Journal of Ubiquitous Computing and Intelligence
Vol. 1, 153–162, 2007

Data-Driven Methods for the Off-Line Generation of Fuzzy Logic Based Agents for an Intelligent Building Environment

A. López[1,*], D. Álvarez[1], F. Doctor[2], H. Hagras[2], and V. Callaghan[2]

[1] *Department of the Electrical Engineering, University of Oviedo, Gijón, Asturias 33204, Spain*
[2] *Department of Computer Science, University of Essex, Colchester, CO4 3SQ, UK*

Ambient Intelligence is nowadays an active research field. As a key part of this concept, learning architectures for the control of the devices in an intelligent building must be developed, where the goal is to control the environment via a set of devices using an intelligent agent which should work in a non-intrusive manner to satisfy the preferences of the user. Mainly, we have focused our attention over fuzzy logic controllers (FLC) for the internal structure of the agent. The main motivation for the work described in this paper is to check different alternatives in order to select a suitable method for the off-line data driven automatic generation of FLC for the agent. We have performed the experiments with real data gathered from the Essex Intelligent Dormitory.

Keywords: Ambient Intelligence, Embedded Agents, Machine Learning, Fuzzy Logic Systems, Ubiquitous Computing.

1. INTRODUCTION

Ambient Intelligence (AmI)[1] is nowadays an active research field. AmI deals with the development of a new paradigm where people are immersed in a digital environment that is aware of their presence and context, and which is sensitive, adaptive and reactive to their desires, habits, and emotions.

AmI builds on three key concepts: Ubiquitous Computing (integration of computers in daily objects), Ubiquitous Communication and Intelligent User Interfaces (voice, gestures). Nevertheless, to give a device (e.g. a lamp) processing and communication capabilities do not make it intelligent. A key concept to answer in AmI is how the system can learn about the user behaviors, and be constantly adapted to them. Autonomous adaptive learning systems must be developed to respond to user preferences and desires. There is a great deal of current research activity in Ambient Intelligence.[2–9] However they primarily focus on intelligent tracking and user interaction, sensing, planning, and time independent context and there is no great emphasis on such learning mechanisms.

The design of generic learning systems for ambient intelligent spaces could be quite challenging. Different scenarios (e.g., industry, workplace, home, machine to machine interaction, etc.) present special characteristics that could imply a specific design. In particular, intelligent buildings present a unique and specific set of characteristics and the aim of our research is to contribute to the development of intelligent mechanisms to address those characteristics.

Our goal in intelligent buildings is to control the environment via a set of devices using an intelligent agent which should work in a non-intrusive manner to satisfy the preferences of the user. The agent should observe the user's interaction with the building and learn how to preempt them automatically from the information provided by a set of environmental sensors (temperature, humidity, etc.) and by the states of the actuators (buttons, heaters, etc.).

We divide the life cycle of the agent in two main phases. In the first phase, the agent simply learns the user's desires and behaviors. It begins by compiling enough information about his behavior with a sampling process where the information provided by the environmental sensors and the corresponding actuator's states are stored (we have sampled the environment for three consecutive days in our experiments). Then the agent adapts his internal structures to fit the underlying knowledge hidden in the sampled data. In the second phase, the agent is used to control the intelligent environment on behalf of the user. It constantly

*Author to whom correspondence should be addressed.
Email: antonio@isa.uniovi.es

checks the information provided by the environmental sensors and decides in each situation the state for the environmental actuators in the basis of the learned knowledge. Nevertheless, the user necessities and desires can change over time. So, the agent should be capable of re-adapt his internal structures to reflect the user's behavioral drifts over time. In our experiments, the agent is aware of a change in the user behavior as the user is allowed to override an agent decision at any moment. Whenever this happens, it reflects a local conflict between the agent's decision and the user's desires and the agent should take the proper actions to correct this malfunction.

Mainly, we have focused our attention over fuzzy logic controllers (FLC) for the internal structure of the agent for two principal reasons. Imprecision is inherent in the sensor measurements and the user's management of the actuators in the environment. This way, FLC seem to be a good alternative because imprecision handling is a key feature of the approach. Also, fuzzy controllers are one of the easiest structures to interpret by a human reader. This could allow one to test if the controller is working properly, performing operations that really reflect the user behavior.

The main motivation for the work described in this paper is to check different alternatives in order to select a suitable method for the automatic generation of FLC for the agent in the initial learning phase. Nevertheless, we will have always in mind the online adaptation necessary in the second phase when discussing about the validity of the compared approaches for the real implementation of the intelligent agent.

From the wide spectrum of alternatives, we have selected (and adapted when necessary) for comparison several well known methods for the induction of rules from data like those based on the combination of genetic algorithms and fuzzy rule based systems (genetic-fuzzy systems),[10] Adaptive-Network-based Fuzzy Inference Systems (ANFIS),[11] and methods previously proposed that have shown to perform well on the target problem like AOFIS (Adaptive Online Fuzzy Inference System).[12] We include also results where Multilayer Perceptron neural networks (MLP) are used for the internal structure of the agent in order to check a different alternative for the internal structure of the agent than that based on FLC.

To validate the comparison, several experiments have been performed using real data gathered from the Essex Intelligent Dormitory (iDorm).

