In the book entitled "Soft Computing agents: New Trends for Designing Autonomous Systems", in the International Series "Studies in Fuzziness and Soft Computing", (Eds: V. Loia, S.Sessa), Springer-Verlag, Volume 75, pp. 117-145, 2002

A Soft-Computing based Distributed Artificial Intelligence Architecture for Intelligent Buildings

Victor Callaghan*, Graham Clarke*, Martin Colley*, Hani Hagras**

* Department of Computer Science, Essex University, Colchester, UK. (email: robots@essex.ac.uk)
** Department of Computer Science, University of HULL, HULL, UK, (email: h.hagras@dcs.hull.ac.uk)

Abstract:

This paper presents an innovative soft computing architecture based on a combination of DAI (distributed artificial intelligence), fuzzy-genetic driven embedded-agents and IP Internet technology applied to the domain of intelligent-buildings. It describes the nature of intelligent buildings (IB) and embedded-agents, explaining the unique control and learning problems they present. We show how fuzzy-logic techniques can be used to create a behaviour-based multi-agent architecture in intelligent-buildings. We discuss how this approach deals with the highly unpredictable and imprecise nature of the physical world in which the system is situated, and how embedded-agents can be constructed that utilise sensory information to learn to perform tasks related to user comfort, energy conservation, and safety. We explain in detail our machine learning methodology that is based on a novel genetic algorithm mechanism referred to as an associative experience engine (AEE) and present the results of practical experiments. We compare results obtained from the AEE approach to that of the widely known Mendel-Wang method. Finally we explain potential applications for such systems ranging from commercial buildings to living-area control systems for space vehicles and planetary habitation modules.

1. Introduction

The building industry use the term *intelligent*, to describe the way the design, construction and management of a building can ensure that the building is flexible and adaptable, and therefore profitable, over its full life span [Robathan, 89], embracing activities that can be unrelated to computers (eg re-configurable internal walls). The computer industry has a slightly different view, regarding *computers* as an essential component of any *intelligent-buildings*, either focusing on the added functionality derived from networks (e.g. CISCO Internet Home [Sherwin 99]) or the use of artificial intelligence techniques to provide buildings with *capabilities that are comparable to intelligence in humans.* Our work falls into this latter category; the so-called 3rd generation intelligent-buildings [Callaghan 99].

In simplified terms, an intelligent-building works by taking inputs from building sensors (light, temperature, passive infra-red, etc), and uses this to control effectors (heaters, lights, electronically-operated windows, etc) throughout the building. If this system is to be intelligent, as we conceive it, an essential feature must be its ability to *learn from experience*, and hence adapt appropriately. The notion of autonomy is important, as it implies a system that can adapt and *generate its own rules* (rather than being restricted to simple automation). Thus we have proposed a computer science definition – "*An Intelligent-Building is one that utilises computer technology to autonomously govern the building environment so as to optimise user comfort, energy-consumption, safety and work efficiency"*. In controlling such a system one is faced with the imprecision of sensors, lack of adequate models of many of the processes and of course the non-deterministic aspects of dealing with the human occupants and their needs. Such problems are well known and there have been various attempts to address them. The most significant of these approaches has been the pioneering work on behaviour-based systems from researchers such as Rodney Brooks [Brooks 91 & Luc Steels [Steels 95]] who have had considerable success in the field of mobile robots.

In our work we embed agents into computer based building devices and use networks to allow them to cooperate and be remotely accessed. We refer to these as *embedded-agents* as they are inseparably integrated into products. The computationally small footprint of computers making up building devices places severe constraints on the AI solutions that are possible. In the remainder of this paper we discuss these issues further and present one such solution based on the use of a hierarchical fuzzy-genetic agent and a hierarchical distributed agent architecture.

1.1 The Challenge of IB Control

A classical control application in a building might be a controller varying heat output in relation to a sensed temperature. Such a single parameter control system would be well suited to traditional PID, fuzzy-logic and even mechanical systems,

such as thermostats. Clearly, using an agent would be overkill if all it did were to replace a thermostat! Thus, the value of an IB agent must lie in other aspects. In our current model this is seen as residing in:

a) The agent’s ability to learn and predict a person’s needs and automatically adjust the system to meet them (we call this *particularisation*).
b) The agent’s ability to do such learning and prediction based on a wide set of parameters.

This latter aspect of an IB agent is crucial. In other words an IB agent needs the ability to modify effectors for environmental variables like heat and light etc on the basis of a *complex multi dimensional input vector,* the dynamics of which we can't specify in advance. For example, an IB agent may have to contend with circumstances such as that where the agent’s actions are contested “unpredictably” by the occupant who is also reacting to the changes in the environment, thereby misleading the cause-effect learning and introducing some non-determinism into the model (see Figure 1a). We refer to this as *Human Transformation.* A more complex situation is that where an agent or person adjusting one apparently independent control loop (e.g. reducing light level) which may cause a person to change behaviour (e.g. sit down) which in turn may result in them effecting another apparently independent control loop (e.g. raising heat). We refer to this effect as human cross-coupling (see Figure 1b).

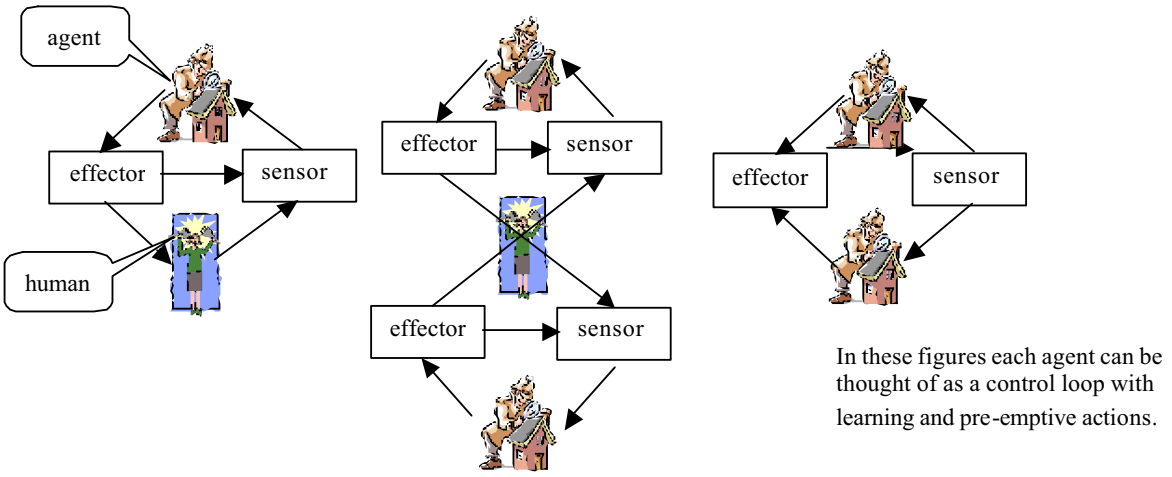

Figure (1) - Some Sources of Non-Determinism In IB based Embedded-Agents

Figure 1c illustrates a similar cross coupling as in figure 1b, except via multi agents acting within a single environment (i.e. action of agent alters world, altered world provokes a reaction from a second agent that may change world which in turn may cause reaction from original agent). Clearly agents that only look at single sensor channels are unable to take these wider issues into account. Thus, the whole control problem is made considerably more difficult by the inclusion of essentiality non-deterministic and highly individual occupants within the control loop plus the need to deal with large and highly dynamic input vectors. When viewed in such terms it is possible to see why simple controllers are unable to deal satisfactorily with the problem.

1.2 The Challenge of IB Learning

Learning in intelligent buildings mostly takes the form of identifying cause-effect relationships based on building occupant actions in response to changes in the environment. The learning mechanism needs to be able to interpret these cause-effect relationships based on the combinational states and temporal sequences of numerous input vectors. The agent uses these relationships as part of a mechanism to serve the occupant’s needs by taking pre-emptive actions. For efficient and robust learning, it is necessary to have mechanisms to identify, combine or resolve similar or contradictory events. IB based learning is focused around the actions of people. People are highly individual and to some degree idiosyncratic (i.e. partially unpredictable). Hence, to better serve individual needs, the learning needs to emphasise particularisation (as opposed to generalisation) and provide some learning inertia to cope with erratic events (e.g. erroneous or one-off actions).

Thus, for example, generalisation is used more sparingly in IB learning, to contain rule size to available resources, rather than minimising rule sets to the lowest possible number. Buildings are, by and large, occupied by people who for a variety of reasons (e.g. time, interest, skills etc) would not wish, or be able to cope with much interaction with the building systems. Thus, in general, learning should, as far as possible, be non-intrusive and transparent to the occupants (i.e. requiring minimal involvement from the occupants). IB agents are sensor rich and it is difficult to be prescriptive about which sensor parameter set would lead to the most effective learning of any particular action. Thus, to maximise the opportunity for the agent to find an optimum input vector set, whilst containing the processing overloads, the ideal agent would be able to learn to focus on a sub-set of the most relevant inputs. Thus, minimally constrained approaches, that maximise agent-learning opportunities, are favoured. Finally, the computationally compact nature of agents in IB (e.g. limited memory) has an effect that permeates all aspects of learning such as altering the extent of particularisation versus generalisation or the granularity of similarity clustering.

