"A Hierarchical Fuzzy Genetic Multi-Agent Architecture for Intelligent Buildings Learning, Adaptation and Control", Journal of Information Sciences, Vol. 150, pp. 33-54, March 2003

A Hierarchical Fuzzy-Genetic Multi-Agent Architecture for Intelligent Buildings Learning, Adaptation and Control

Hani Hagras[†], Victor Callaghan[‡], Martin Colley[‡], Graham Clarke[‡]

[†]The Computer ScienceDepartment, The University of Hull, Hull, HU6 7RX, UK.

[‡] The Computer ScienceDepartment, Essex University, Colchester, CO43SQ, UK.

Email: hani@essex.ac.uk

Abstract:

In this paper, we describe a new application domain for intelligent autonomous systems – Intelligent Buildings (IB). In doing so we present a novel approach to the implementation of IB based on a hierarchical fuzzy genetic multi embedded-agent architecture comprising a low-level behaviour based reactive layer whose outputs are co-ordinated in a fuzzy way according to deliberative plans. The fuzzy rules related to the room resident comfort are learnt and adapted online using our patented Fuzzy-Genetic techniques (British Patent 99-10539.7). The learnt rule base is updated and adapted via an iterative machine-user dialogue. This learning starts from the best stored rule set in the Experience Bank thereby decreasing the learning time and creating an intelligent agent with memory. We discuss the role of learning in building control systems, and we explain the importance of acquiring information from sensors, rather than relying on pre-programmed models, to determine user needs. We describe how our architecture, consisting of distributed embedded agents, utilises sensory information to learn to perform tasks related to user comfort, energy conservation, and safety. We show how these agents, employing a behaviour-based approach derived from robotics research, are able to continuously learn and adapt to individuals within a building, whilst always providing a fast, safe response to any situation. In addion we show that our system learns similar rules to other offline supervised method such as the Mendel-Wang method but that our system has the addional capability to rapidly learn and optimise the learnt rule base. Applications of this system include personal support (e.g. increasingr independence and quality of life for older people), energy efficincy in commercial buildings or living-area control systems for space vehicles and planetary habitation modules

1. Introduction

The building industry uses 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. A definition which finds favour with many building managers and architects is that "*An Intelligent-Building is one that provides a productive cost-effective environment through the optimisation of four basic elements; systems, structures, services, management and the inter-relationship between them*" [Robathan 89].

Computer scientists, however, hold a different view of intelligence. They are more concerned with giving machines management, analytic and control capabilities that are *comparable to intelligent human activity*. We prefer the definition - "*An Intelligent-Building is one that utilises computer technology to autonomously govern and adapt the building environment so as to optimise user comfort, energy-consumption, safety and work efficiency*". In the context of a building, a system works by taking inputs from building sensors (light, temperature, passive infra-red, etc), and using this and other information to control effectors (heaters, lights, electronically-operated windows, etc). If this system is to be intelligent, an essential feature must be its ability to learn and adapt appropriately. For this a system which can adapt and generate its own rules (rather than being restricted to simple automation) is required. Kasabov provides a set of 7 criteria that define an intelligent systems [Kasabov 98]. An intelligent system should be able to learn quickly from large amounts of data therefore using fast training. An intelligent system should also adapt in real-time and in an on-line mode where new data are accommodated as it arrives. Also the system should be able to accommodate in an incremental way any rules that will become known about the problem. It should be memory-based, plus possess data and exemplar storage and retrieval capacities. It also should be able to learn and improve through active interaction with the user and the environment. It should have parameters to represent short and long term memory, age, forgetting, etc. It should also analyse itself in terms of behaviour, error and success. To the authors knowledge no system in the field of intelligent building has satisfied these conditions.

We view intelligent buildings as computer-based systems, akin to robots, gathering information from a variety of sensors, and using embedded intelligent agent techniques to determine appropriate control actions. In controlling such systems one is faced with the imprecision of sensors, lack of adequate models of many of the processes and the non-deterministic aspects of human behaviour. 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 Brooks [Brooks 91] & Steels [Steels 95] who have had considerable success in the field of mobile robots. It might not seem obvious that a building can be looked upon as a machine; indeed "*a robot that we live within*", but, in other work [Callaghan 2000] we have justified the view that intelligent buildings, are analogous to robots, gathering information from a variety of sensors, and able to use behaviour-based techniques to determine appropriate control actions.

Fuzzy logic offers a framework for representing imprecise, uncertain knowledge. Similar to the way, in which people make their decisions, fuzzy systems use 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, fuzzy systems have a well-known problem concerning the determination of their parameters. In most fuzzy systems, the fuzzy rules are determined and tuned through "trial and error" by developers, taking much iteration to determine and tune them. As the number of input variables increases (which is the case of Intelligent buildings) the numbers of rules increase further magnifying the difficulty.

Evolutionary algorithms constitute a class of search and optimisation methods guided by the principles of natural evolution. Genetic Algorithms (GA) are optimisation methods inspired by principles of natural evolution and genetics. GA has been successfully applied to solve a variety of difficult theoretical and practical problems by imitating the underlying processes of evolution such as selection, recombination and mutation. Their capability of learning enables a GA to adapt a system to deal with any task [Goldberg 89]. There are numerous reports in the scientific literature on designing fuzzy controllers using GA [e.g. Fukuda 99, Linkens 95, Bonarini 96, Hoffmann 98]. However, most work uses simulation to overcome the time overhead problem caused by the large number of iterations needed for a conventional GA to develop a good solution. Thus it is not feasible for a simple GA to learn online and adapt in real-time. The situation is worsened by the fact that most evolutionary computation methods developed so far assume that the solution space is fixed (i.e. the evolution takes place within a pre-defined problem space and not in a dynamically changing and open one), thus preventing them from being used in real-time applications [Kasabov 98].

