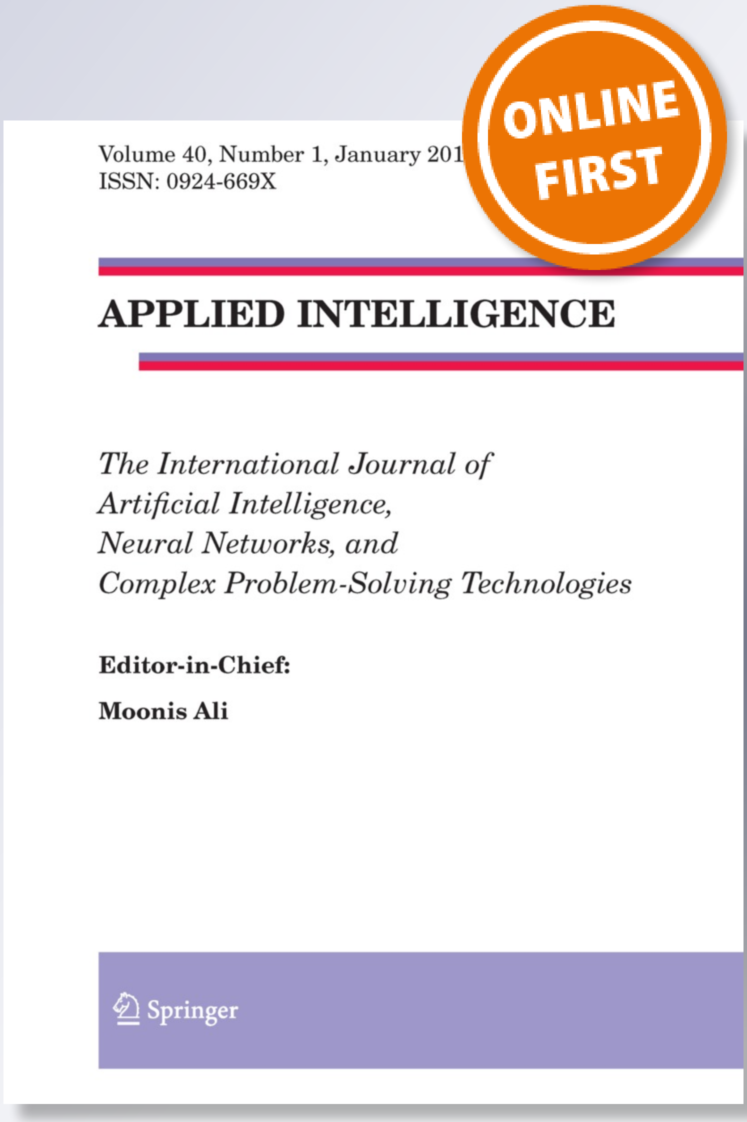
GABF: genetic algorithm with base fitness for obtaining generality from partial results: study in autonomous intersection by fuzzy logic

E. Onieva, E. Osaba, X. Zhang & A. Perallos

Applied Intelligence
The International Journal of Artificial Intelligence, Neural Networks, and Complex Problem-Solving Technologies

ISSN 0924-669X

Appl Intell
DOI 10.1007/s10489-013-0498-5


Volume 40, Number 1, January 201
ISSN: 0924-669X
ONLINE FIRST
APPLIED INTELLIGENCE
The International Journal of Artificial Intelligence, Neural Networks, and Complex Problem-Solving Technologies
Editor-in-Chief:
Moonis Ali
Springer

Appl Intell
DOI 10.1007/s10489-013-0498-5

GABF: genetic algorithm with base fitness for obtaining generality from partial results: study in autonomous intersection by fuzzy logic

E. Onieva · E. Osaba · X. Zhang · A. Perallos

Abstract Many applications of optimization techniques, such as classification and regression problems, require long simulations to evaluate the performance of their solutions. Problems where the fitness function can be divided into smaller pieces—problem partitioning—demand techniques that approximate the overall fitness from that obtained in a small region of the problem space. This means that less time is spent evaluating individual solutions, which makes such approaches computationally efficient.

In this work, a method is proposed to deal with a dynamically calculated fitness function; it is called Genetic Algorithm with Base Fitness (GABF). This method is built over a Genetic Algorithm (GA) to optimize a Fuzzy Rule-Based System (FRBS). The proposed method works by partitioning training data into smaller subsets. The main idea is to assign fitness values derived from part of the training set (or a short simulation) to individuals in the current generation. This fitness value is then inherited and combined with those obtained in subsequent generations.

To test the proposal, a scenario in which two vehicles are approaching an intersection is implemented. One vehicle is presumed to be driven by a human and does not change its speed, whereas the other implements an autonomous speed regulator based on fuzzy logic. The regulator must maneuver the *autonomous* vehicle in a safe and efficient manner. The objective is to optimize both the membership functions and the rule base of the fuzzy system controlling the *autonomous* vehicle.

Keywords Intelligent transportation systems · Autonomous vehicles · Intelligent intersections · Genetic algorithms · Efficiency enhancement techniques · Fuzzy rule-based systems

1 Introduction

Genetic Algorithms (GAs) are search techniques based on the process of evolution. They provide robust search capabilities in complex spaces, and thereby offer a valid approach to problems requiring efficient and effective search processes [13, 19]. Once a problem has been represented for treatment by the GA, the key aspect is the method of distinguishing between good and bad solutions (called the fitness function). This can either be interactively determined by a human, or be the result of a computer simulation. GAs do not differentiate between these different modes of fitness assignments, as long as good solutions have better fitness values than bad ones. Unlike many traditional optimization techniques, there is no requirement for the fitness function of a GA to be continuous, convex, or well-formed [30].

A promising area of research in genetic and evolutionary algorithms is the design and development of competent GAs, which solve hard problems quickly, reliably and accurately [14]. When the fitness function requires a complex simulation or computation, a single evaluation might take a large amount of time. This places a premium on Efficiency Enhancement Techniques (EETs) [30]. These techniques can be broadly classified into four categories:

E. Onieva (✉) · E. Osaba · X. Zhang · A. Perallos
Deusto Institute of Technology (DeustoTech), University of Deusto, Bilbao 48007, Spain
e-mail: enrique.onieva@deusto.es

E. Osaba
e-mail: e.osaba@deusto.es

X. Zhang
e-mail: xiao.zhang@deusto.es

A. Perallos
e-mail: perallos@deusto.es

Published online: 17 January 2014

1. *Parallelization*: GAs are run on multiple processors, and the computational resources are distributed among them [6, 7].
2. *Hybridization*: Domain-specific knowledge and other techniques are coupled with a GA to create a search bias and accelerate the search process. The most common form is to couple GAs with local search techniques. Such approaches are known as memetic algorithms [17, 24].
3. *Time continuation or time utilization*: A tradeoff is sought between the search for solutions with a large population and a single convergence epoch, or a small population with multiple convergence epochs [31, 32].
4. *Evaluation relaxation*: Sometimes, a complex fitness function can be replaced by a simple and inexpensive one. This can reduce the total number of costly fitness evaluations [23].

