International Journal of Soft Computing 1 (1) : 52-60, 2006
© Medwell Online, 2006

Hierarchical Reinforcement Learning with Grammar-Directed GA-P

[1]Santi Garcia Carbajal and [2]Nouhad J.rizk
[1,2]Department of Computer Science, University of Oviedo,Gijon, Spain
[2]NDU University Jounieh -Lebanon, Lebanon

Abstract: This article proposes a grammatical approach to hierarchical reinforcement learning.It is based on the grammatical description of a problem,a complex task,or objective.The use of a grammar to control the learning process,constraining the structure of the solutions generated with standard GP, permits the inclusion of knowledge about the problem in a straightforward manner,if this knowledge exists.When the problem to be solved involves the use of fuzzy concepts,the membership functions can be evolved simultaneously within the learning process using the advantages of the GA-P paradigm. Additionally,the inclusion of penalty factors in the evaluation function allows us to try to bias the search toward solutions that are optimal in safety or economical terms,not only taking into account control matters.We tested this approach with a real problem,obtaining three difierent control policies as a consequence of the different fitness functions employed.So,we conclude that the manipulation of fitness function and the use of a grammar to introduce as much knowledge as possible into the search process are useful tools when applying evolutionary techniques in industrial environments.The modified fitness functions and genetic operators are discussed in the paper,too.

Key words: Hierarchical, reinforcement, grammar, knowledge

INTRODUCTION

Reinforcement learning comprises a large family of algorithms dealing with the problem of maximiz- ing the performance of an agent in unknown environments. At any time during the learning process,the agent can observe the state of the environment, denoted by $s \in S$ and apply an action, a $a \in A$. Actions change the state of the environment and also produce a scalar payo. value,denoted by $r_{s,a} \Re$.Any reinforcement learning algorithm will look for an action policy,$\pi : S \rightarrow A$ that maximizes the expected discounted sum of future payoff according to:

$$R = E\left[\sum \gamma^{t-t_0} r_t\right] \quad (1)$$

where γ is a discount factor that favors payoff s reaped sooner in time.In general,payoff. uses to be delayed,so the algorithm has to solve a temporal credit assignment problem

The most widely used algorithm for learning problems with delayed payoff is Q-Learning,that learns a value function denoted by:

$$Q : S \otimes A \rightarrow \Re \quad (2)$$

When the training is finished,Q represents a mapping from {state,action }couples into rewards and allows the agent to maximize R by picking actions in a greedy way with respect to Q.

$$\pi(s) = \text{argmax}_{\hat{a} \in A} Q(s, \hat{a}) \quad (3)$$

The value function Q is learned on-line through experimentation.Initially,all values Q (s,a)areset to zero.Suppose that during learning the agent executes action a at state s ,whichleadsto anew state s and the immediate payoff is $r_{s,a}$ Q-Learning uses this state transition to update Q (s,a):

$$Q(s,a) \leftarrow (1-\alpha).\, Q(s,a) + \alpha.\left(r_{s,a} + \gamma.V(s')\right) \quad (4)$$

with V (s´) = max âQ(s´, a)

The scalar α ($0 < \alpha \leq 1$)is the learning rate which is typically set to a small value that is decayed over time.

Hierarchical reinforcement learning: In order to scale reinforcement learning to complex real-world tasks,one possible way is to discover the structure of the problem during the learning process,or to include existing knowledge in the algorithm,to reduce the dimension of the searching space.Research in classical planning has shown that hierarchical methods can provide exponential reductions in the computational cost of finding good plans.Many researchers[1-6], have experimented with different

Corresponding Author: Santi Garcia Carbajal, Department of Computer Science, University of Oviedo,Gijon, Spain

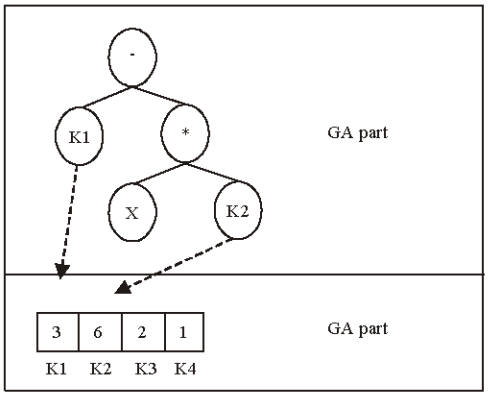

Fig. 1: Representation of a generic individual in GA-P techniques. GA-P individuals consist of two parts: a tree and a set of numerical constants. Genetic operators can take place in the tree (GP part), or in the set of constants (GA part)

methods of hierarchical reinforcement learning and hierarchical probabilistic planning. Recently,T.G. Dietterich proposed the MAXQ method as a way to specify the decomposition of a given reinforcement learning problem into a set of sub problems, simultaneously with a decomposition of the value function[7,8]. In this study we formalize a different approach for solving given reinforcement problem in a hierarchical manner.

