Computing with Actions: The case of driving a car in a simulated car race

Sergio Guadarrama [*]
Fundamentals for Soft Computing Unit
European Centre for Soft Computing
Mieres, Spain
sergio.guadarrama@softcomputing.es

Abstract— *Describing and modeling a complex system is very hard and time consuming, so, we propose to use a linguistic approach introduced by Zadeh. By taking a linguistic approach we can use linguistic variables and approximate rules to characterize approximately phenomenons which are too complex or too ill-defined to be done in numerical terms.*

Starting from the results obtained in our previous works, in this paper we propose to decompose complex behaviors in simple behaviors, and context dependent rules for the behaviors coordination, which are described by set of fuzzy rules. The use of fuzzy rules for behavior coordination and behavior description has allowed us to have a shared framework that makes easy to develop and to compute with linguistic descriptions of actions.

We propose to use a mixture of competitive and cooperative co-evolutionary algorithms to manage the big search space and tackle a complex and decomposable problem.

We use the Fuzzy-IEEE 2007 Car Racing Competition as an example of a problem that allows to show the benefits of our approach. In the example presented the results obtained are better and the flexibility is increased with respect to previous works.

Keywords— Computing with Words, Computing with Actions, Fuzzy Linguistic Descriptions, Fuzzy Controllers, Co-evolutionary Algorithms, Behavior coordination.

1 Introduction

Real-life problems with solutions that are complex and unknown in advance require to build and test several prototypes before being able to build a good enough solution. Prototypes allow designers to devise what an appropriate solution might look like while they work at intermediate stages of development. In this way, designers require to test potential solutions, following a design-test-update cycle before reaching a good enough solution (as suggested in the Agile software development [1]). Computers are useful tools to develop and test these prototypes, which in turn serve designers to explore, build and test different ideas.

Describing and modeling a complex system is very hard and time consuming. So, we propose to use a linguistic approach introduced by Zadeh in [2, 3, 4, 5, 6] to deal with complex systems and complex concepts. By taking a linguistic approach we can use linguistic variables and approximate rules to characterize approximately phenomenons which are too complex or too ill-defined to be done in numerical terms [7, 8]. We use linguistic descriptions to describe the desired solution, problem decomposition techniques and fuzzy controllers to build the solution, and co-evolutionary algorithms to tune and improve the solutions.

Indeed, in this work we are introducing a tentative proposal to compute with actions by computing with linguistic descriptions of actions, paralleling Zadeh's Computational Theory of Perceptions [9], in which he proposed to compute with perceptions by means of computing with linguistic descriptions of perceptions.

To cope with these problems, we combine our previous ideas, improvements and results in the areas of computing with words [10], fuzzy controllers [11] and co-evolutionary algorithms [12].

The rest of the paper is organized as follows. In section 2, we provide details of the problem tackled, the FUZZ-IEEE 2007 Car Racing Competition. In section 3, the solution based on linguistic description of the behaviors is presented. The coordination between behaviors is presented in section 5. The mixture of co-evolutionary algorithms is presented in section 6. In section 7 we present the results and finally, we summarize our work and provide conclusions in section 8.

2 The problem: FUZZ-IEEE 2007 Car Racing Competition

Car Racing Competition is a well-known sport that gets attention from many people in a number of different scenarios, ranging from popular real-life Fomula One to virtual computer games. Particularly, we tackle the challenge of building a car pilot for the FUZZ-IEEE 2007 Car Racing Competition.

Driving is a complex problem and how to find a general solution is unknown. Everyone need a lot practice before becoming a good driver. Actually, people become driving experts by practicing in many different scenarios. So we will use the simulator provided by the FUZZ-IEEE 2007 Car Competition to simulate different races that will allow us to learn and test solutions in many different scenarios.

In this problem, the challenge was to develop the best fuzzy car driver controller for a basic form of car race. A competition in which a pair of simulated cars race through a series of waypoints within a 2d plane, and compete between them to reach the maximum number of waypoints. Basic aspects of the car race are the following:

1. A car only sees the next three waypoints at a given time.
2. Waypoints have to be visited in order and the next one to be visited is highlighted.
3. Waypoints are randomly generated on a 2d plane.
4. The car that first reaches the next waypoint scores.
5. Cars cannot collide with each other, so two cars can follow the same path at the same time.

