Proceedings of the 1996 IEEE International
Symposium on Intelligent Control
Dearborn, MI • September15-18, 1996 pp. 127-132

EVOLUTIONARY DESIGN OF FUZZY LOGIC CONTROLLERS

Carlos Cotta, Enrique Alba and José Mª Troya

Dpto. de Lenguajes y Ciencias de la Computación
E.T.S.I. Informática, Universidad de Málaga
Campus de Teatinos (2.2.A.6), 29071-Málaga (SPAIN)
ccottap@lcc.uma.es

Abstract

An evolutionary approach to fuzzy logic controller design is presented in this paper. We propose the use of a class of genetic algorithms to produce suboptimal fuzzy rule-bases (internally represented as constrained syntactic trees). This model has been applied to the cart centering problem. The obtained results show that a good parameterization of the algorithm and an appropriate evaluation function lead to near-optimal solutions.

1. Introduction

Genetic Algorithms (GAs) [1] are heuristic optimization procedures based on the principles of natural evolution. They combine the concepts of adaptation and survival of the fittest to produce suboptimal solutions for an arbitrary problem. This is achieved by means of the repeated application of mating and reproduction operators onto a population representing potential solutions. *Genetic Programming* (GP) [2] is a subset of the GA paradigm in which each individual in the population is a computer program (usually represented as a syntactically correct tree). These programs try to solve a certain task, given as a set of examples expressing the desired behavior. Programs are built up by the composition of some functions (internal nodes of the tree) operating on a set of terminal symbols (leaves).

This work presents an approach in which individuals (trees) do not represent traditional computer programs but fuzzy rule-bases intended to solve a control problem (the cart centering problem, described further in this paper). A fuzzy interpreter is used to test the resulting fuzzy rules on the target problem.

This paper is organized as follows: a brief review of fuzzy logic controllers and a description of the cart centering problem is first outlined, followed by the GP model used for the experiments. Finally, we present a comparative performance analysis of our approach and extract some important conclusions regarding the reproductive policy and the evaluation function.

2. Fuzzy Logic Controllers

Fuzzy Logic Controllers (FLCs) are rule-based systems that successfully incorporate the flexibility of human-decision making by means of the use of fuzzy set theory [3]. Natural-language terms (e.g. pressure is low) are used in fuzzy rules. Their ambiguity is modeled by membership functions, intended to represent human expert's conception of the linguistic terms. A membership function gives a measure of the confidence with which a precise numeric value is described by a linguistic label.

Rules take the form **IF** [*conditions*] **THEN** [*actions*], where *conditions* and *actions* are linguistic terms describing the values of input and output variables (e.g. **IF** *pressure* **IS** *low* **THEN** *valve-opening* **IS** *small*). The fuzzy rule-base of the FLC is composed of a number of such fuzzy rules. This rule-base is used to produce precise output values according to actual input values. This control process is divided into three stages (Figure 1):

a) *fuzzification*: calculate fuzzy input, i.e. evaluate input variables with respect to the corresponding linguistic terms in the condition side.

b) *fuzzy inference*: calculate fuzzy output, i.e. evaluate activation strength of every rule and combine their action sides.

c) *defuzzification*: calculate actual output, i.e. convert fuzzy output into a precise numerical value.

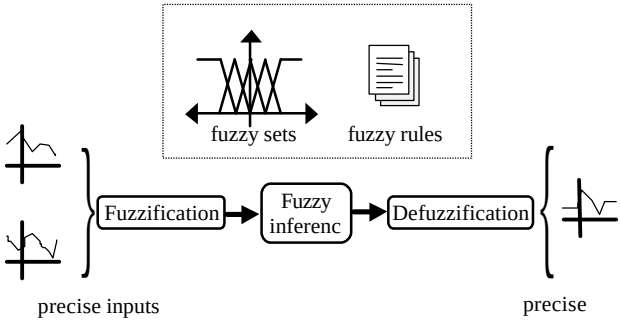


Figure 1. *The Fuzzy Control Process.*

The fuzzy interpreter used in this work performs fuzzification via triangular membership functions, uses the *min* intersection operator, the *maxmin* method for fuzzy inference and the *centroid* procedure for defuzzification.