1.3 Why Use Robotic Techniques in Intelligent Buildings ?

At a simple level, it can be seen that modern buildings have strong physical similarities to machines, in that they contain a myriad of mechanical, electrical, electronic, computing and communications devices. As building services become increasingly sophisticated they contain ever more sensors (to gain information about the environment within the building), effectors (to make changes to conditions within the building), computer based devices (to increase automation) and networks (to facilitate remote control and more efficient management).

We contend that there are enough similarities between machines (particularly mobile robots), and buildings to justify such techniques being applied to building control systems to make them behave more intelligently. For example, both are dealing with a highly dynamic, unpredictable world, in which people and items move about, natural phenomena occur, and people may behave idiosyncratically or even irrationally. As has been shown by other researchers [Brooks 91] this situation makes it almost impossible to model the world, or to plan in advance for every possible occurrence (making traditional model-based control techniques particularly difficult to apply).

Further similarities are revealed when we consider how the intelligent mechanisms of both systems work. A machine such as a mobile robot activates effectors in response to sensor information, and in doing so moves safely and efficiently from one point in Cartesian space to another. Work at Essex has shown that buildings may be regarded as “navigating” safely

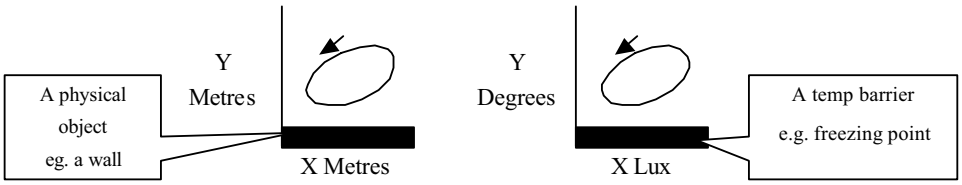

Figure (2) – Navigation Spaces

and efficiently through analogous landscapes to robots – “*sensory maps*” (s-maps*)*. To illustrate the principle, a highly simplified 2D illustration of these maps is provided in Figure (2). From this it is possible to view a building as navigating like a mobile-robot within a constrained world populated by distinctive features. In robotic machines (and by analogy in IBs), these distinctive features are used to trigger distinct behaviours, with the interaction of behaviours and distinctive features giving rise to the emergent intelligent behaviour that provides pseudo reasoning and planning. At the end of this paper, in the section devoted to future work, these “s-maps” are considered as a possible means of enabling learning without interaction with the user

Both practical and market-driven factors require building control systems to have a small computational footprint (i.e. to be small and relatively cheap). Hence in intelligent-buildings, centralised, traditional AI, with bulky planners and reasoning systems, becomes less attractive and using techniques from mobile robots techniques, offers more promise.

Thus, from the considerations above we contend that buildings might be regarded as machines or even ‘*robots that we live inside’.*

2. Distributed Architecture

As people's work or leisure is usually *room-based* (i.e. different activities take place in different rooms) and rooms are often devoted to specific purposes, we argue that both the physical and logical unit of a building is therefore a room. We have accordingly chosen to distribute control at room-level This mirrors the architect’s perception of the functionality of the building. Thus, each room contains an *embedded-agent*, which is responsible, via sensors and effectors for the local control of that room as shown in Figure (1). All embedded-agents are connected via a high level network (IP-ethernet in our case), allowing collaboration or sharing of information to take place where appropriate [Sharples 99]. Within a room, devices such as sensors and effectors are connected together using a building services network (Lontalk in our case). Internet networks (ethernet-TCP/IP) have advantages over existing building services networks in that they are much higher bandwidth (10-100Mb/s versus 1Mb/s) and are in widespread use (with attendant economies of scale). However, they generally suffer the disadvantage of being non-deterministic and having bulky computational overheads (i.e. support for functions not often needed in building service). Currently there is much work underway to bring Internet network technologies into building services. In our approach we address this dilemma by utilising a hierarchy with the agent forming the bridge between Lonworks at the lower level and IP at the higher level; thus allowing us to benefit from the best aspects of both worlds while the technology and market is developing. This DAI architecture is illustrated in Figure (3). Other work we have undertaken, beyond the scope of this paper, extends to mobile and body based agents [Callaghan 2000].

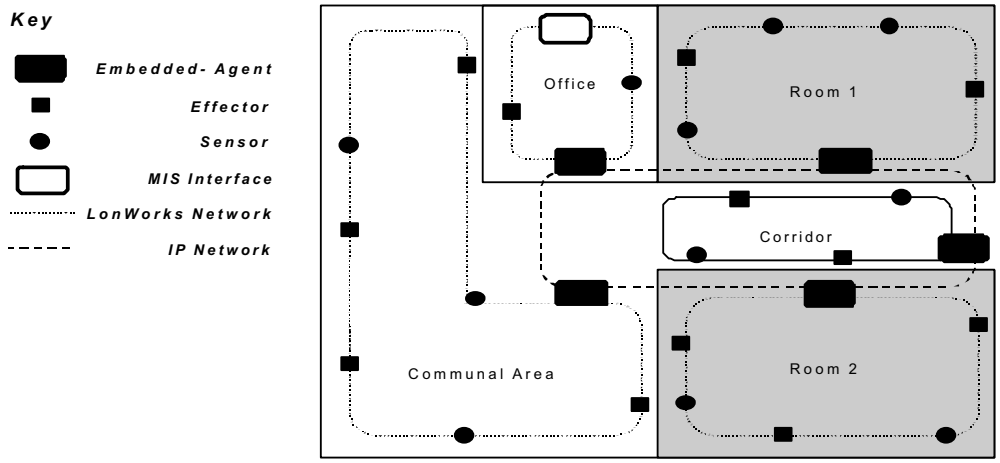

Figure (3): The DAI Building-Wide Architecture (Simplified Example)

As far as we are aware, this approach differs significantly to that adopted by other researchers working on interactive intelligent environments and related agent architectures for IB. Examples of such work [Coen 97] include research in Sweden [Davisson 98] that utilises multi-agent principles to control an Intelligent Building. Their primary goal is energy-efficiency, and although their system does adjust the heating and light level to suit individual preferences, these settings must be pre-defined. Their agents are built from traditional AI (ie not behaviour based) and their work does not address issues, such as occupant based learning. The system, so far implemented in simulation only, managed to achieve energy savings of 40% over the same building being controlled manually by occupants. A group in Colorado [Mozer 98] are using a soft computing approach - neural networks - focusing solely on the intelligent control of lighting within a building, by

anticipating when particular *zones* (regions in a room) will be occupied or unoccupied. Their system, implemented in a building with a real occupant, also achieved a significant energy reduction, although this was sometimes at the expense of the occupant’s comfort. They use a centralised control architecture which differs from our multi-agent approach and which intrinsically seeks a generalised solution rather than a *particularised* solution, as is the case for us. A third group based at the MIT Artificial Intelligence Lab in Massachusetts is working on an Intelligent Room project. They employ a mix of cameras, microphones and multiplexer to enable people to interface with room-based systems in a natural way using speech, gesture, movements, and context information [Brooks 97]. This primary focus on facility of the user interface differs to our work where ideally the agent remains more or less invisible to the user of the building.

Distributed computing, programming and communication models such as MEX [Lehikoinen 99], Java (including Jini & JavaSpaces, JAFMAS, JATLite etc) [Jeon 2000] KQML/FIPA [Labrou 99] offer important infrastructure support for distributed agent systems. Java is proving particularly popular and useful in programming such systems due to its focus on network support, in part driven by its role as a main Internet and Web programming tool. The HIVE project at MIT [Minar 99] is an example of a particularly forward-looking distributed agent model. The model differs to ours principally in respect that their agents are soft (rather than our hard embedded-agents) with access to hard devices being via coded objects referred to shadows. The soft agents reside on servers (eg PCs) which thereby de-emphasis minimalist aspects of agent design which is a central focus of our work. A particularly attractive feature of the HIVE model is the ad-hoc nature of the multi-agent agent structures supported and that it can be integrated with standard services such as Java derivatives. Whilst the HIVE work is not at a point that we could adopt it into this work, we are hoping it will evolve to become an option for us. Agent communication languages form another essential component in the overall framework. In traditional soft agent work the most widely used standards are KQML & FIPA. A consequence of being designed for non-minimal agents is that they attract a large computational footprint that makes them unsuitable for compact embedded-agents. To overcome this problem we propose using a Distributed Intelligent Building Agent Language (DIBAL) being developed at the University of Essex, that has been tailored to the needs of intelligent-building based embedded-agents. The main feature of DIBAL is that has a versatile hierarchical tagged data format, which provides highly compact representation [Cayci 2000].