Our work is concerned with utilising an intelligent *embedded-agent* approach based on a double hierarchical Fuzzy-Genetic system, similar to the approach already taken in our previous work in mobile robotics [Hagras 99a, Hagras 99b], to create an integrated and semi-autonomous building control system. There are other research projects concerned with applying AI to buildings most notably [Brooks 97, Coen 97, Davidsson 98, Mozer 98]. We believe our approach is unique in seeking to apply a hierarchical fuzzy control architecture with genetic learning. This approach allows the learning or adaptation to be performed through interaction with the environment (with the occupant of the room being seen as part of the environment). In this approach there is no need for simulation or direct human intervention into the rule setting system thus satisfying the definitions of Intelligent Autonomous Agents in robotics [Steels 95] and intelligent systems as explained above.

Broadly speaking, the agent work described here is situated in the recent line of research that concentrates on the realisation of embedded real-time agents grounded in the physical world. Traditional approaches such as real-time control and expert systems rely on predictive models and thus have difficulty in environments that are of intractable complexity (in terms of the dynamics and number of variables) being essentially (if not actually) non-deterministic. However, a number of robotic based researchers have had notable success in dealing with this type of problem based on techniques that can broadly be categorised by the term behaviour based systems. We take this approach as our starting point and have developed a method of implementing this as a hierarchical fuzzy system with numerous consequent advantages. Thus, our macro-control architecture belongs the "behaviour based control architecture" school pioneered by Rodney Brooks of MIT in the late 80's [Brooks 92]. In this approach a number of concurrent behaviours (mechanisms to attain goal or maintain state) are active (sensing environment, effecting machine) to a degree determined the relationship of the machine and environment. We have extended this approach by developing a number of mechanisms that address both behaviour-integration, in the form of a hierarchical fuzzy controller, and genetic learning, combined into a single architecture which we label an *"Associative Experience Engine"* [Hagras 99a, Hagras 99b]. In addition to dealing with control problems that are difficult to model, the self-programming nature of the system has benefits in reducing software development costs throughout the lifetime of the product. In this paper we give an overview of both the hierarchical fuzzy control and the genetically based Associative Experience Engine applied to develop an embedded-agent within a distributed intelligent buildings architecture.

2. Distributed Architecture

The granularity of our computational distribution is room-based. Thus, each room contains an *embedded-agent*, which is then responsible, via sensors and effectors for the local control of that room as shown in Figure (1). The logic behind this is that it mirrors the architects' vision of the functionality of the building and thereby provides a natural segmentation of agent types and functions. All embedded-agents are connected via a high level network (IP-ethernet in our case), thereby enabling collaboration or sharing of information to take place where appropriate. Within a room, devices such as sensors and effectors are connected together using a building services network (Lontalk in our case) and IP at the higher level. This Distributed Artificial Intelligence (DAI) architecture is illustrated in Figure (1).

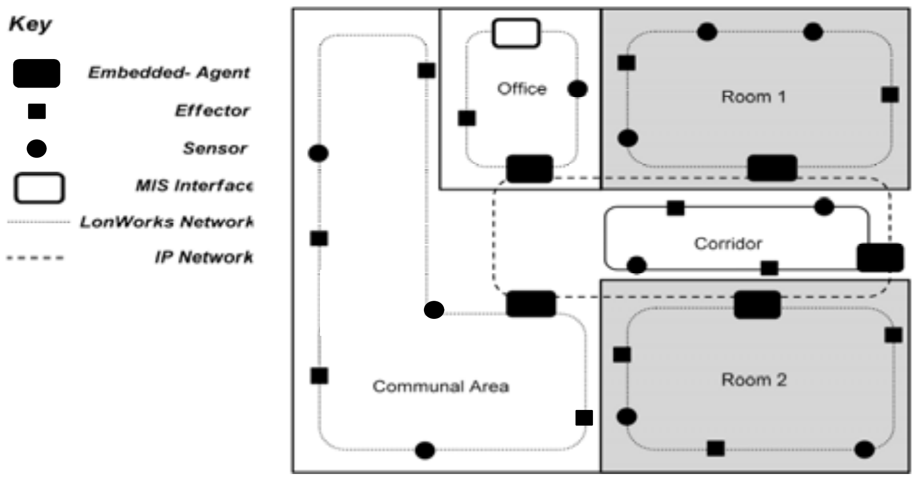

Figure (1): The DAI Building-Wide Architecture

3. The Embedded-Agents

Figure (2) shows the internal of the behaviour-based agent. Controlling a large integrated building system requires a complicated control function resulting from the large input and output space and the need to deal with many imprecise and unpredictable factors, including people. In our system we simplify this problem by breaking down the control space into multiple behaviours, each of which responds to specific types of situations, and then integrating their recommendations.