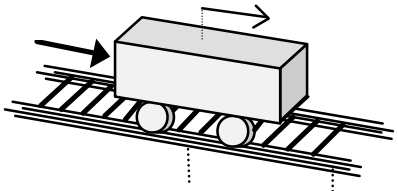
3. The Cart Centering Problem

In this section a description of the cart centering problem will be made. Next, it will be shown how the principles of

fuzzy control can be applied to this problem, outlining an ad-hoc solution.

3.1. Description of the Problem

The cart centering problem definition involves a cart with mass *m* moving along an infinite one-dimensional frictionless track. The problem consists in centering the cart, in minimal time, by applying an external time-variant force (of maximal magnitude F) so as to accelerate the cart in either the positive or the negative direction of the track. Given the values for the position $x(t)$ and velocity $v(t)$ of the cart at time t, this problem can be further formulated as calculating a force $F(t) = F[x(t), v(t)]$ in order to place the cart in position $x(t_f)=0$ with velocity $v(t_f)=0$ in minimal t_f.



$$F(t) = \begin{cases} +F & \text{if } -x(t) > \dfrac{v(t)^2 \operatorname{sgn}[v(t)]}{2\frac{F}{m}} \\ -F & \text{otherwise} \end{cases}$$

Figure 2. *The Cart Centering Problem and its optimal solution*.

Though simple, this is a typical optimal control problem and has a well-known solution (the *bang-bang* rule [2]).

This system is simulated using Euler's method, i.e. taking time steps of τ sec. and applying Newton's Laws:

$$x(t+\tau) = x(t) + \tau \cdot v(t)$$
$$v(t+\tau) = v(t) + \tau \frac{F(t)}{m}$$

As in [4] and [5], we consider τ=0.02 sec. and m=2.0 kg.

3.2. Designing a Fuzzy Logic Controller

As an optimal control problem, the cart centering problem is well-suited for fuzzy control. The development of an FLC for this system involves the following three steps [6]:

a) Determine *condition* (input) variables and specify the fuzzy sets describing them.

b) Determine *action* (output) variables and specify the fuzzy sets describing them.

c) Produce the rule-base.

It is clear that position $x(t)$ and velocity $v(t)$ of the cart are the input variables, and the force $F(t)$ applied to the cart is the only output variable in this problem. The membership functions defining the fuzzy sets over these variables are shown in Figure 3: five partitions representing the linguistic terms negative large (NL), negative small (NS), zero (ZE), positive small (PS) and positive large (PL).

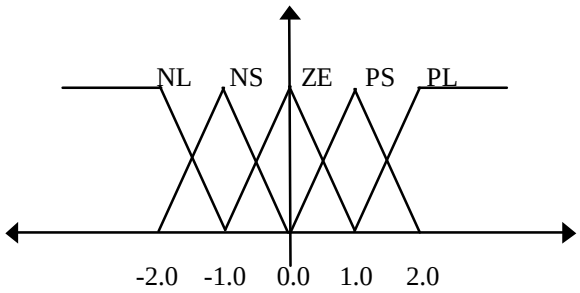


Figure 3. *Fuzzy sets for position, velocity and force.*

The third step (production of the rule-base) may be accomplished by a human expert. The use of fuzzy linguistic terms allows an easier incorporation of human knowledge. For example, it seems intuitive in this problem that the applied force must almost always be directed towards the origin (since we want to place the cart in the 0 position) and, just when we are close to that point, we must reverse the force to stop the cart. The following fuzzy rules express these rules-of-thumb:

```
IF pos IS PL THEN for IS NL
IF pos IS PS THEN for IS NL
IF pos IS ZE AND vel IS PL THEN for IS NL
IF pos IS ZE AND vel IS PS THEN for IS NL
IF pos IS ZE AND vel IS NS THEN for IS PL
IF pos IS ZE AND vel IS NL THEN for IS PL
IF pos IS NS THEN for IS PL
IF pos IS NL THEN for IS PL
```

where *pos*, *vel* and *for* stand for position, velocity and force respectively.

The FLC described above is not the optimal one. Improvements in the rule-base or in the interpretation of the linguistic terms may be done by means of GAs (see [4], [5], [6], [7]). We propose the use of GP to produce rule-bases that can be used as a starting point for further refinements by a human expert or even as a final solution.