The simple fitness function used in evaluation relaxation can be either *exogenous*, as for surrogate (or approximate) fitness [20], or *endogenous*, as for fitness inheritance [3, 12] whereby fitness is based on that of the parents. In classification or regression problems, the approximate fitness is calculated by dividing the training set into a smaller, representative set [22], thus reducing the computational time required to evaluate individuals. This process, known as stratification, can also be done by means of evolutionary techniques [4, 5]. In contrast, fitness inheritance allows the GA to estimate the fitness of a portion of the population from the fitness of its parents, again saving computational effort [10].

Fuzzy logic techniques [38] have demonstrated the ability to solve different kinds of problems, such as: classification [11], modeling [21, 27] and control [29]. Fuzzy Rule-Based Systems (FRBSs) that rely on evolutionary optimization techniques are known as Genetic Fuzzy Systems (GFSs) [18]. Thus, a GFS is a FRBS augmented by a learning process based on a GA. GFSs have been used in a variety of applications, such as the classification of high-dimensional data [1] and control systems for air conditioners [11].

This work presents a combination of evaluation relaxation and fitness inheritance in a GA-optimized FRBS [35]; thus, the method is called Genetic Algorithm with Base Fitness (GABF). The FRBS determines the speed of a vehicle when traversing an unregulated intersection. The algorithm considers both the approximate fitness, in the evaluation of individuals, and the fitness inheritance after crossover and mutation. The objective is to use the results from a small set of test cases to obtain a FRBS that can deal with any intersection scenario.

The FRBSs are tested in scenarios involving an intersection and two approaching vehicles. One of them is presumed to be human-driven (with no obligation to cooperate), whereas the other is an autonomous vehicle which implements automatic speed regulation. The latter takes responsibility for adapting its own speed according to the position and speed of the former. The FRBS is in charge of providing speed indications to the autonomous vehicle with the objective of crossing the intersection without sudden speed variations or risk. To obtain generality in the obtained FRBSs, multiple training scenarios must be used. However, the use of a large number of scenarios might affect the execution time of the GA. At this point, the proper management and inheritance of the approximate fitness function could be a considerable advantage, as demonstrated in this work.

The remainder of this paper is structured as follows: Sect. 2 presents the proposed method before Sect. 3 introduces the problem and the fitness function used in the proposed approach. Section 4 describes both the experimental setup and results obtained. Finally, Sect. 5 presents concluding remarks and ideas for future developments.

2 Genetic algorithm with base fitness

The purpose of including the concept of base fitness in a GA is to theoretically preserve information about an individual's performance in a subset of cases across different generations. To achieve this goal, the fitness function is presumed to be suitable for partitioning into smaller pieces, whose aggregation leads to the final fitness value. Examples of partitioning are:

– Optimization problems whose objective is to minimize certain measures related to a dataset, or a set of examples. Most classification–regression problems can be included in this group. The objective is the minimization of errors between an individual's output and that stored in a training set [9]. In these cases, the training set can be divided into smaller sets of examples, thus reducing the computational cost of evaluating large amounts of data to calculate the fitness of a single individual.
– Optimization problems in which the objective function must be calculated multiple times to test as many situations as possible. Control or decision optimizations can be included here. These problems are usually tested under different conditions to verify their robustness [37]. In these situations, the fitness function can be decomposed into the result for the controller in each possible situation.

In general, the problem must be suitable for partitioning into a set of small and correlated pieces. This means that an individual who can solve a small part of the problem is more likely to solve the remainder. For example, a classifier that has zero error over 10 % of examples is more likely to properly classify the other 90 % than another classifier that is not so accurate. To be more exact, the main idea lies in using a different set of problems to evaluate individuals at each generation. This forces *changes* to the fitness function at each generation, because the fitness of an individual changes with the set of problems to be evaluated.

This work seeks to derive a base fitness that summarizes the performance of individuals in different cases (even those in which individuals have not been tested). To achieve this goal, the fitness function is modified to estimate the performance of the individual in both the previous and current generations. Finally, offspring inherit the fitness function from their parents, thus preserving information about the parents' performance.

The objectives of the GABF are: to reduce the execution time by evaluating individuals in small portions of the problem for each generation, and to use a fitness function that summarizes the performance of individuals in all cases. These objectives are satisfied by implementing the following procedures:

- *Cases Selection*: this procedure picks the subset of cases that are used to calculate the fitness function in the current generation. This step is performed prior to the evaluation of the current population.
- *Fitness Aggregation*: this procedure updates the fitness of an individual for the current cases with its actual (inherited) value. This takes place once the fitness of the individuals under the current test cases has been calculated.
- *Fitness Inheritance*: this procedure assigns the offspring a deteriorated fitness from their parents. This is implemented after the crossover and mutation step.

Figure 1 shows how the proposed procedures are included in the classical flowchart of a generational GA [19]. Note that the GABF almost retains the flow of the GA, but differs in the way in which the fitness value of each individual is calculated and treated. For this reason, the methodologies proposed in this work can easily be extended to other optimization techniques. The only modification is that, because cases vary from one generation to the next, the best individual must be re-evaluated to determine a new fitness value.

The following sections propose alternatives for each procedure involved in the maintenance of this *base fitness*. Sections 2.1, 2.2 and 2.3 present alternatives to the *Cases Selection*, *Fitness Aggregation* and *Fitness Inheritance* methods, respectively.

2.1 Cases selection

Given a problem that can be divided and sub-optimized, this method aims to select N_c sub-problems from which to evaluate the accuracy of the entire set of individuals. For example, in a classification problem, the function would be in charge of selecting N_c examples to measure the classification error of the individuals, whereas for control systems, it would

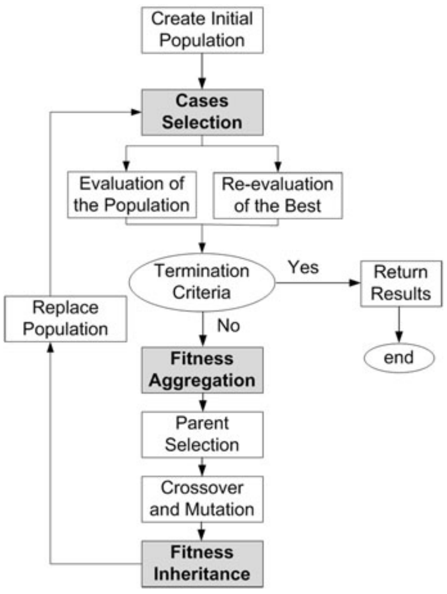


Fig. 1 Flowchart of the proposed GABF

be in charge of determining the situations in which the controller responses must be evaluated. This selection can be performed in different ways, for example:

- The simplest way to implement *Cases Selection* is to randomly select N_c cases from which to evaluate individuals.
- Cases can also be selected in accordance with their (real or expected) difficulty, using easy cases in early generations and hard ones in later stages of the GA.
- Another alternative is to maintain a register of cases and fitness values obtained for individuals, and to select cases according to the difficulty individuals experience in overcoming them.
- In a more problem-oriented way, cases can be selected depending on the characteristics of the problem.
- Finally, to enhance the precision of the GA, the number of cases used can be varied for each generation. In this way, N_c becomes a function of t.