[*]His work has been supported by the Spanish Department of Science and Innovation (MICINN) under program Juan de la Cierva JCI-2008-3531, and the European Social Fund.

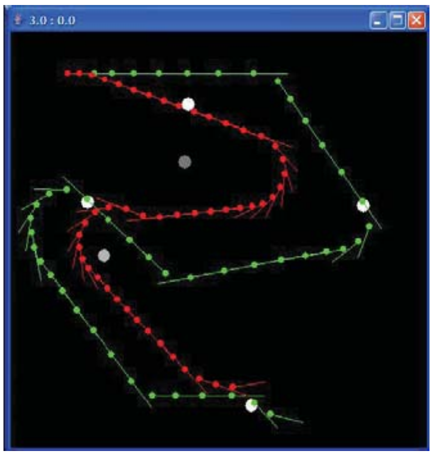

Figure 1: Car Racing Simulation on a 2d Plane.

6. Cars have imperfect sensors and actuators: gaussian noise is used to simulate the lack of perfect observations on position and velocity.

7. General driving ability rather than specialization to a particular track it is tested.

3 The solution: Behaviors described by linguistic actions

We describe a desired behavior by describing linguistically a set of actions modeled by fuzzy rules. So each behavior will be described by a set of fuzzy rules, and a solution will be composed by one or more behaviors coordinated appropriately.

Fuzzy rules are linguistically interpretable units that can serve our purposes of designing and testing prototypes easily and quickly, since they are easily human-interpretable, as well as, adjustable.

We have used fuzzy controllers [13, 14, 15] to implement individual behavior units. Fuzzy controllers incorporate heuristic control knowledge in the form of if-then rules, whose membership functions are later tuned by the co-evolutionary algorithm, similarly as it was proposed by Karr in [16].

We have used Xfuzzy 3.0, a fuzzy system development environment, to design our fuzzy inference system [17]. This system contains a set of tools that allows not only to easily describe a fuzzy inference model but also to implement it, and export it to programming languages such as JAVA, C and C++.

In this section we present the three different behaviors that we introduced in our previous paper [12]. The first behavior "Go straight to next target" was a local solution in the sense that the other targets were not considered (the other two way-points). The second behavior "Follow a spline curve" takes into account the next three way-points, thus following a more global approach. The third behavior "Avoiding ellipses around targets" cope with unwanted situation of going around a target indefinitely.

3.1 Behavior 1: Go straight to the target

Basically, this fuzzy controller takes as input variables: velocity, distance to next target and angle to next target; and has as output variables: acceleration and turn angle. The final controller is composed of two rule-bases:

- Acceleration rules:
  If Distance is Large then Acceleration is Very Positive.
  If Distance is Medium then Acceleration is Positive.
  If Distance is Small and Velocity is Very Positive then Acceleration is Very Negative.
  If Distance is Small and Velocity is Positive then Acceleration is Negative.
  If Distance is not Small and Velocity is Zero then Accelerations is Positive.

- Turn rules:
  If Angle is Zero then Turn Zero
  If Angle is Positive then Turn Positive
  If Angle is Negative then Turn Negative
  If Angle is not Zero and Distance is Very Small then Turn Zero

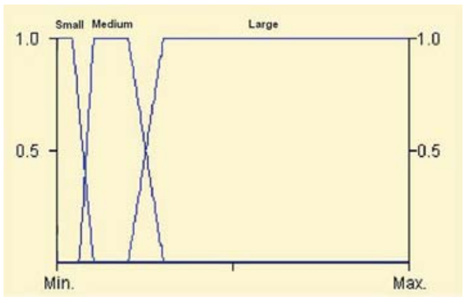

Figure 2: Distance Linguistic Variable

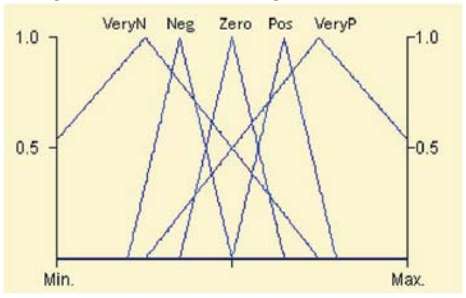

Figure 3: Acceleration Linguistic Variable

We impose the following ordering constrains [18] between the linguistic labels of the linguistic variables to maintain the interpretability of the rules, although these don't impose that they should form a partition:

- Distance = $\{$ Small $\prec$ Medium $\prec$ Large $\}$
- Velocity = $\{$ Very Negative $\prec$ Negative $\prec$ Zero $\prec$ Positive $\prec$ Very Positive $\}$
- Acceleration = $\{$ Very Negative $\prec$ Negative $\prec$ Zero $\prec$ Positive $\prec$ Very Positive $\}$

After tuning parameters of the fuzzy membership functions using a genetic algorithm (see figures 2 and 3) of the fuzzy rules this behavior was able to defeat the heuristic controller in all races (see table 2 for comparison). So from now on we took it as the new baseline for comparison with the rest of the solutions.

3.2 Behavior 2: Follow a spline curve

We observed that the previous behavior was able to align with the first target (next target) quite well, but it did not consider the positions of second and third targets. In other words, direction of the car was perfect for the next target, but it was not exploiting the knowledge about the rest other targets.

To do that, we use a spline curve instead of following straight lines, in order to follow a global strategy and gain advantage over the opponent, as soon as possible. We built a spline curve that goes through four points on a 2d plane: the position of our car, and the next three targets. We use the spline curve to calculate/approximate intermediate points between our car and the next target, and then follow them using these fuzzy rules:

- Acceleration rules:
  If Distance is Large then Acceleration is Very Positive.
  If Distance is Medium then Acceleration is Positive.
  If Distance is not Small and Velocity is Zero then Accelerations is Positive.
- Turn rules:
  If Angle is Zero then Turn Zero
  If Angle is Positive then Turn Positive
  If Angle is Negative then Turn Negative

On the negative side of this approach we can say that the distance that the car is covering is higher than in our previous behavior, that is, without using a spline curve, so when we set up a competitive co-evolutionary strategy this behavior (B2) was always loosing, unable to defeat the previous one (B1) (see table 3 for comparison).

3.3 Behavior 3: Avoid ellipses around targets

We also noticed that some races were ended when both cars were unable to reach a target. This situation occurred when a new target appeared and both cars were: 1, too close to the new target; and 2, with such a high angle to the target that it could not be corrected enough, given the short distance and the velocity of the cars. As a result, cars spent their last moments turning around the target.

To avoid that, we add this behavior (B3), that detects the unwanted situation and takes a different action. In order to detect that unwanted situation, we keep track of what happened in last n iterations. Particularly, we consider three facts during the last n iterations: 1, the target remained the same; 2, the angle to target was around 80 degrees; 3, the distance to target was small.

To address the described scenario, when the unwanted situation is detected then this new behavior is activated during a number m of iterations. This new behavior was built following the same rules that *"Go straight to the target"* with one sole difference: it did not go towards the target but in the opposite direction (it has the same rules as behavior 1 but with opposite sign).

4 Tuning Fuzzy Membership Functions by Co-evolution

Developing a good prototype from scratch is very time consuming and difficult since it involves a good understanding of the problem to do a good design, and a good understanding of the solution to adjust all its parameters. We did this initially manually and then used genetic algorithms, and co-evolutionary algorithms.

Table 1: Summary of previous results

	HC	GA	$Comp_1$	$Coop_\alpha$
HC	-	15	0	0
GA	83	-	1	5
$Comp_1$	100	99	-	47
$Coop_\alpha$	100	81	47	-

To avoid complexity and exploit the features of co-evolutionary algorithms we defined different species with different attributes to optimize. Actually, we divided our population into three species. Species number 1 consisted of individuals of behavior 1 *"Go straight to target"*, species number 2 was formed by individuals belonging to behavior 2 *"Follow a spline curve"*, and species number 3 was formed by individuals of behavior 3 *"Avoid ellipses"*.