3.1 The Hierarchical fuzzy control architecture

The behaviour based approach, pioneered by Brooks, consists of many simple co-operating units and has produced very promising results when applied to the control of robotics, which we argue includes IB.

The problem of how to co-ordinate the simultaneous activity of several independent behaviour-producing units to obtain an overall coherent behaviour have been discussed by many authors [Brooks 91, Saffiotti 97]. The work described in this paper suggests a solution based on using fuzzy logic to both implement individual behaviour elements and the necessary arbitration (allowing both fixed and dynamic arbitration policies to be implemented). We achieve this by implementing each behaviour as a fuzzy process and then using fuzzy agents to co-ordinate them. In the resultant architecture, a hierarchical fuzzy logic controller (HFLC) takes a hierarchical tree structure form and is shown in Figure (2). This hierarchical approach has the following advantages:

- It facilitates the design of the agent controller and reduces the number of rules to be determined. It uses the benefits of fuzzy logic to deal with imprecision and uncertainty.
- Using fuzzy logic for the co-ordination between the different behaviours which allows more than one behaviour to be active to differing degrees thereby avoiding the drawbacks of on-off switching schema (i.e. dealing with situations where several criteria need to be taken into account). In addition, using fuzzy co-ordination provides a smooth transition between behaviours with a consequent smooth output response.
- It offers a flexible structure where new behaviours can be added or modified easily. The system is capable of performing different tasks using identical behaviours by changing only the co-ordination parameters to satisfy a different high level objective without the need for re-planning.

Our room-based decomposition of behaviours consists of the following meta-functions. A S*afety behaviour*, which ensures that the environmental conditions in the room are always at a safe level. An *Emergency behaviour,* which in the case of a fire alarm or another emergency, might for instance open the emergency doors and switch off the main heating and illumination systems. In the case of an emergency this will be the only active behaviour. An *Economy behaviour* that ensures that energy is not wasted so that if a room is unoccupied the heating and illumination will be switched to a sensible minimum value. All of the previous behaviours are fixed but settable, there is however a set of behaviours that the system learns from the occupant of the room and these are called *Comfort behaviours.* These behaviours try to ensure that the conditions the occupant prefers (subject to being safe) are set. The learning process 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, inside illumination level, outside illumination level, inside temperature, outside 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.

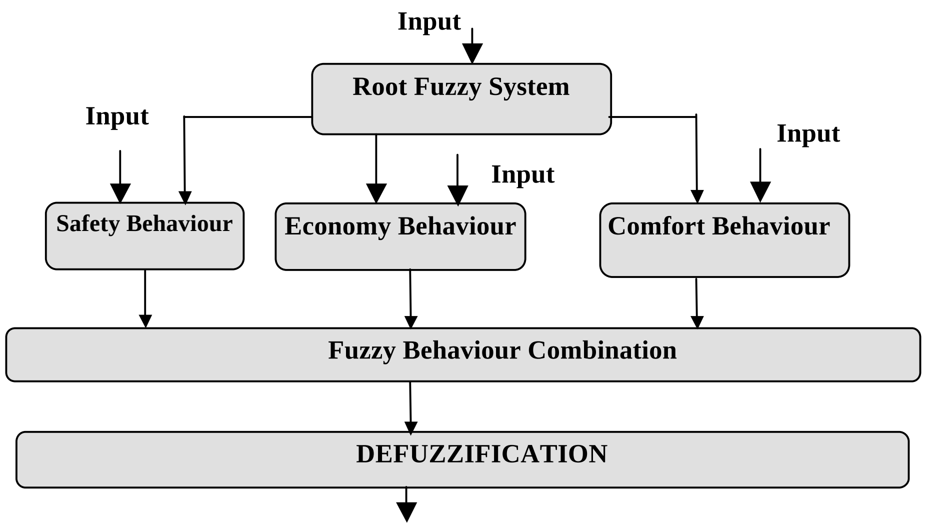

Figure (2): The Hierarchical Fuzzy Control System.

In our prototype system each agent has six inputs made up of four environmental variables - a Room Temperature (RTEMP), the Outside Natural Temperature (ONTEMP), the Room Illumination (RILLUM) and the Outside Natural

Illumination (ONILLUM) each of which have the fuzzy membership functions shown in Figure (3). Each input will be represented by three fuzzy sets as these were found to be the smallest number that give satisfactory results. The two remaining inputs to the system are the room occupancy flag and another input indicating whether there is an emergency alarm. The system has two outputs which are the Room Heating (RH) setting and a Room Illumination (RI) setting, these outputs have the membership functions shown in Figure (4). Seven fuzzy sets were used as it was found to be smallest number to give a satisfactory result. If the alarm input is active then the Emergency behaviour is dominant and the room illumination and heat are switched off. The Economy behaviour is active to a fuzzy degree depending on occupancy and the outside temperature and light. Behaviour co-ordination is done in a fuzzy way as will be explained later and the fuzzy membership for behaviour co-ordination are shown in Figure (5).