2. THE iDORM

The intelligent Dormitory (iDorm) shown in Figure 1 (top) is situated in the University of Essex. The iDorm is a real test bed used for conducting research into intelligent embedded agent technologies. The iDorm is a multi-user inhabited space that is fitted with a plethora of embedded sensors, actuators, processors, and heterogeneous networks that are cleverly concealed (buried in the walls and underneath furniture) so that the user is completely unaware

Fig. 1. The iDorm general overview (top row). The iFridge, a refrigerator with internet capabilities hosting the intelligent agents to control the iDorm (center row). Different possibilities to interface the intelligent agent: SNAP controller (left), PDA interface (center), mobile phone interface right (bottom row).

of the hidden intelligent infrastructure of the room. The iDorm looks and feels like an ordinary study/bedroom environment containing a mix of furnishings such as a bed, work desk, and wardrobe. This splits the room into areas of different activity such as sleeping, working, and entertaining. There is a standard multi-media PC that combines a flat screen monitor and a multi-media video projector which can be used for both working and entertainment.

The iDorm is built around three networks, Lonworks, 1-wire (TINI), and IP which provide a diverse infrastructure allowing the development of network independent solutions. A common interface to the iDorm and its devices is implemented through Universal Plug & Play (UPnP) which is an event-based communication middleware for allowing devices to be plug & play enabling automatic discovery and configuration. A gateway server is used to run the UPnP software devices that interface the hardware devices on their respective networks. The agent that controls the intelligent space is built on top of the low level UPnP control architecture enabling it to communicate with the UPnP devices in the iDorm and thus allowing it to monitor and control these devices.

Any networked embedded computer that can run a standard Java process can access and control the devices in the iDorm directly. Our agent system is currently embedded in an internet Fridge (iFridge) located in the iDorm as shown in Figure 1 (center). The iFridge incorporates an intelligent embedded user friendly server with touch screen capability.

Our agent could also be embedded into any other part of the environment using embedded hardware controllers (see Fig. 1, bottom) such as the Imsys technologies Simple Network Application Platform (SNAP) (left), portable computational artefacts that can monitor and control the iDorm wirelessly such as handheld PDAs (center), or mobile phones using a WAP interface which is a simple extension of a web interface (right). This integration of embedded agents and wireless communications creates the pervasive transparent infrastructure that is characteristic of an Ambient Intelligent Environment.

3. METHODS

In this section a description of the techniques selected for comparison is provided. Special attention will be paid to those methods that are relatively new or have been adapted in some way to the problem at hand. All the techniques will be run over data collected by monitoring a user in the iDorm over a period of time.

The first selected method is called AOFIS,[12] a two phase unsupervised data-driven one-pass approach for extracting fuzzy rules and membership functions from data that has shown to be adequate for the control of the devices of the target environment. We will pay special attention to this technique later. Let's say for the time being that this technique extracts from the data a set of rules represented in a multi-input/multi-output format, where all the sampled inputs and outputs are included in each rule and joined by means of the AND connective in both the antecedent and consequent parts of the rules (see Fig. 2).

If x_l is A_1 AND ... AND x_n is A_n
THEN y_1 is B_1 AND ... y_m is B_m
$x_1, \ldots, x_n$: Input Variables
$y_1, \ldots, y_m$: Output Variables
$A_1, \ldots, A_n$: Labels for Input
$B_1, \ldots, B_m$: Labels for Output

Fig. 2. Multi-input/multi-output rule format.

We have selected as a second technique Genetic Algorithms (GA),[13,14] a popular method used over the last decade to learn or tune fuzzy rule based systems.[10] We have applied a modified version of a state of the art approach[15] in order to learn both fuzzy rules and membership functions for the FLC. We have considered a multi-input/multi-output format for the representation of the rules like that used in AOFIS (see Fig. 2).

Going deeply into the application of genetic algorithms to the generation of FLC for the agent, we have applied a previously used technique[16] based on Genetic Programming (a Genetic Algorithm Program, GA-P).[17] A remarkable characteristic of this approach is that the evolution is based on similar principles than those employed in the GA-based approach, but a different format and representation for the rules, described next, is used. This way we are able to compare not only methods for the generation of controllers but also rule formats.

Besides the GA-P-based technique, we have applied simulated annealing optimization principles to evolve the same structures as that used in GA-P, applying so a SA-P[18] technique to learn the FLC for the agent. The idea is to check if the evolutive scheme used in the GA-P approach can be improved with other techniques.

We have also compared the performance of these techniques with ANFIS,[11] a well known method for the extraction of fuzzy controllers from data.

Lastly, we have considered a data-based learning technique based on MLP neural networks[19] in order to know if a different internal structure for the agent controller than that based on FLC could be of interest.

3.1. AOFIS

The Adaptive Online Fuzzy Inference System (AOFIS)[12] technique is an unsupervised data-driven one-pass approach for extracting fuzzy rules and membership functions from data. Moreover, this technique is capable of incrementally adapting an initially provided/learnt rule base online as new data samples are provided. AOFIS has been previously proposed and tested for the control of intelligent environments like the proposed in this paper, showing good performance.

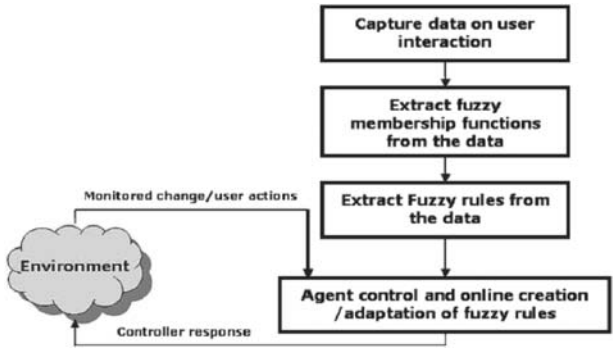

Fig. 3. Flow diagram showing the phases of AOFIS.