So, we built a population with three different species with 20 individuals each, which evolve during 1000 generations. These species only met when racing (to calculate the fitness functions of individuals), while for the rest (cross, mutation, selection of individuals) were kept apart.

In [12] we compared how the use of a competitive or a cooperative co-evolutionary algorithm affects to the final solution. Obtaining that both approaches can find very good solutions, once the appropriate crossover operator is chosen. Table 6 recall the summary of results.

1. *HC*, hand coded model;
2. *GA*, complete model including three pilots, automatically tuned using a genetic algorithm;
3. $Comp_1$, complete model including three pilots, automatically tuned using a competitive coevolutionary approach, with 1-point cross operator;
4. $Coop_\alpha$, complete model including three pilots, automatically tuned using a cooperative coevolutionary approach, with BLX-alpha cross operator.

5 Behavior coordination

The main two problems of behavior coordination are:

1. Arbitration: deciding which behavior should be activated in each situation. This can be decided a priori or dynamically by establishing variable priorities. The simplest solution is a switching scheme that selects one behavior and ignores the rest. More flexible arbitration can be obtained using fuzzy context rules, that allow to activate concurrently several behaviors with different degrees of activation.
2. Command fusion: If only one behavior is active at a time, then there is not need to do it, but if there are more than

one behavior active simultaneously, then it is needed to combine in some way the different commands generated by them. It is needed to fuse, or aggregate, properly these commands in order not to send a set of contradictory commands but to send a coherent fused command.

Context dependent blending is a flexible and general form of solving the problem of behavior coordination, since it solves both arbitration and command fusion by using a common scheme, fuzzy context rules and fuzzy aggregation. It was initially proposed by Ruspini in [19] and later expanded by Saffioti and himself in [20, 21].

5.1 *Context dependent behavior coordination*

In our previous work we defined a fix arbitration scheme to switch between behaviors. And compared competitive and cooperative co-evolutionary algorithms to tune the membership functions of the linguistic variables used in the fuzzy rules.

In this work we use context dependent blending, using fuzzy rules for both behavior coordination and behavior description, since that allowed us to have a shared framework that made easier to develop complex behaviors.

To address the coordination problem, we defined context rules for each behavior, so they could be activated in different situations, and serve to merge their outputs when are activated at the same time.

- Context Rule 1:
  If Distance is Small then “Go Straight to the target”.
- Context Rule 2:
  If Distance is Large then “Follow a spline curve”.
- Context Rule 3:
  If Distance is Small and Angle is approx. 80^o then “Avoid ellipses”.

This allows to follow a local strategy when the car is near to the next target, a global strategy when the car is far to the next target, and avoiding ellipses around targets

We set up a mixture of competitive and cooperative co-evolutionary strategy, the individual behaviors cooperate to form the complex behavior (B123) while competing with the best solution obtained previously (B1). (see table 5 for comparison).

In figure 4 we can see a diagram of how the coordination of the different behaviors works. Using a weighted fusion between the commands of behaviors 1 and 2 and a switching scheme with behavior 3.

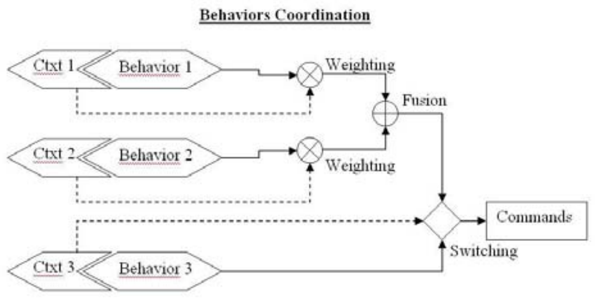

Figure 4: Behaviors Coordination

6 Mixture of Co-evolutionary algorithms

In some way, approaches pointing in the same direction that ours have already been proposed. In fact, many papers on fuzzy systems tuned with genetic algorithms have been published (for a good review see [22]).

Genetic algorithms, of single-population algorithms, are not well suited to problems which have solutions that are long, complex and interdependent [23]. In the last years, co-evolutionary algorithms have shown their ability to manage big search spaces and to tackle complex problems. Specially when the solution to the problem may be decomposable, or when there exist different competitive solutions [24, 25].