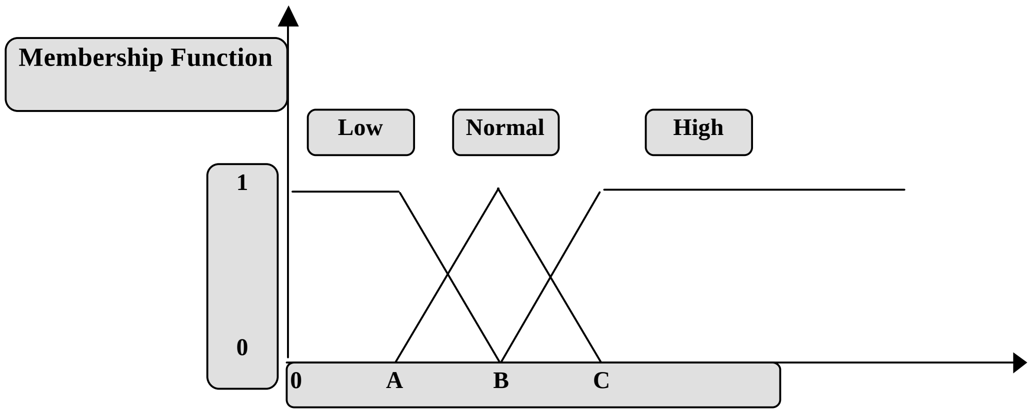

Figure (3): The input membership functions for RTEMP and ONTEMP A= 10°, B= 20°, C= 30°, for RILLUM and ONILLUM A= 300 Lux, B= 400 Lux, C= 500 Lux.

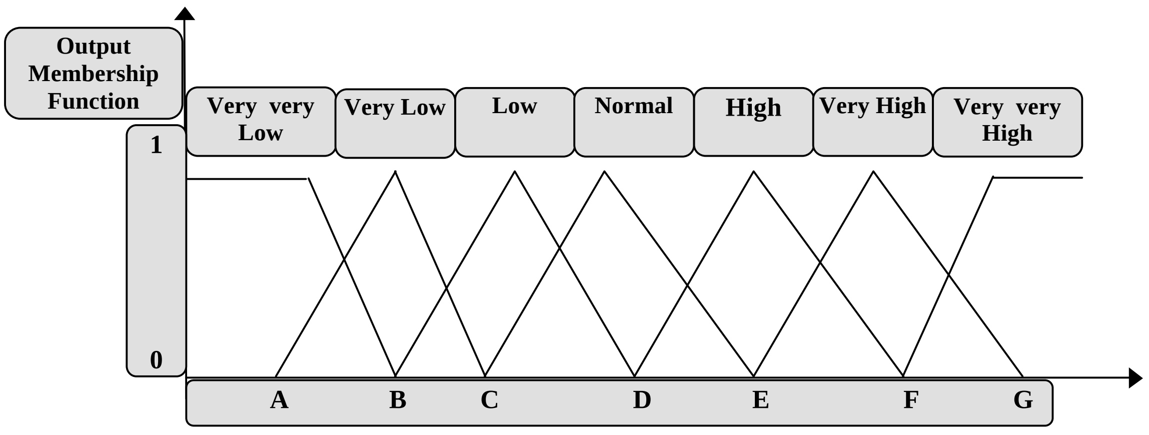

Figure (4): The output membership functions for RI A= 0 %, B= 20 %, C= 35 %, D=40%, E=50%, F=70%, G=100%, for RH A=0%, B=30%, C=40%, D=50% , E= 70%, F=85%, G=100%.

Each separate behaviour will use 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, αAi is the product of the membership functions of each rule input and G is the number of inputs. In this hierarchical architecture we will use a fuzzy operator to combine the preferences of different behaviour into a collective preference. According to this view, 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 final output equation is [Saffiotti 97]:

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

Here i = economy, comfort, C_i is the behaviour command output (room heat and 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 and then using fuzzy behaviour combination we obtain a collective fuzzy output which is then deffuzzified to obtain a final crisp output. The fuzzy values are used as inputs to the context rules which are suggested by the high level planner according to the mission to be performed by *the agent. Which is in general :*

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 context rules determine which behaviour is fired, and to what degree, depending on the fuzzy membership functions in Figure (5). The final output is a mixture of the different behaviour outputs, each weighted by the degree of its importance, and the final output is calculated using Equation (2).

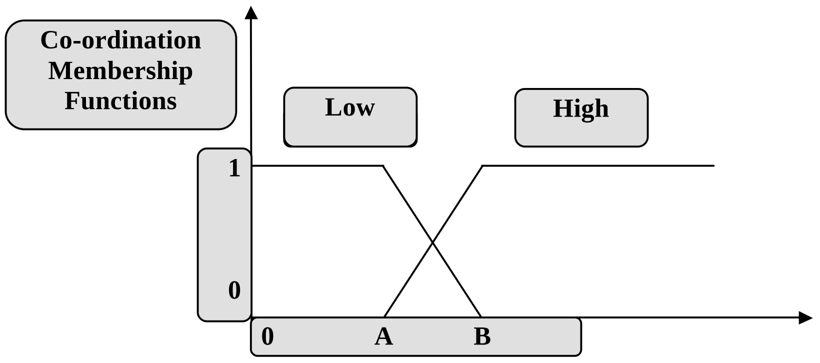

Figure (5) The co-ordination parameters for ONILLUM A= 350Lux and B= 400 Lux, for ONTEMP A= 15° and B=25°.

4. Overview of the Genetic Learning Architecture

For learning and adapting the dynamic comfort rule base according to the occupant behaviours we use an evolutionary computing approach based on a development of novel hierarchical genetic algorithm (GA) technique. This mechanism operates directly on the fuzzy controller rule-sets. We refer to any learning conducted without user interaction, in isolation from the environment as *offline* learning. In our case learning will be done *online* in real-time through interaction with the actual environment and user.