2.2 Fitness aggregation

This procedure redefines the fitness value of an individual to be an aggregation of the fitness obtained in current and previous cases, that is, as the *base fitness*. Once the individuals in the current generation have been evaluated, the information from this evaluation should be merged with the stored values (inherited from the parents). In this way, two different alternatives can be proposed. The simplest is a weighted average of the current and stored fitness, as presented in Eq. (1).

$$f(t) = \omega \cdot f(t) + (1 - \omega) \cdot f(t-1) \quad (1)$$

Here $f(t)$ refers to the fitness of the individual in the current generation, and $f(t-1)$ represents the stored fitness of the individual. The parameter ω weights the importance of the current fitness value over the inherited value. The main advantage of this approach is that there is no need for additional structures beyond those required to implement a GA.

The alternative involves using a specific *memory* to maintain the fitness obtained in a certain number of previous generations. This memory allows more complex aggregations to be implemented by using the fitness obtained in scenarios from more than one generation ago, as stated in Eq. (2).

$$f(t) = \omega_0 \cdot f(t) + \omega_1 \cdot f(t-1) + \cdots + \omega_k \cdot f(t-k) \quad (2)$$

2.3 Fitness inheritance

This procedure maintains the base fitness in the offspring following the crossover and mutation operations, thus preserving the accumulated fitness of the parents in the children. To achieve this, the following possibilities are suggested:

– Fitness is inherited directly from parents; where each offspring receives the fitness value from one parent.
– Fitness is inherited from the most similar of the child's parents.
– Fitness is inherited from the most similar individual in the whole mating pool.

Other possibilities could be designed depending on the specific features of the problem or technique in which the *base fitness* operators are added. After an offspring inherits a fitness value, it must be degraded to counteract modifications with respect to the individual from whom the fitness has been inherited. The simplest way to achieve this is by multiplying the received fitness value by a factor $\rho > 1$ (for minimization problems).

3 Problem statement

The proposed scenario involves an unregulated intersection being approached by two vehicles at the same time. One vehicle is driven by an human, whereas the other is operating under autonomous speed management. The human-driven vehicle has no obligation to cooperate with the autonomous one and is supposed to not alter its speed or execute any turn. So the latter must independently adapt its speed to complete its maneuver quickly and without collision risk.

To adapt its speed, the autonomous vehicle implements a FRBS that indicates an appropriate speed. As inputs, the FRBS takes both vehicle speeds ($S_{\{A|M\}}$) and their distances to the exit of the *crossing area* ($D_{\{A|M\}}$). A schematic view of the scenario is shown in Fig. 2, where the *crossing area* represents the zone in which the vehicles must be prevented from colliding. The aim of the FRBS is to provide reference speeds that the autonomous vehicle should follow.

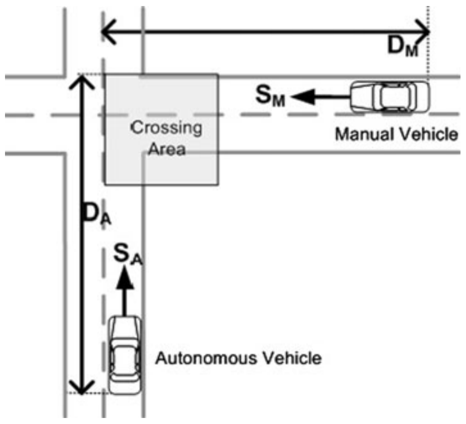


Fig. 2 Proposed scenario: an autonomous vehicle approaches an intersection at the same time as a human-driven vehicle

The FRBS must ensure that the autonomous vehicle completes its maneuver without colliding with the human-driven vehicle, and with a minimum change in speed.

Because the FRBS must be capable of dealing with any plausible situation (represented as a four element tuple $\{D_A, S_A, D_M, S_M\}$), assuming the human-driven vehicle does not alter its speed or execute any turn during the maneuver, a large initial configuration space $\{D_A, S_A, D_M, S_M\}_{t=0}$ should be defined. Using the proposed GABF, individuals from only a small number of initial configurations need be evaluated in each generation.

3.1 Longitudinal behavior of the autonomous vehicle

Under the management of the FRBS, the speed of the autonomous vehicle evolves according to Eq. (3). This equation models the longitudinal behavior of an autonomous vehicle (for slowly varying dynamics and on a flat surface) when a proportional-integral controller is in charge of tracking the given reference speed. Tejado et al. [36] describes this model in detail. The model has previously been used to simulate the speeds of an autonomous vehicle under FBRS management [26].

$$\begin{aligned} S_A(t) = {} & a_2 S_R(t-1) + a_1 S_R(t-2) \\ & + a_0 S_R(t-3) - b_2 S_A(t-1) \\ & - b_1 S_A(t-2) - b_0 S_A(t-3) \end{aligned} \quad (3)$$

Here $S_R(k)$ and $S_A(k)$ are the reference and actual velocities at instant k, with $a_0 = -5.467 \cdot 10^{-5}$, $a_1 = -0.2041$, $a_2 = 0.2495$, and $b_0 = 0$, $b_1 = 0.7421$, $b_2 = -1.697$. Typical changes in speed for two initial speed settings are illustrated in Fig. 3.

3.2 FRBS codification

The proposed FRBS uses trapezoidal membership functions to codify the input variables and singletons, which are punc-

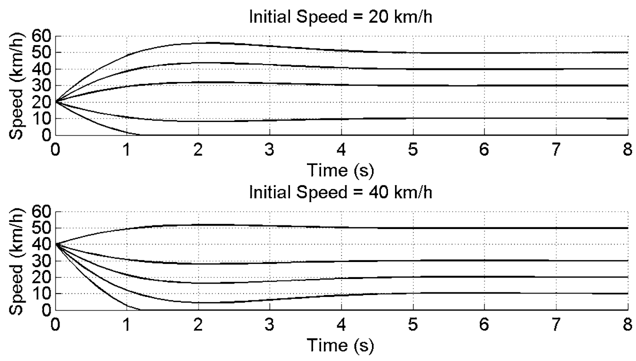


Fig. 3 Evolution of speed for an autonomous vehicle starting with initial speeds {20, 40} km/h to final reference speeds {0, 10, 20, 30, 40, 50} km/h

tual values, to codify the output variable. The FRBS results in a zero-order Takagi-Sugeno-Kang (TSK) fuzzy controller [35]. The use of trapezoids and singletons for codifying the input and output variables guarantees that execution of the control stage will be fast, because only basic operations are needed to infer the output value. This feature is sufficient for implementing real-time systems such as those involved in managing vehicles [25, 34].

Each of the four input variables (D_A, S_A, D_M, S_M) passed to the FRBS is fuzzified by means of three Membership Functions (MFs), designated as $\{Low, Medium, High\}$. The representation to be optimized by the GABF is given as three real values (x_1, x_2, x_3), each of which is within the range $[0, 50]$. Values are given in m for distance-related quantities (D_A and D_M), and in km/h for speed-related quantities (S_A and S_M). The first value represents the point from which the *Low* MF is extended; the second value sets the vertex of the *Medium* MF, and the third marks the start of the *High* MF, as illustrated in Fig. 4. It is important to note that the restriction $(x_1 \le x_2 \le x_3)$ must hold after any modification of these values.