3. The Embedded-Agents

Nikola Kasabov offers useful criteria for intelligent systems, based on seven requirements [Kasabov 98]. They include fast learning from large data sets, on-line incremental adaptation, accommodation of new knowledge, memory based, interaction with environment (and other systems), adequate representation and an ability to analyse their own performance. In our agent design, we aim to meet as many of these criteria as we can. The internal architecture of the embedded-agents we use is illustrated in Figure (4). It is based on the behaviour-based approach, pioneered by Brooks, and composed of many simple co-operating units [Brooks 91]. This approach has produced very promising results when applied to the control of robots, which, as we explained above, can include intelligent-buildings. Controlling a large integrated building system requires a complicated control function because it involves both a large input and output space and the need to deal with many imprecise and unpredictable factors, including people. This function can be made more manageable by breaking down the space for analysis into multiple behaviours, each of which responds to specific types of situations, and then integrating their recommendations.

3.1 Embedded-Agent Control Architecture

Our work is broadly situated within the behaviour based architecture work, pioneered by Brooks, consisting of many simple co-operating sub-control units [Brooks 91]. This has produced very promising results when applied to the control of robots [Hagras 99a, Hagras 99b, Hagras 2000a, Hagras 2000b], which we argue includes IB. We have extended this work to include a double hierarchy of behaviours (implemented as fuzzy controllers) and learning (implemented using genetic algorithms).

We use a room-based DAI decomposition with each agent having behaviours consisting of two groups of meta-functions. The first group are fixed (pre-programmed) behaviours which are not subject to adaptation and consist of; a *Safety behaviour* (this ensures environmental conditions are always at a safe level), an *Emergency behaviour* (reactions to fire, burglary etc), and an *Economy behaviour* (this ensures that energy is not wasted). The second group of meta-functions are dynamic behaviours which the system seeks to learn from the occupant based on his/her actions; the main one being a *Comfort behaviour* which attempts to set the room to a state that matches examples of the occupants previous preferences. These dynamic meta-functions have an adaptable rule base, which learns from the room occupant’s behaviour. The

management of these dynamic behaviours offers the main challenge to our research due to there numerous, dynamic, imprecise and uncertain nature.

Fuzzy logic offers a framework for representing imprecise and uncertain knowledge. It has similarities to the way people make decisions as it uses a mode of approximate reasoning, which allows it to deal with vague and incomplete information. In addition fuzzy controllers exhibit robustness with regard to noise and variations of system parameters. However, it is often difficult to determine parameters for fuzzy systems. In most fuzzy systems, the fuzzy rules were determined and tuned through trial and error by human operators. It normally takes many iterations to determine and tune them. As the number of input variables increases (IBs have very large numbers of rules due to particularisation) the number of rules increases disproportionately, which can cause difficulty in matching and choosing between large numbers of rules.

In our approach we implement each behaviour as a fuzzy process and then use higher level fuzzy process to co-ordinate them. The resultant architecture takes the form of a hierarchical tree structure form (see Figure (3)). This approach has the following advantages:

- It simplifies the design of the embedded-agent, reducing the number of rules to be determined (in previous work we have given examples of rules reduction of two orders of magnitude via the use of hierarchies).
- It uses the benefits of fuzzy logic to deal with imprecision and uncertainty.
- It provides a flexible structure where new behaviours can be added (eg comfort behaviours) or modified easily.
- It utilises a continuous activation scheme for behaviour coordination which provides a smoother response than switched schema

The learning process involves the creation of Comfort behaviours. This is done *interactively* using reinforcement where the controller takes actions and monitors these actions to see if they satisfy the occupant or not, until a degree of satisfaction is achieved. This process would be acceptable in a hotel or apartment block but would probably require the intervention of a care assistant in housing for the elderly or those with learning difficulties. The behaviours, resident inside the agent, take their input from a variety of sensors in the room (such as occupancy, internal illumination level, external illumination level, internal temperature, external temperature etc), and adjust device outputs (such as heating, lighting, blinds, etc) according to pre-determined, but settable, levels. The complexities of training and negotiating satisfactory values for multiple use rooms would depend upon having a reliable means of identifying different users.

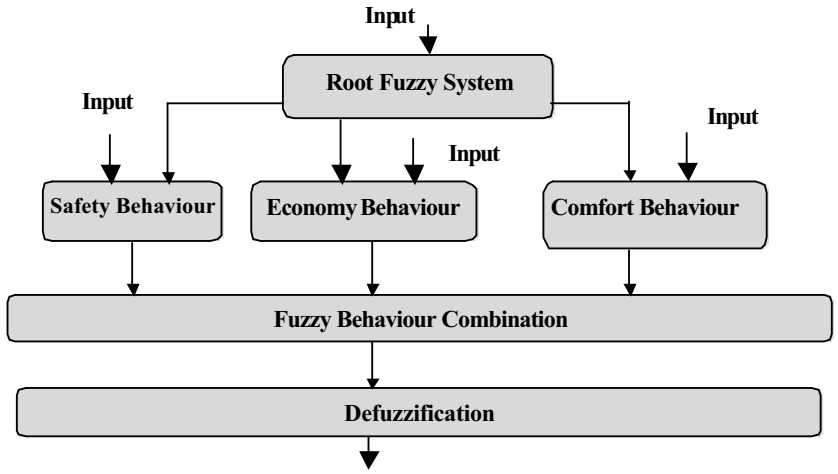

Figure (4): The Hierarchical Fuzzy System

In our prototype system each agent has six inputs made up of four environmental variables - a Room Temperature (RTEMP), the External Temperature (ONTEMP), the Room Illumination (RILLUM) and the External Illumination (ONILLUM) each of which have the fuzzy membership functions shown in Figure (5). Each input is represented by three fuzzy sets, as this was the minimum number that gave satisfactory results. The two remaining inputs to the system are a room occupancy indicator and an emergency alarm flag. The system has two outputs; Room Heater (RH) setting and a Room Illuminator (RI) setting which have the membership functions shown in Figure (6). Seven fuzzy sets were found to be the minimum needed to provide satisfactory results. Whenever an alarm input is activated the Emergency behaviour

becomes dominant and the room illumination is switch to max and heat is switched off (a pre-determined plan). The Economy behaviour is active to a fuzzy degree dependent on occupancy, outside temperature and light. Behaviour is fuzzily co-ordinated as shown in Figure (7).

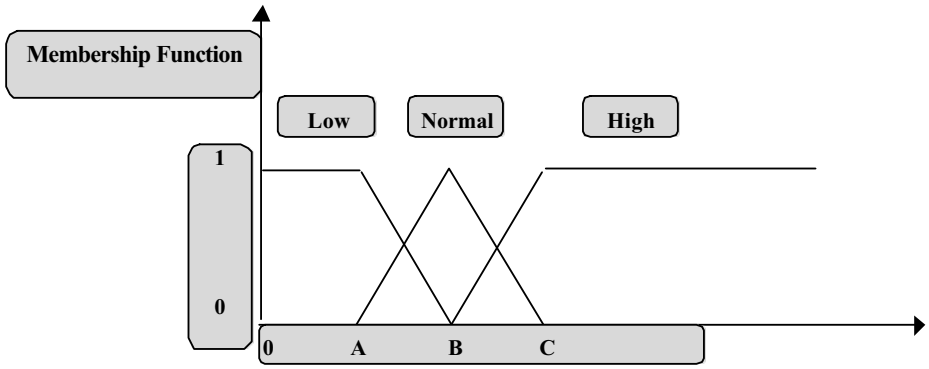

Figure (5): Input membership functions for RTEMP and ONTEMP
A= 10°, B= 20°, C= 30°, for RILLUM and ONILLUM. A= 300 Lux, B= 400 Lux, C= 500 Lux.

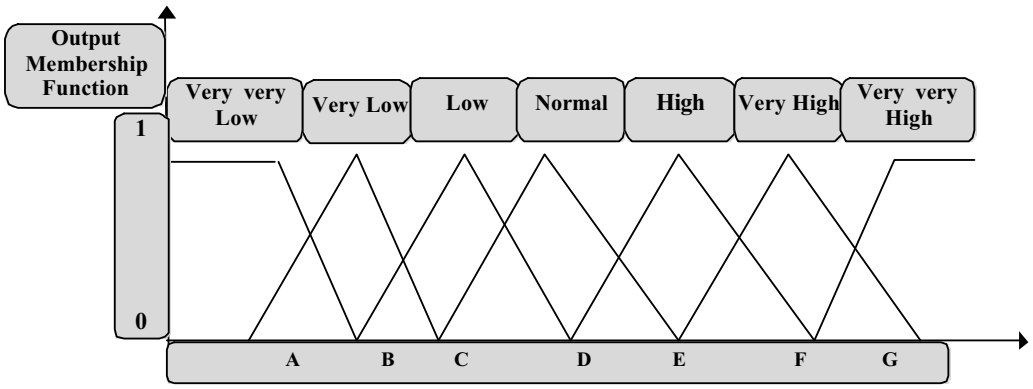

Figure (6): Output membership functions for:
RI A= 0 %, B= 20 %, C= 35 %, D=40%, E=50%, F=70%, G=100%
RH A=0%, B=30%, C=40%, D=50% , E= 70%, F=85%, G=100%.