Co-evolutionary algorithms can be seen as an extension of traditional evolutionary algorithms, where there are, at least, two species that are simultaneously evolved and their fitness functions are coupled [24, 25]. Therefore, each species has its own coding scheme and reproduction scheme, but each individual is evaluated considering individuals from other species. Depending on the type of interaction between species, we can distinguish between competitive or cooperative species.

6.1 *Competitive Co-evolution*

In this context, each individual represents a candidate solution to the problem and competes with individuals of other species to find better solutions. Thus, if one individual increases its fitness, then fitness functions from the rest individuals decrease [24]. This may be seen as a kind of struggle between species, which presses individuals to compete between them for finding better solutions.

6.2 *Cooperative Co-evolution*

In this case, each individual represents a component of the solution and requires the rest components to build a complete solution. Therefore, an individual needs to cooperate with others to find a better solutions [25]. Thus, an individual fitness is linked with fitness values of its cooperative individuals. This is often seen as a kind of symbiosis between species to preserve existence, leading individuals to cooperate for finding better solutions.

6.3 *Mixture of Competitive and Cooperative Co-evolution*

We have different solutions that compete or cooperate to obtain better solutions. Following the two co-evolutionary approaches we have defined a mixed environment, where alternative solutions were competing to obtain better solutions, and where complementary solutions were cooperating to obtain better solutions. At the same time that the three different behaviors evolve in a cooperative way, they are competing with the best previous solutions.

The main features of the co-evolutionary algorithm used to tune the parameters of the membership functions are the following:

generations = 1000.
population size = 20.
crossover probability = 0.9.
mutation probability = 0.1.
crossover operator = BLX-α.
$\alpha = 0.7$.
species = 3.
co-evolutionary strategy = mixed.

ISBN: 978-989-95079-6-8

Each specie have its own coding as follows:

- Go Straight to Target (51 genes) + Context 1 (2 genes)
- Following a Spline Curve (39 genes) + Context 2 (2 genes)
- Avoiding Ellipses (51 genes) + Context 3 (5 genes)

7 Results

In this section it is shown the sum of the results we obtained during the development of the system and a comparative evaluation of the obtained solutions (for previous results see [12]).

Table 2:
Heuristic vs Behavior 1

Race	HC	B1
1	18	37
2	15	35
3	12	42
4	16	44
5	11	43
6	19	41
7	13	46
8	12	42
9	20	44
10	17	39
Total	153	413

Table 3:
Behavior 1 vs Behavior 2

Race	B1	B2
1	47	4
2	44	2
3	44	5
4	43	7
5	42	6
6	44	3
7	43	9
8	42	8
9	42	5
10	44	5
Total	435	54

Table 2 shows the number of way-points (targets) got in 10 races by the hand coded (HC) against the behavior 1 (B1) “Go straight to the next target”, in which we can see that the behavior 1 is much better than the hand coded controller, and it is wining all the races by a big difference.

Table 3 shows the number of way-points (targets) got in 10 races by the behavior 1 (B1) “Go straight to the next target” against the behavior 2 (B2) “Follow the spline curve”, in which we can see that behavior 2 is performing very bad against behavior 1, loosing all the races by a huge difference of points, and performing even worse than the hand coded controller.

These results indicate that the best individual behavior is “Go straight to the next target”, and that to improve these results we need to combine it with other behaviors, as we did in the next experiments.