The FRBS will contain a rule for each of the possible AND-combinations of the antecedents. Thus, the rule base is composed of 81 $(3 \times 3 \times 3 \times 3)$ rules in the form presented in Eq. (4).

$$\text{IF } (D_A \text{ is } \widehat{D_A}) \text{ AND } (S_A \text{ is } \widehat{S_A}) \text{ AND}$$
$$(D_M \text{ is } \widehat{D_M}) \text{ AND } (S_M \text{ is } \widehat{S_M}) \text{ THEN } \left(S_R = S_R^i\right) \qquad (4)$$

Here $[\widehat{D_A}, \widehat{S_A}, \widehat{D_M}, \widehat{S_M}]$ represents the MF used in the i-th rule (*Low*, *Medium*, or *High*), $[S_R^i] \in [0, 50]$ represents the consequent assigned to the i-th rule, and $i \in \{1, 2, \ldots, 81\}$ denotes the index of the rule. Hence, all possible combinations of input values are covered by at least one rule. Because the whole rule base is represented in each chromosome, the codification follows a Pittsburgh approach for the GFS [33].

Thus, the FRBS codification to be optimized by GABF consists of 93 real values. The first 12 define the distribution of the MF for each of the four inputs (three per input), and the remaining 81 define the positions of the singletons used as consequents in the rules. These values are normalized to provide a representation in the interval [0, 1].

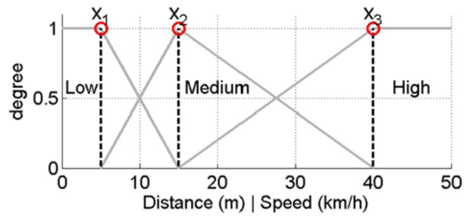


Fig. 4 Encoding of three membership functions by three real values used

3.3 Fitness function for evaluating frbs behavior

Recall that any case used to evaluate individuals is defined by means of state variables in the initial configuration of the scenario. This section presents the method of evaluating the performance of a FRBS for a set of cases. Section 4.1 explains how these cases are generated. Given a certain initial setting $\{D_A, S_A, D_M, S_M\}_{t=0}$, the crossing scenario is simulated twice:

1. Without considering output values coming from the FRBS (free execution, or E_f), where the autonomous vehicle maintains a constant speed.
2. Considering the indications of the FRBS (controlled execution, or E_c), where the vehicle adapts its speed to the given reference.

In this way, E_f enables the process to determine whether the given scenario leads to a collision, whereas E_c allows the process to consider whether (and how) the FRBS avoids that situation. Each execution of the FRBS (E_f, E_c) can report one of three different results, depending on the circumstances of each vehicle's entry to the *crossing area* (defined as 5 m before the exit line of the intersection):

1. The vehicles do not collide in the crossing area (there is no collision, C_0). In this case, the FRBS is expected to instruct the vehicle to maintain its speed.
2. A *lateral* collision occurs, that is, the vehicles collide in the crossing area, but the autonomous vehicle has entered before the other (C_L). Here, the FRBS is expected to slightly increase the speed of the autonomous vehicle so as to exit the crossing area before the human-driven vehicle enters.
3. A *frontal* collision occurs, that is, the vehicles collide in the crossing area, but the human-driven vehicle has entered before the other (C_F). The FRBS is expected to slightly decrease the speed of the autonomous vehicle in this situation, thus entering the crossing area once the human-driven vehicle has exited.

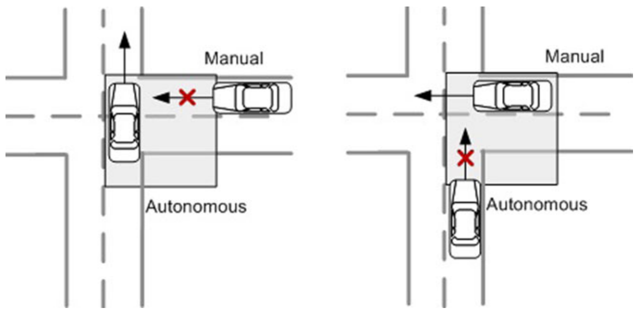


Fig. 5 Situations leading to lateral (*left*) and frontal (*right*) collisions

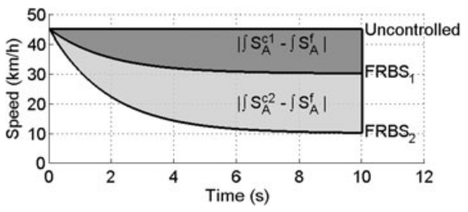


Fig. 6 Example execution of two FRBSs: the speed is reduced less by $FRBS_1$ than by $FRBS_2$, hence receiving a better (lower) partial fitness

Scenarios that produce collisions (lateral or frontal) are illustrated in Fig. 5. The evaluation of each FRBS in a particular scenario yields a partial fitness value for the solution ($F(FRBS, Case)$). All partial fitness values are averaged to obtain a final value. In other words, given N_c cases, the fitness function for a certain FRBS is calculated as presented in Eq. (5).

$$fitness(FRBS) = \frac{1}{N_c} \sum_{i=1,\ldots N_c} F(FRBS, Case_i) \tag{5}$$

Partial fitness is calculated according to the nine possible combinations derived from the results of the controlled and uncontrolled simulations ($\{E_f, E_c\} \in \{C_0, C_F, C_L\}^2$). Calculation of the partial fitness is based on the Average Absolute Speed Difference ($AASD$), which is calculated as shown in Eq. (6).

$$AASD = \frac{|\int S_A^c - \int S_A^f|}{50} \tag{6}$$

Where S_A^c and S_A^f represent the speed of the controlled/uncontrolled autonomous vehicle, respectively. Given two FRBSs that produce the same result (for example, not to collide), this formula allows the process to penalize the system that alters the original speed of the autonomous vehicle more, as illustrated in Fig. 6. Because $AASD$ is normalized to the maximum permitted speed (50 km/h), $AASD \in [0, 1]$, which is useful when penalizing undesired behavior. Therefore, the partial fitness function $F(FRBS, Case)$ is calculated in accordance with Table 1, where cases can be grouped in the following way:

– Cases 1 to 3 represent situations where the FRBS has avoided a collision by behaving as desired. Specifically, no collision occurs, or the collision is avoided by increasing (when lateral) or decreasing (when frontal) the speed of the autonomous vehicle.
– Cases 4 and 5 represent situations in which the FRBS avoids a collision that occurs in the uncontrolled simulation, but does so through an incorrect response. That is, decreasing the speed in the case of lateral collision and increasing the speed in the case of a frontal collision. A penalty is added in this case.
– Case 6 penalizes the FRBS more than before, because the FRBS is not able to avoid a collision that occurred in the uncontrolled simulation.
– Case 7 gives the maximum penalty, as the collision was caused by the FRBS management. That is, there was no collision in the uncontrolled simulation, but a collision occurred in the controlled situation.