Each behaviour uses a singleton fuzzifier, triangular membership functions, product inference, max-product composition and height defuzzification. The selected techniques were chosen due to their computational simplicity. The equation that maps the system input to output is given by:

$$Y_t = \frac{\sum_{p=1}^{M} y_p \prod_{i=1}^{G} \alpha_{Aip}}{\sum_{p=1}^{M} \prod_{i=1}^{G} \alpha_{Aip}} \qquad (1)$$

In this equation M is the total number of rules, y is the crisp output for each rule, $\ddot{\imath}_{i=1}^{g}\alpha Ai$ is the product of the membership functions of each rule input and G is the number of inputs. In this hierarchical architecture we utilise a fuzzy operator to combine the preferences of different behaviour into a collective preference. Command fusion is decomposed into two steps: preference combination, decision and in the case of using fuzzy numbers for preferences, product-sum combination and height defuzzification. The total control output C is [Saffiotti 97]:

$$\mathbf{C}= \frac{\sum_i (BW_i * C_i)}{\sum_i BW_i} \qquad (2)$$

In Equation (2) i = economy, comfort, C_i is the behaviour command output (room temperature). BW_i is the behaviour weight. The behaviour weights are calculated dynamically taking into account the context of the agent. In Figure (2) each behaviour is treated as an independent fuzzy controller, which is fuzzily combined to provide a single output, which is then deffuzzified to give the final crisp output. The fuzzy values form an input to context rules, which directly govern when, which, and to what extent behaviours are fired, depending on fuzzy membership functions in Figure (7). The default rule we use is:

IF ONTEMP IS HIGH AND ONILLUM IS HIGH AND THE ROOM IS OCCUPIED THEN ECONOMY.
IF ONTEMP IS LOW AND ONILLUM IS LOW THEN COMFORT
IF THE ROOM IS VACANT THEN RH IS LOW AND RI IS LOW

The final output is a mix of the different behaviour outputs, each weighted by the degree of its importance, and the final output is calculated using Equation (2).

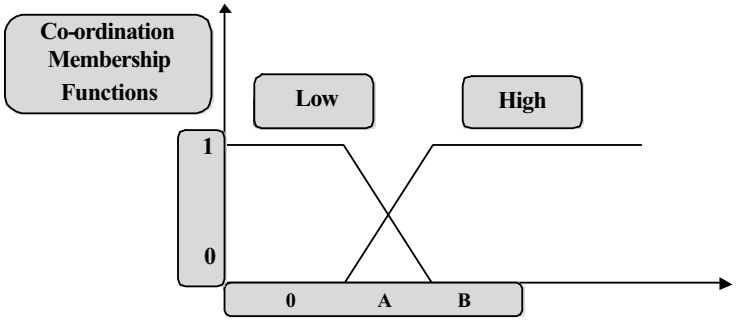

Figure (7) Co-ordination parameters for ONILLUM A= 350Lux and B= 400 Lux, for ONTEMP A= 15° and B=25°.

3.2 Embedded-Agent Learning Architecture

It is clear that, in order for an agent to autonomously particularise its service to an individual, some form of learning is essential. In our agent learning takes the form of adapting the dynamic Comfort behaviour’s rule base, according to the occupants actions. To do this we utilise an evolutionary computing mechanism based on a novel hierarchical genetic algorithm (GA) technique which modifies the fuzzy controller rule-sets through interaction with the environment and user.

The hub of the GA learning architecture is what we refer to as an *Associative Experience Engine* [British patent 99-10539.7]. Each behaviour is a fuzzy logic controller (FLC) that has two parameters that can be modified; a *Rule Base* (RB) and its associated *Membership Functions* (MF). In our learning we will modify the rule-base. The architecture, as adapted for IB embedded-agents, is given in Figure (8). The behaviours receive their inputs from sensors and provide outputs to the

actuators via the c*o-ordinator* that weights their effect. When the system fails to have the desired response (e.g. an occupant manually changes an effector setting), the learning cycle begins.

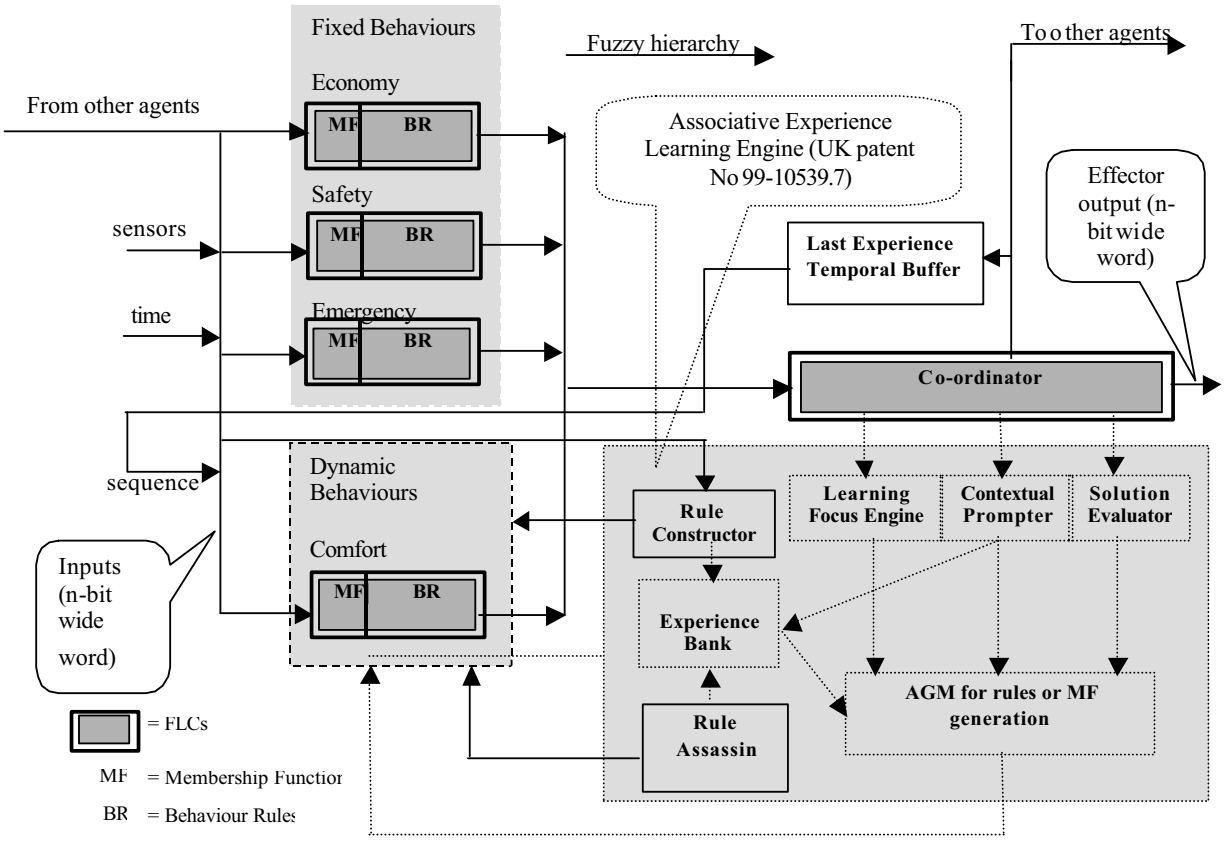

Figure (8): Embedded-Agent Architecture

When a learning cycle is initiated, the most active behaviour (i.e. that most responsible for the agent behaviour) is provided to the *Learning Focus* from the *Co-ordinator* (the fuzzy engine which weights contributions to the outputs), which uses the information to point at the rule-set to be modified (i.e. learnt) or exchanged. Initially, the *Contextual Prompter* (which gets a characterisation of the situation, an experience, from the Co-ordinator) is used to make comparison to see whether there is a suitable behaviour rule set in the *Experience Bank*. If there is a suitable experience, it is used. When the past experiences do not satisfy the occupant's needs we use the best-fit experiences to reduce the search space by pointing to a better starting point, which is the experience with the largest fitness. We then fire an *Adaptive Genetic Mechanism (AGM)* using adaptive learning parameters to speed the search for new solutions. The AGM is constrained to produce new solutions in a certain range defined by the *Contextual Prompter* to avoid the AGM searching options where solutions are unlikely to be found. By using these mechanisms we narrow the AGM search space massively, thus improving its efficiency. After generating new solutions the system tests the new solution and gives it fitness through the *Solution Evaluator*. The AGM provides new options via operators such as crossover and mutation until a satisfactory solution is achieved.

The system then remains with this set of active rules (an experience) until the occupant's behaviour indicates a change of preference (e.g. has developed a new habit), signalled by a manual change to one of the effectors when the learning process described above is repeated. In the case of a new occupant in the room the *Contextual Prompter* gets and activates the most suitable rule base from the *Experience* Bank or if this proves unsuitable the system re-starts the learning cycle above. The *Solution Evaluator* assigns each stored rule base in the *Experience Bank* a fitness value. When the *Experience Bank* is full, we have to delete some experiences. To assist with this the *Rule Assassin* determines which rules are removed according to their importance (as set by the *Solution Evaluator*). The *Last Experience Temporal Buffer* feeds back to the inputs a compressed form of the n-1 state, thereby providing a mechanism to deal with temporal sequences.

