Beating uncertainty in racing bot evolution through enhanced exploration and pole position selection

Mohammed Salem*, Antonio M. Mora† and Juan J. Merelo‡
*Dept. of Computer Sciences, University of Mascara, Algeria.
Email: salem@univ-mascara.dz
†Dept. of Signal Theory, Telematics and Communications, ETSIIT-CITIC, University of Granada, Spain.
Email: amorag@ugr.es
‡Dept. of Computer Architecture and Computer Technology. University of Granada, Spain.
Email: jmerelo@geneura.ugr.es

***Abstract*—One of the main problems in the design through optimization of car racing bots is the inherent noise in the optimization process: besides the fact that the fitness is a heuristic based on what we think are the keys to success and as such just a surrogate for the ultimate objective, winning races, fitness itself is uncertain due to the stochastic behavior of racing conditions and the rest of the (simulated) racers. The fuzzy-based genetic controller for the car racing simulator TORCS we have defined in previous works is based on two fuzzy sub-controllers, one for deciding on the wheel steering angle and another to set the car target speed at the next simulation tick. They are both optimized by means of an Evolutionary Algorithm, which considers an already tested fitness function focused on the maximization of the average speed during the race and the minimization of the car damage. The noisy environment asks for keeping diversity high during evolution, that is why we have added a Blend Crossover (BLX-α) operator, which is, besides, able to exploit current results at the same time it explores. Additionally, we try to address uncertainty in selection by introducing a novel selection policy of parents based in races, where the individuals are grouped and compete against others in several races, so just the firsts ranked will remain in the population as parents. Several experiments have been conducted, testing the value of the different controllers. The results show that the combination of a dynamic BLX-α crossover operator plus the *pole position selection* policy clearly beats the rest of approaches. Moreover, in the comparison of this controller with one of the participants of the prestigious international Simulated Car Racing Championship, our autonomous driver obtains much better results than the opponent.**

***Index Terms*—Simulated Car Racing, TORCS, Fuzzy Controllers, Autonomous Drivers, Genetic Algorithms, Optimization, BLX-α Crossover, Race-based Selection, Uncertainty**

I. Introduction

Games are, in many cases, closed and controlled environments, mini or simulated worlds that allow you to test techniques that will then eventually be applied in real life, probably combined with other different technologies to tackle its complexity and variability. Car racing simulation includes many of the factors that are present in autonomous driving: tracks are very different and not known in advance, there are other vehicles present on the track, and the conditions change according to weather, and car deteriorates with damage. Car is also aware of this only through a set of limited sensors, and it will have to take a decision on speed and steering that is optimal in several different senses [15], including, depending on the context, the possibility of beating a set of opponents in a simulated race.

Since testing different autonomous driving methodologies in real life is usually reserved to just a few big players, methodologies as well as algorithms are usually tested in simulated environments; these simulated environments, at the same time, offer the incentive of competition among your system and others. In this paper, we will be using The Open Racing Car Simulator (TORCS) [32], a very realistic racing simulator which offers a great testbed for the implementation and evaluation of autonomous drivers. It has been used several times for the celebration of Artificial Intelligence (AI) competitions, where the aim is to create the best autonomous driver for racing [17]–[19]. Besides being able to test your car against other cars that have been published, it can be used as a standalone environment to optimize driving in a solo race.

Evolutionary Algorithms (EAs) [2] have been frequently applied as a general-purpose optimization method in this area, generally combined with behavioural engines that rule different parts of the car [13], [24], [25]. These driving engines have included lately fuzzy controllers [9], [16], [23]. These controllers use fuzzy Logic [7], a technique that is quite suitable for defining this kind of autonomous agents, since they are in part inspired by the human reasoning when driving. A fuzzy controller works with linguistic variables, and will for instance turn *slightly* to the right when the next curve is *close*, but these controllers have to be designed to map properly inputs to desired outputs in particular situations.

From the point of view of optimization, one of the main problems is that the environment is always going to change; this is correctly reflected in the simulator used, and it means that the score (and thus the ranking, if that's the eventual target) will always change, making selection of the *best* or *winning* controller probabilistic at best. This uncertainty is a challenge from two points of view: non optimal (non-winning) controllers might be selected just by chance, since they were assigned a score favored by that uncertainty, and, once they are selected, that might make the algorithm explore zones around some controllers that have been selected just casually, and

978-1-7281-1884-0/19/$31.00

leave others unexplored. For these two reasons, there are two challenges that we intend to approach in this paper: reduce uncertainty in the selection of the "best", and keep diversity high to not exploit just the areas around those individuals whose score might have been favored by chance at a certain point in the evolution process.

Previously, the authors presented an approach combining two specialized fuzzy controllers, designed by hand, that were able to decide the car's proper steering angle and desired speed at every single point (or tick) during a race [29]. This driver was later improved [30] optimizing the parameters of their membership functions by means of a Genetic Algorithm [8]; this automated design improved manual one obtaining several controllers that were able to beat the initial hand-designed controller in a race, as well as other published controllers. Finally, the authors enhanced the controller in the last paper [28] by means of the definition of new fitness functions. The selection of the best controller at the end of the evolution was based on a set of races among the best 4 solutions, getting a better driver than in previous studies.