Table 4 shows the number of way-points (targets) got in 10 races by the behavior 1 (B1) “Go straight to the next target” against the combined behavior 1 and 2 (B12) “Go straight to the next target; Follow the spline curve”, in which we can see that the combined behavior 12 is much better that the behavior 1 alone, and it is wining all the races by some difference. Table 5 shows the number of way-points (targets) got in 10 races by the behavior 1 (B1) “Go straight to the next target” against the combined behavior 1, 2 and 3 (B123) “Go straight to the next target; Follow the spline curve; Avoid ellipses”, in which

Table 4:
Behavior 1 vs Behavior 12

Race	B1	B12
1	26	35
2	27	37
3	25	32
4	31	32
5	24	38
6	32	31
7	28	36
8	26	36
9	28	35
10	30	40
Total	277	352

Table 5:
Behavior 1 vs Behavior 123

Race	B1	B123
1	19	35
2	18	45
3	13	43
4	22	33
5	18	32
6	20	34
7	16	37
8	26	28
9	24	32
10	20	36
Total	196	355

we can see that behavior 123 is performing very well against behavior 1 wining all the races by a big difference of points, and performing even better than the combined behavior 1 and 2 (B12).

These results indicate that the best behavior is the combined behavior 123. But after these comparisons and to evaluate the impact of the context blending behavior coordination, we compared it with the best solutions from our previous work $Comp_1$ and $Coop_\alpha$.

We set up a league competition among all of them, by playing 100 races between each pair of solutions. Table 6 shows the results of the five solutions considered: *Hand-coded*, $Comp_1$, $Coop_\alpha$ and $B123$, and their associated wins for each one in the 100 races between each pair of solutions.

Table 6: Wins Against Each Other

	HC	$Comp_1$	$Coop_\alpha$	$B123$	Total
HC	-	0	0	0	0
$Comp_1$	100	-	47	20	167
$Coop_\alpha$	100	51	-	31	182
$B123$	**100**	**74**	**63**	–	**237**

Table 6 shows the associated wins of each solution. Both behaviors, $Comp_1$ and $Coop_\alpha$, obtained good performance but the new solution $B123$ outperformed them with a greater number of wins than the other approaches, which indicates that: 1, the context dependent blending provides a great advantage compared to fixed arbitration coordination; 2, automatically tuning of fuzzy membership functions yield better results than using hand-coded solutions.

8 Conclusions

In the particular context of the FUZZ-IEEE 2007 Car Racing Competition we have introduced a tentative proposal to compute with actions by computing with linguistic descriptions of behaviors. In this context we have identified and tackled the following key aspects: how to define different behaviors by describing them linguistically and how to coordinate several behaviors using context dependent rules.

We have set up a mixture of competitive and cooperative co-evolutionary algorithms to model the membership functions of the linguistic variables used. On one hand, competitive co-evolution allows to find alternative solutions that compete among them to find better solutions and to avoid being trapped in local minima. On the other hand, cooperative co-evolution allows to find complementary solutions that cooperate among them to find complex and better solutions.

By following an incremental approach, we were able to decompose the solution to this problem in simple behaviors with a lower degree of complexity. The use of fuzzy rules for behavior coordination and behavior description has allowed us to have a shared framework that makes easy to develop complex behaviors and to compute with linguistic descriptions of actions. The use of co-evolutionary algorithms has allowed us to test, compare, combine and tune the different behaviors appropriately and obtain a better solution.

In general, it is not possible to predict a priori all the scenarios, problems, or solutions. So a “learning by doing” approach is a need in these cases. The know-how obtained by testing and playing with different solutions is irreplaceable in order to build systems that compute with actions described by words. This approach allows to mix designer’s knowledge and knowledge learned from the data, maintaining a good interpretability of the solution. This represents a very important step-stone that would allow to change and improve the system in front of new environments and situations.

Acknowledgment

The author want to thank to Claudio Moraga, David Perez and to the anonymous reviewers for their comments and fruitful discussion about the paper.

References

[1] J. Highsmith and A. Cockburn. Agile software development: The business of innovation. *Computer*, 34(9):120–127, 2001.

[2] L.A. Zadeh. Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transaction on Systems, Man and Cybernetics*, 3:28–44, 1973.

[3] L.A. Zadeh. On the analysis of large-scale systems. In *Fuzzy Sets, Fuzzy Logic, and Fuzzy Systems: Selected Papers*, pages 195–209. World Scientific Pub Co Inc, 1996.

[4] L.A. Zadeh. A fuzzy-algorithmic approach to the definition of complex or imprecise concepts. *Int. J. Man-Machine Studies*, 8:249–291, 1976.