Multi-Agent operation is supported by making this compressed information available to the wider network. The compressed data takes the form of which behaviours are active (and to what degree). The general philosophy we have adopted is that data from remote agents is simply treated in the same way as all other sensor data. As with any data, the processing agent decides for itself which information is relevant to any particular decision. Thus, multi-agent processing is implicit to this paradigm, which regards remote agents as simply more sensors. We have found that receiving high level processed information from remote agents, such as “the room is occupied” is more useful than being given the low level sensor information from the remote agent that gave rise to the high-level characterisation. This is because the compressed form both relieves agent processing overheads and reduces network loading. Inter-agent communication also requires appropriate networking and programming infrastructure together with standardised agent communication languages. This is a large and complex subject beyond the scope of this paper but we refer interested readers to our work concerned with intelligent-building and agent communication languages [Cayci 2000].

From a users viewpoint the system functions interactively as follows. A user is asked to select his preference for any given programmable setting. The system then tries to adapt its rules to achieve this setting. The user is prompted to confirm or deny his satisfaction with the result. If the occupant is dissatisfied the system tries to re-adjust the rules. If the user is satisfied, the current rule set is accepted. Experiments to date show the *experience engine* achieves a satisfactory solution in a small number of iterations. Our experiments show this takes an average of twenty-one iterations. As we mentioned earlier this process would probably have to be undertaken by a care assistant for some groups of occupants. In the next section we will explain the techniques in more detail

4. The Associative Experience Learning Engine in Detail

Most automated fuzzy controller design employing conventional GAs use simulation to overcome problem of lengthy training periods caused by the testing of numerous generations of possible solutions [e.g. Fukuda 99, Linkens 95, Bonarini 96, Hoffmann 98]. As explained above, we employ a combination of domain constraints and environmental cues to reduce the search space and thereby substantially speed up the GA process, eliminating the need for simulation. The following paragraphs explain these methods in detail.

4.1 Identifying Poorly Performing Rules

The rule-base is initialised to have all the outputs switched off. The GA population consists of all the rules consequents contributing to an action, which is usually a small number of rules. As is the case for classifier systems, in order to preserve the system performance, the GA is allowed to replace a subset of the classifiers (the rules in this case). The worst m classifiers are replaced by the m new classifiers created by the application of the GA on the population [Dorigo 93]. The new rules are tested by the combined action of the performance and apportionment of credit mechanisms. We will replace all the rules that participated in this action for a given input.

In the learning phase, the agent is introduced to different situations (e.g. low temperature & illumination both inside and outside the room), and the agent, guided by the occupant, attempts to discover the rules needed for each situation. The learning system consists of learning different situations. The model to be learnt is small, as is the search space, and in each situation only small number of rules will be fired. In our agent model, control is dominated by activity physically close to the agent. Knowledge *of* remote agents (and their activity), and information *from* remote agents is less important and smaller in quantity (e.g. only information on active behaviours is passed on). The accent on local models at all levels implies the possibility of learning by focusing at each step on a small part of the search space only, thus reducing interaction among partial solutions. The interaction among local models, due to the intersection of neighbouring fuzzy sets means local learning reflects on global performance [Bonarini 96]. Moreover, the smooth transition among the different controllers implemented by fuzzy rules implies robustness with respect to data noise. Thus, we can have global results coming from the combination of local models, and smooth transition between close models. Also dividing the learning into local situations can reduce the number of learnt rules. For example, in one situation we started learning with 81 rules, and the agent discovered that during its interactive training with the occupant, it needed only 49 rules.

4.2 Fitness Determination and Credit Assignment

The system fitness is determined by the *Solution Evaluator* and is evaluated by how much the system satisfies the room occupant’s desired target value (such as desired temperature) in a specific situation and how it reduced the normalised absolute deviation (d) from the normal value. This is given by:

$$d = \frac{|normal\ \ value - \ \ deviated\ value|}{\max\ \ deviation} \qquad (3)$$

Here, the normal value will correspond to that desired by the occupant. The deviated value corresponds to the actual measured value. The maximum deviation is the theoretical maximum that can occur. Hence the fitness of the solution may be found from the difference d1 - d2, where d2 is the normalised absolute deviation before introducing a new solution and d1 is the normalised absolute deviation following the new solution. The deviation is measured using the physical sensors, which gives the agent the ability to adapt to the imprecision and the noise found in the real sensors rather than relying on estimates from previous simulations. The fitness of each rule for a given situation is calculated as follows. The crisp output Y_t can be written as in (1). If the agent has two output variables, then we have Y_{t1} and Y_{t2}. The normalised contribution of each rule p output (Y_{p1}, Y_{p2}) to the total output Y_{t1} and Y_{t2} can be denoted by S_{r1}, S_{r2} where S_{r1} and S_{r2} is given by:

$$\mathbf{S_{r1}} \quad \frac{Y_{p1}\prod_{i=1}^{G}\alpha_{Aip}}{\sum_{i=1}^{m}\frac{\prod_{i=1}^{G}\alpha_{Aip}}{Y_{t1}}}, \qquad \mathbf{S_{r2}} = \frac{Y_{p2}\prod_{i=1}^{G}\alpha_{Aip}}{\sum_{i=1}^{m}\frac{\prod_{i=1}^{G}\alpha_{Aip}}{Y_{t2}}} \qquad (4)$$

We then calculate each rule’s contribution to the final action $S_c = \frac{S_{r1} + S_{r2}}{2}$. Then the most effective rules are those that have the greatest values of S_c. The fitness of the rule in a given solution is supplied by the *Solution Evaluator* and is given by:

$$S_{rt} = \text{Constant} + (d_1 - d_2)\ .S_c \qquad (5)$$

$d_1 - d_2$ is the deviation improvement or degradation caused by the adjusted rule-base produced by the algorithm. If there is improvement in the deviation, then the rules that have contributed most will be given more fitness to boost their actions. If there is degradation then the rules that contributed more must be punished by reducing their fitness w.r.t to other rules giving other useful actions an opportunity to produce better solutions.

4.3 Memory Based Mechanisms

Zhou [Zhou 90] presented the CSM (Classifier System with Memory) system that addressed the problem of long versus short-term memory (i.e. how to use past experiences to ease the problem solving activity in novel situations). Zhou’s approach was to build a system in which short and long-term memory are simultaneously present. The short-term memory is simply the standard set of rules found in every learning classifier system (the fuzzy rule base in our case). The long-term memory is a set of rule clusters, in which every rule cluster represents a generalised version of problem solving expertise acquired in previous problem solving activity. Each time the agent is presented a problem it starts the learning procedures trying to use long-term experience by means of an appropriate initialisation mechanism. Thereafter, the system works as a standard classifier system (except for some minor changes) until an acceptable level of performance has been achieved. It is at this point that a generalising process takes control and compresses the acquired knowledge into a cluster of rules that are stored for later use in the long-term memory.

In our system, when the agent begins learning it has no previous experience and the *Experience Bank* is empty. But as it begins GA enabled learning, it begins filling the memory with different rule bases, each associated to different users. Each

stored rule base consists of rules and the actions (consequents) that were learnt by the GA. With a new user, after monitoring the user’s action for a period the agent matches the rules fired during this time to sets of rule bases for different users stored in the *Experience Bank.* The system tries to identify which rule base is appropriate to the user on the basis of actions taken by him during this time, and the rule base containing the most similar actions to the occupants is chosen as a starting point for learning and adaptation.

Each time the agent is presented with a situation to solve, it begins checking if the consequents of firing the rules from a rule base extracted from the *Experience Bank* suits the new user or not. If these rules are suitable for the user then they are used for the Comfort behaviours. If some actions are not suitable for the user, the system begins identifying the poorly performing rules as described in Section (4.1), then it fires the Adaptive Genetic Mechanisms (AGM) to change these rules. This action helps to speed up the genetic search as it starts from the search from the best known point in the search space instead of starting randomly. In this way the system does not need the “matcher calculations” used by [Zhou 90]. This is because we do not use the binary message coding, or “don’t care”, conditions but instead utilise perfect matches; hence we don’t need the generaliser. The clusters are arranged in a queue starting from the most recent experiences.

Problems occur as the system begins accumulating experience that exceeds the physical memory limits. This implies that we must delete some of the stored information as the acquired experience increase. Clearly not all experiences are of equal value. One that are frequently used or difficult to learn are clearly of more value that others. Thus, for every rule base cluster we attach a *difficulty counter* to count the number of iterations taken by the agent to find a suitable rule base for a given user, we also attach a *frequency counter* to count how often they have been retrieved. The *degree of importance* of each rule base cluster is calculated by the *Experience Survival Valuer* and is given by the product of the *frequency counter* and the *difficulty counter*. This approach tries to keep the rules that have required a lot of effort to learn (due to the difficulty of the situation) and also the rules that are used frequently. When there is no more room in the *Experience Bank*, the rule base cluster that had the least *degree of importance* is selected for removal. If two rule base clusters share the same importance degree, tie breaking is resolved by a least-recently-used strategy. Thus an *age parameter* is also needed for each rule base cluster. We can also operate the *Experience Survival Valu*er in an “Assassin Mode”. In this mode it periodically, decrements the frequency counter by one (e.g. once a day etc) thereby proactively forcing death on little used rules or ageing out rules.