Besides an initial data gathering phase, AOFIS comprises of three fundamental phases (see Fig. 3):

(1) *Fuzzy membership function extraction.* AOFIS begins by categorizing the accumulated input/out put data into a set of fuzzy membership functions which quantify the raw crisp values of the inputs and outputs into linguistic labels such as 'normal,' 'low,' or 'high.' A Double Clustering approach[20] combining Fuzzy-C-Means and hierarchical clustering is used for extracting gaussian fuzzy membership functions from data. AOFIS needs in advance a parameter specifying the maximum overlap allowed between two adjacent fuzzy sets.

(2) *Fuzzy rule extraction.* The defined set of membership functions is combined with the existing data to extract fuzzy rules. This process is based on an Enhanced version of the Mendel Wang (MW) method,[21] a one pass technique for extracting fuzzy rules from data. As a consequence of the applied approach, AOFIS extracts a set of multi-input multi-output rules (see Fig. 2) which describe the relationship among every input and output variable.

(3) *Online adaptation.* AOFIS can also be used to incrementally adapt the fuzzy controller rule base when new data instances are provided. If the new data instance enters a conflict with rules in the rule base (controller actions contradict the expected actions specified in the new data instance), fired rules are selected and their consequents are replaced with the consequent membership functions that give the highest membership values to the values specified in the new data instance. If the behavior dictated by the new data instance is not enclosed in rules in the rule base (none of the rules fire using the combination of inputs included in the data instance), AOFIS generates new rules using the process mentioned in the second phase and adds them to the rule base.

3.2. GA-Based Approach

Genetic algorithms[13,14] are iterative stochastic optimization methods premised on the evolutionary ideas of natural selection and genetics. They emulate evolutionary processes in natural systems, specifically those that follow the principles of survival of the fittest.

The GA used in this paper is based on an adapted version of a Pittsburgh-based GA presented in Ref. [15] that would learn both the membership function parameters and rules to form the FLC.

This GA used multi-chromosome representation (see Fig. 4). Numerical vectors represent the numerical parameters of the membership functions (one vector for centers and one for standard deviations as we have chosen to use gaussian membership functions). This way the GA learns the number of rules and the rules itself. Each single rule is represented by means of a vector containing the linguistic labels associated with each input and output in the rule. As we are using the multi-input/multi-output format employed in AOFIS, no linguistic label can be empty.

The GA evolves the membership functions and rules together. It uses a modified two phase crossover operator which will affect the parameters for membership functions or the rules based on predefined probabilities. Classical real value based crossover is applied to evolve the numerical coefficients. One point crossover operator is applied to evolve rules with the single condition that a valid set of rules is generated after its application (the variable length chromosome generated after the application of the operator must contain an integer number of rules; we do not admit

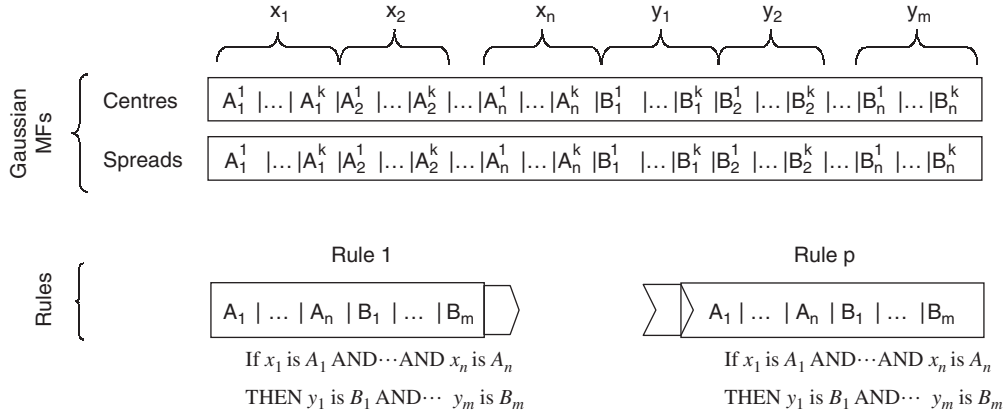

Fig. 4. Representation used in the GA-based approach.

```
Algorithm GA/GA-P is
    Generate random population
    Calculate fitness
    While not finished do
        Generate a new population by
            Selecting individuals from the actual population
            Applying genetic operators to generate new individuals
        Calculate fitness for the new population
        Replace the current population with the new population
    EWhile
    Result is the best individual found during the evolution
EAlgorithm
```

Fig. 5. GA/GA-P pseudocode.

the operator to generate incomplete rules into the chromosome). Mutation operators work in a similar manner. They are applied over the centers, spreads or rule vectors based on predefined probabilities. In all cases, the mutation is defined as the crossover between the selected individual and a randomly generated one.

We used the classical generational approach (see Fig. 5) for the evolution with a selection process based on tournament selection. We run the genetic algorithm for a limited number of generations and we select the best individual found during the evolution as the result of the search process.

3.3. GA-P-Based Approach

GA-P[17] algorithms are based on Genetic Programming (GP) algorithms.[22,23] In GP individuals to be evolved are usually represented by a tree, where the leaf nodes (terminals) usually represent input variables and the internal nodes represent functions that transform the input information to provide the output. The problem arises when the individual is dependant on numerical coefficients, which must be learnt in the canonical approach by complex operations with the functions and the terminals (see Fig. 6, left). An initial attempt to solve it is to include terminals containing numerical parameters and employing special operators to evolve such special kind of terminals[22] (see Fig. 6, centre).