[5] L.A. Zadeh. Precisiated natural language (PNL). *AI MAGAZINE*, 25(3):74–91, 2004.

[6] L.A. Zadeh. A New Frontier in Computation? Computation with Information Described in Natural Language. In *Proceedings Symposium on Fuzzy Systems in Computer Science FSCS*, pages 1–2, 2006.

[7] L. A. Zadeh. Fuzzy logic = computing with words. *IEEE Trans. Fuzzy Systems*, 4:103–111, 1996.

[8] L.A. Zadeh. From computing with numbers to computing with words.from manipulation of measurements to manipulation of perceptions. In P.P. Wang, editor, *Computing with words*, pages 35–68. John Wiley & Sons, 2001.

[9] L.A. Zadeh. A New Direction in AI: Toward a Computational Theory of Perceptions. *AI Magazine*, 22(1):73, 2001.

[10] S. Guadarrama. *A Contribution to Computing with Words / Perceptions*. PhD thesis, Universidad Politécnica de Madrid, Madrid, April 2007. http://oa.upm.es/448/.

[11] C. Moraga, E. Trillas, and S. Guadarrama. Multiple-valued logic and artificial intelligence: Fundamentals of fuzzy control revisited. *Artif. Intell. Rev.*, 20(3-4):169–197, 2003.

[12] S. Guadarrama and R. Vazquez. Tuning a fuzzy racing car by coevolution. In *Genetic and Evolving Systems, 2008. GEFS 2008. 3rd International Workshop on*, pages 59–64, 2008.

[13] L.A. Zadeh. A rationale for fuzzy control. *Jr. of Dynamic Systems, Measurements and Control*, 34:3–4, 1972.

[14] S. Assilian and E.H. Mamdani. Learning control algorithms in real dynamic systems. In *Int. IFAC/IFIP Conf. on Digital Computer Appl. to Process Control*, ZÜrich, 1974. IFAC/IFIP.

[15] E.H. Mamdani. Advances in the linguistic synthesis of fuzzy controllers. *Int. Jr. Man-Machine Studies*, 8:669–678, 1976.

[16] C. Karr. Genetic algorithms for fuzzy controllers. *AI Expert*, 6(2):2633, 1991.

[17] F.J. Moreno-Velo, I. Baturone, S. Sanchez-Solano, and A. Barriga. XFUZZY 3.0: A Development Environment for Fuzzy Systems. In *Proc. International Conference in Fuzzy Logic and Technology*, pages 93–96, 2000.

[18] JJ Saade and H. Schwarzlander. Ordering fuzzy sets over the real line: an approach based on decisio making under uncertainty. *Fuzzy sets and systems*, 50(3):237–246, 1992.

[19] E.H. Ruspini. Fuzzy logic in the flakey robot. In *Procs. of the Int. Conf on Fuzzy Logic and Neural Networks*, pages 767–770, Iizuka, Japan, 1990.

[20] A. Saffiotti, K. Konolige, and E.H. Ruspini. A multivalued logic approach to integrating planning and control. *Artificial Intelligence*, 76(1-2):481–526, 1995.

[21] A. Saffiotti, E.H. Ruspini, and K. Konolige. Blending reactivity and goal-directedness in a fuzzy controller. In *Fuzzy Systems, 1993., Second IEEE International Conference on*, pages 134–139, 1993.

[22] O. Cordon, F. Herrera, F. Gomide, F. Hoffmann, and L. Magdalena. Ten years of genetic fuzzy systems: current framework and new trends. *Fuzzy Sets and Systems*, 141(1):5–31, 2004.

[23] C.A. Pena-Reyes and M. Sipper. Fuzzy CoCo: a cooperative-coevolutionary approach to fuzzy modeling. *Fuzzy Systems, IEEE Transactions on*, 9(5):727–737, 2001.

[24] C.D. Rosin and R.K. Belew. New Methods for Competitive Coevolution. *Evolutionary Computation*, 5(1):1–29, 1997.

[25] M.A. Potter and K.A.D. Jong. Cooperative Coevolution: An Architecture for Evolving Coadapted Subcomponents. *Evolutionary Computation*, 8(1):1–29, 2000.