4.4 Producing New Solutions

If the rule base extracted from the *Experience Bank* is not suitable for the user, the GA starts its search for new solutions (i.e. new rules). The fitness of each rule in the population is proportional to its contribution in the final action. If the proposed action by the new solution results in an improvement in performance then the rules that have contributed most will have their fitness increased more than the rules that have contributed less in this situation (and vice-versa for negative results). This allows us to move away from points in the search space that cause no improvement (or even degradation) in the performance. The parents for any new solution are chosen proportional to their fitness using the roulette-wheel selection process together with genetic operations of crossover and mutation. The proposed system can be viewed as a double hierarchy system in which the fuzzy behaviours are organised in hierarchical form. The learning algorithm can also be seen as a hierarchy. At the higher level we have a population of solutions stored in the *Experience Bank*. If the stored experiences leads to a solution then the search ends. If none of these stored experiences leads to a solution then each of these experiences acquires a fitness assigned by the *Experience Assessor* that finds how many rules in the stored rule-base are similar to the user’s action in the test period. At this lower level the highest fitness experience is used as a starting position for the GA.

The Adaptive Genetic Mechanism (AGM) is the rule discovery component for our system (as in the classifier system). We used Srinivas method [Srinivas 96] to adapt the control parameters (mutation and crossover probabilities). The strategy used for adapting the control parameters depends on the specification of the performance of the GA. In a non-static environment (which is our case), where the optimal solution changes with time, the GA should also possess the capacity to track optimal solutions. The adaptation strategy needs to vary the control parameters appropriately whenever the GA is not able to find the optimum. It is essential for GAs to have two characteristics for optimisation. The first characteristic is the capacity to converge to an optimum (local or global) after locating the region containing the optimum. The second characteristic is the capacity to explore new regions of the solution space in search of the global optimum. In order to vary P_c (crossover probability) and P_m (mutation probability) adaptively, for preventing premature convergence of the GA, it is essential to be able to identify whether the GA is converging to an optimum. One possible way of detecting convergence is to observe the average fitness value f' of the population in relation to the maximum fitness value f_{max} of the population. f_{max}-f' is likely to be less for a population that has converged to an optimum solution than that for a population scattered in the solution space. The equations that determine P_c, P_m are given by:

$$P_c = (f_{max}-f')/(f_{max}-f') \quad \text{when } f' \geq f'$$
$$P_c = 1 \quad \text{when } f' < f' \qquad (6)$$

$$P_m = (f_{max}-f)/2.(f_{max}-f') \quad f \geq f'$$
$$Pm = 0.5 \qquad (7)$$

Where f’ is the larger of the fitness values of the solutions to be crossed, f is the fitness of the individual solutions. The method means that we have p_c and p_m for each chromosome. We chose a one-point crossover for computational simplicity and real time performance. In [Srinivas 96] this method was superior to the simple GA and gave a rapid convergence rate of 8:1. We use this adaptive method for finding the values of crossover and mutation probabilities. We use an elite strategy, meaning that the best individual is automatically promoted to the next generation, and used to generate new populations. We also use constrained GA search, in the form of a *contextual prompter* based on the occupant’s needs. For example, if a temperature is too high for the room occupant then the AGM will be constrained so as not to suggest solutions involving increasing the temperature. In this way we can minimise the search space of the GA and achieve faster conversion.

In order to justify these techniques we have conducted various Comfort behaviour leaning experiments using both open and constrained Adaptive GA (AGA) operation plus Simple GA (SGA) with constrained operation. The results are shown in Figure (9).

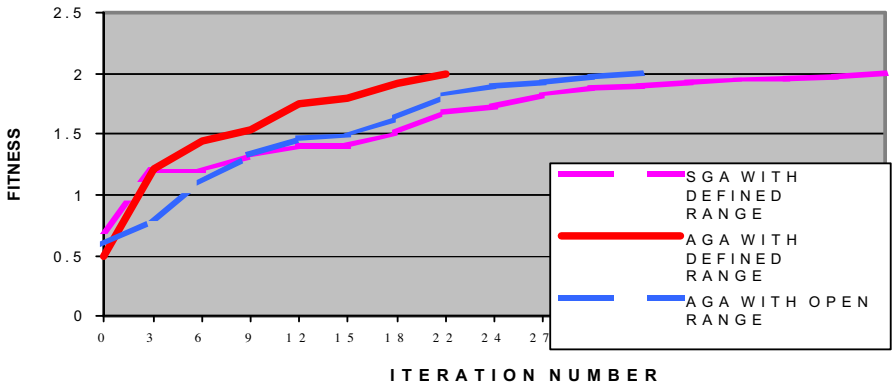

Figure (9): The best fitness plotted against the number of iterations for different GA learning.

The SGA was tried with different parameters in the range [0.5 1.0] for p_c and [0.001 0.1] for p_m. The best performance was found to occur at p_c =0.7 and p_m=0.002 (see figure 8). It was found that constrainedAGA converges to a solution in average of *22 iterations.* The AGA with open operation converged after larger number of iterations (33 iterations in average), as it needs longer to explore the search space and determine its limits. The SGA with defined limits, p_c =0.7 and p_m=0.002, converged to a solution after an average of 48 iterations. These experiments show the constrained GA methodology results in the quicker convergence. We use binary coding in the GA. For each rule there are two actions, room heating and illumination. As we have 7 output membership function, we decode each action by three bits as follows, *Very Very Low* is 000, Very *Low is 001, Low 010, Normal is 011, High is 100 Very High 101 Very Very High 110.* By doing this we have a chromosome length of 6 bits.

Figure (10) illustrates GA operation where actions of rule number 5 and rule number 7 of the comfort behaviour are chosen for reproduction by roulette wheel selection due their high fitness. They have contributed more with their actions to improvement, or contributed less to degradation. The adaptive crossover and mutation probabilities have been applied to both chromosomes. The resultant offspring were used to replace the consequents of rules 1 and rule 2, which were blamed more than the others for the unsatisfactory responses.

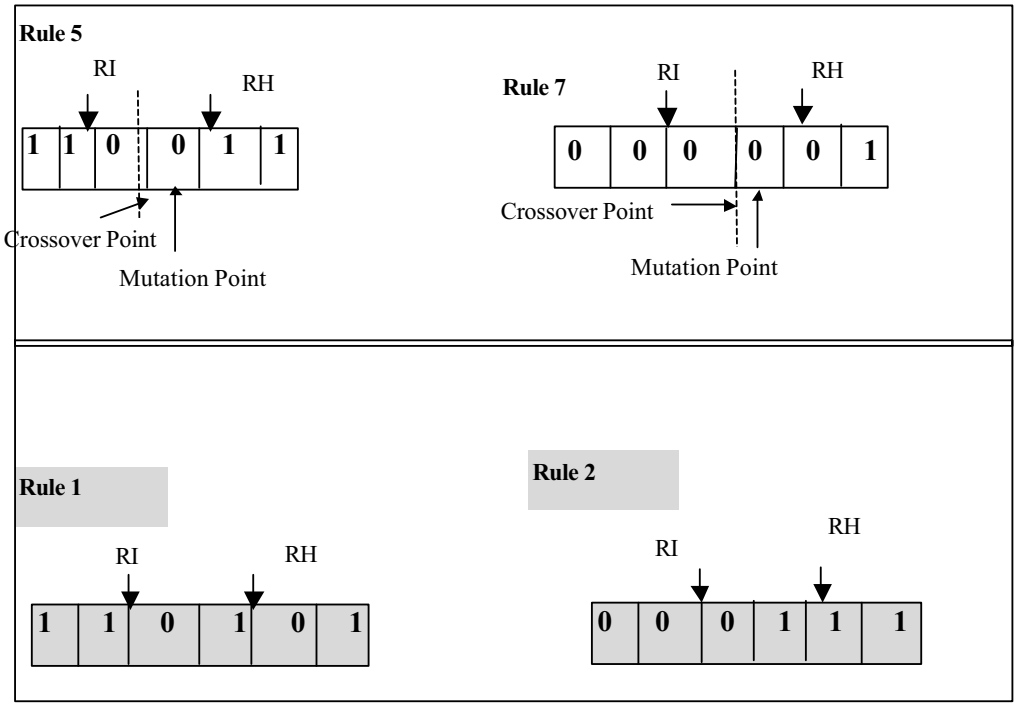

Figure (10): GA process example: rule 5 and rule 7 (selected due to higher fitness) generate new actions for rules 1,2.

5. Experimental Results

In our preliminary experiments we have used an IB agent based on a 68000 Motorola processor, see photo 1. The agent is equipped with light and heat sensors and effectors in the form of a heater and a light source. The room is subject to various conditions such as multiple occupancy, differing levels of natural light/temperature, varying times of day and different human preferences. Whilst we used a real physical agent and sensors, in order to accelerate the passage of time the agent was operated in an emulation mode, where the sensors were subjected to controlled stimulation thereby allowing days to be cycled in hours. The agent shown in Figure (11) was tried, under different conditions such as hot, sunny days and cold, dull days.