This proved that evolutionary algorithms were able to get the fuzzy controller parameters better than a hand-made design, but at the same time revealed several challenges. In general, evolutionary algorithms optimize the fitness function that is used; evolved fuzzy controllers (hereafter FCs) will be eventually as good as the fitness function allows. But in this particular case we cannot use as fitness function the position obtained by the FC in every possible race on every possible track with every possible opponent, so we have to settle for a *surrogate* of the fitness in a very limited environment. First we opted for eliminating opponents and making evaluations in solo races; then we chose a particular track that combined straight segments as well as some curves and did not take too long to run, and eventually we had to decide what factors related to speed, damage and lap time were going to be effectively included in the final fitness function.

The two new techniques introduced in this paper, namely, *"Pole" Position selection* (race-based) and $BLX - \alpha$ crossover, try to improve on previous results by first relying less on the surrogacy of the fitness function to select the best individuals. The Pole selection will use a parameter-less fitness function to select a few individuals that will race against each other; racing will *smooth out* randomness in the fitness by putting them in a more real environment; racing cars against each other will offer a result that varies much less than simply comparing fitness. But, even so, uncertainty is present in the fitness and we should avoid excessive exploitation of the results. The $BLX-\alpha$ crossover we have introduced takes care of this aspect.

With these methods we aim to obtain more reliable and competitive controllers and we will test them against some tough opponents, including a controller from the state of the art in Simulated Car Racing Competitions.

II. State of the Art

TORCS has become one of the main environments for research on AI since its launch in 2007 [32]. *Autonomous* cars, or *bots*, created to win races in this environment need to set the optimal parameters for the cars [14] with the ultimate objective of participating in one of the Simulated Racing Car Competitions [18], [19]. However, the problem of racing a car itself is a challenge, and thus it has been used as a subject of research from the beginning even without the intention of participating in the competition; in any case, published controllers are always used as the state of the art reference, and any new bot should be compared against at least those that are available.

There are many possible ways of approaching the design of an automatic driver for a car, and through the use of different simulators and techniques, they have probably been pursued in one way or the other. They have to provide for a way to drive the car, in real time, from the inputs gathered by the car. TORCS offer a rich array of sensors, but it also includes vision [33], although most papers do rely only on the sensors, since they offer enough information for driving the car.

The way sensors and effectors are connected also varies. The simulator itself offers a baseline controller, but that is also configurable and you can choose to wire it in some other way [3]. While this might be effective for some specific purposes like the one used in the paper, evaluate the dangerousness of the track, in general working with an already wired controller that is able, for instance, to work towards a target speed instead of dealing directly with throttle and braking is much more convenient. This default controller, as a matter of fact, lets you choose what you actually want to change or optimize. Some people opt for changing just the steering [22], [23], [25].

Although you can simply create a controller and optimize a series of probabilities or rates by hand, generally meta-heuristics such as neural networks [33] or fuzzy controllers [1]. Besides, working towards the automatic configuration via optimization of the steering wheel and the target speed also allows a bit of more leverage to obtain the maximum performance out of the racing cars [16], [24]. The main intention of these authors was to imitate human driving patterns; however, in our case, we aim to optimize the performance.

Evolutionary algorithms have been often used as an optimization mechanism, but any optimization mechanism must take into account the uncertainty in selecting the best. For instance, Fernández-Ares et al. [5] use a mechanism called *Joust* selection, avoiding inherently noisy fitness by subsuming selection in a situation as close to real as possible. Several different mechanisms have been applied in the context of TORCS racing including application of filters [27] and averaging [12]. Our own work tried to take the state of the art further by introducing a fuzzy controller evolved via evolutionary algorithms in [30], which evolved a fuzzy-based driver considering the target speed in addition to controlling steering (two fuzzy sub-controllers). We optimized the parameters of the membership functions by means of a real

coded genetic algorithm, obtaining a noticeable improvement in performance. Lately, we presented [28] an improvement on our proposal considering parameter-less fitness functions and a final selection of the best based in additional races involving the top 4 individuals and other rivals.

In this study we build on this last approach, focusing on the best fitness function to date and the final selection process, but also involving two new mechanisms during the evolution: a *Blend or BLX-α Crossover operator* to control the balance between exploration and exploitation, and a *Pole Selection policy of parents* inspired by the Joust selection mentioned above and aiming to choose real good drivers in races, rather than just select those with the highest fitness values.

III. The Open Racing Car Simulator

The framework in which this study has been conducted is TORCS [32]. It is an open source, multi-player, modular and portable racing simulator that allows users to compete against other computer-controlled opponents. Its high degree of modularity and portability, together with the realistic and real-time driving simulation, make it an ideal testbed for artificial intelligence research, as stated in previous section.

Every car in TORCS includes a large set of sensors [17], whose values the car can use during a race, such as distances to track borders, to rivals, current fuel, current gear, position in the race, speed, or damage, among others. See Figure 1.

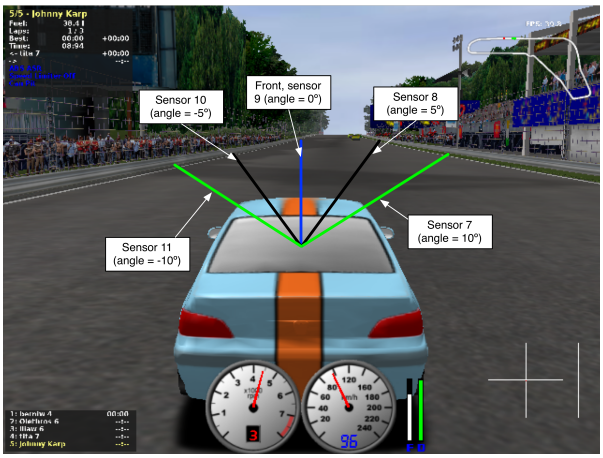

Fig. 1. TORCS capture showing some of the sensors that the car includes.