4.1 The Associative Experience Engine (AEE)

Figure (6) provides an architectural overview of what we term an *Associative Experience Engine* which forms the learning engine within the control architecture and is the subject of British patent 99-10539.7. Behaviours are represented by parallel Fuzzy Logic Controllers (FLC). Each FLC has two parameters that can be modified which are the *Rule Base* (RB) of each behaviour and its *Membership Functions* (MF), however in this paper we will concentrate on learning the Rule Base. The behaviours receive their inputs from sensors. The output of each FLC is then fed to the actuators via the *Co-ordinator* that weights its effect. When the system response fails to have a desired a response, the learning cycle begins.

The learning depends on the *Learning Focus* which is supplied by the *Co-ordinator* (the fuzzy engine which weights contributions to the outputs). When the *Learning Focus* is learning an individual rule base for a behaviour, then each rule base for each behaviour is learnt alone, which is the case when learning the Comfort behaviour rule base in the IB agent.

After the initial process of interactively learning the occupant's preferred behaviours, if the occupants behaviour indicates that there are changes in attitude or that a new situation has arisen, the system can add or delete rules to satisfy the occupant by re-entering the interactive mode. In the case of a new occupant in the room the system interactively monitors their actions during the first few minutes and after that the system takes control and fires the most suitable rule base from the *Experience Bank.* If this rule base is not totally appropriate, the interactive system learning is started from the previously found best point in the search space rather than starting from random, this results in the speeding up of learning and adaptation. The Experience Assessor 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 *Experience Survival Evaluator* determines which rules are removed according to their importance (as set by the Experience Assessor).

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 solution 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 certain range defined by the *Contextual Prompter* which is supplied by human constrains 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 until a satisfactory solution is achieved.

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. The system then either tries to re-adjust the rules if the occupant is dissatisfied or, 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 (typically 22). As we mentioned earlier this process would probably have to undertaken by a care assistant for some populations. In the next section we will explain the techniques in more detail.

4.2 Comparison of Results with Mendel-Wang Approach

We have compared our method, which is an online proactive method, with an offline supervised learning system - the Mendel-Wang approach. [Mendel 92] The Mendel-Wang approach learns by constructing fuzzy rules from input and output values. We found, from the experimental results, that our system gives comparable results to that of the Mendel-

Wang approach while our system learns online through interaction with the user and has the ability to adapt to new users or situations. Offline methods like Mendel-Wang have to repeat the learning cycle from the beginning and require that the initial training set plus any newly acquired data be used. Our 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 in an interactive manner. To gather all the given results we used an AEE specially adapted for IB control, shown in Figure (7).