Grammar Directed GA-P for Hierarchical Reinforcement Learning: In this section we formalize the characteristics of the problems this methodology is oriented to and set the terminology employed.Previously,a brief knowledge about Genetic Algorithms (GA),Genetic Programming,(GP)[9] and hybrid techniques like GA-P is required in order to understand the capabilities of our system.We will focus in the description of GA-P as GP and GA are well known techniques.

GAP-P techniques: GA-P technique[10] is an hybrid between genetic algorithms and genetic programming, which was first used in symbolic regression problems.Individuals in GA-P have two parts: a tree based representation and a set of numerical parameters. Different from canonical GP, the terminal nodes of the tree store not numbers,but linguistic identifiers that are pointers to the chain of numbers (Fig. 1). The behavior of the GA-P algorithm is mainly due to its crossover operator.Either or both parts of the individual may be selected and crossed.We have employed GA-P algorithms in the identification and control of complex dynamical processes and in classification problems[11].

Problem description: The generic control situation is defined by the following elements (Fig. 2):

- A set of continuous variables, E i showing the actual state of any relevant state variable of the system.
- A set of logical fiags, Fi Each fiag will be automatically activated by one safety restriction,specific to each problem. Flags are used to prevent the system from coming into unstable or dangerous states.
- A set of actuators, A i Every mechanical component of the system that can be modified by the control policy will be represented in the grammar by an actuator symbol.
- A set of local controllers, C i As we want to obtain hierarchical control policies,a set of partial controllers will be distributed over the system with no connection between them until the GP system finds the optimal hierarchy.
- The coordinates of the state variables and local controllers:e ix e iy C ix and C iy Connection cost will be a factor when calculating fitness function,so we take into account the position of each variable, actuator and local controller.
- A set of goals,expressed as predicates on the state variables,which we shall detach in:
- Goals that must be satisfied concurrently during the operation of the system,R i This means that there will be some safety restrictions that any policy must satisfy in order to be evaluated by the fitness function of the Genetic Programming System.

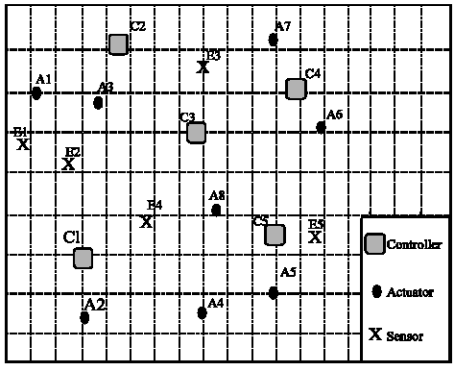

Fig. 2: Initial configuration of the system

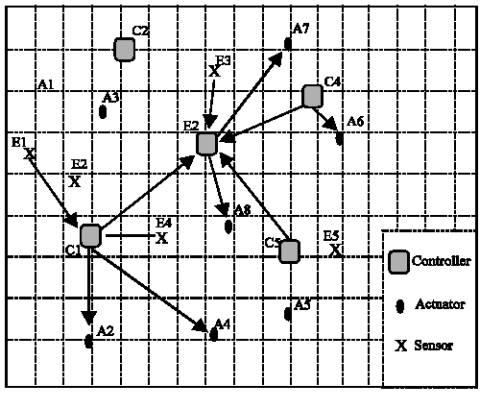

Fig. 3: Induced hierarchy of control

- Objectives that must as far as possible be satisfied, O i Finally,a good policy will be one that, satisfying safety restrictions,obtains an optimal control of the plant,with low connection costs.The way all these objectives are introduced into the fitness functions is discussed in section III-C.
- Three fuzzy sets,named LOW , MEDIUM and HIGH for each state or control variable of the system. When the learning process has finished,the system will produce an output consisting of (Fig. 3):
- The control hierarchy,in terms of the set of local controllers used,the connections between them and the state variables and actuators connected to each one.
- The set of fuzzy rules for each controller.
- The optimal values for the parameters that define the shapes of the fuzzy sets involved in the process.Each fuzzy set associated with a state variable will be defined by the position of three points (Fig. 1).The value of these points is represented in the GA part of the individuals (Fig. 1).

The genetic evolution of the solutions for a problem defined this way will be directed by afitness function that takes into account the following factors:

Safety Conditions: defined as logic predicates on the state variables of the system.Depending on the characteristic of the system to be controlled,a set of safety restrictions will be defined by the expert.