4 Experiments and results

This section presents the results obtained by applying some of the concepts already presented in Sect. 2 and the decision problem stated in Sect. 3. The results are compared with those obtained by two implementations of a classical GA and one co-evolutionary GA [16, 28]. For both GA and GABF, similar operators and parameters were used, so the only difference between them is the addition of the proposed *base fitness*. This highlights the advantages in using such techniques in GABF. This section is organized as follows: Sect. 4.1 describes the experimental setup, and Sect. 4.2 presents the results obtained by the different methods.

4.1 Experimental setup

All the methods considered in this experiment have the following common properties:

– As explained in Sect. 3.2, each chromosome is codified as a string of 93 real values in the interval [0, 1], where the first 12 define the MF distribution and the remaining 81 define the consequences of the rules. The fitness functions used were explained in Sect. 3.3.
– *Selection* is performed using a binary tournament [2, 15], whereby two individuals are randomly chosen, their fitness values are compared, and the individual with the best value is selected as the parent.
– *Crossover* is carried out using the *BLX-α* operator [8] that, given two genes from parents p_1 and p_2, randomly generates new offspring in the interval $[\min(p_1, p_2) - \alpha \cdot |p_1 - p_2|, \max(p_1, p_2) + \alpha \cdot |p_1 - p_2|]$. In this work, a value of $\alpha = 0.5$ is used.

Table 1 Partial fitness calculated according to results from both controlled and uncontrolled simulations

Case	Description	Conditions	$F(FRBS, Case)$
1	FRBS neither avoids nor produces a collision	$(E_f = C_0)\&(E_c = C_0)$	$AASD$
2	FRBS avoids a lateral collision by increasing speed	$(E_f = C_L)\&(E_c = C_0)\&(\overline{S_A^f} < \overline{S_A^c})$	$AASD$
3	FRBS avoids a frontal collision by decreasing speed	$(E_f = C_F)\&(E_c = C_0)\&(\overline{S_A^f} > \overline{S_A^c})$	$AASD$
4	FRBS avoids a lateral collision by decreasing the speed	$(E_f = C_L)\&(E_c = C_0)\&(\overline{S_A^f} > \overline{S_A^c})$	$1 + AASD$
5	FRBS avoids a frontal collision by increasing speed	$(E_f = C_F)\&(E_c = C_0)\&(\overline{S_A^f} < \overline{S_A^c})$	$1 + AASD$
6	FRBS is not able to avoid a collision	$(E_f = C_{\{F\|L\}})\&(E_c = C_{\{F\|L\}})$	$2 + AASD$
7	FRBS produces a collision	$(E_f = C_0)\&(E_c = C_{\{F\|L\}})$	$3 + AASD$

– *Mutation* is performed using the random resetting method: for each gene, the current value is changed by a random value in the permissible interval with probability $P_m = 2/93$.

Both the GA and the GABF follow a generational scheme [19] whereby, at each generation, an entire new population is generated to replace the previous generation. Three versions of the GA are implemented. The first, called GA_{static}, generates cases used to evaluate individuals at the beginning of the process, and these cases do not change during the process. In the second ($GA_{dynamic}$), new random cases are generated in each iteration of the process. The third follows a competitive co-evolutionary approach (GA_{coevol}) in which two populations are maintained. One population is composed according to the FRBS, and the other depends on a series of cases. Cases are selected, crossed and mutated with the same operators used by the FRBS, but with the objective of maximizing the penalties received by the FRBS. This can be expressed as presented in Eq. (7).

$$fitness(Case) = \frac{1}{N_{FRBS}} \sum_{i=1,\dots,N_{FRBS}} F(FRBS_i, Case) \tag{7}$$

Apart from the common operators, GABF uses its own methods, which are implemented as follows:

– Cases Selection (Sect. 2.1): In this work, the scenarios used to evaluate individuals are randomly selected. However, given it is far more probable (about 75 %) that a generated scenario does not lead to a collision in the uncontrolled simulation ($E_f = C_0$), the selection of cases is restricted to scenarios that evolve to the three possible results ($E_f \in \{C_0, C_L, C_F\}$) with the same probability.

– Fitness Aggregation (Sect. 2.2): For this problem, Eq. (1) with $\omega = 0.25$ is used. This value was chosen after testing different configurations, as it was observed to lead to more regular results because the dependence on the current generation is smaller than that on the previous *base fitness*.

• Fitness Inheritance (Sect. 2.3): After crossover and mutation, two versions of fitness inheritance are tested. In the first, individuals inherit a fitness value directly from their parents ($GABF_{direct}$), whereas in the second, they inherit fitness from the parent with the more similar fitness ($GABF_{similar}$). Fitness is degraded according to the factor $\rho = 1.1$.

Each method has a population size of 50 individuals, and evolves until the maximum number of 10000 *evaluations*. An evaluation is considered to be the simulation of one intersection scenario, rather than the evaluation of an individual (which implies more than one case). To obtain statistically comparable results, each test was repeated 20 times. The results are compared in terms of their average and standard deviation over the 20 repetitions.

The number of cases used by each method to evaluate individuals (N_c) was set to one of $\{3, 6, 12, 24, 48\}$. Thus, considering that GABF must *re-evaluate* the best individual in each generation, the total number of generations to be run by each method is 67, 34, 17, 9 and 5, respectively. To ensure comparable results, the cases used by the GAs were also selected such that $E_f \in \{C_0, C_L, C_F\}$ had the same probability.

4.2 Experimental results

Because different scenarios were used for each generation carried out by the GABF technique (and $GA_{\{dynamic|coevol\}}$), each technique used the same 2000 intersection scenarios as the test set. These were randomly generated by ensuring $E_f \in \{C_0, C_L, C_F\}$ had the same probability. In this way, all the obtained FRBSs were tested under the same scenarios, ensuring that their performances could be compared.

Table 2 shows the average proportion of maneuvers finished without collision for each of the executed tests. In addition, the standard deviation and best value are shown. For emphasis, average values above 0.80 and best values above 0.95 are boldfaced. From Table 2, it can be seen that the average GABF values are higher than those obtained by classic GAs. In particular, the difference is more evident for lower values of N_c. The results obtained by GA are around 0.60, whereas those from the GABF are closer to 0.75. Note

Table 2 Average proportion of collision free maneuvers over 2000 cases

N_c	3	6	12	24	48
GA_{static}	0.577	0.678	0.767	**0.812**	**0.844**
σ	0.081	0.113	0.111	0.120	0.083
Best	0.754	0.907	0.911	**0.991**	**0.966**
$GA_{dynamic}$	0.538	0.667	0.745	0.793	**0.872**
σ	0.121	0.122	0.094	0.108	0.080
Best	0.905	0.893	0.922	**0.981**	**0.999**
GA_{coevol}	0.623	0.619	0.678	**0.836**	**0.868**
σ	0.102	0.095	0.096	0.104	0.103
Best	0.844	0.781	0.937	**0.987**	**0.999**
$GABF_{direct}$	0.737	**0.802**	**0.835**	**0.880**	**0.895**
σ	0.082	0.100	0.099	0.091	0.086
Best	0.894	**0.981**	**0.999**	**1.000**	**0.985**
$GABF_{similar}$	0.751	0.783	**0.805**	**0.833**	**0.895**
σ	0.121	0.084	0.090	0.119	0.070
Best	0.939	0.907	**0.977**	**0.997**	**0.998**

also that GA obtains better results in the *static* case, with the exception of $N_c = \{24, 48\}$. With respect to the GABF methods, *direct* fitness inheritance gives slightly better results than *similar* fitness inheritance.