4. Evolutionary Approach

The genetic programming model is exposed in this section. First, the encoding is described. Next, the necessity of a type system is shown. Finally, some general details about the evaluation function are outlined.

4.1. Two Ways of Encoding a Rule-Base

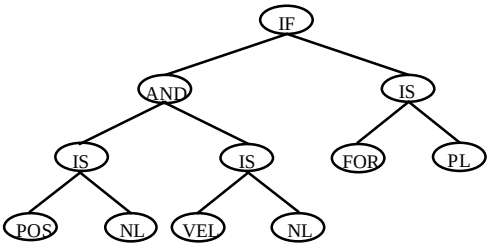
There are two approaches when trying to encode a rule-base, the Pitt approach and the Michigan approach [9]. In the former, every individual in the population represents a whole rule-base that competes with the other ones. In the latter, a single rule is contained in each individual; the behavior of the FLC is determined by the cooperation between all individuals. It must be taken into account that selecting any of these two schemes conditionates other aspects of the algorithms (e.g. fitness assignation).

A Pitt approach has been chosen following a simplicity criterion: fitness can be calculated via a short simulation and no advanced genetic mechanism (like *sharing* or *crowding*) is needed.

4.2. Encoding Rule-Bases

Given that every individual contains a list of rules, an encoding for single rules and subsequently a mechanism to join them is needed.

A single rule can be easily represented as a binary tree: the root being an `IF` node, and left and right branches representing the condition and the action sides respectively. Likewise, both conditions and actions can be expressed as trees. On the one hand, a variable paired with a fuzzy set can be represented as a tree with an `IS` root-node: the variable name in the left branch and the fuzzy set in the right branch. On the other hand, a conjunction of such terms can be expressed as a tree with an `AND` root-node and two branches representing nested conjunctions or pairs (variable, fuzzy set). Figure 4 shows an example.



IF *pos* **IS** NL **AND** *vel* **IS** NL **THEN** *for* **IS** PL

Figure 4. *A syntactic tree and the rule it represents.*

A list of rules might be represented as a list of trees. However, it is more appropriate to use a single tree representation because it allows rules to be tightly connected and therefore facilitates the propagation of good functional blocks. Two additional symbols are required. The `EL` symbol represents an empty list. The `RLIST` symbol combines rules in the same fashion that the `AND` symbol joins terms, i.e. a list of rules is a tree with an `RLIST` root-node and two branches representing single rules or analogous lists of rules.

This is a very flexible and powerful representation of FLC rule-bases. It deals well with the fact that an FLC may have rules with very different structures among them as well as a variable number of rules. This aspect contrasts with other GA-based approaches in which the structure [4] [7] or the number of rules in the FLC [5] is arbitrarily prefixed.

4.3. Necessity of a Type-System

The flexibility of this tree representation makes crossover a very powerful tool since it allows interchange at several levels: rules, terms or even variable names. However, the traditional GP crossover operator may produce nonsense rules. For example, consider the case in which a variable name is substituted by a whole rule. There are three possible solutions for that problem:

a) Define closed functions, so those ill-defined rules are reinterpreted in some predefined way [2].

b) Repair the tree, deleting incorrect subtrees and adding new subtrees if necessary.

c) Select crossover points in such a way that the resulting trees be correct [9].

The third option has been chosen in order to avoid the creation of a high number of useless trees and the production of very non-causal changes [10] in offspring due to the repairing mechanism.

Every symbol has a type attribute, determined by means of the partitions defined by the equivalency relationship "is interchangeable with". For example, since an `EL` node may be swapped with an `RLIST` node, they have the same type. Figure 5 shows all types.

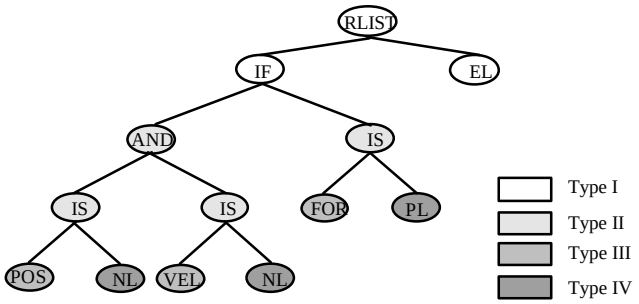


Figure 5. *Basic types.*

The crossover operator first selects a random node in one parent; then, a node of the same type is randomly selected in the other parent and the corresponding subtrees are swapped, thus producing two valid offspring. However, considering just the types above leads to the creation of syntactically correct but semantically incorrect trees with output variables appearing in the condition side and/or input variables appearing in the action side (e.g. **IF** *for* **IS** NL **THEN** *pos* **IS** ZE). These situations must be avoided to make a good use of computational resources. To achieve this, two subtypes for both type II and III are defined.