For example: _gValve Displacement Speed will never be greater than 0.5 m/s _h. Each rule will imply the existence of an associated fiag. Control Error, E C: defined as follows:

$$Ec = \int_0^T \left(c_i(t) - s_i(t)\right)^2 dt \tag{5}$$

where $c_i(t)$is the desired value of each variable,E i attimet and $s_i(t)$ is the real value.T is the length of the simulation period.

Control Cost, C :an efiort measure for the actuators during a simulation period.Typically,the sum of the displacements of all the valves of the system during the whole simulation.

$$C = \int_0^T \left(\frac{\partial \alpha(t)}{\partial t}\right)^2 dt \tag{6}$$

Implementation Costs, *CI* :an estimation of the implantation costs of the controller,mainly,due to the costs derived from the topology of the resulting control hierarchy.

$$C_I = \sum_{i=0}^{i=N} \text{length}(S_i) \tag{7}$$

where N is the total number of connection segments (S i needed to compose thefinal controller and length (S i is the length of each segment.

MATERIALS AND METHODS

The methodology we present consists of the steps shown in Fig. 4,which are described in detail in the following sections.

- The process starts with a description of the problem,made by the expert.
- Based on this description,we perform a deeper analysis that ends up with a grammatical description of the possible control solutions for the system.The solutions may or not be hierarchical,depending on the nature of the problem,but this point has no effect on the subsequent steps of the method.
- We choose a fitness function that gives us an estimation of the quality of solutions generated by the grammar defined in step 2.
- The genetic process obtains solutions according to this fitness function.
- The solutions are studied in order to determine their feasibility and if necessary, the grammar and fitness function are modi fied.GA optimization of the parameters that describe the shapes of fuzzy

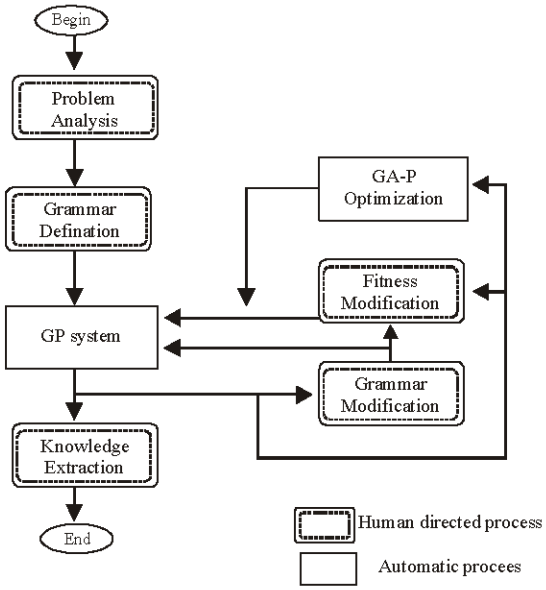

Fig. 4: The method starts from the grammatical definition of the problem and ends up with a set of rules and a control hierarchy

sets involved in the control can be performed concurrently within the genetic evolution of solutions due to the GA-P coding if the individuals,or off -line,as shown in Fig. 4.

Creating population: A simplistic manner to generate N strings of a given language would be to construct them randomly from the terminal symbols of the grammar.Obviously,this kind of mechanism would lead,in most cases,to a situation where an important fraction of the population would be formed by invalid strings. In some of the works of Andreas Geyer Schultz,a mechanism to generate an initial population uni- formly distributed regarding derivation-depth and form of the individuals is described[12].After some experimental runnings,we decided to implement a simpler way of generating the initial population:

- Choose population size, N and maximium derivation depth, P
- Select the initial symbol of the grammar
- Choose production and expand its right side.
- Repeat steps 2 and 3 for each non-terminal symbol appearing as a resultsof the substitution, untill derivstion depth, P, is reached.
- If the string is formed only by terminal symbol, put it into population.
- Return to 2 untill N is reached.

Genetic operators: When implementing a grammar directed Genetic Programming System,a new problem,related to the genetic operators implementation rises up:the individuals generated by these operators must be syntactically valid strings of the language,too[13,14]. Independently of the fact that some acceptable solutions could exist without complying with the restrictions imposed by the grammar,the learning process must be centered in the syntactically correct ones.In this work we implemented the genetic operators defined in[12].