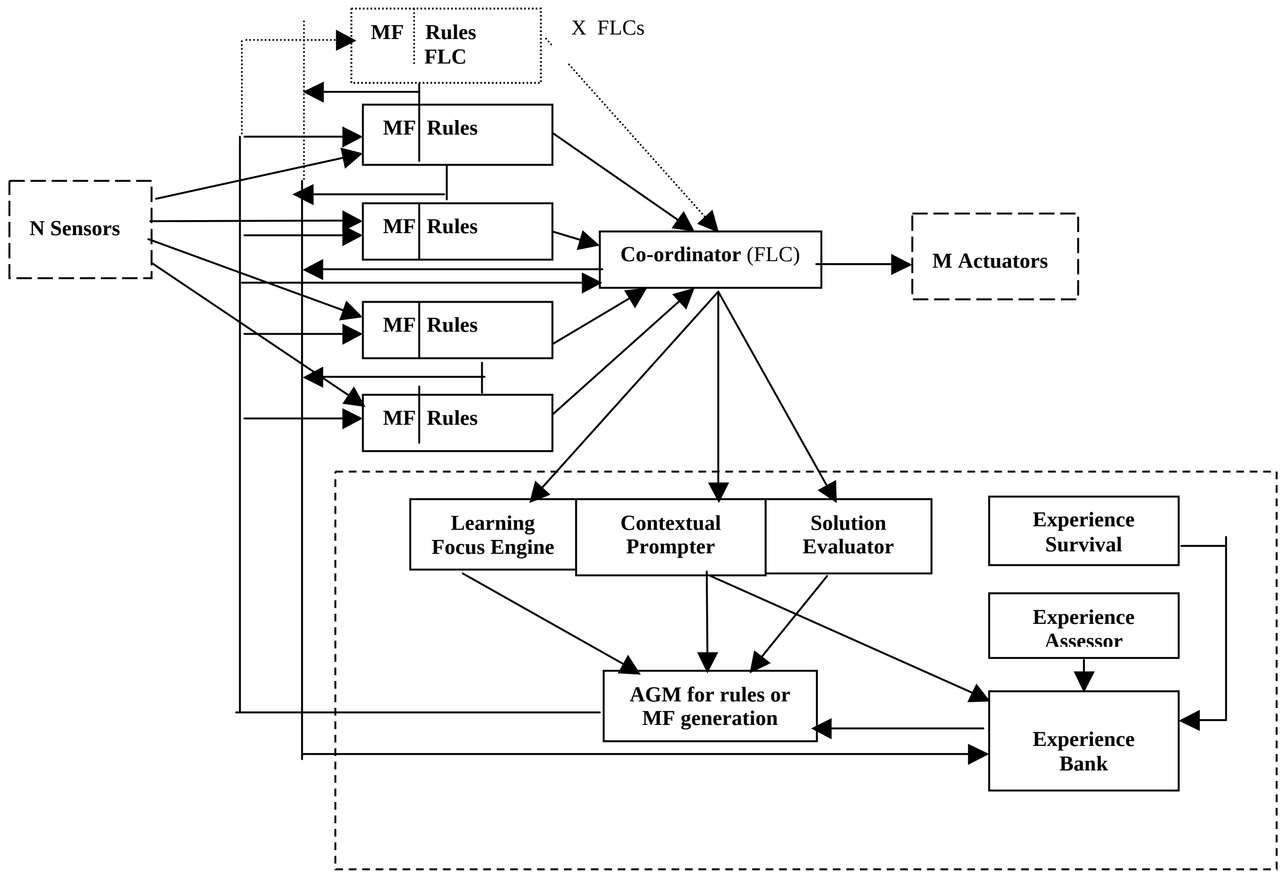

Figure (6): Architectural Overview of Associative Experience Learning Engine (UK patent No 99-10539.7)

5. The Associative Experience Learning Engine applied in IB in detail

5.1 Identifying Poorly Performing Rules

The rule-base is intialised 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 in the case with 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 actions of the rules that participated in this action at a given input.

In the learning phase the agent is Introduced to different situations, such as having low temperature inside and outside the room and low illumination level, and the agent, guided by the occupants needs attempts to discover the rules needed in each situation. The learning system consists of learning different situations or episodes; in each situation only small number of rules will be fired . The model to be learnt is small and so is the search space. The accent on local models 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 99]. Moreover, the smooth transition among the different models implemented by fuzzy rules implies robustness with respect to data noise. So 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 if we started learning with 81 rules, the agent discovered that during its interactive training with the occupant, it needed only 49 rules.

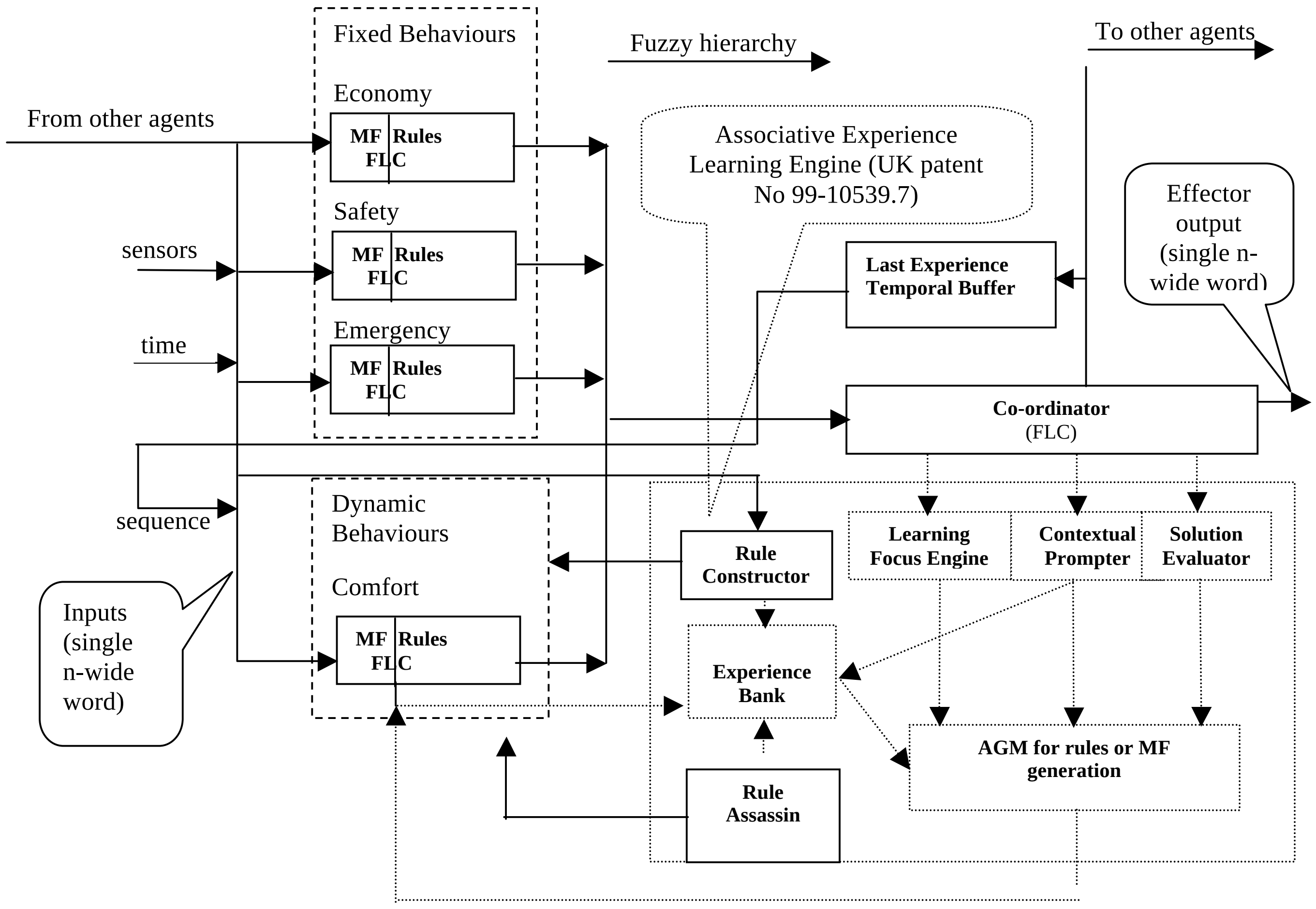

Figure (7): Modified AEE Embedded-Agent Architecture

5.2 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 desired input value or normal value (such as desired temperature) in a specific situation and how it reduces the normalised absolute deviation (d) from the normal value. This is given by:

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

Where the normal value will correspond to the value that gives the value desired by the human. The deviated value is any value deviating from the normal value. The maximum deviation corresponds to the maximum deviation that can occur. So the fitness of the solution is given by 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 agent's 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 at 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:

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

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 so we can give the chance to other useful actions and go away from the current bad actions.

5.3 Memory Application

Zhou [Zhou 90] presented the CSM (Classifier System with Memory) system that addresses 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 is to build a system in which a short and long term memory are simultaneously present. The short term memory is just 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, where 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 generalizer process takes control and compresses the acquired knowledge into a cluster of rules that are memorised for later use in the long term memory.

In our system, as the agent begins learning, it has no previous experience and the *Experience Bank* is empty. But as it begins learning by GA, it begins filling the memory with different rule bases of different users. Each stored rule base is consisting of the rules that were learnt and the actions (consequences) that were learnt by the GA.

After monitoring the user's action for period, the agent then matches the fired rules during this time to sets of rules bases for different users stored in the *Experience Bank*. The system tries to identify which rule base is appropriate to the user based on the basis of actions taken by him during the initial period, and the rule base that contains the most similar actions to the occupant 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 consequences 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 (5.1), 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 found point in the search space instead of starting from a random point.

By doing this, our system does not need the matcher calculations used by [Zhou 90]. This is because our system does not use the binary message coding, or "don't care", conditions but uses perfect matches, hence we don't need the generalizer. 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 get rid of some of the stored information as the acquired experience increases. However we don't favour this, as this means that some of the experiences acquired by the agent will be lost. So 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 much this rule base 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; derived from a "life invocation" flag, that is updated each time a rule is activated.

5.4 Using GA to Produce New Solutions

If the rule base extracted from the *Experience Bank* is not suitable for the user, the GA starts its search for new consequences for the poorly performing 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. If the result was a decrease in performance then the rules that have contributed most to this action will have their fitness less than the rules that have contributed less to this action. This allows us to go away from those points in the search space that cause no improvement or degradation in the performance. The parents for the new solution are chosen proportional to their fitness using the roulette-wheel selection process and the genetic operations of crossover and mutation are applied.

The proposed system can be viewed as a double hierarchy system in which the fuzzy behaviours are organised in a hierarchical form and the online learning algorithm is also a hierarchy. In 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* by finding how many rule in the stored rule base are similar to the user's action in the first two minutes. The highest fitness experience is used as a starting position to the lower level GA that is used to produce new solution to the current situation.

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 definition of the performance of the GA. In a non-stationary environment (which is our case), where the optimal solution changes with time, the GA should possess the capacity to track optimal solutions, too. The adaptation strategy needs to vary the control parameters appropriately whenever the GA is not able to track the located optimum. It is essential to have two characteristics in GA 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 determines P_c, P_m are given by :

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

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

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. The type of crossover was chosen to be one point crossover for computational simplicity and real time performance.

In [Srinivas 96] this method was superior to the simple GA and gave faster convergence of rate 8:1. We use this adaptive method for finding the values of crossover and mutation probabilities. This way leads to fast convergent solutions, and adapts the GA for non-stationary environments and relives the designer from determining these values, heuristically.

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, which is constrained by the *contextual prompter* according to the occupant's needs so that if, for example, a temperature is too high for the room occupier then the AGM won't suggest solutions 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 experiments with this Adaptive GA (AGA) with open range and AGA with constrained range and Simple GA (SGA) with constrained range for the problem of learning the comfort behaviour of different human attitudes as shown in Figure (8).