Every subtype can be seen as the variety of the corresponding type that appears in either the condition side or the action side. For example, for type `VBLE` (type III), we define the subtypes `VBLEIN` and `VBLEOUT`, representing input and output variables respectively. See Figure 6 for a complete taxonomy of types and subtypes.

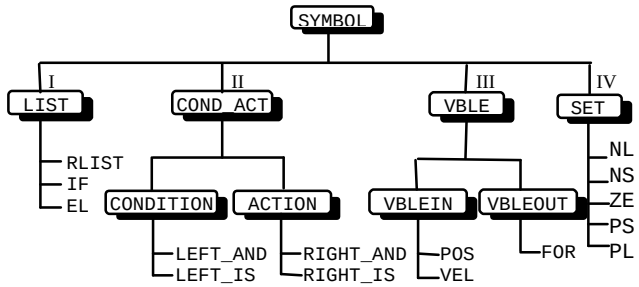


Figure 6. *Types and symbols used.*

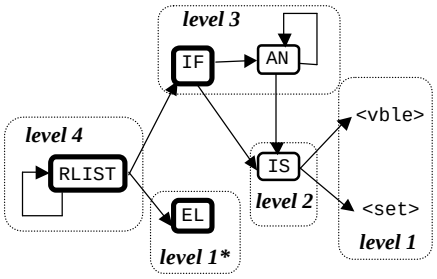
4.4. Generating Rule-Bases

Trees are bounded to have a maximum depth of 6 levels in the initial generation and 15 in subsequent generations. Initially, the population is generated using a *half-ramping* mechanism [2]. We follow the BNF grammar shown in Figure 7 to produce correct trees. Notice that only the last two rules (describing input and output variables of the problem) are problem-dependent. The rest of rules can be used in any other fuzzy problem.

```
<TREE>    ::= EL | <IF> | <RLIST>
<RLIST>   ::= <TREE> <TREE>  RLIST
<IF>      ::= <COND> <ACT> IF
<COND>    ::= <L_IS> | <L_AND>
<L_IS>    ::= <VBLEIN> <SET> LEFT_IS
<L_AND>   ::= <COND> <COND> LEFT_AND
<ACT>     ::= <R_IS> | <R_AND>
<R_IS>    ::= <VBLEOUT> <SET> RIGHT_IS
<R_AND>   ::= <ACT> <ACT> RIGHT_AND
<SET>     ::= NL | NS | ZE | PS | PL
<VBLEIN>  ::= VEL | POS
<VBLEOUT> ::= FOR
```

Figure 7. *BNF grammar of correct trees (described in postorder).*

Every tree is generated in a top-down fashion, i.e. choosing a root-node and then producing appropriate subtrees in a similar way. To generate a node, a random type is chosen and subsequently a random symbol belonging to that type is assigned to that node. The selection of this random symbol depends on the level of the node (e.g. an `IF` node cannot appear in levels below 3 because its branches must hang two levels down -an `IS` subtree- at least). Figure 8 shows which symbols may be root of a subtree according to its parent and the maximal depth of the tree. The root node of the whole tree may be only one of the thick nodes.



**** not allowed in complete trees***

Figure 8. *This graph shows which symbols may be arguments of another one and the minimal level at which they may appear in a complete tree.*

4.5. Evaluating Rule-Bases

Individuals are trained through 16 simulations of the problem in which the initial conditions are equally spaced points in the input surface. Fitness is assigned according to the performance on these test cases. This evaluation technique is preferred to selecting random starting points because it is less noisy and assures a better coverage of all possible cases. However, there may be situations in which the computational complexity of a simulation makes it too expensive to perform such a complete sampling and performing random tests may be a good tradeoff.