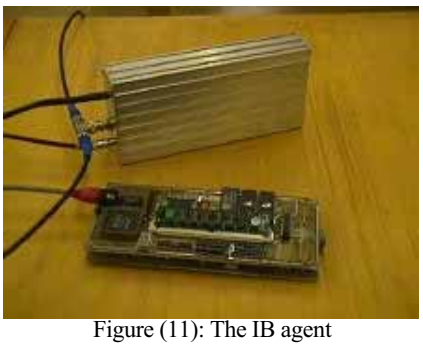

Figure (11): The IB agent

Economy behaviour seeks to minimise heat and light when the room is vacated. Safety behaviour prevents the heat going below a minimum safe level (e.g. zero degrees that would result in pipes freezing). The Comfort behaviour generation

mechanism proactively serves the needs and desires of the human occupant(s). Where necessary (e.g. setting up the system) the agent interacts with occupants.

The rules generated are presented in Table (1). Whilst operating the AEE method the agent proved itself able to rapidly deduce appropriate rules (an average of 21 iterations). It was noted that the method also optimised the number of rules by using only rules demonstrated to be important to the room occupant. The AEE optimised the number of rules from an expected $3^4 = 81$ rule base to only 49 rules.

To provide a benchmark for the AEE we implanted and evaluated the performance of an IB embedded-agent using the Mendel-Wang fuzzy rule learning method (which outperforms the ANFIS network). The Mendel-Wang approach learns by constructing fuzzy rules from input and output values. This is a widely known method, which will not be described here but can be found in Mendel-Wang’s own papers [Mendel 92].

				AEE		Mendel-Wang	
RTEMP	**ONTEMP**	**RILLUM**	**ONILLUM**	**RH**	**RL**	**RH**	**RL**
Low	Low	Low	Low	Very High	High	Very Very High	Very High
Low	Low	Low	Norm	Very Very High	Very Very High	Very High	Very High
Low	Low	Norm	Low	Very High	Norm	Very High	Norm
Low	Low	Norm	Norm	Norm	Norm	Norm	Norm
Low	Low	Norm	High	Very High	Very Low	Very High	Very Low
Low	Low	High	Low	High	Very Low	Very High	Very Low
Low	Low	High	Norm	High	Very Very Low	Very High	Very Very Low
Low	Low	High	High	High	Very Very Low	Very High	Low
Low	Norm	Low	Low	Norm	Very Very High	Norm	Very High
Low	Norm	Low	Norm	Norm	Norm	Norm	High
Low	Norm	Norm	Low	Very Very High	Very Low	Very Very High	Very Low
Low	Norm	Norm	Norm	High	Very Low	Very High	Very Very Low
Low	Norm	Norm	High	High	Very Very Low	Very High	Very Low
Low	Norm	High	Low	Very High	Low	Very Very High	Very Low
Low	Norm	High	Norm	Norm	Very Very Low	High	Very Low
Low	Norm	High	High	Very High	Very Very Low	Very Very High	Low
Low	High	Low	Low	Norm	Very High	Very Very High	Very High
Low	High	Low	Norm	Very Very High	High	Very Very High	High
Low	High	Norm	Low	High	Norm	Very Very High	High
Low	High	Norm	Norm	Very Very Low	Very Low	Very Very Low	Very Low
Low	High	Norm	High	Very Very Low	Very Low	Very Very Low	Very Low
Low	High	High	Norm	Very Very Low	Very Low	Very Very Low	Very Low
Low	High	High	High	Very Very Low	Very Low	Very Very Low	Very Low
Norm	Low	Low	Low	Norm	High	Norm	High
Norm	Low	Low	Norm	Norm	Very Very High	High	High
Norm	Low	Norm	Low	Norm	Norm	Norm	Norm
Norm	Low	Norm	Norm	Norm	Norm	High	High
Norm	Low	Norm	High	Very Very Low	Very Low	Very Very Low	Low
Norm	Low	High	Norm	Low	Norm	Very Very Low	Low
Norm	Low	High	High	Very Very Low	Very Low	Very Low	Norm
Norm	Norm	Low	Low	Very Low	Very High	Very Very Low	Very Very High
Norm	Norm	Norm	Low	Low	Norm	Very Low	High
Norm	Norm	Norm	Norm	Very Low	Very Very	Very Very Low	Norm

					Low		
Norm	Norm	Norm	High	Very Very Low	Low	Very Low	Very Very Low
Norm	Norm	High	Norm	Low	Very Low	Very Very Low	Low
Norm	Norm	High	High	Very Low	Low	Very Very Low	Low
Norm	High	Low	Low	Very Very Low	High	Very Very Low	Norm
Norm	High	High	High	Very Very Low	Very Low	Very Very Low	Very Low
High	Low	Low	Low	Very Low	High	Very Low	Very High
Norm	High	High	High	Very Very Low	Very Low	Very Very Low	Very Low
High	Low	Low	Low	Very Low	High	Very Low	Very High
High	Low	Norm	Low	Very Low	Very Very High	Very Low	Very High
High	Low	Norm	Norm	Very Very Low	Low	Very Low	Very Low
High	Low	Norm	High	Norm	Very Very Low	Norm	Very Low
High	Low	High	Norm	Norm	Low	Very Very Low	Low
High	Low	High	High	Very Low	Low	Very Low	Low
High	Norm	Low	Low	Norm	High	Very Low	High
High	Norm	Norm	Low	Low	Very High	Very Low	Very High
High	Norm	Norm	Norm	Very Very Low	Low	Low	Very Low
High	Norm	Norm	High	Very Very Low	Very Low	Very Very Low	Low
High	Norm	High	High	Very Very Low	Very Low	Low	Very Very Low

Table (1): The rule base learnt by the AEE and Mendel-Wang methods.

The rules generated by Mendel-Wang are presented in Table (1) where they can be compared to the AEE method. Although both systems appear to give comparable results, the AEE system out-performs Mendel-Wang in at least one important aspect; Mendel-Wang uses essentially off-line learning, in which each learning cycle needs to repeat from the beginning, requiring both the initial training set together with any newly acquired data. In contrast the AEE method directly interacts with the user in an essentially on-line way, continuing the learning cycle from an advanced point rather than starting afresh making it much faster. Thus the AEE system works by cause-effect actions in the form of fuzzy rules, based on the occupant’s actions. The advantage of this is that the system responds and adapts to the users needs interactively.

Techniques, such as Mendel-Wang, have the disadvantage that the interface with the user is based on the provision of a set of desired values rather than simply interacting with the user to obtain a satisfactory result. Whilst the user may eventually acquire a feel for what figures to supply to get the right result, but even if a computer program was used to assist, the process is far from intuitive. Thus, although the rules extracted by the AEE method are similar to the Mendel-Wang rules, the difference is that with AEE learning the occupant can interact directly with the agent until satisfied with the actual settings.

6. Conclusion

6.1 Summary

In this paper we have introduced innovative fuzzy-genetic distributed agent architecture for intelligent buildings. We have outlined the difficult and unique control and learning problem that IB based agents need to cope with, in particular, dealing with large numbers of sensory inputs which display complex dynamics due to interactions with the environment, people and other agents.

We have also described a novel soft-computing architecture that solves this problem and is based on the use of hierarchical fuzzy controls. The fuzzy controllers form a behaviour-based architecture comprising three fixed behaviours - the Safety, Emergency and Economy behaviours and a dynamic (adaptable) rule-set that forms what we term a Comfort behaviour.

We have explained the importance of learning in IB agents and in particular the emphasis on particularisation rather than generalisation that is required to tailor the agents activities to the individual needs of differing occupants, moods and occasions. To address this challenge we have described a novel constrained GA learning methodology (Associative Experience Engine - AEE) that uses both past experiences and contextual information to find solutions more efficiently. In practical experiments we have conducted we found that AEE based agents interactively learn optimised rule bases for the comfort behaviour in approximately 21 iterations. In a comparative study to the Mendel-Wang method we showed that the AEE could produce similar rules and had the significant advantage of being able to interactively adapt to environmental (occupant driven) changes. Other notable characteristics are incremental rules processing (adding rules as more about the problem and solution is discovered), memory based exemplar processing (including short and long term processing such as aging) and self-analysis in terms of behaviour, error or success.

6.2 Future Work

Our current work is aimed at (1) establishing a standard communication framework for distributed embedded-agents (e.g. DIBAL), (2) development of better simulation/emulation tools for distributed embedded-agents, (3) the application of emerging technologies (eg embedded-internet, Mex, Java, Jini, JavaSpaces etc) and (4) exploring the use of alternative agent architectures (e.g. neural networks, Fuzzy Neural Networks, Instance Based etc). Concerning the associative experience engine, whilst it has allowed us make some significant progress towards meeting the challenges we set ourselves in sections 1.1 and 1.2 above, we clearly have work left to achieve the ideal embedded-agent for intelligent-buildings. Our original and continuing goal is a system that learns from the occupant without the need for any explicit input (i.e. non-intrusive online learning). Our current experimental system requires explicit interaction in order to develop its rules. We need a system that is capable of carrying out a recalculation of the appropriate rule base after the trigger of occupant intervention (i.e. changing an effector setting) without having to engage the occupant in the process. This would make the learning process totally transparent to the occupant.