To determine whether the proposed method achieves significantly better results than the GAs, a Student's t-test is performed. The t-statistic was calculated according to Eq. (8). A confidence interval of 95 % is used, hence $t_{0.05} = 2.021$. Table 3 shows the results of the Student's t-test for all of the comparable techniques, where each cell (row, column) can contain three different values:

- Positive $+$ when $technique_{row}$ is significantly better than $technique_{column}$.
- Negative $-$ when $technique_{row}$ is significantly worse than $technique_{column}$.
- Empty when $technique_{row}$ is not significantly different to $technique_{column}$.
- In addition, items inside parentheses indicate that $technique_{row}$ and $technique_{column}$ share the same N_c value.

$$t = \frac{\overline{X_1} - \overline{X_2}}{\sqrt{\frac{(n_1-1)\sigma_1^2+(n_2-1)\sigma_2^2}{n_1+n_2-2} \cdot \frac{n_1+n_2}{n_1 n_2}}} \tag{8}$$

Table 3 is divided into *smaller* tables to enhance its readability. These are referred to as sub-table(row,column), with *row* and *column* being the numbers written before the name of the technique. For instance, sub-table(1,4) compares GA_{static} with $GABF_{direct}$.

In addition, in Table 4, techniques are ranked according to the number of positive and empty symbols appearing in Table 3. In the table, R_T represents the total ranking, calculated according to the number of positive values, and using the number of empty values to break deadlocks. R_P is a partial ranking that is calculated in the same manner, but considering only techniques with similar N_c.

Analyzing the results shown in Tables 3 and 4 in more depth yields the following principal conclusions:

- Each of the implemented techniques improves (or does not degrade) their results as the number of cases (N_c) increases. This can be elicited from the cells under the diagonal in Table 3, sub-tables (1, 1), (2, 2), (3, 3), (4, 4) and (5, 5), being filled with positive or empty values. This is logical, because the more cases are used to evaluate the individuals in the population, the more general will be the results. The resulting FRBS will then be able to deal satisfactorily with more situations. In addition, in Table 4, R_T can be seen to always decrease as N_c increases, for all techniques.
- GABF-based techniques are ranked higher than the GA-based methods (see Table 4). Three GABF-based methods are among the top five, whereas the lowest-ranked five are GA-based methods. In addition, GABF techniques attain $R_P = \{1, 2\}$, with the exception of GA_{coevol} for $N_c = 24$.
- No significant differences can be seen when comparing GA-related techniques (Table 3, sub-tables (2, 1), (3, 1) and (3, 2)); there are no significant differences for equal N_c, and, when unequal, the technique with higher N_c usually produces better results. Studying Table 4, it can be seen that GA_{static} has a higher partial ranking than $GA_{dynamic}$ in most instances, with the exception of $N_c = 48$. In contrast, GA_{coevol} obtains a better ranking than both GA_{static} and $GA_{dynamic}$ when $N_c = \{3, 24, 48\}$.
- When comparing $GABF_{direct}$ with GA_{static} (Table 3, sub-table (4, 1)), the proposed technique improves results from the classic one in most cases, provided N_c of the GABF is not greater than that of the GA. One exception is for $N_c = 48$, where there is no significant difference. In cases when the N_c used by GA_{static} is greater than that for $GABF_{direct}$, the latter results are not significantly different, with the exception of $GABF_{direct}$ with $N_c = 3$, for which the results are worse than for GA_{static} with $N_c = \{24, 48\}$.
- A very similar analysis can be found when comparing $GABF_{direct}$ with $GA_{dynamic}$ (Table 3, sub-table (4, 2)), with some particularities. In this case, $GABF_{direct}$ with $N_c = 3$ gets better results than $GA_{dynamic}$ with $N_c = \{3, 6\}$, and is only out-performed by $GA_{dynamic}$ with $N_c = \{48\}$.
- If $GABF_{direct}$ is compared with GA_{coevol} (Table 3, sub-table (4, 3)), all the $GABF_{direct}$ techniques outperform GA_{coevol} with $N_c = \{3, 6, 12\}$. GA_{coevol} with $N_c = 24$ only improves results of $GABF_{direct}$ with

Table 3 Results of the Student's t-test comparing performance between each technique. + when $technique_{row}$ is significantly better than $technique_{column}$, − when $technique_{row}$ is significantly worse than $technique_{column}$, and *empty* for no significant difference

Method		(1) GA_{static}					(2) $GA_{dynamic}$					(3) GA_{coevol}					(4) $GABF_{direct}$					(5) $GABF_{similar}$				
	N_c	3	6	12	24	48	3	6	12	24	48	3	6	12	24	48	3	6	12	24	48	3	6	12	24	48
(1) GA_{static}	3	( )	−	−	−	−	( )	−	−	−	−	( )		−	−	−	(−)	−	−	−	−	(−)	−	−	−	−
	6	+	( )	−	−	−	+	( )	−	−	−		( )		−	−		(−)	−	−	−	−	(−)	−	−	−
	12	+	+	( )		−	+	+	( )		−	+	+	(+)	−	−			(−)	−	−			( )		−
	24	+	+		( )		+	+	+	( )		+	+	+	( )		+			(−)	−				( )	−
	48	+	+	+		( )	+	+	+		( )	+	+	+		( )	+				( )	+	+			(−)
(2) $GA_{dynamic}$	3	( )	−	−	−	−	( )	−	−	−	−	(−)	−	−	−	−	(−)	−	−	−	−	(−)	−	−	−	−
	6	+	( )	−	−	−	+	( )	−	−	−		( )		−	−	−	(−)	−	−	−	−	(−)	−	−	−
	12	+	+	( )	−	−	+	+	( )		−	+	+	(+)	−	−			(−)	−	−			(−)	−	−
	24	+	+		( )		+	+		( )	−	+	+	+	( )	−				(−)	−				( )	−
	48	+	+	+		( )	+	+	+	+	( )	+	+	+		( )	+	+			( )	+	+	+		( )
3) GA_{coevol}	3	( )		−	−	−	(+)		−	−	−	( )			−	−	(−)	−	−	−	−	(−)	−	−	−	−
	6		( )	−	−	−	+	( )	−	−	−		( )		−	−	−	(−)	−	−	−	−	(−)	−	−	−
	12	+		(−)	−	−	+		(−)	−	−			( )	−	−	−	−	(−)	−	−	−	−	(−)	−	−
	24	+	+	+	( )		+	+	+	( )		+	+	+	( )		+			( )		+			( )	−
	48	+	+	+		( )	+	+	+	+	( )	+	+	+		( )	+	+			( )	+	+	+		( )
4) $GABF_{direct}$	3	(+)			−	−	(+)	+			−	(+)	+	+	−	−	( )	−	−	−	−	( )		−	−	−
	6	+	(+)				+	(+)			−	+	(+)	+		−	+	( )		−	−		( )			−
	12	+	+	(+)			+	+	(+)			+	+	(+)			+		( )		−	+		( )		−
	24	+	+	+	(+)		+	+	+	(+)		+	+	+	( )		+	+		( )		+	+	+	( )	
	48	+	+	+	+	( )	+	+	+	+	( )	+	+	+		( )	+	+	+		( )	+	+	+		( )
5) $GABF_{similar}$	3	(+)	+			−	(+)	+			−	(+)	+	+	−	−	( )		−	−	−	( )			−	−
	6	+	(+)			−	+	(+)			−	+	(+)	+		−		( )		−	−		( )			−
	12	+	+	( )			+	+	(+)		−	+	+	(+)		−	+		( )	−	−			( )		−
	24	+	+		( )		+	+	+	( )		+	+	+	( )		+			( )		+			( )	−
	48	+	+	+	+	(+)	+	+	+	+	( )	+	+	+	+	( )	+	+	+		( )	+	+	+	+	( )