Fitness function: In this section we define a generic .tness function for a Hierarchical Fuzzy controller.We call this Gradual Fitness Function since the value returned for a controller that leads the system out of the safety limits is as good as the length of the period during which the plant is working between safe, or acceptable conditions. The longer is this period,the higher is the value of the .tness function. We also de .ne two additional measures of the quality of a controller,naming their modularity and overlapping,in order to study the possibility of directing the search process through the introduction of these factors in the .tness function. For any given Hierarchical Fuzzy Controller J ,we compute the Gradual Fitness Function, $F_g(J)$
as:

$$F_g(J) = \begin{cases} \text{System between secure margin until } t_d \\ -\left(C_I + C + E_C + \left(T - t_d\right) * \left(D + A\right)\right) \\ \text{System between limits during the whole simulation} \\ -\left(C_I + C + E_C\right) \end{cases}$$

where C_I ,C ,E_c are the terms defined in section II and T is the total length of the simulation period.
t_d is the instant in which the system abandons the safety limits.
D is the maximum possible deviation regarding the goal state of the system.
A is the maximum possible control action.

C.1 Modularity: Let J be a hierarchical fuzzy controller. Let V Ci be the set of state variables that appear in the antecedents of any local controller C i Let |V |be the carnality of any set of variables,V. We de .ne the modularity coe .cient of a hierarchical fuzzy controller, $C_M(J)$, as

$$C_M(J)\frac{\left|\cup V_{C_i}\right| - \max\left\{\left|V_{C_i} \cap V_{C_j}\right|\right\}}{\left|\cup V_{C_i}\right|} \quad (8)$$

$C_M(J)$ reaches its maximum value (1)when the local controllers that form J share no variable. The modi .ed fitness function is:

$$F_g(J) = \begin{cases} \text{System between secure margin until } t_d \\ -(((C_I + C + E_C) * (1 - C_M) + (T - t_d) * (D + A) \\ \text{System between limits during the whole simulation} \\ -(C_I + C + E_C) * (1 - C_M) \end{cases}$$

Overlapping: The different local controllers included in a Hierarchical Fuzzy Controller J make their contribution to the .nal control action as a function of the membership degree of the state variables to the fuzzy sets referenced in their rule bases.Let V Ci be the subset of working variables for C i andlet μCi be the value returned by the membership function of the fuzzy region coupled with C i forthatsetof variables. We define ne the overlapping factor of a hierarchical fuzzy controller J,$G_S(J)$,as:

$$G_S(J) = \frac{\int_t \min\left\{\left(\mu_{C_i}\right)\right\} dt}{\int_t \max\left\{\left(\mu_{C_i}\right)\right\} dt} \quad (9)$$

that can be approximated by:

$$\frac{\sum \min\left\{\left(\mu_{C_i}\right)\right\}}{\sum \max\left\{\left(\mu_{C_i}\right)\right\}} \quad (10)$$

The fitness function modified to minimize overlapping,will be:

$$F_{gm}(J) = \begin{cases} \text{System between secure margin until } t_d \\ -(((C_I + C + E_C) * (G_S)) + (T - t_d) * (D + A) \\ \text{System between limits during the whole simulation} \\ -(C_I + C + E_C) * (G_S) \end{cases}$$

Starting from the de .nitions of Modularity and Overlapping,we will study how the introduction of these factors into the .tness function can help the genetic programming system to .nd solutions whose behavior when applied to a real problem could be interesting in any way (linguistic readability of the controller,quality of the control,optimization of connection costs,etc). Also,the GA-P optimization applied to the parameters that de .ne the fuzzy sets involved in the control process[11] and the inclusion of the (possibly inexistent) knowledge about the problem using the grammatical description of the solutions will be tested in a real world problem.

An illustrative example: electric plant control policy induction: The real problem that motivated the development of this methodology and the experimental results obtained using three di .erent .tness functions to learn the best hierarchical control policy.

Problem Description: The Abono energy central (Fig. 5)has two groups of coal boilers with 360 and 540 Mw of nominal Power Station in service since 1974 and 1985.They use national and imported coal and siderurgical residual gases from the CSI factory in Gij _Lon.CSI stands for Corporaci´ on Sider´ urgica Integral, i. e. Integral Siderurgical Corporation Nominal consumption of gas in the boilers of the energy central is 250000 $m^3 \frac{N}{hour}$ N hour and 360000 $m^3 \frac{N}{hour}$ N hour. Since 1974 both companies work together in order to guarantee the best conditions of e .ciency in the production and consumption of gas .ow.The CSI factory and the Abono Plant are connected by a pipeline 3 meters in diameter and 4 kilometers long.