The sensor values are considered by any TORCS autonomous driver, or *controller*, to manage the car using actuators [17]: the steering wheel, the accelerator, the brake pedal and the gearbox.

IV. Fuzzy sub-controllers

We initially proposed a controller [29] with the same modular architecture as the simple TORCS driver; however, the target speed and steering angle are computed by means of two modular and specialized fuzzy sub-controllers, which consider five position sensors. This is the controller which will be improved by means of a GA in this work.

The **fuzzy target speed sub-controller** aims to estimate the optimal target speed of the car, both in straight parts and curves of the track, taking into account two criteria: move as fast as possible and be safe. This estimation is based on two general cases: if the car is in a straight line, the target speed will take a maximum value (*maxSpeed* km/h). However, if it is close to a curve, the controller will decrease the current speed to a value included in the interval *[minSpeed, maxSpeed]* km/h.

This fuzzy controller has an output, the speed, and three input values (See Figure 1):

- Front = Track_9: front distance to the track border (angle 0°).
- M5 = max (Track_8, Track_10): max distance to the track border in an angle of +5°and -5°with respect to Front.
- M10 = max (Track_7, Track_11): max distance to track border in an angle of +10°and -10°.

It is a Mamdani-based fuzzy system [11] with three trapezoidal Membership Functions (MF) for every input variable. In [30] the different sets of parameters which define the membership functions were improved using a Genetic Algorithm to obtain the best results.

Moreover, the controller is based in a set of fuzzy rules, designed to maximize the car speed depending on the distance to the track border. These rules can be consulted in [29]. The second is the **fuzzy steering sub-controller**, which aims to define the steer angle estimating and determining the target position of the car. The structure of this sub-controller is similar to the speed one, but it has the steering as output. Thus, the set of sensors considered is the same as in the speed case.

Then, as general rules: if the car is in a straight line, it will set as target position half width of the race track (central position of the lane). Whereas, if the car is near a right curve, it will approach the path leading to the right, with a space between the car and the border of the track to avoid the loss of control. The same approach is considered if the car is near a left curve.

In order to detect the curves, the controller focuses on the sensor values (M10, M5, and Front). So, if the value on Front sensor is the longest, there is a straight road; whereas if the values of M5 and M10 with positive angles (+5 and +10) are the longest, there is right curve; and the other way round.

It uses a base of rules which has been defined trying to model the behavior of a human driver [29].

As stated, the designed fuzzy controllers have trapezoidal membership functions given by Equation 1. In such a controller, fuzzy rules are applied to linguistic terms. These terms, which qualify a linguistic variable, are defined through membership functions, which, in turn, depend on a set of parameters that 'describes' their shape (and operation). Using a GA we will optimize the parameters of the membership functions that constitute the fuzzy partition of the linguistic variable [31]. The input linguistic variables in our problem, *Front, Max5* and *Max10*, are represented by three trapezoidal membership functions.

A trapezoidal membership function in a finite universe of discourse *[a, b]* can be defined by:

$$\mu_A(x) = \begin{cases} \frac{x-x_1}{x_2-x_1}, & x_1 \leq x \leq x_2 \\ 1, & x_2 \leq x \leq x_3 \\ \frac{x_4-x}{x_4-x_3}, & x_3 \leq x \leq x_4 \\ 0, & else \end{cases} \quad (1)$$

with:

$$x_1 \leq x_2 \leq x_3 \leq x_4 \quad (2)$$

This MF function is defined by four parameters x_1, x_2, x_3 and x_4 taking their values in the interval *[a, b]*.

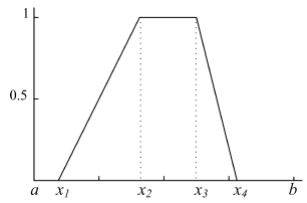

Fig. 2. Trapezoidal MFs

And a fuzzy partition with *n* trapezoidal membership functions is defined by *2n* variables (x_1,x_2,. .., x_{2n})(Equation 4). In this case, the representation is given by Figure 3.

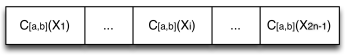

Fig. 3. Trapezoidal-shaped MFs coding

With:

$$x_1 \leq x_2 \leq ... \leq x_{2n-1} \leq x_{2n} \quad (3)$$

The first variable x_1 is chosen to be equal to the lower boundary of the range (*a* while x_{2n} is equal to *b*.

$$\mu_{A1}(x) = \begin{cases} 1, & x_1 \leq x \leq x_2 \\ \frac{x_3-x}{x_3-x_2}, & x_2 \leq x \leq x_3 \\ 0, & x > x_3 \end{cases}$$

$$\mu_{Ai}(x) = \begin{cases} 0, & x \leq x_{2i-2} \\ \frac{x-x_{2i-2}}{x_{2i-1}-x_{2i-2}}, & x_{2i-2} \leq x \leq x_{2i-1}, n = 2, ..., i-1 \\ 1, & x_{2i-1} \leq x \leq x_{2i} \\ \frac{x_{2i+1}-x}{x_{2i+1}-x_{2i}}, & x_{2i} \leq x \leq x_{2i+1} \\ 0, & x > x_{2i+1} \end{cases}$$

$$\mu_{An}(x) = \begin{cases} 0, & x \leq x_{2n-2} \\ \frac{x-x_{2n-2}}{x_{2n-1}-x_{2n-2}}, & x_{2n-2} \leq x \leq x_{2n-1} \\ 1, & x > x_{2n-1} \end{cases} \quad (4)$$

This is the base of the optimization conducted by the Genetic Algorithm, as it is described in the following section.