GA-P algorithms[17] mix ideas from genetic algorithms and genetic programming. Individuals in GA-P are composed of two parts: a tree and a vector of numerical parameters (see Fig. 6, right). Terminals never store numerical values. When needed, they store linguistic identifiers that act as pointers to positions in the vector of numerical parameters. Initially applied in symbolic regression problems, GA-P has shown to surpass GP when the structure to evolve is dependant of a relatively high number of numerical coefficients.

In GA-P, both the tree and the vector of coefficients are evolved during the process in the basis of the usual genetic operators (reproduction, crossover, and mutation). This way, a parallel search in both the structural and parametrical components is made.

We have adapted the canonical GA-P to the generation of fuzzy logic controllers suited for our purposes. On the basis of previous work,[16] a grammar (see Fig. 7) has been declared to define the genotype of the controller for a single output. It is defined from a set of M inputs (m analog inputs and m' binary inputs). From our purposes, the controller will be a valid chain from the context free grammar defined by the shown production rules, where $a_1, \ldots, a_m$ are analog inputs, $d_1, \ldots, d_{m'}$ are binary inputs, and p is the number of linguistic labels for any analog variable, either input or output (a generalization that could allow to specify a different number of linguistic partitions for different variables is straightforward. In this work we will define the same number of partition for every analog variable).

As shown, fuzzy membership functions for analog input variables are represented by gaussians, taking the limits (centre $v\_a_i^k$ and spread $s\_a_i^k$) from the CONSTANTS vector. Membership function for binary input variables are defined by means of crisp functions *zero* () or *one* (), that return a membership value of one if the associated input variable takes the proper value.

Fuzzy membership functions for the outputs are defined by singletons. If the output variable is binary, singletons are not evolved taking the prefixed values 0.25 and 0.75. If the output variable is analog, singletons are included in the CONSTANTS vector (k_o^j) and are evolved together with the rest of numerical parameters.

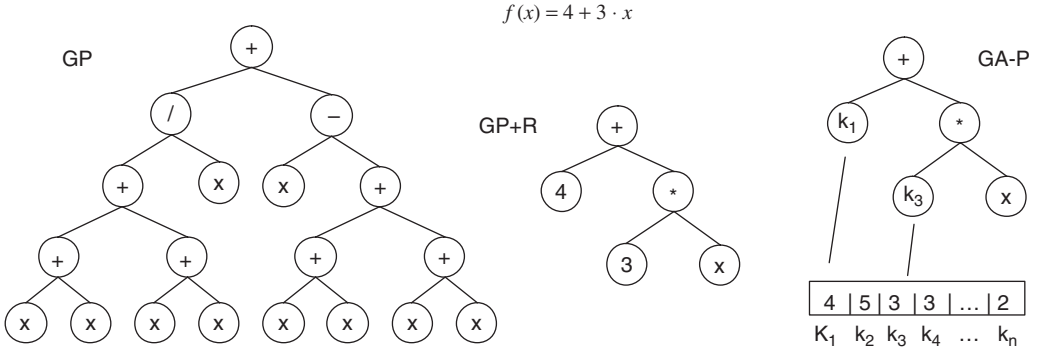

Fig. 6. Different representation in GP/GA-P.

CONTROLLER:	if CONDITION then (y is 0.25) if CONDITION then (y is 0.75) | if CONDITION then (y is k_o^1) if CONDITION then (y is k_o^2) … if CONDITION then (y is k_o^p)
CONDITION:	ASSERT_a_1 |... | ASSERT_a_m | ASSERT_d_1 |... | ASSERT_$d_{m'}$ | CONDITION OR CONDITION | CONDITION AND CONDITION
ASSERT_a_i:	gaussian(a_i, v_a_i^1, s_a_i^1) | ... gaussian(a_i, v_a_i^p, s_a_i^p)
ASSERT_d_i:	zero(d_i) | one(d_i)
CONSTANTS:	v_a_1^1,...,v_a_1^p, …, v_a_m^1,...,v_a_m^p, s_a_1^1,...,s_a_1^p, …, s_a_m^1,...,s_a_m^p, k_o^1, k_o^2 , .., k_o^p

Fig. 7. Grammar for the fuzzy logic controller of a single output.

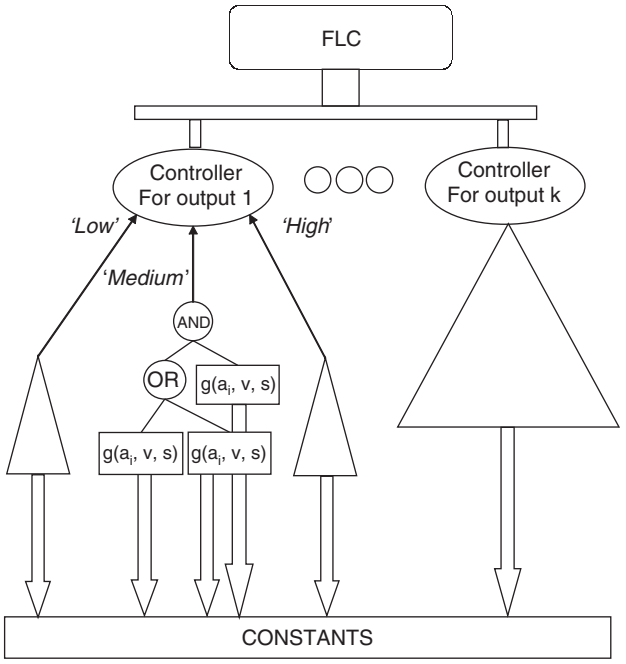

Fig. 8. Representation of a controller derived from the grammar.