Since 1995, the production in CSI is being increased,so the production of gas is being raised in the same proportion.In order to keep the best conditions of e .ciency in the managing of the gas,the capacity of the gas holders has been increased and the control policy must be adapted to this new situation.In[15]we developed a fuzzy controller to keep the gas flow to the boilers stable.Now we are working on a hierarchical reinforcement learning approach to minimize the implementation cost of the control policy due to the connection costs and to develop a predictive controller useful to train the human operators working in the plant. Schematically,this problem consists of the following elements(Fig. 6):
Inputs:

- Incoming gas flow.
- Position angle of system main valve.
- Position angles of the valves from gas holder 1 and 2.
- Position angles of the valves from Burners 1 and 2.
- Number of active burners in group 1 and 2.

Outputs:

- Gas flow derived to group 1.
- Gas flow derived to group 2.

The block diagram for the controller we are trying to induce is shown in Fig. 7

Fig. 5: Energy center of Abono

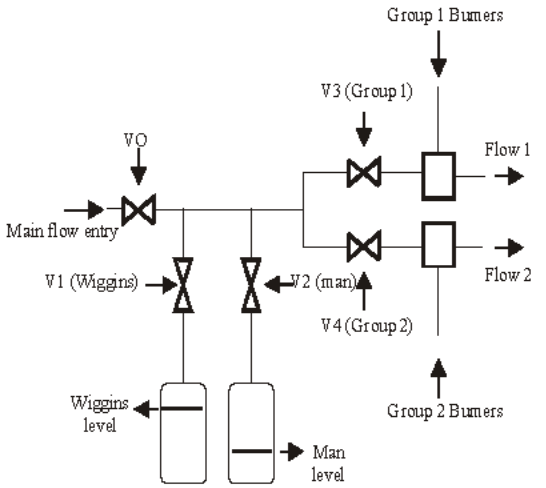

Fig. 6: Abono Plant. Synopsis

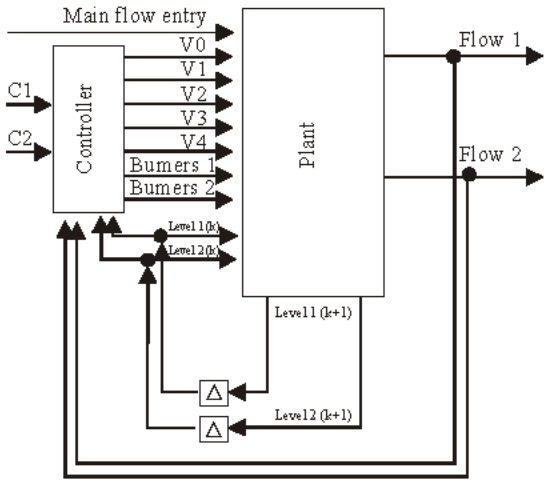

Fig. 7: Aboño. Block diagram for the controlle

Table 1: Aboño system variable

Symbol	Meaning	Type
V0	Main valve Position	Continuous
V1	Wiggins gas holder valve	Continuous
V2	Mann gas holder valve	Continuous
V3	Group 1 Valve Position	Continuous
V4	Group 2 Valve Position	Continuous
Q1	Active Burners of Group 1	Continuous
Q2	Active Burners of Group 2	Continuous
C1	Goal Group 1	Continuous
C2	Goal Group 2	Continuous
N1	Wiggins gas holder level	Continuous
N2	Mann gas holder level	Continuous
Flow	1 Actual Flow Group 1	Continuous
Flow	2 Actual Flow Group 2	Continuous

Table 2: Aboño. Flags

Symbol.	Meaning	Type	Val.
F_1	Wiggins out of range	Crisp	0,1
F_2	Mann out of range	Crisp	0,1
F_3	Pressure at Collector 1	Crisp	0,1
F_4	Pressure at Collector 2	Crisp	0,1
F_5	Piston 1 Speed	Crisp	0,1
F_6	Piston 2 Speed	Crisp	0,1

Flags: Table 2 summarizes the identified flags of this system.The special control policies started by the activation of each flag are:

- **F 1:** Close valve V1.Stop simulation.
- **F 2:** Close valve V2.Stop simulation.
- **F 3:** Stop simulation.Unbreeded boilers at Group 1.
- **F 4:** Stop simulation.Unbreeded boilers at Group 2.
- **F 5:** Close valve V1.Excessive speed at piston 1.
- **F 6:** Close valve V2.Excessive speed at piston 2.

Actuators: In this system there is an actuator by each valve of the plant,plus one by each group of boilers, that sets the number of active burners.

Fitness function definition: The definition of the fitness functions F_g F_{gm} and F_{g}.

RESULTS

The results obtained by applying this methodology to Abo .no Plant with three different fitness functions are collected.The main parameters of the genetic programming system are summarized in Table 3.