$N_c = 3$, and GA_{coevol} with $N_c = 48$ gets better results than $GABF_{direct}$ with $N_c = \{3, 6\}$.

- When comparing $GABF_{similar}$ with GA_{static}, $GA_{dynamic}$ and GA_{coevol} (Table 3, sub-tables (5, 1), (5, 2) and (5, 3)), very similar results are found: the proposed method yields better or similar results in most cases, except for $N_c = 48$, where GA-based techniques obtain better results than GABF for low N_c.
- Finally, comparisons among the GABF-based techniques (Table 3, sub-table (5, 4)) show that slightly better results are obtained by $GABF_{direct}$. Indeed, this technique occupies three of the five top rankings in Table 4.

In general, results show that GABF-related techniques produce remarkable improvements over GA-based techniques. Even for small N_c, they obtain, in most cases, equal or better results than GA-based techniques using high N_c.

5 Conclusions and future work

In the present study, both exogenous and endogenous fitness estimations are combined in a single algorithm to solve a decision problem involving two vehicles at an intersection. The method presented is based on the well-known genetic algorithm. It adds three specific operators to allow the GA to estimate the fitness of an individual based on a small subset of problem cases. In addition, the proposed method allows the calculated fitness to be inherited and aggregated in subsequent generations.

To test the performance of the proposed method, the generation of FRBSs that guide an autonomous vehicle through an intersection being approached by a vehicle driven by an inattentive human is considered. In this case, the FRBS must determine and calculate the proper speed to navigate through the intersection without causing risk or delay.

Ultimately, the problem was stated as a real 93-dimensional optimization problem. Subsets of cases were used to

Table 4 Number of positive and empty values obtained from the t-test for each technique. The table presents the method used, the number of positive or empty symbols (from Table 3), and two rankings: a total ranking (R_T) and the partial ranking (R_P) based on methods with equal N_c

Method	N_c	'+'	'='	R_T	R_P
$GABF_{similar}$	48	20	5	1	1
$GABF_{direct}$	48	17	8	2	2
$GABF_{direct}$	24	16	9	3	1
GA_{coevol}	48	15	10	4	3
$GA_{dynamic}$	48	15	10	4	4
GA_{static}	48	12	12	5	5
GA_{coevol}	24	11	13	6	2
$GABF_{direct}$	12	11	12	7	1
$GABF_{similar}$	24	10	14	8	3
GA_{static}	24	9	13	9	4
$GABF_{similar}$	12	9	11	10	2
$GABF_{direct}$	6	8	12	11	1
$GA_{dynamic}$	24	7	13	12	5
$GABF_{similar}$	6	7	12	13	2
GA_{static}	12	7	10	14	3
$GABF_{similar}$	3	7	9	15	1
$GA_{dynamic}$	12	7	7	16	4
$GABF_{direct}$	3	6	7	17	2
GA_{static}	6	2	6	18	3
GA_{coevol}	12	2	5	19	5
$GA_{dynamic}$	6	2	5	19	4
GA_{coevol}	6	1	6	20	5
GA_{coevol}	3	1	6	21	3
GA_{static}	3	0	4	22	4
$GA_{dynamic}$	3	0	2	23	5

evaluate individuals, as it is computationally impractical to test each FRBS for all possible situations. Two different approaches of the proposed technique were compared with two GA implementations, and results demonstrated that the proposed method out-performs the GAs.

Future work will consider the use of different methods to inherit and aggregate fitness, and use different strategies to select which cases are used by the algorithm. In addition, the operators presented here will be implemented in different optimization techniques, such as ant colony systems and particle swarm optimization.

Acknowledgements The authors would like to thank the EU Intelligent Cooperative Sensing for Improved Traffic Efficiency (ICSI) project (FP7-ICT-2011-8) for its support in the development of this work.

References

1. Alcalá R, Gacto MJ, Herrera F (2011) A fast and scalable multi-objective genetic fuzzy system for linguistic fuzzy modeling in high-dimensional regression problems. IEEE Trans Fuzzy Syst 19(4):666–681
2. Brindle A (1981) Genetic algorithms for function optimization. PhD thesis, University of Alberta
3. Bui L, Abbass H, Essam D (2005) Fitness inheritance for noisy evolutionary multi-objective optimization. In: Proceedings of the conference on genetic and evolutionary computation, pp 779–785
4. Cano JR, Herrera F, Lozano M (2006) On the combination of evolutionary algorithms and stratified strategies for training set selection in data mining. Appl Soft Comput 6(3):323–332
5. Cano JR, Herrera F, Lozano M (2007) Evolutionary stratified training set selection for extracting classification rules with trade off precision-interpretability. Data Knowl Eng 60(1):90–108
6. Cantu-Paz E (2000) Efficient and accurate parallel genetic algorithms, vol 1. Kluger Academic, Boston
7. Cantú-Paz E, Goldberg D (1999) On the scalability of parallel genetic algorithms. Evol Comput 7(4):429–449
8. Eshelman L, Schaffer J (1993) Real coded genetic algorithms and interval schemata. Foundation of genetic algorithms, vol 2. Morgan Kaufmann, San Mateo
9. Ferri C, Hernández-Orallo J, Modroiu R (2009) An experimental comparison of performance measures for classification. Pattern Recognit Lett 30(1):27–38
10. Fonseca L, Lemonge A, Barbosa H (2012) A study on fitness inheritance for enhanced efficiency in real-coded genetic algorithms. In: IEEE congress on evolutionary computation, pp 1–8
11. Gacto MJ, Alcalá R, Herrera F (2012) A multi-objective evolutionary algorithm for an effective tuning of fuzzy logic controllers in heating, ventilating and air conditioning systems. Appl Intell 36(2):330–347