The derivations of CONTROLLER used depend on the class of the output variable. The first two have only sense when the output variable is binary, and the rest are used when evolving a controller for an output analog variable.

The FLC for the agent, possibly used to control more than one output variable, will be composed of a controller derived from the grammar for every output variable. They all share the same membership functions described by the parameters in the CONSTANTS vector, so sharing the same definition for the linguistic labels, giving this way the same linguistic sense to all the rules in the controller.

The implementation of an individual departing from the grammar is straightforward. Figure 8 shows a high level representation of a FLC. A controller is attached to every output. Within these controllers, a branch is generated for every partition of the output (as dictated by the production rules in CONTROLLER), containing the rules that activate such partition of the output. These rules are composed (as dictated by the production rules from CONDITION) of AND-OR associations of either gaussian membership functions attached to specific analog input variables or *zero* () or *one* () crisp functions attached to binary input variables (as shown in the grammar in the production rules ASSERT_).

The numerical value of the FLC for an output is calculated using the following equation:

$$\frac{\sum_{j=1}^{p} w_j k_o^j}{\sum_{j=1}^{p} w_j}, \quad \text{where} \begin{cases} w_j \text{ is the activation value for branch } j \\ k_o^j \text{ is the singleton value for partition } j \end{cases}$$

Using this representation, it is straightforward to attach linguistic labels to ASSERT_ derivations and so to input variable values. For example, if you think about three partition for analog input variable i $(p = 3)$, *gaussian* $(a_i, m\_a_i^1, s\_a_i^1)$ means 'low,' *gaussian* $(a_i, m\_a_i^2, s\_a_i^2)$ means 'medium,' and *gaussian* $(a_i, m\_a_i^3, s\_a_i^3)$ means 'high.' If you think about binary input variable j, *zero* (a_j) means OFF and *one* (a_j) means ON.

It is also straightforward to attach linguistic labels to output variables. Labels (for instance 'low,' 'medium,' and 'high') are assigned to k_o^j values.

A highly relevant question for our analysis is to empathize that not all the inputs need to be included in the antecedent of one rule: it depends on the length of the branch that is variable and on the causalities of the evolution (terminals can be repeated and even contradictory). It is also relevant that the number of rules to activate a partition of an output depends on the number of OR operators included in the branch: the higher the number of OR operators, the higher the number of rules (see Fig. 9).

It can be easily seen how we are using a completely different format for the rule base and for the rules themselves than that proposed in the two former approaches: AOFIS and GA-based.

We have applied the classical generational approach for the evolution (see Fig. 5). Reproduction, crossover,

Rule:
If A is A_1 AND B is B_1 or C is C_1
THEN O is k_0

Is equivalent to:
If A is A_1 AND B is B_1
THEN O is k_0
If C is C_1
THEN O is k_0

Fig. 9. The number of rules in GA-P/SA-P representation depends on the number of OR operators.

and mutation operators are applied to evolve the population based on predefined probabilities of application.[24] We apply subtree crossover for the rules and real value based crossover for the numerical coefficients of the membership functions. We define mutation as the crossover of an individual with a randomly generated one. We run the algorithm for a predefined number of generations and we select the best individual found during the evolution as the result of the process.

3.4. SA-P-Based Approach

Simulated Annealing (SA)[25] is a well known optimization method based on an emulation of the way metals recrystalize in a process of annealing. It is basically an iterative process where a solution is randomly modified (mutated in the jargon) in each iteration. If the result of the mutation is better than the current one, the new solution overwrites it for the next iteration. If it is worse, it is usually discarded. Nevertheless, an opportunity is given to such worse solutions to overwrite the actual one, and then be used for the next iteration, based on a probability that is usually progressively reduced as the search goes on. The habitual implementation of this procedure is based on a numerical parameter known as temperature. Initialized to a specific value, it is reduced as the iterations goes on using usually an exponential policy. The probability of acceptance of a worse solution than the current one depends on the actual value of the temperature.

An hybridisation between simulated annealing and genetic programming has been proposed in the literature to apply the approach to the optimization of tree based structures.[18] The idea is to use the operator of mutation proposed in GP under the SA scheme of optimization. Also, when individuals are composed of a structural component (usually a tree) and a parametric component (usually a vector of parameters), the representation and mutation operators proposed in GA-P can be applied under simulated annealing evolution principles to produce the hybrid SA-P algorithms.

In this paper, we have followed this approach to evolve FLC for the iDorm. Details of the algorithm are the same as those described for the GA-P (representation and genetic operators are identical) but for the scheme of evolution, based now on SA principles (see Fig. 10).

3.5. ANFIS and MLP Approaches

At last, we have compared the results of the previously described techniques with those obtained using the well known ANFIS (Adaptive-Network-based Fuzzy Inference Systems)[11] and MLP (Multilayer Perceptron) neural networks[19] techniques. We are not paying attention to these techniques in this paper as they are quite popular and they have been applied as proposed in the canonical approaches, without any modification. Parameterization

```
Algorithm SA-P is
  Generate random individual
  Initialize temperature
  While not finished do
    Create new individual (mutate current individual)
    Replace current individual with new individual if:
      - It is best
      - A random experiment function of temperature
      returns true
    Decrease temperature
  EWhile
  Result is the best individual found during the evolution
EAlgorithm
```

Fig. 10. SA-P pseudocode.

and details of their application will be given when necessary in the experimental section.