V. Genetic Algorithm

We proposed an optimization approach based in Genetic Algorithms (GAs) [8] aiming to find the optimal parameters of the membership functions of the two sub-controllers previously introduced.

Thus, every individual/chromosome is a vector of 18 values/parameters, 6 per variable, as Figure 4 shows.

The initialization of the chromosomes (first population) is performed by assigning random values inside a range of variation ($[0, 100]$) [8], in order to start from feasible values [29]. Since our work requires some precision and the variation interval of each parameter is not well known, we have considered a real coding implementation [4] in a vector that includes all variables to optimize.

The overall process is summarized in Figure V. As it can be seen, TORCS is used during the evaluation step of every individual in the evolutionary process.

The evaluation of the individuals is based in different fitness functions, which we have tested in previous works [30]. In this study we will consider the one which yielded the best results in our previous paper [28], namely:

$$f_{AVS} = \frac{AVG(Speed)}{Damage+1} \quad (5)$$

It is a parameter-less approach (no weights in the terms) [10], which is also more focused on the real objectives for a driver during a race, rather than the overall target of winning or not, in order to obtain more 'human-like' controllers. It depends on two variables, so the function aims to obtain drivers reaching the highest average speed as possible on the whole track while avoiding damage:

- $AVG(Speed)$: pursues a combination of good driving in the difficult zones of the tracks (e.g. curves) and also on easy or straight parts; i.e. considers the overall behavior in the whole track.
- $Damage$: aims to create 'safe' controllers, as it is mandatory being able to finish the race.

So, the fitness of each candidate solution is computed by injecting its gene values to the parameters of the membership functions of the two fuzzy sub-controllers. The defined autonomous controller is used to drive a car in a 20 laps race in a circuit without opponents, and the results (Maximum, Minimum and Average speed, Damage) are used to compute the fitness value. As the objective of the car controller is to win as many races as possible, we tried to optimize the most general case by carrying out solo *training races*, which will be less sensitive to the presence of noise/uncertainty due to the participation of other controllers [20]. The selected track for this evaluation will be one with a combination of curves and straight parts in order to obtain an 'all-terrain behaviour'.

With regard to the genetic operators, mutation has remained the same as in previous approaches of our genetic controller, i.e. **non uniform mutation** [21].

A new **Pole Position Selection policy** (or race-based selection) has been implemented in this approach, aiming to get better or more reliable individuals/controllers to be parents of the following population. To this end all the individuals are arranged in groups of 10, then some different races of several laps are simulated using every individual as a controller (with

Fig. 4. Chromosome description

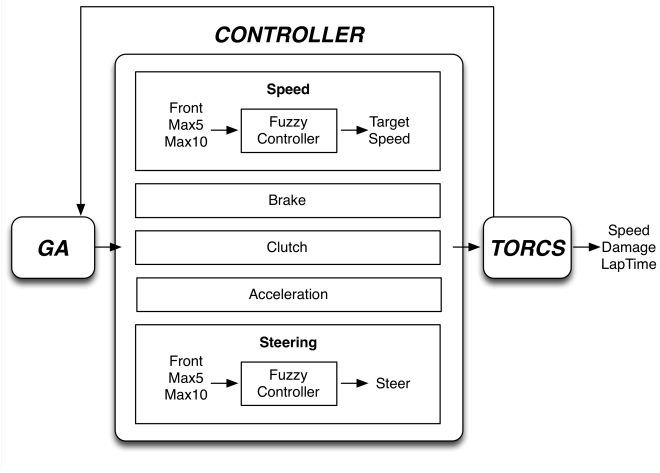

Fig. 5. Flowchart of the optimization process of a TORCS fuzzy controller. To evaluate an individual we put the parameter values of the two sub-controllers in the corresponding chromosome, then we launch a race in TORCS with this configuration, obtaining the resulting values of Damage, Top Speed and Mean Lap Time. Individual's fitness value is computed using these values.

the same car) in a track of TORCS. After every race, the participants obtain different scores depending on their position in the final rank. The best 5 controllers in the sum of all the races are selected as parents for the following offspring.

This way, the best individuals will be selected to reproduce with higher probability. It is not possible to assure they are absolutely the best, due to the uncertainty present in this type of environments, i.e. games against non-deterministic opponents [20]. However, we argue that this selection policy will be 'less sensitive' to that uncertainty (or noise), and thus, it will be fairer and more reliable than an approach purely based on the fitness values. Thus, we think that this proposed selection mechanism would benefit getting a good optimization process.

BLX-α Crossover operator [6] has also been added to the GA (instead of previous two-point operator). The Blend crossover operator starts by choosing randomly a number from the interval $[x_i - \alpha(y_i - x_i)..y_i + \alpha(y_i - x_i)]$, where x_i and y_i are the i^{th} parameter values of the parent solutions x,y and $x_i < y_i$. See Figure 6.