Gradual fitness: Figure 8 shows the structure of the best controller induced by F g This controller only uses variables Goal Group 1 Goal Group 2 and Actual Flow Group 2 Fig. 9 shows the system behavior when this controller is applied to the plant in training mode (Fig. 9) and test (Fig. 9). Figure 10 shows the position of valves in Group 1 and 2 during a simulation period in test mode.

Table 3: Aboño experimental setup

Popsize	3000
Maximum Generations	50
Mutation Probability	0.05
Fitness	
Fg,Fgm,Fgs	
Termination	G=50
Training cases	100
Test Cases	100

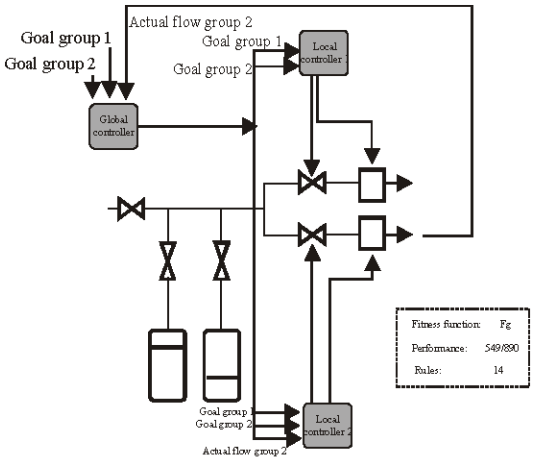

Fig. 8: Block diagram of Fuzzy Controller induced by F_g

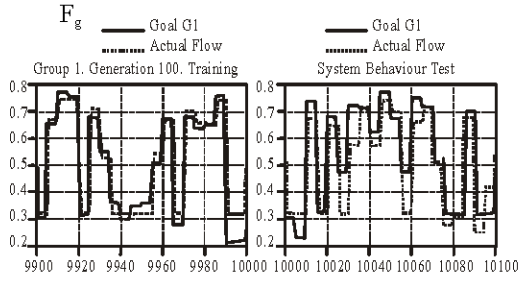

Fig. 9: Training and test behavior of the best controller induced by F_g

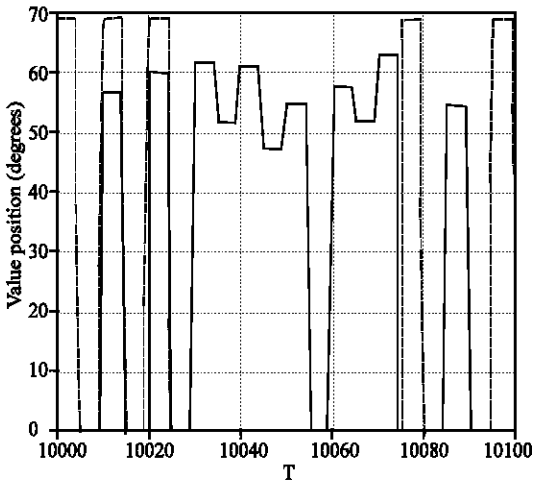

Fig. 10: Valves 1 and 2. Position during simulation using the best controller induced by F_g

Gradual fitness plus modularity: Figure 11 shows the structure of the best controller obtained using F gm The main controller of the hierarchy needs variables Goal Group 1 Goal Group 2 and Actual Flow