4. EXPERIMENTAL RESULTS AND DISCUSSION

In this section a detailed description and discussion of the results of a set of unique experiments using real data collected from a user inhabiting the iDorm for three consecutive days is included. We begin describing data collection and preparation processes, followed by the details of application of each technique. The final subsections include quantitative and qualitative analysis of the results.

4.1. Data Collection and Preparation

Data for the experiments is collected from the iDorm in such a way that whenever the user changes actuator settings a 'snapshot' of the current inputs (sensor states) and the outputs (actuator states with the new altered values of whichever actuators were adjusted by the user) is stored. These 'snapshots' are accumulated over a period of time (three days in the case of our experiments) so enough information is provided to the technique about the user's interactions within the environment. The dataset obtained from the iDorm during the monitoring phase was comprised of 408 instances.

Seven inputs (internal light level, external light level, internal temperature, external temperature, chair pressure, bed pressure, and time measured as a continuous input on an hourly scale) and ten outputs (four variable intensity spot lights, a desk and bed side lamps, window blinds, a heater and two PC based applications comprising of a word processing program and a media playing program) were sampled for the experiments.

We have decided to use the ASRMSE value for comparison, calculated as follows: first, using the test sample, we calculate for each output the RMSE (Root Mean Square Error) between the values proposed by the learning approach and the real values selected from the user included in each data instance. In a second step, SRMSE (Scaled Root Mean Square Error) is calculated, where RMSE values are scaled to account for the different ranges

of variation of the output variables. As a final step, all the SRMSE for the ten outputs are averaged, giving the ASRMSE value we are going to use for comparison.

In order to make the experiments as independent as possible of the data configuration, the 408 available samples were randomized into six samples, and each of them was then split into a training and test set consisting of 272 and 136 instances, respectively. Learning techniques were independently applied using the data contained in the six samples.

4.2. Configuration of the Experiments

AOFIS, GA, GA-P, and SA-P based approaches were applied such that analog variables were partitioned into three partitions, as it favours the linguistic interpretability of solution permitting to divide such spaces into the linguistic labels 'low,' 'medium,' and 'high.'

AOFIS needs an extra parameter to specify the degree of overlap allowed between adjacent fuzzy sets, which were fixed at 0.5.[12]

The GA-based technique was run for 500 generations involving a population of 500 individuals. Crossover and mutation were applied with a probability 0.9 and 0.05, respectively. They were applied over the rules, centers, or standard deviation parts of the individuals after a random experiment with a predefined probability of 0.5 for the first and 0.25 for each of the numerical parameter parts. Tournament selection involving five individuals was used for each selection process. We have limited the number of rules in each controller to 200.

The GA-P-based technique was also run 500 generations involving again a population of 500 individuals. Probabilities used for the genetic operators were 0.9 for crossover and 0.05 for mutation. Both operators were applied over the structure or numerical parts of the individuals with an equal probability of 0.5 for each. Tournament selection involving five individuals was also used for each selection process. The maximum depth specified for each branch of the rule structures was fixed at 5.

The SA-P based approach was run such that the number of structures evaluated was similar as in the GA-, and GA-P-based approaches (about 250000), giving this way similar opportunities to all the approaches to find good controllers. The mutation operator affects the structural or parametric components of the solution with an equal probability of 0.5. Initial temperature was fixed at 0.000001 and progressively reduced using a factor of 0.9999.

In ANFIS we used subtractive clustering to generate an initial TSK-type fuzzy inference system. Back propagation was used to learn the premise parameters while least square estimation was used to determine the consequent parameters. An iteration of the learning procedure consists of two parts where the first part propagates the input patterns and estimates optimal consequent parameters through an iterative least squares procedure. The second part uses back propagation to modify the antecedent membership functions. The MLP back-propagation neural network was set up using a single hidden layer.

ANFIS and MLP results were obtained after ten independent experiments where the cluster radii and the number of hidden networks (unknown parameters in advance) were varied respectively between 0.3 and 0.9 with a step of 0.2 and 2 and 400 with a variable step.[12] The best result obtained was selected as representative of the performance of the technique.

4.3. Analysis of Results

In the following subsection we will analyze and discuss the results of the experiments in terms of several relevant factors such as the accuracy of each technique, the linguistic interpretability of the obtained FLC and other particularities such as the required speed of convergence or some interesting features of particular approaches. We will use the results of the first analysis (accuracy) to discard some of the techniques for further analysis.

4.3.1. Accuracy

Figure 11 shows quantitative results of the compared techniques, where the ASRMSE has been averaged over the six sample partitions used.

We are going to take as a basis for comparison the results obtained from AOFIS, a previously proposed technique that exhibited a good performance in the target environment. We can see how the GA approach provides remarkably worse results. But when GA-P is applied errors are dramatically reduced.

GA-P evolution is not very different from GA evolution (see Fig. 5): a population of rules and membership function parameters are evolved by means of genetic operators; these operators are applied to population individuals in a random fashion with a bias to the most fitted individuals; etc. So, to explain the difference in the FLC performance we must search in other directions. The main difference among both approaches is the rule format. On the one hand, GA searches for rules in the multi-input/multi-output format used in AOFIS (every input and every output is present in each rule). On the other hand, GA-P learns a

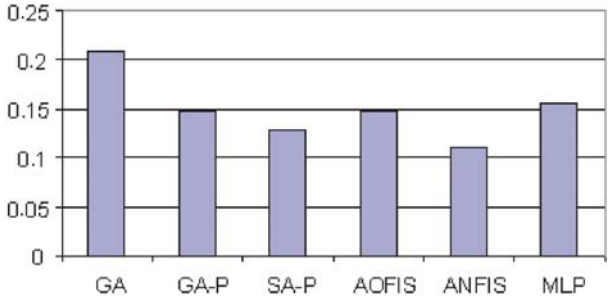

Fig. 11. ASRMSE for all the compared methods. Averaged results over six sample partitions.

set of rules for each partition of each output variable, but not all the input variables are mandatory included in the antecedents of such rules.