Thus, this operator is based on the random generation of genes from the associated neighborhood of the genes in the parents. Three generated descendants are different among them and also among them and their parents, leading to a higher exploration factor in the generation of the offspring. This operator is suitable for real coded genetic algorithms and it has proved to achieve a good balance between exploration and exploitation [6].

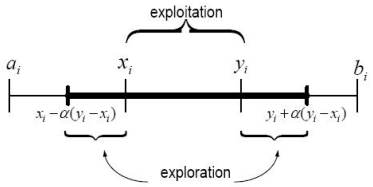

Fig. 6. Blend crossover operator ($BLX - \alpha$)

In the Blend crossover operator, the α parameter values can control the exploration/exploitation rate. So, in order to ensure the balance between exploitation and exploration of the search space, $\alpha = 0.5$ could be selected.

In GAs, the search process needs a high exploration rate in the first generations to explore multiple parts of the search space so to obtain high diversity but in the last generations, high exploitation is preferred to ensure the optimal solution.

We have considered two different approaches in the experiments, one taking a constant value of α, and another with a variable scheme, in which the value of α is decreased over the generations (getting sequentially more exploitation and less exploration). Thus, its value is obtained as:

$$\alpha = 1 - \frac{g}{g_{max}} \tag{6}$$

Where g is the current generation and g_{max} is the maximum number pf generations. We think that this approach can achieve an effective balance between exploitation and exploration and therefore, better solutions may be reached.

VI. Experiments and results

On the basis of the results of our previous paper [28], the selection of an appropriate track for training is an important factor in order to obtain competitive bots. *Alpine 2* circuit has been selected for the experiments, since it combines multiple turns with straight parts (See Figure 7).

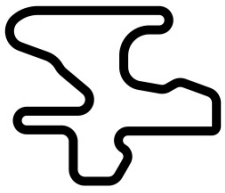

Fig. 7. Alpine 2 Track: Slow mountain road. Length: 3773.57m, Width: 10m

As in our others studies, we have used the vehicle *car1-tbr1* for our controllers, since it has a moderate performance, which will lead our controller to be prepared to drive in the most usual conditions.

We have evaluated the Genetic Fuzzy Controller (GFC) with the proposed fitness function: f_{AVS} (Equation 5). We have run the algorithm with a population size of 60 individuals. The rest of parameters are: Generations=50, Crossover rate=0.85, Mutation rate=0.09, and 10 different runs per configuration.

New pole position selection has been conducted considering *Alpine 2* track, 5 races and 20 laps per race. We have defined a score function based in Formula 1 schema, so the obtained punctuations depend on the car position in the final rank: 1 - 25 points, 2 - 18, 3 - 15, 4 - 12, 5 - 10, 6 - 8, 7 - 6, 8 - 4, 9 - 2, 10 - 1. The the starting grid (initial positions of cars) on these races was set randomly. Due to the high-demanding time this method is, this race-based selection process has just been performed every 5 generations (not in all of them).

At the end of the evolution (in the last generation) an additional race-based selection process is applied as in previous work [28]. So, the best 10 individuals (according to their fitness) of the final population compete in 5 races (of 5 laps) in the *Alpine 2* track. Formula 1 scores are again applied and the winner will be selected as the best controller of the run.

The evolution process was applied in separate bunches of runs to obtain the following controllers:

- GFC: Controller from our previous work [28] using race-based selection only in the final generation and with fitness f_{AVS} (Equation 5).
- $GFC - RS$: A controller obtained by applying the classical two points crossover operator (as in previous works) and race-based selection once every 5 generations and fitness f_{AVS} in the others.
- $GFC-FA$: A controller obtained by applying $BLX-\alpha$ crossover operator with a constant value of $\alpha = 0.5$ and race-based selection once every 5 generations and fitness f_{AVS} in the others.
- $GFC-VA$: A controller obtained by applying $BLX-\alpha$ crossover operator with a varying value of α using Equation 6 and race-based selection once every 5 generations and fitness f_{AVS} in the others.

Once the 10 runs have finished, the obtained 10 best controllers compete again in a similar set of races as those conducted in the last generation of the algorithms, in order to choose the best controller overall per approach, i.e. the best $GFC - RS$, $GFC - FA$ and $GFC - VA$.

The final best GFCs (one per approach) are evaluated in some races together in a kind of Formula 1 *mini championship*, consisting of 10 races, each one for 20 laps, and with a total of 10 participants per race: the 4 GFCs and also 6 standard bots from TORCS. We have choose two controllers between `tita` (a conservative driver), `berniw` (known by its aggressive overtaking policy) and `inferno` (the fastest one). The first 5 races are conducted in *Alpine 2* track (used during training/optimization); and the other 5 races took place in *E-Track 5* track (not trained for the new controllers). Finally, in order to do it fairer, we have defined an *additional score*, so the controller which gets the fastest lap or the minimum damage in each race is given 5 extra points. The starting grid was again set randomly.

The results of this comparison are shown in Table I and summarized graphically in Figure 8.

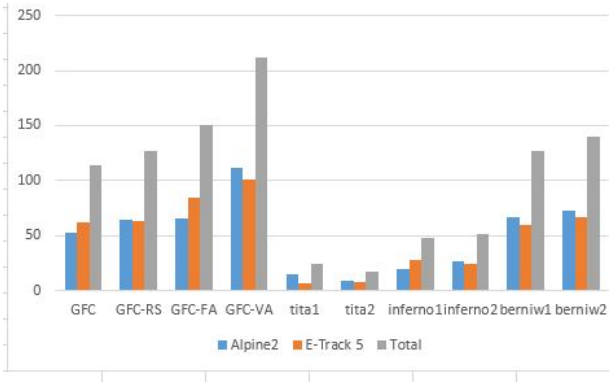

Fig. 8. Scores obtained by the different Genetic fuzzy-based controllers in two different tracks.