The simulations are executed for a maximum number of 500 time steps. The cart is considered to be centered if the distance to the goal state in the input surface is small enough. Notice that considering a particular distance function notably influences the final results, as shown in the next section. If the cart could not be centered, the maximum number of steps is taken. The mean time of all simulations is subtracted from 500 to obtain a fitness value (to be maximized).

5. Results

In this section we show the results the GP model has produced. For comparison purposes, we also include the results of the intuitive solution described in section 3.2. and some results from related works.

5.1. Optimal and Intuitive Solutions

To measure the performance of a given FLC, the mean time to center the cart throughout 100 simulations of the problem, starting from random initial points in the domain [-2.5, 2.5] is used. Situations in which the initial state is also a solution state are excluded. As stated in section 3.1., the parameters of the simulations were $\tau = 0.02$ sec, $m = 2.0$ kg. Force has a cut-off value of 2.5 N. Simulations are executed for a maximum number of 500 time steps. The cart is considered centered when $\mathbf{max}(|pos|, |vel|)<0.5$.

Whilst the optimal solution averaged 129 time steps to center the cart, our *naive* solution took 249 time steps, timing-out 14 times. In Figure 9, the control surfaces of both solutions are shown.

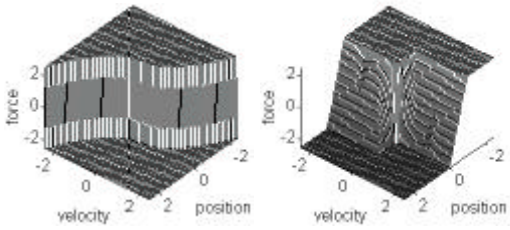


Figure 9. *Optimal (left) and naive (right) control surfaces.*

The 18 rule-FLC described in [4] is tested too. Its performance is better than our *naive* solution. The mean time to center the cart is 149 time steps (for the parameters of our simulator).

5.2. The GP solution

10 runs of the GP system were performed. Following [2], a population size of 500, for a maximum number of 51 generations (the initial one plus 50 additional generations) was chosen. The reproductive policy was generational and elitist. Selection was 50% fitness-proportionate, 50% random. This mechanism was used as a tradeoff between the exploration of the search space and the exploitation of known solutions. Figure 10 shows the best and the mean fitness in every generation, averaged for the 10 runs.

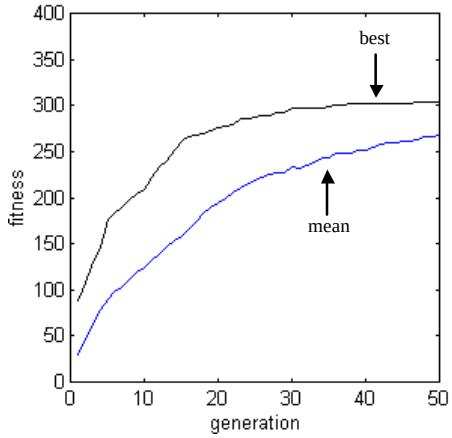


Figure 10. *Best and mean fitness averaged for 10 runs.*

The best generated individual obtained a fitness value of 321 (averaging 158 time steps to dock the cart), and corresponded to the following 9 rule-FLC:

1) **IF** *pos* **IS** PL **THEN** *for* **IS** NL
2) **IF** *pos* **IS** PS **THEN** *for* **IS** NL
3) **IF** *pos* **IS** NL **THEN** *for* **IS** PL
4) **IF** *vel* **IS** PL **THEN** *for* **IS** NL
5) **IF** *vel* **IS** NS **THEN** *for* **IS** PL
6) **IF** *vel* **IS** NL **THEN** *for* **IS** PL
7) **IF** *pos* **IS** ZE **AND** *vel* **IS** PS **THEN** *for* **IS** NL
8) **IF** *pos* **IS** NS **AND** *vel* **IS** ZE **THEN** *for* **IS** PL
9) **IF** *pos* **IS** PS **AND** *pos* **IS** PL **AND** *vel* **IS** ZE **AND** *vel* **IS** NS **THEN** *for* **IS** NL **AND** *for* **IS** NS **AND** *for* **IS** PL

The rule-base above was obtained after deleting duplicated rules. Rules #2, #3 and #6 appeared twice and rules #1 and #4 were repeated three times. This indicates they are important functional blocks for the FLC. The best individual also contained two dummy rules whose condition side is equivalent to an empty fuzzy set, so they carry no influence on the output.