Moreover, the GA approach needs to learn the number of rules by means of a concatenation of rules, whilst the GA-P changes the number of rules by the use of the OR connective in the antecedent. These features seem to improve the GA-P search as it has more freedom for searching the space of candidate FLC.

GA-P and AOFIS performances are similar, but errors can still be reduced with the application of SA-P. It can be explained as SA-P is more capable of exploring the search space.

It can also be seen from the results (Fig. 11) how MLP errors are in the higher level of all the tested approaches while ANFIS is the most accurate approach.

In the basis of these results, we are going to limit the following analysis to ANFIS, AOFIS, and SA-P. We will not pay attention to GA as they have shown to be quite inaccurate and to GA-P, as its accuracy has been improved using SA-P, being very similar the foundations of both techniques. We will neither pay attention to MLP, as the motivation for comparing the results with this technique was to contrast FLC with other structures for the agent, showing quantitative results how MLP does not clearly beat them and so accepting FLC as a good candidate for the agent implementation.

4.3.2. Linguistic Interpretability

While quantitatively the most accurate approach, ANFIS presents two relevant drawbacks from the linguistic interpretability point of view that make it useless for our purposes:
(i) ANFIS generates TSK FLC, where the consequent parameters are represented as either linear functions of the input variables or constant values, rather than linguistic variables. Thus linguistic interpretation of consequents of rules is not easy.
(ii) ANFIS only learns a multi-input single-output (MISO) FLC. For the experiments described in this paper, ANFIS was independently applied for each output variable and therefore used different membership functions for representing the same linguistic labels in the rules generated for each output. As a consequence, a sentence such as "ExternalLightLevel is LOW" could mean different things depending whether we are talking about the controller of the ceiling spotlight or a desk lamp. Thus there is no semantic consistency through the controllers for all the outputs.

AOFIS, and SA-P-based technique learn rules in a Multi-Input/Multi-Output Mamdani format which makes it more suited to be understood by a human reader. Nevertheless, the number of rules generated using AOFIS is relatively high. While strongly dependant on the data sample, we have obtained 155 rules for the 272 training instances. This high number of rules can again seriously handicap the rule base understanding. SA-P, on the other hand, apart from being one of the most accurate approaches, provides an easier to interpret set of rules. As an example, we include in Figure 12 the rules for two controllers (one binary-blind state- and one analog output—an intensity variable light point-) extracted from one of the runs of the SA-P. It can be seen how rules are easy to read and (we think) they reflect a normal behavior of a user when handling such devices.

IF (BedPressure is ON) **OR**
(ExternalLightLevel is Low *AND*
InternalLightLevel is Medium) **OR**
(ExternalLightLevel is Low *AND*
ChairPressure is High)
THEN Light1 is Low

IF (ChairPressure is OFF *AND*
ExternalLightLevel is Low *AND*
InternalLightLevel is Medium)
THEN Light1 is Medium

IF (BedPressure is ON *AND*
ExternalTemperature is Low) **OR**
(ExternalLightLevel is low)
THEN Light1 is High

IF (BedPressure is ON) **OR**
(InternalLightLevel is Low) **OR**
(ExternalTemperature is Low)
THEN Blind is OFF (closed)

IF (ExternalLightLevel is Low)
THEN Blind is ON (Open)

Fig. 12. SA-P learnt rules for two outputs.

4.3.3. Parallel Factors

Parallel but highly relevant aspects are:
(i) SA-P is slower than ANFIS, being AOFIS (a one pass method) the fastest approach.
(ii) AOFIS is able to incrementally adapt the rule base on line when necessary, while the other techniques can not be easily applied to this task.

5. CONCLUSIONS

At this work we have tested several techniques for the off-line generation of fuzzy controllers for an intelligent environment, being AOFIS and SA-P the most adequate approaches. Nevertheless, we have shown that there are significant differences between them, and so further considerations are necessary to settle which one is the best technique.

AOFIS provides controllers that are quite good in terms of accuracy at modeling the user behavior. It is a one pass approach and so it learns FLC very fast. Rules in AOFIS are Mamdani formatted. So, linguistic interpretability could be possible as rules are expressed in

human language. But actually the high number of rules usually obtained using AOFIS makes it difficult to reach such goal, being this the main drawback of the technique.

The SA-P-based approach seems to be a serious opponent, as linguistic interpretability of solutions is easier and the accuracy of the learnt controllers is even better. But SA-P is much slower than AOFIS, employing several minutes to converge in contrast to a few seconds necessary to apply AOFIS. Nevertheless, the speed of convergence would not be a problem if we focus the attention in the offline generation of controllers: a few minutes to train could be admissible. So, from this point of view, we could settle that SA-P is the most adequate approach for the initial phase of off-line learning.

