost-team competition. In: Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on. pp. 1–1 (Sept 2009)
37. Luke, S., Panait, L., Balan, G., Paus, S., Skolicki, Z., Bassett, J., Hubley, R., Chircop, A.: Ecj: A java-based evolutionary computation research system (2006), downloadable versions and documentation can be found at the following url: http://cs.gmu.edu/eclab/projects/ecj
38. Merelo, J.J., Castillo, P.A., Mora, A., Fernández-Ares, A., Esparcia-Alcázar, A.I., Cotta, C., Rico, N.: Studying and tackling noisy fitness in evolutionary design of game characters. In: Rosa, A., Merelo, J.J., Filipe, J. (eds.) ECTA 2014 - Proceedings of the International Conference on Evolutionary Computation Theory and Applications. pp. 76–85 (2014)
39. Merelo, J., Chelly, Z., Mora, A., Fernández-Ares, A., Esparcia-Alcázar, A.I., Cotta, C., de las Cuevas, P., Rico, N.: A statistical approach to dealing with noisy fitness in evolutionary algorithms. In: Computational Intelligence, pp. 79–95. Springer (2016)
40. Merelo-Guervós, J.J., Prieto, A., Morán, F.: Optimization of classifiers using genetic algorithms, chap. 4, pp. 91–108. MIT press (2001), iSBN: 0262162016; draft available from http://geneura.ugr.es/pub/papers/g-lvq-book.ps.gz
41. Miller, B.L., Goldberg, D.E.: Genetic algorithms, selection schemes, and the varying effects of noise. Evolutionary Computation 4(2), 113–131 (1996)
42. Mora, A.M., Fernández-Ares, A., Merelo-Guervós, J.J., García-Sánchez, P., Fernandes, C.M.: Effect of noisy fitness in Real-Time Strategy games player behaviour optimisation using evolutionary algorithms. J. Comput. Sci. Technol. 27(5), 1007–1023 (2012)
43. Mora, A.M., Montoya, R., Merelo, J.J., Snchez, P.G., Castillo, P.A., Laredo, J.L.J., Martnez, A.I., Espacia, A.: Evolving Bots AI in Unreal. In: di Chio et al., C. (ed.) Applications of Evolutionary Computing, Part I. Lecture Notes in Computer Science, vol. 6024, pp. 170–179. Springer-Verlag, Istanbul, Turkey (7 - 9 Apr 2010)
44. Ong, Y.S., Zhou, Z., Lim, D.: Curse and blessing of uncertainty in evolutionary algorithm using approximation. In: Evolutionary Computation, 2006. CEC 2006. IEEE Congress on. pp. 2928–2935. IEEE (2006)
45. Ontañón, S., Synnaeve, G., Uriarte, A., Richoux, F., Churchill, D., Preuss, M.: A survey of real-time strategy game AI research and competition in starcraft. IEEE Trans. Comput. Intellig. and AI in Games 5(4), 293–311 (2013)
46. Paredis, J.: Coevolutionary computation. Artificial life 2(4), 355–375 (1995)
47. Parras-Gutierrez, E., Arenas, M.G., Rivas, V.M., del Jesus, M.J.: Coevolution of lags and rbfns for time series forecasting: L-co-r algorithm. Soft Computing 16(6), 919–942 (2012), http://dx.doi.org/10.1007/s00500-011-0784-2
48. Peñalver, J.G., Merelo, J.J.: Optimizing web page layout using an annealed genetic algorithm as client-side script. In: Proceedings PPSN, Parallel Problem Solving from Nature V. pp. 1018–1027. No. 1967 in Lecture Notes in Computer Science, Springer-Verlag (1998), http://www.springerlink.com/link.asp?id=2gqqar9cv3et5nlg
49. Qian, C., Yu, Y., Jin, Y., Zhou, Z.H.: On the effectiveness of sampling for evolutionary optimization in noisy environments. In: Bartz-Beielstein, T., Branke, J., Filipic, B., Smith, J. (eds.) Parallel Problem Solving from Nature PPSN XIII, Lecture Notes in Computer Science, vol. 8672, pp. 302–311. Springer International Publishing (2014), http://dx.doi.org/10.1007/978-3-319-10762-2_30
50. Qian, C., Yu, Y., Zhou, Z.H.: Analyzing evolutionary optimization in noisy environments. CoRR abs/1311.4987 (2013)
51. Rada-Vilela, J., Johnston, M., Zhang, M.: Population statistics for particle swarm optimization: Resampling methods in noisy optimization problems. Swarm and Evolutionary Computation 17, 37–59 (2014), http://www.sciencedirect.com/science/article/pii/S2210650214000261

52. Rakshit, P., Konar, A., Nagar, A.: Artificial bee colony induced multi-objective optimization in presence of noise. In: Evolutionary Computation (CEC), 2014 IEEE Congress on. pp. 3176–3183 (July 2014)
53. Rattray, M., Shapiro, J.: Noisy fitness evaluation in genetic algorithms and the dynamics of learning pp. 117–139 (1998)
54. Rudolph, G.: A partial order approach to noisy fitness functions. In: Proceedings of the IEEE Conference on Evolutionary Computation, ICEC. vol. 1, pp. 318–325 (2001)
55. Squillero, G.: Microgp-an evolutionary assembly program generator. Genetic Programming and Evolvable Machines 6(3), 247–263 (2005), http://dx.doi.org/10.1007/s10710-005-2985-x
56. Stroud, P.D.: Kalman-extended genetic algorithm for search in nonstationary environments with noisy fitness evaluations. Evolutionary Computation, IEEE Transactions on 5(1), 66–77 (2001)
57. Wilcoxon, F.: Individual comparisons by ranking methods. Biometrics Bulletin 1(6), 80–83 (1945)