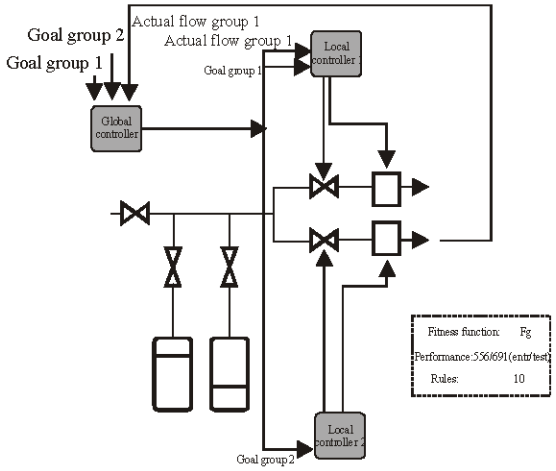

Fig. 11: Block diagram of Fuzzy Controller induced by F_{gm}

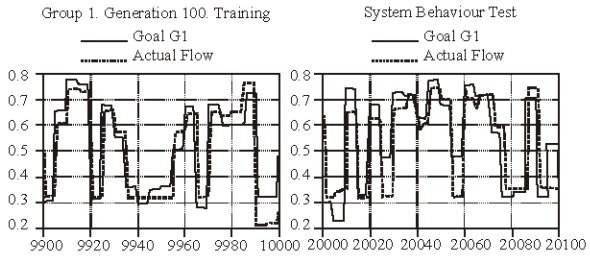

Fig. 12: Training and test behavior of the best controller induced by F_{gm}

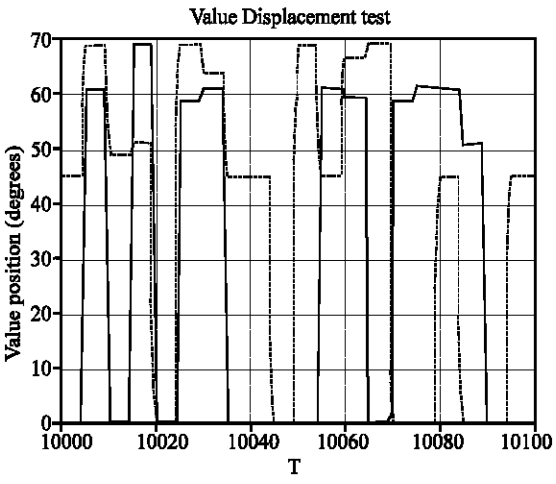

Fig. 13: Valves 1 and 2. Position during simulation using the best controller induced by F_{gm}

Table 4: Aboño. Average results 100 runs

Fitness function	Training	Test	Rules	CM	GS
F_g	680.25	750.12	16.51	0.27	0.71
F_{gm}	702.14	790.23	11.4	0.83	0.83
F_{gs}	649.11	739.41	14.2	0.61	0.65

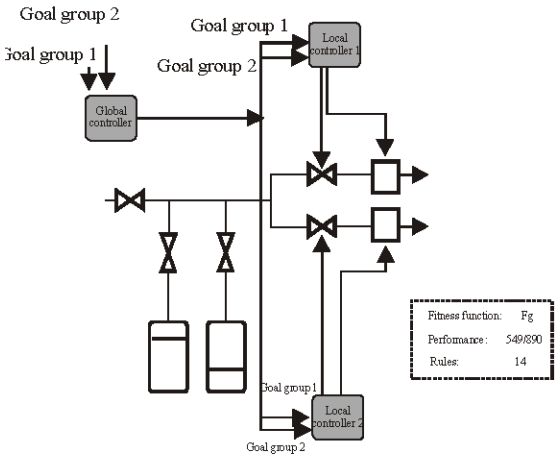

Fig. 14: Block diagram. of Fuzzy Controller induced by F_{gs}

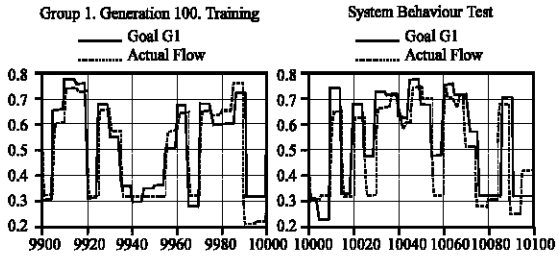

Fig. 15: Training and test behavior of the best controller induced by F_{gs}

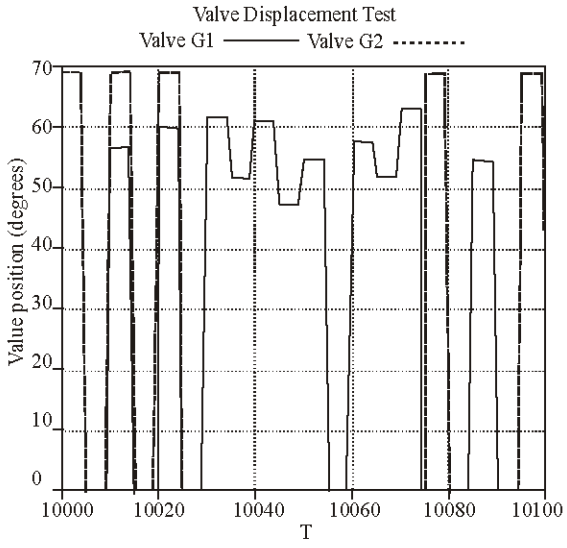

Fig. 16: Valves 1 and 2. Position during simulation using the best controller induced by F_{gs}

Group 1 It is important to note that the sets of variables used by the two local controllers are disjunct sets.This circumstance has great in .uence in decreasing connection costs of the controller. Fig. 12 shows the behavior of the system in training (left)and test (right)mode,under this controller.The displacement of valves in Group 1 and 2 is shown in Fig. 13.