Figure 11 shows the control surface of this solution. It can be seen that it is a rather poor approximation of the optimal solution (Figure 9). Since it is well-known that the optimal output for this problem is always saturated, a wrapper converting positive values of $F(t)$ to $+F$ and negative values to $-F$ was added. The average time to center the cart is then reduced down to 131 time steps. However, this FLC did not evolve under that condition and therefore the evolutionary process did not optimize it for that purpose. Consequently, a new set of 10 runs was performed adding this wrapper to the evaluation function. Figure 12 shows how the quality of the solutions is very notably improved.

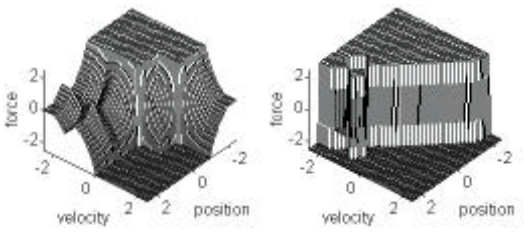


Figure 11. *Control surfaces of the evolved FLC (left) and its saturated counterpart (right).*

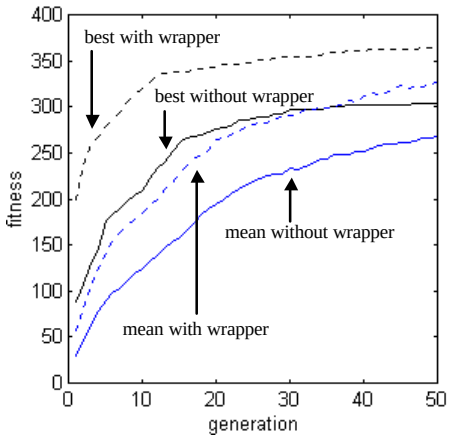


Figure 12. *Improvement in fitness adding a wrapper.*

The best evolved FLC has 13 rules. Its control surface is show in Figure 13.

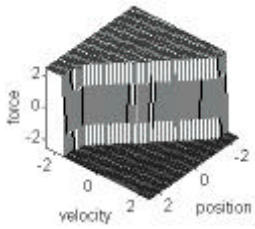


Figure 13. *Control surface of the best FLC evolved using a wrapper in the evaluation function.*

The average time it takes to center the cart is 120 time steps. Though it may seem that this FLC performs better than the optimal solution (129 time steps), it is only a consequence of the distance function that has been used. The **max** function considers as a goal state any point within a square centered in the origin. The GA learned that reaching a corner of the target zone is faster, in some situations, than approaching any other point of its perimeter. Figure 14 shows an example.

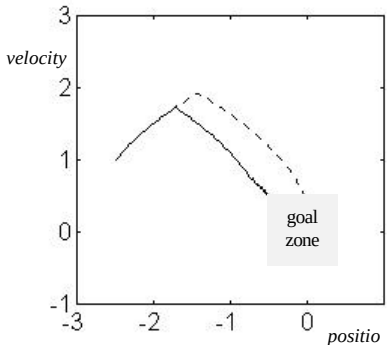


Figure 14. *Example for x(0)=-2.5 and v(0) = 1 of the optimal solution (dotted line) and the evolved FLC (solid line).*

If an Euclidean distance function is taken, the average time to dock the cart increases up to 133 time steps. To test the effect of this new stopping criterion, another set of 10 experiments were done. A best FLC with 21 rules was obtained. Its control surface is show in Figure 15.

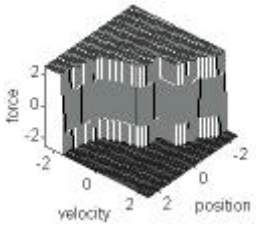


Figure 15. *Control surface of the best FLC evolved with a wrapper and Euclidean stopping criterion.*

It represents a very good approximation to the optimal solution. This FLC averages 127 time steps to dock the cart. This small *improvement* with respect to the optimal solution reflects the fact that the analytical solution is optimal to place the cart exactly in the origin, but not for moving it to an area around it.

Finally, a set of experiments were performed using a steady-state reproduction policy (i.e., generating just 2 individuals in every iteration and inserting them into the population substituting the worst ones). Figure 16 shows a comparison of this model with the generational one. Notice how the steady-state GA has a much higher convergence rate and therefore good solutions may be obtained earlier.