It is clear from the table and figure that the $GFC - VA$ controller yields the best results in the evaluation process. Indeed, The proposed controller won three races in Alpine 2 track and has been ranked in the second place in the two other racing tracks. In the E-Track 5 circuit, it won two races, has been second twice and third in the last race.

The second controller using the $BLX - \alpha$ (constant value of 0.5) operator came second in all races. It won one race and was ranked second in two more and third in three races. The other races were won by the $berniw$ controller, always a tough rival. We can notice that the $BLX - \alpha$ based controllers won three out of the five races in the Alpine 2 track used in the selection and were ranked at least in fourth place. The same results or even better were obtained for the other track, which is supposed to be unknown for our controllers.

These results confirm the effectiveness and strength of the pole position selection policy used to evaluate individuals as well as to select candidates for crossover. Although this policy has been applied only once every 5 generations due to its time-consuming, it has clearly affected the performance of the obtained controllers looking to the large gap between the results of the GFC controller against $GFC - RS$ one.

This proposed selection policy combined with the $BLX-\alpha$ operator, has boosted the performance of the $GFC - FA$ controller. The introduction of a variable α parameter along the generations in $GFC - VA$ bot has made it possible to better control the exploration/exploitation ratio during the evolutionary process, allowing to generate descendants different from their parents in genes and more efficient than them.

In order to check the value of our best controller, we have conducted an additional experimentation.

We have considered an opponent from the state of the art, which participated in several Simulated Car Racing Competitions in past editions. It was proposed by Pérez-Liébana, Sáez,

TABLE I
RESULTS OF THE MINI-CHAMPIONSHIP WITH 10 DRIVERS AND 10 RACES IN TWO DIFFERENT TRACKS. `tita`, `berniw` AND `inferno` ARE EXAMPLE CONTROLLERS INCLUDED WITH THE TORCS SIMULATOR [32]

Driver	Races in *Alpine 2* track (20 laps each)						Races in *E-Track 5* track (20 laps each)						Total Score
	R1	R2	R3	R4	R5	Track Score	R6	R7	R8	R9	R10	Track Score	
GFC	6	8	8	15	15	52	12	15	10	10	15	62	134
$GFC-RS$	12	10	18	12	12	64	10	8	12	25	8	63	127
$GFC-FA$	18	12	15	10	10	65	15	12	25	15	25	92	157
$GFC-VA$	25	18	25	25	18	111	25	18	15	18	18	94	205
$tita1$	4	2	6	4	1	15	2	1	2	1	1	7	22
$tita2$	2	1	2	2	2	9	1	2	1	2	2	8	17
$inferno1$	8	4	1	1	6	20	4	4	6	8	6	28	48
$inferno2$	1	6	4	8	8	27	6	6	4	4	4	24	51
$berniw1$	10	25	10	18	4	67	18	10	8	12	12	60	127
$berniw2$	15	15	12	6	25	73	8	25	18	6	10	67	140

Recio and Isasi [26] and later refined in the work [16]. We have baptised it as PSRI in honor of its authors' surnames.

This controller behaves mainly using a Finite State Machine (FSM), defining the main states in which the driver can be (for instance turning, overtaking a rival). The transitions in the FSM are governed by a set of fuzzy rules, based on the information read from different sensors. There is also a classifier module (J48 decision tree), able to analyse the inputs from some sensors in order to predict parts of the track, to anticipate the following actions to perform. The fuzzy rules and also some parameters of the FSM were optimized by means of a NSGA-II algorithm.

As stated, PSRI controller competed in 2009 edition of the Simulated Car Racing Championship [18], where it was ranked 4th considering the scores obtained in three different Competitions (held at CEC, GECCO and CIG 2009 conferences). It performed on average very well, reaching good scores and positions in several races.

Table II presents a comparison between the two $BLX-\alpha$ genetic based fuzzy controllers presented in this paper, $GFC-FA$ and $GFC-VA$ with PSRI controller. The results are the average values of $damage$, $MaxSpeed$ and $Speed$ of 10 races in the *Alpine 2* and *E-Track 5* tracks.

TABLE II
AVERAGE DAMAGE AND SPEED RESULTS OF 5 RACES IN **ALPINE 2** AND 5 RACES IN **E-TRACK 5** TRACKS

Alpine 2			
	$GFC-FA$	$GFC-VA$	$PSRI$
Average Speed (km/h)	187.11	199.65	176.94
Max Speed (km/h)	225.07	231.91	217.83
Damage	126.82	117.55	131.99
Won races	0	4	1
E-Track 5			
	$GFC-FA$	$GFC-VA$	$PSRI$
Average Speed (km/h)	161.11	170.23	160.89
Max Speed (km/h)	262.88	270.17	266.54
Damage	18.12	14.67	28.09
Won races	1	3	1

The results of the $PSRI$ controller and $GFC-FA$ are very close, they won two and one race respectively among 10. Their average speeds are similar but as for the damage, the controller $GFC-FA$ suffered the minimum because of the inclusion of the variable $damage$ in the fitness evaluation. The results of the $GFC-VA$ controller are very satisfactory. Indeed, it won 7 races and got the lowest value of damage 117.55 and 14.67 for both circuits, the highest average speed 199.65 and 170.23.