Gradual fitness plus overlapping minimization: Figure 14 shows the structure of the best controller induced by the use of F gs This controller receives no feedback about the actual .ow circulating throw Groups 1 or 2 as a consequence of the action of its own control action.This points explains the low performance reached by this controller during the test phase. Figure 15 (left)shows the behavior of the system under the best controller obtained using F gs after 100 generations.Same .gure (right)shows the behavior of the system under the test conditions.The displacement of valves in Group 1 and 2 is shown in Fig. 16.

CONCLUSIONS

There have been many approaches to the supervised learning of fuzzy controllers by means of evolutive techniques. However, as far as we know,distal learning of control policies has not been deeply studied.The main conclusions of our work after applying the approach to a real problem,can be summarized as follows:

- The inclusion of modularity and overlapping factors directs the search process to solutions that promote di .erent behaviors of the induced hierarchical controllers.
- As can be seen in Table 4, modularity and overlapping minimization are opposite goals.The use of F_g leads to fuzzy controllers where there is low modularity and high overlapping. When using F_{gm} ,we obtained solutions that have high modularity,but with high overlapping. Finally,when F_{gs} was used, we obtained lower overlapping,but loosing modularity.
- For the real situation of control used to test the methodology,the best medium results were reached using F gm obtaining controllers with high modularity, this is, sharing only a few variables between the controllers implied in the hierarchy.

This work proposes a methodology to extract semantic rule bases of control and knowledge from mathematical models of complex systems.We have

obtained three different control policies by means of altering .tness function to bias the search process towards solutions satisfying safety and economical restrictions.As the internal structure of each local controller is based on a set of fuzzy rules, these policies can be used to extract additional knowledge about the problem and to make predictive modeling and control.

REFERENCES

1. Singh, S.P., 1992. Transfer of learning by composing solutions of elemental sequential tasks, Machine Learning, 8: 323–339.
2. Sutton, R., D. Precup and S. Singh, 1998. Between mdps and semi-mdps: Learning, planning and representing knowledge at multiple temporal scales. J. Articial Intelligence Res., 1: 1-39.
3. Kaelbling, L.P., 1993. Hierarchical learning in stochastic domains: Preliminary results, in ICML-93. San Francisco, CA, pp: 167-173.
4. Lin, L.J., 1993. Hierarchical learning of robot skills by reinforcement, in IEEE International Conference on Neural Networks. IEEE Computer Society Press, pp: 181–186.
5. Dayan, P. and G.P. Hintin, 1993. Feudal reinforcement learning, in NIPS. Morgan Kaufmann, 5: 271-278.
6. Parr, R. and S. Russell, 1998. Reinforcement learning with hierarchies of machines, in Advances in Neural Information Processing Systems, Michael I. Jordan, Michael J. Kearns and Sara A. Solla, Eds. 1998. The MIT Press.
7. Dietterich, T.G., 2000. Hierarchical reinforcement learning with the maxq value function decomposition, J. Artificial Intelligence Res., 13: 227–303, 2000.
8. Dietterich, T.G., 1998. The maxq method for hierarchical reinforcement learning, in Proceedings of the Fifteenth International Conference on Machine Learning. Morgan Kaufmann, pp: 118–126.
9. Koza, J.R., 1994. Genetic programming: On the programming of computers by means of natural selection, Statistics and Computing, pp: 4-2.
10. Howard, L.M. and D.J.D. Angelo, 1995. The GA–P: A genetic algorithm and genetic programming hybrid, IEEE Expert, 10: 11–15.
11. Garcia, S., F. Gonzalez and L. Sanchez, 1999. Evolving fuzzy rule based classifiers with GAP: A grammatical approach, in Genetic Programming, Proceedings of EuroGP'99, Riccardo Poli, Peter Nordin, William B. Langdon and Terence C. Fogarty, Eds., Goteborg, Sweden, of LNCS, Springer-Verlag, pp: 203–210.
12. Geyer-Schulz, A., 1995. Fuzzy Rule-Based Expert Systems and Genetic Machine Learning, Studies in Fuzziness,Physica-Verlag, Heidelberg,
13. David J. Montana, 1994. Strongly typed genetic programming, BBN Technical Report #7866, Bolt Beranek and Newman, Inc., 10 Moulton Street, Cambridge, MA 02138, USA.
14. Montana, D.J., 1995. Strongly typed genetic programming, Evolutionary Computation, 3: 199-230.
15. Garc´ýa. S. and L. S´anchez, 1998. Fuzzy control applied to a gas transport network in a siderurgycal environment, in Proceedings of IPMU'98, Paris, France, pp: 203-210.