We are exploring various possibilities for giving the AEE such a non-intrusive learning capability. We are currently investigating a mechanism we refer to as *Incremental Synchronous Learning* (ISL) which would work as follows: when an occupant changes an effector setting manually, the system would respond by immediately carrying out the action, setting the building to the requested state and generating a new rule based on that instance. In a manner comparable to the use of the AGM in the experimental system such a change of behaviour would initiate a learning sequence. In this case the learning sequence would be the equivalent of one iteration of the experimental system. At this point any further action would be suspended until there was another interaction with the occupant. That is, there would be no forced interactions with the occupant but rather the occupants spontaneous interactions would be used to trigger a simple learning process. It is hoped that such a modification to the system would allow the system to learn in the same way as the experimental prototype but unobtrusively by spreading the iterations over an extended period using the natural interactions of the user with the system. Thus for example, considering a temperature controller, each day the occupant might make an adjustment to the system (i.e. one learning iteration) thereby completing a learning cycle in an average of 21 days (according to our experimental data) which we would argue would be a most acceptable time for an agent to learn to particularise it services to a person (given in a manual system the user will always need command the system, whereas in the agent-assisted system the manual load upon the occupant reduces over time). In addition to providing a non-intrusive learning mechanism, this approach also places the user in prime control as it unfailingly and immediately responds to his command.

Another method we intend to examine is how s-maps (see section 1.2) might be used as an intermediate representation (i.e. a target system behaviour template) against which the AGM might explore actions to develop a new set of rules without having to interact with the occupant (i.e. it interacts with the template in a virtual space). This would be done as a background task (off-line learning in the strict sense), with the existing rules, including the newly acquired rule, in operation until they could be replaced by the newly learnt rules based upon the changes to the navigation spaces that the independent action of the occupant has occasioned. This should lead to the acceleration of learning to the same sort of speed as the prototype system but without the need for explicit occupant interaction.

With respect to applications, our interests include situations as diverse as mobile phones, wearable agents, through white/black goods to space-based transport and habitats.

Acknowledgements: We are pleased to acknowledge the contribution of Malcolm Lear (Essex University) who built the agent hardware, sensors and test rig. We would also like to thank, Anthony Pounds-Cornish, Sue Sharples, Gillian Kearney, Robin Dowling and Filiz Cayci with whom we have had many stimulating discussions on embedded-agent architectures. Finally, we would like to express our gratitude to Martin Henson for his assistance in translating the original WORD version of this paper into Latex.

References

[Bonarini 96] A. Bonarini, F. Basso, “ Learning Behaviors Implemented As Fuzzy Logic And Reinforcement Learning”, 2nd Online Workshop On Evolutionary Computation, 1996.
[Brooks 91] R Brooks, “Intelligence Without Representation”, Artificial Intelligence 47, pp139-159, 1991.
[Brooks 97] R. Brooks, “Intelligent Room Project”, Proc 2nd Int’l Cognitive Technology Conference, Japan 1997.
[Callaghan 2000] Callaghan V, Clarke, G, "Buildings As Intelligent Autonomous Systems: A Model for Integrating Personal and Building Agents", Proc. 6th International Conference on Intelligent Autonomous Systems, Venice, Italy; July 25 - 27, 2000.
[Cayci 2000] Cayci F, Callaghan V, Clarke G, "DIBAL - A Distributed Intelligent Building Agent Language", Proc. 6th International Conference on Information Systems Analysis and Synthesis, Orlando, Florida, July 2000.
[Coen 97] M.H.Coen, “Building Brains for Rooms: Designing Distributed Software Agents”, Proc. Ninth Innovative Applications of AI Conference, AAAI Press, 1997.
[Davisson 98] P. Davisson “Energy Saving and Value Added Services; Controlling Intelligent-Buildings Using a Multi-Agent System Approach” in DA/DSM Europe DistribuTECH, PennWell, 1998.
[Dorigo 93] M. Dorigo, "Genetics-Based Machine Learning And Behaviour Based Robotics: A New Synthesis", IEEE transactions on Systems, Man, Cybernetics, pp. 141-154, 1993.
[Fukuda 99] T. Fukuda , N. Kubota , "An Intelligent Robotic System Based On Fuzzy Approach", Proceedings of the IEEE, Vol. 87, No. 9, pp.1448-1470, September 1999.
[Hagras 99a] H.Hagras, V Callaghan, M Colley, “A Fuzzy-Genetic Based Embedded-Agent Approach to Learning and Control in Agricultural Autonomous Vehicles”, IEEE International Conference on Robotics and Automation, pp. 1005-1010, Detroit- U.S.A, May 1999.
[Hagras 99b] H.Hagras, V Callaghan, M Colley, “Online Learning of Fuzzy Behaviours using Genetic Algorithms & Real-Time Interaction with the Environment”, IEEE International Conference on Fuzzy Systems, Seoul-Korea, pp. 668-672, August 1999.
[Hagras 2000a] Hagras H, Callaghan V, Colley M, “Online Learning Of Fuzzy Behaviour Co-Ordination For Autonomous Agents Using Genetic Algorithms And Real-Time Interaction With The Environment” IEEE International Conference on Fuzzy Systems in San Antonio, Texas, USA, 7-10 May 2000.
[Hagras 2000b] Hagras H, Callaghan V, Colley M, “On -Line Learning Of The Sensors Fuzzy Membership Functions In Autonomous Mobile Robots”, IEEE International Congress on Robotics and Automation, San Francisco, April 2000.
[Hoffmann 98] F. Hoffmann, "Incremental Tuning Of Fuzzy Controllers By Means Of Evolution Strategy”, GP-98 Conference, pp. 550-556, Madison, Wisconsin, 1998.
[Jeon 2000] Jeon H, Petrie C, Cutkosky M.R, "JATLite: A Java Agent Infrastructure with Message Routing", University of Stanford, IEEE Internet Computing, March/April 2000.
[Kasabov 98] Kasabov N “The ECOS Framework and the ECO Learning Method for Evolving Connectionist Systems”, J. Advanced Computational Intelligence, Vol 2, No 6, 1998.
[Labrou 99] Labrou Y, Finin T, Peng Y, "The Current Landscape Of Agent Communication Languages", IEEE Intelligent Systems, Vol. 14, No. 2, March/April 1999.
[Lehikoinen 99] J. Lehikoinen, J. Holopainen, M. Salmimaa, and A. Aldrovandi "MEX: A Distributed Software Architecture for Wearable Computers" 3rd International Symposium on Wearable Computers, San Francisco, California 18-19 October 1999.
[Linkens 95] G.Linkens, O. Nyongeso, "Genetic Algorithms For Fuzzy Control, Part II: Online System Development And Application", IEE proceedings Control theory applications, Vol.142, pp.177-185, 1995.
[Minar 99] M Nelson, M Gray, O Roup, R Krikorian, P Maes "HIVE: Distributed Agents For Networking Things", Proc. First International Symposium on Agent Systems and Applications and Third International Symposium on Mobile Agents, Rancho Las Palmas Marriott's Resort and Spa, Palm Springs, California, October 3 - 6 1999.
[Mozer 98] M. Mozer “The Neural Network House: An Environment That Adapts To Its Inhabitants”, Proc of American Association for Artificial Intelligence Spring Symposium on Intelligent Environments, pp110-114, AAAI Press, 1998.
[Robathan, 89] P. ROBATHAN, “Intelligent Buildings Guide”, Intelligent Buildings Group and IBC Technical Services Limited, 1989.
[Saffiotti 97] A. Saffiotti, "Fuzzy Logic In Autonomous Robotics: Behaviour Co-Ordination", Proc. 6th IEEE International Conference on Fuzzy Systems, Vol.1, pp. 573-578, Barcelona, Spain, 1997.
[Sharples 99] S. Sharples, V. Callaghan, G. Clarke, "A Multi-Agent Architecture for Intelligent Building Sensing and Control" International Sensor Review Journal, May 1999 .
[Srinivas 96] M. Srinivas, L. Patnaik, “Adaptation In Genetic Algorithms”, Genetic Algorithms For Pattern Recognition”, (Eds Pal & Wang), CRC press, pp. 45-64, 1996.
[Steels 95] L. Steels, "When Are Robots Intelligent Autonomous Agents", Journal of Robotics and Autonomous Systems, Vol. 15, pp.3-9, 1995.
[Mendel 92] J. Mendel, L.Wang, "Generating Fuzzy Rules by Learning Through Examples", IEEE Trans. on Systems, Man and Cybernetics, Vol. 22, pp. 1414-1427, December 1992.
[Sherwin 99] Sherwin A “Internet House Offers a Life of Virtual Luxury”, The Times, p10, 3rd Nov 1999.
[Zhou 90] H. Zhou, “ A Computational Model of Cumulative Learning”, Machine Learning Journal, pp. 383-406, 1990.