Looking at these and previous results, we can conclude that the proposed controllers are very successful, due to the new included mechanisms to deal with uncertainty and to perform a more convenient search of the space of solutions.

VII. CONCLUSIONS AND FUTURE WORK

In this paper we have tried to get winning racing car drivers by first improving the selection process so that, from time to time, uncertainty is eliminated by using actual competitions instead of fitness-based evolution, and second, keep the balance between exploration and exploitation high, and also variable, by using a fixed and adaptive version of the BLX-α operator.

The fuzzy genetic controller is subject to uncertainties in the track especially in case of presence of rivals so in order to overcome this problem and thus design a robust and reliable bot, we proposed to apply a *Pole Position Selection policy* where the selection of parents in the evolutionary process is carried out according to the results of a set of mini-championships organized among the individuals of the population, which looks like a car racing tournament selection. At the same time, and aiming to intensify the exploration process in the search space, we used the $BLX-\alpha$ crossover operator with decreasing values of the α parameter throughout the generations.

The evaluation was performed by comparing the proposed controller with bots of the TORCS platform, yielding very good results. The other evaluation of our controller was a confrontation with a real bot ($PSRI$ controller), which participated in several Simulated Car Racing Competitions. In this case, the BLX operator and the new selection policy have had a lot of impact in helping our controller to win three quarters of the races by getting the lowest damage, average speed and maximum speed values.

These results let us to think that our controller could have reached a very good rank in the Simulated Car Racing Competition, which is unfortunately over since 2015. Anyway, we think that the findings of this study (and previous ones) could be applied successfully to other car racing simulators,

such as those used in current eSports Competitions, such as iRace (https://www.iracing.com/).

As future lines of work, this controller can be improved in some ways: We can extend the selection policy to all generations while overcoming the computation time drawback by means of a parallel implementation. We can also explore other parameter-less fitness functions to evaluate individuals including other factors affecting the performance of the car. Another perspective is to use multiple tracks (instead of just one) in the selection process in order to train a more general controller, able to deal with many different situations.

ACKNOWLEDGMENTS

This work has been funded by projects TIN2017-85727-C4-2-P, RTI2018-102002-A-I00 (Spanish Ministry of Science, Innovation and Universities) and TEC2015-68752 (Spanish Ministry of Economy and Competitiveness + FEDER).

REFERENCES

[1] E. Armagan and T. Kumbasar, "A fuzzy logic based autonomous vehicle control system design in the torcs environment," in *2017 10th International Conference on Electrical and Electronics Engineering (ELECO)*. IEEE, 2017, pp. 737–741.

[2] T. Bäck, *Evolutionary algorithms in theory and practice*. Oxford University Press, 1996.

[3] S. Cussat-Blanc, J. Disset, and S. Sanchez, "Dangerousness metric for gene regulated car driving," in *European Conference on the Applications of Evolutionary Computation*. Springer, 2016, pp. 620–635.

[4] S. M. M. Elsayed, R. Sarker, and D. L. Essam, "A genetic algorithm for solving the CEC2013 competition problems on real-parameter optimization," in *IEEE Congress on Evolutionary Computation, CEC 2013*, Cancun, Mexico, 21-23 June 2013 2013, pp. 356–360.

[5] A. Fernández-Ares, P. García-Sánchez, A. M. Mora, P. A. Castillo, and J. J. Merelo, "There can be only one: Evolving RTS bots via joust selection," in *Applications of Evolutionary Computation - 19th European Conference, EvoApplications 2016, Proceedings, Part I*, 2016, pp. 541–557.

[6] C. García-Martínez, M. Lozano, F. Herrera, D. Molina, and A. Sánchez, "Global and local real-coded genetic algorithms based on parent-centric crossover operators," *European Journal of Operational Research*, vol. 185 (3), pp. 1088–1113, 2008.

[7] S. Godil, M. Shamim, S. Enam, and U. Qidwai, "Fuzzy logic: A 'simple' solution for complexities in neurosciences?" *Surg Neurol Int.*, pp. 2 – 24, 2011.

[8] D. E. Goldberg, *Genetic Algorithms in search, optimization and machine learning*. Addison Wesley, 1989.

[9] S. Guadarrama and R. Vazquez, "Tuning a fuzzy racing car by coevolution," in *Genetic and Evolving Systems, GEFS 2008*, March 2008.

[10] G. R. Harik and F. G. Lobo, "A parameter-less genetic algorithm," in *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation - Volume 1*, ser. GECCO'99. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, pp. 258–265.

[11] I. Iancu, *A Mamdani Type Fuzzy Logic Controller*. InTech, 2012, pp. 325–352.

[12] S. J.L., K. Chin, T. Jason, T.G.Tan, and A. Rayner, "Evolving controllers for simulated car racing using differential evolution," *Asia-Pacific Journal of Information Technology and Multimedia*, 2013.

[13] T. S. Kim, J. C. Na, and K. J. Kim, "Optimization of an autonomous car controller using a self-adaptive evolutionary strategy," *International Journal of Advanced Robotic Systems*, vol. 9, no. 3, p. 73, 2012.