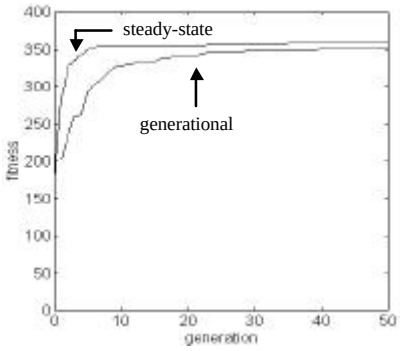


Figure 16. *Comparison of convergence rates in the generational and the steady-state models.*

6. Conclusions

A GP approach to FLC design has been presented. This class of genetic algorithms is more appropriate than other more traditional evolutionary models due to the use of high-level data structures; complex rule-bases may be easily represented using trees. This encoding is much more flexible than the fixed-length binary encoding of simple genetic algorithms since there may be any number of rules in an FLC. Moreover, rules may have any structure as long as they are syntactically and semantically correct. Therefore any good underlying (known or unknown) structure can evolve to improve the GP result.

This model has an additional advantage: its extensibility. Some GA-based systems work on encodings in which the rules are constrained to have a determined structure or even the number of rules is fixed. In such models, increasing the number of variables involved in the test problem implies a growth in chromosome length, and therefore large populations are required to keep diversity. Since most computational effort is devoted to simulate the system to obtain a fitness measure, a large population leads to very long runs of the GA. In the GP system, it would be necessary to increase the maximum depth of trees, thus allowing more complex rules and FLCs to evolve. However, since loss of diversity is not a main issue in GP [2], the population size could be kept constant (or just slightly increased).

The results are very promising. FLCs have been generated clearly outperforming an intuitive solution and other evolutionary approaches. Their quality is comparable with the optimal solution.

Advanced GP techniques (like Automatically Defined Functions [11]) will be tried in future work. More difficult test problems (e.g. the cart-pole balancing) may be used as a touch-stone for that purpose.

References

[1] Holland, J.H. *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975.

[2] Koza, J.R. *Genetic Programming*, MIT Press, Cambridge MA, 1992.

[3] Lee, C.C. Fuzzy Logic in Control Systems: Fuzzy Logic Controller-Parts I & II, *IEEE Transacts. on Systems, Man and Cybernetics*. 20:2. Pages 404-435, 1990.

[4] Thrift, P. Fuzzy Logic Synthesis with Genetic Algorithms. In Belew R.K., Booker L.B. (Eds.), *Procs. of the Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann Pub., San Mateo CA. Pages 509-513, 1991.

[5] Feldman, D.S. Fuzzy Network Synthesis with Genetic Algorithms. In Forrest S. (Ed.), *Procs. of the Fifth International Conference on Genetic Algorithms*, Morgan Kaufmann Pub., San Mateo CA. Pages 312-317, 1993.

[6] Karr, C.L. Design of an Adaptive Fuzzy Logic Controller using a Genetic Algorithm. In Belew R.K., Booker L.B. (Eds.), *Procs. of the Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann Pub., San Mateo CA. Pages 450-457, 1991.

[7] Lee, M.A. Takagi, H. Dynamic Control of Genetic Algorithms using Fuzzy Logic Techniques. In Forrest S. (Ed.), *Procs. of the Fifth International Conference on Genetic Algorithms*, Morgan Kaufmann Publ., San Mateo CA. Pages 76-83, 1993.

[8] Bonarini A. Evolutionary Learning of Fuzzy Rules: Competition and Cooperation. In Pedrycz W. *Fuzzy Modelling: Paradigms and Practice*, Kluwer Academic Press, Norwell MA, 1995.

[9] Montana, D.J. *Strongly Typed Genetic Programming*, Bolt Beranek & Newman, Inc. Technical Report #7866, 1993.

[10] Rosca J.P., Ballard D.H. Causality in Genetic Programming. In the *Procs. of the Sixth International Conference on Genetic Algorithms*, Morgan Kaufmann Publ., San Francisco CA, 1995.

[11] Kinnear, K.E. *Advances in Genetic Programming*, Chapters 5-6, MIT Press, Cambridge MA, 1994.