12. Ghosh Nee De S, Ghosh A, Pal S (2003) Incorporating ancestors' influence in genetic algorithms. Appl Intell 18(1):7–25
13. Goldberg D (1989) Genetic algorithms in optimization, search and machine learning. Addison Wesley, Reading, MA
14. Goldberg D (1998) The race, the hurdle, and the sweet spot: lessons from genetic algorithms for the automation of design innovation and creativity. Tech Rep 98007, University of Illinois at Urbana-Champaign
15. Goldberg D, Deb K (1991) A comparative analysis of selection schemes used in genetic algorithms. Morgan Kaufmann, San Mateo
16. Gu J, Gu M, Cao C, Gu X (2010) A novel competitive co-evolutionary quantum genetic algorithm for stochastic job shop scheduling problem. Comput Oper Res 37(5):927–937
17. Hart W, Krasnogor N, Smith J (2004) Recent advances in memetic algorithms. Studies in fuzzyness and soft computing series, vol 166. Springer, Berlin
18. Herrera F (2008) Genetic fuzzy systems: taxonomy, current research trends and prospects. Evol Intell 1:27–46
19. Holland J (1975) Adaptation in natural and artificial systems. The University of Michigan Press, Ann Arbor
20. Jin Y (2005) A comprehensive survey of fitness approximation in evolutionary computation. Soft Comput 9(1):3–12
21. Kaya M, Alhajj R (2006) Utilizing genetic algorithms to optimize membership functions for fuzzy weighted association rules mining. Appl Intell 24(1):7–15
22. Liu H, Motoda H (2002) On issues of instance selection. Data Min Knowl Discov 6(2):115–130
23. Luong H, Nguyen H, Ahn C (2012) Entropy-based efficiency enhancement techniques for evolutionary algorithms. Inf Sci 188:100–120
24. Moscato P, Cotta C, Mendes A (2004) Memetic algorithms. In: New optimization techniques in engineering. Studies in fuzziness and soft computing, vol 141. Springer, Berlin, pp 53–85
25. Onieva E, Naranjo J, Milanés V, Alonso J, García R, Pérez J (2011) Automatic lateral control for unmanned vehicles via genetic algorithms. Appl Soft Comput 11(1):1303–1309
26. Onieva E, Milanés V, Villagrá J, Pérez J, Godoy J (2012) Genetic optimization of a vehicle fuzzy decision system for intersections. Expert Syst Appl 39(18):13,148–13,157
27. Panoutsos G, Mahfouf M (2010) A neural-fuzzy modelling framework based on granular computing: concepts and applications. Fuzzy Sets Syst 161(21):2808–2830
28. Paredis J (2000) Coevolutionary algorithms. Evol Comput 2:224–238
29. Precup RE, Hellendoorn H (2011) A survey on industrial applications of fuzzy control. Comput Ind 62(3):213–226
30. Sastry K (2001) Evaluation-relaxation schemes for genetic and evolutionary algorithms. PhD thesis, University of Illinois at Urbana-Champaign
31. Sastry K, Goldberg DE (2004) Designing competent mutation operators via probabilistic model building of neighborhoods. In: Deb K (ed) Genetic and evolutionary computation—GECCO 2004. Lecture notes in computer science, vol 3103. Springer, Berlin, pp 114–125
32. Sastry K, Goldberg DE (2004) Let's get ready to rumble: crossover versus mutation head to head. In: Deb K (ed) Genetic and evolutionary computation—GECCO 2004. Lecture notes in computer science, vol 3103. Springer, Berlin, pp 126–137
33. Smith SF (1980) A learning system based on genetic adaptive algorithms. PhD thesis, University of Pittsburgh
34. Sugeno M (1999) On stability of fuzzy systems expressed by fuzzy rules with singleton consequents. IEEE Trans Fuzzy Syst 7(2):201–224
35. Takagi T, Sugeno M (1985) Fuzzy identification of systems and its applications to modeling and control. IEEE Trans Syst Man Cybern 15(1):116–132
36. Tejado I, Milanés V, Villagrá J, Godoy J, HosseinNia H, Vinagre B (2011) Low speed control of an autonomous vehicle by using a fractional PI controller. In: IFAC world congress, vol 18, pp 15025–15030
37. Yaochu J, Branke J (2005) Evolutionary optimization in uncertain environments—a survey. IEEE Trans Evol Comput 9(3):303–317
38. Zadeh L (1965) Fuzzy sets. Inf Control 8(3):338–353

E. Onieva received the B.E. degree in computer science engineering, the M.E. degree in soft computing and intelligent systems and the Ph.D. degree in computer science from the University of Granada, Spain, in 2006, 2008 and 2011, respectively. From 2007 to 2012, he has been with the AUTOPIA Program at the Centre of Automation and Robotics, Consejo Superior de Investigaciones Científicas, Madrid, Spain. During 2012 he has been with the Models of Decision and Optimization group, at the University of Granada. From the beginning of 2013, he is with the Mobility Unit at the Deusto Institute of Technology (DeustoTech), where he carries out cutting-edge research in the application of soft computing techniques to the field of intelligent transportation systems. His research record includes more than 25 publications in journals of the highest level and more than 25 publications in international conferences. He has been awarded about his research several times. His research interest is based on the application of Soft Computing Techniques to Intelligent Transportation Systems, including fuzzy-logic based decision and control and evolutionary optimization.

E. Osaba is a Ph.D. student at the Deustotech research center at the University of Deusto. He is Computer Engineer since 2010 by the University of Deusto, and he received in 2011 the title of Master Degree in Development and Integration of Software Solutions, having studied at the same university. His doctoral thesis is focused on artificial intelligence, specifically in the field of combinatorial optimization, studying and developing heuristics and meta-heuristics solving routing problems. Since 2010, he has participated in several research projects, and has also contributed to scientific production, publishing several papers in international conferences and journals.

X. Zhang received his Bachelor of Technology (Software Engineering) and Master degree (Computer Science) from South-Central University for Nationalities, China, in 2009 and 2011 respectively. Between July 2010 and Sep 2010, Xiao was with City University of HongKong as a research assistant. After graduated from university, he worked in IBM Corporation in China. Since Sep 2012, he joined DeustoTech Mobility research team, and currently he is working towards his Ph.D. in the field Intelligent Transportation System and Artificial Intelligence.

A. Perallos hold a Ph.D., M.Sc. and B.Sc. in Computer Engineering from the University of Deusto. Since 1999 he has been working as a lecturer in the Faculty of Engineering at the University of Deusto. His teaching focuses on software design and distributed systems, having taught several B.Sc., M.Sc. and Ph.D. courses. He is currently the Head of the Computer Engineering Department and Director of M.Sc. in Software Engineering at the University of Deusto. Apart from these academic activities, he worked in the IT department of the University of Deusto (from 2003 to 2004), as a project manager and functional analyst in software development projects. He is also Head Researcher at the DeustoTech Mobility Unit at the Deusto Institute of Technology (DeustoTech), where he coordinates the research activities of around 25 researchers. This research unit promotes the application of ICT to address smarter transport and mobility. In particular, Perallos' research background is focused on telematic systems, vehicular communication middleware and intelligent transportation systems. He has over a decade of experience in R&D management, with tens of projects and technology transfer actions led, more than 50 publications in the area of intelligent transport systems, more than a dozen of JCR indexed publications and one start-up company created.