[14] M. Köle, A. S. Etaner-Uyar, B. Kiraz, and E. Özcan, "Heuristics for car setup optimisation in TORCS," in *12th UK Workshop on Computational Intelligence (UKCI)*, Sept 2012, pp. 1–8.

[15] S. Kolski, D. Ferguson, C. Stacniss, and R. Siegwart, "Autonomous driving in dynamic environments," in *In Proceedings of the Workshop on Safe Navigation in Open and Dynamic Environments at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Beijing, China, 2006.

[16] D. P. Liébana, G. Recio, Y. Sáez, and P. Isasi, "Evolving a fuzzy controller for a car racing competition," in *Proceedings of the 2009 IEEE Symposium on Computational Intelligence and Games, CIG 2009, Milano, Italy, 7-10 September, 2009*, 2009, pp. 263–270.

[17] D. Loiacono, L. Cardamone, and P. Lanzi, "Simulated car racing championship competition. software manual," *TORCS news*, 2013.

[18] D. Loiacono, P.-L. Lanzi, J. Togelius, E. Onieva, D. A. Pelta, M. . Butz, T. D. Lonneker, L. Cardamone, D. Perez, Y. Saez, M. Preuss, and J. Quadflieg, "The 2009 simulated car racing championship," *IEEE Trans. Comput. Intell. AI Games*, vol. 2(2), pp. 131–147, 2010.

[19] D. Loiacono, J. Togelius, P. L. Lanzi, L. Kinnaird-Heether, S. M. Lucas, M. Simmerson, D. Perez, R. G. Reynolds, and Y. Saez, "The wcci 2008 simulated car racing competition," in *2008 IEEE Symposium On Computational Intelligence and Games*, Dec 2008, pp. 119–126.

[20] J. Merelo, Z. Chelly, A. Mora, A. Fernández-Ares, A. I. Esparcia-Alcázar, C. Cotta, P. de las Cuevas, and N. Rico, "A statistical approach to dealing with noisy fitness in evolutionary algorithms," in *Computational Intelligence*. Springer, 2016, pp. 79–95.

[21] A. Neubauer, "A theoretical analysis of the non-uniform mutation operator for the modified genetic algorithm," in *Proceedings of the IEEE International Conference on Evolutionary Computation*, 1997.

[22] V. Nikulin, A. Podusenko, I. Tanev, and K. Shimohara, "Evolving the autosteering of a car featuring a realistically simulated steering response," in *Proceedings of the Genetic and Evolutionary Computation Conference*, ser. GECCO '18. New York, NY, USA: ACM, 2018, pp. 1326–1332.

[23] E. Onieva, J. Alonso, J. Pérez, and V. Milanés, "Autonomous car fuzzy control modeled by iterative genetic algorithms," in *Fuzzy Systems*, 2009, pp. 1615 – 1620.

[24] E. Onieva, D. Pelta, J. Godoy, V. Milanés, and J. Rastelli, "An evolutionary tuned driving system for virtual car racing games: The autopia driver," *International Journal of Intelligent Systems*, vol. 27, pp. 217–241, 2012.

[25] E. Onieva, D. A. Pelta, J. Alonso, V. Milanés, and J. Pérez, "A modular parametric architecture for the torcs racing engine," in *Proceedings of the 5th IEEE Symposium on Computational Intelligence and Games (CIG'09)*. Piscataway, NJ, USA: IEEE Press, 2009, pp. 256–262.

[26] D. Perez, Y. Saez, G. Recio, and P. Isasi, "Evolving a rule system controller for automatic driving in a car racing competition," in *2008 IEEE Symposium On Computational Intelligence and Games*, Dec 2008, pp. 336–342.

[27] M. Preuss, J. Quadflieg, and G. Rudolph, "Torcs sensor noise removal and multi-objective track selection for driving style adaptation," in *2011 IEEE Conference on Computational Intelligence and Games (CIG'11)*. IEEE, 2011, pp. 337–344.

[28] M. Salem, A. M. Mora, and J. J. Merelo, "The evolutionary race: Improving the process of evaluating car controllers in racing simulators," in *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, Aug 2018, pp. 1–8.

[29] M. Salem, A. M. Mora, J. J. Merelo, and P. García-Sánchez, "Driving in TORCS using modular fuzzy controllers," in *Applications of Evolutionary Computation. EvoApplications 2017, LNCS, vol 10199*, S. K. Squillero G., Ed. Springer, Cham, 2017, pp. 361–376.

[30] M. Salem, A. M. Mora, J. J. Merelo, and P. García-Sánchez, "Evolving a TORCS modular fuzzy driver using genetic algorithms," in *Applications of Evolutionary Computation*, K. Sim and P. Kaufmann, Eds. Cham: Springer International Publishing, 2018, pp. 342–357.

[31] H. D. Thang and J. M. Garibaldi, "A novel fuzzy inferencing methodology for simulated car racing," in *IEEE International Conference on Fuzzy Systems, Hong Kong, China, 1-6 June, 2008, Proceedings*. IEEE, 2008, pp. 1907–1914.

[32] B. Wyman, E. Espie, C. Guionneau, C. Dimitrakakis, R. Coulom, and A. Sumner, "TORCS the open racing car simulator," 2000. [Online]. Available: http://www.torcs.org

[33] Y. Zhu and D. Zhao, "Driving control with deep and reinforcement learning in the open racing car simulator," in *International Conference on Neural Information Processing*. Springer, 2018, pp. 326–334.