A different situation arises if we think about a method to be used during the two phases of the cycle of life of the agent: the initial learning phase and the subsequent phase where online control and adaptation to changing preferences is faced. In such situation, it is highly relevant that AOFIS provides built in mechanisms to online adapt the rule base when inconsistencies between the controller decisions and the user desires are detected. The others, including SA-P, do not. The use of SA-P to solve such circumstances could be to retrain the FLC using all the data instances collected over time, making it useless as an alternative for online life-long learning in ambient intelligent environments for questions of data storage and processing power requirements.

As a conclusion, we can settle that AOFIS is the best alternative to be used for online life-long learning and control in ambient intelligent environments. Nevertheless, a potentially promising alternative could be to use SA-P to initially learn the FLC in the first phase and then applying similar concepts to that used in AOFIS to re-adapt FLC over time during the online control. We defer these considerations for future work.

References

1. K. Ducatel, M. Bogdanowicz, F. Scapolo, J. Leijten, and J. C. Burgelman, Ambient intelligence: From Vision to Reality, IST Advisory Group Draft Report to the European Commission **(2003)**.
2. M. Trivedi, K. Huang, and I. Mikic, Dynamic context capture and distributed video arrays for intelligent spaces. *IEEE Transactions on Systems, Man and Cybernetics. Part A—Systems and Humans* **(2005)**, Vol. 35, pp. 145–163.
3. T. Barger, D. Brown, and M. Alwan, Health-status monitoring through analysis of behavioral patterns. *IEEE Transactions on Systems, Man and Cybernetics. Part A—Systems and Humans* **(2005)**, Vol. 35, pp. 22–27.
4. K. Eng, R. Douglas, and P. Verschure, An interactive space that learns to influence human behavior. *IEEE Transactions on Systems, Man and Cybernetics. Part A—Systems and Humans* **(2005)**, Vol. 35, pp. 66–77.
5. G. Abowd, M. Eibling, G. Hunt, and H. Lei, Context aware computing. *IEEE Pervasive Computing* **(2002)**, Vol. 1, pp. 22–23.
6. B. Eisenberg, People-Centric Computing: MIT's Project Oxygen, Pacific Connection Web Site **(2002)**, http://www.gihyo.co.jp/magazine/SD/pacific/SD_0210.html
7. M. Coen, Design principles for intelligent environments. *Presented at 15th National Conference on Artificial, Intelligence*, Madison, Wisconsin **(1998)**.
8. SONY Interaction Laboratory. Interaction Lab, http://www.csl.sony.co.jp/IL/index.html
9. U. Rutishauser, J. Joller, and R. Douglas, Control and learning of ambience by an intelligent building. *IEEE Transactions on Systems, Man and Cybernetics. Part A—Systems and Humans* **(2005)**, Vol. 35, pp. 121–132.
10. O. Cordon, F. Gomide, F. Herrera, F. Hoffmann, and L. Magdalena, Ten years of genetic fuzzy systems: Current framework and new trends. *Fuzzy Sets and Systems* **(2004)**, Vol. 141, pp. 5–31.
11. J. JANG, ANFIS—Adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man and Cybernetics* **(1993)**, Vol. 23, pp. 665–685.
12. F. Doctor, H. Hagras, and V. Callaghan, A fuzzy embedded agent-based approach for realizing ambient intelligence in intelligent inhabited environments. *IEEE Transactions on Systems, Man and Cybernetics. Part A—Systems and Humans* **(2005)**, Vol. 35, pp. 55–65.
13. J. Holland, Adaptation in Natural and Artificial Systems, MIT Press **(1992)**.
14. D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley Longman Publishing Co., Inc. **(1989)**.
15. L. Magdalena and F. Monasterio, A fuzzy logic controller with learning through the evolution of its knowledge base. *Int. J. Approx. Reasoning* **(1997)**, Vol. 16, pp. 335–358.
16. A. Lopez, L. Sanchez, F. Doctor, H. Hagras, and V. Callaghan, An evolutionary algorithm for the off-line data driven generation of fuzzy controllers for intelligent buildings. *Presented at IEEE SMC: International Conference on Systems, Man and Cybernetics*, The Hague, The Netherlands **(2004)**.
17. L. Howard and D. Dangelo, The GA-P-A genetic algorithm and genetic programming hybrid. *IEEE Expert-Intelligent Systems and Their Applications* **(1995)**, Vol. 10, pp. 11–15.
18. L. Sanchez, I. Couso, and J. Corrales, Combining GP operators with SA search to evolve fuzzy rule based classifiers. *Information Sciences* **(2001)**, Vol. 136, pp. 175–191.
19. S. Haykin, Neural Networks: A Comprehensive Foundation, Prentice Hall, PTR **(1994)**.
20. G. Castellano, A. M. Fanelli, and C. Mencar, Generation of interpretable fuzzy granules by a double-clustering technique. *Archives of Control Sciences. Special Issue on Granular Computing* **(2002)**, Vol. 12, pp. 397–410.
21. L. Wang and J. Mendel, Generating fuzzy rules by learning from examples. *IEEE Transactions on Systems, Man and Cybernetics* **(1992)**, Vol. 22, pp. 1414–1427.
22. J. R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press **(1992)**.
23. W. Banzhaf, F. Francone, R. Keller, and P. Nordin, Genetic Programming: An Introduction on the Automatic Evolution of Computer Programs and Its Applications, Morgan Kaufmann Publishers Inc. **(1998)**.
24. A. Lopez and L. Sanchez, Genetic search of block-based structures of dynamical process models. *Computational Methods in Neural Modeling, Part 1* **(2003)**, Vol. 2686, pp. 526–533.
25. S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, Optimization by simulated annealing. Neurocomputing: Foundations of Research, MIT Press **(1988)**, pp. 551–567.