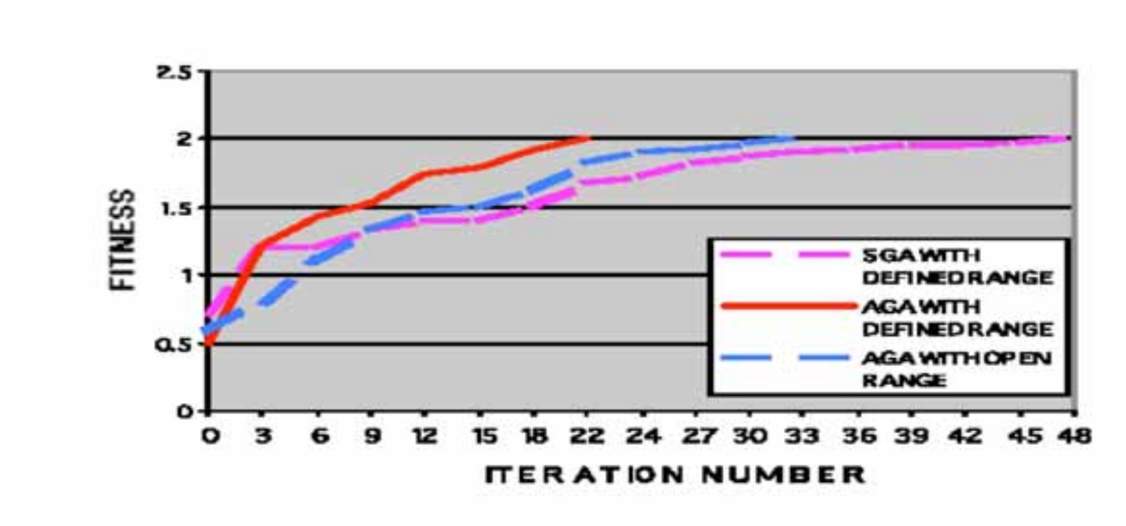

Figure (8): The best fitness plotted against the number of iterations for different learning.

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. And the best performing one was found to be p_c =0.7 and p_m=0.002 and is plotted in Figure (8). It was found that the one with constrained AGA converges to a solution in average after only 22 iterations. The AGA with open rage converges also but after larger number of iterations (33 iterations in average) as it needs more time to explore the search space and determine its limits this takes about 11 minutes of the agent time to converge to a solution. The SGA with defined limits and p_c =0.7 and p_m=0.002 converge to a solution after in average 48 iterations. These experiments justify that our proposed algorithm converges in a very short time interval thanks to the AGA and the constrained GA.

We use binary coding in the GA. For each rule there are two actions which are the 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 (9) shows a description of the GA operation in which the actions of rule number 5 and rule number 2 of the comfort behaviour are chosen for reproduction by roulette wheel selection due their high fitness. They have contributed more with their actions to the final action which caused improvement, or contributed less with their actions to final action which caused degradation. The adaptive crossover and mutation probabilites have been applied to both chromosomes. The resultant offspring were used to replace the consequences of rules 1 and rule 3 which were mostly blamed for the unstatisfactory responses.

5. Experimental Results

In our preliminary experiments we used an IB agent based on 68000 Motorola processor, the agent is equipped with light and heat sensors and actuators in the form of a heater and a light source; the IB agent is shown in Figure (10).

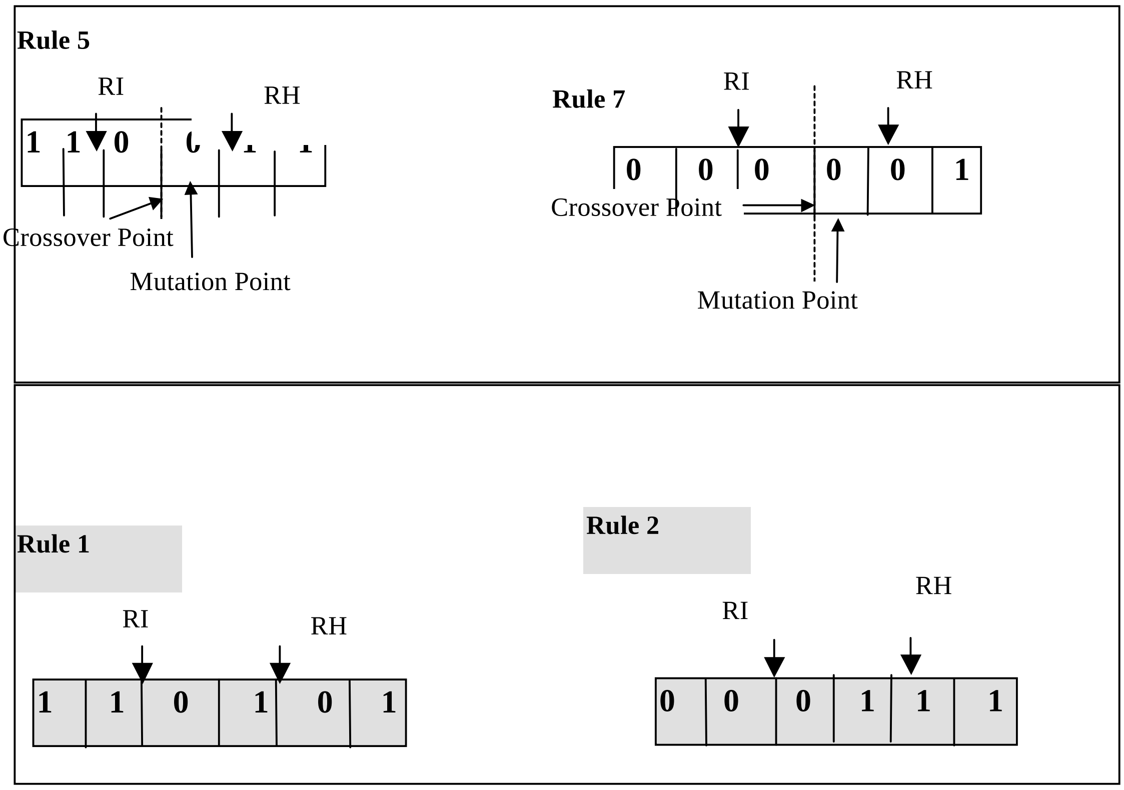

Figure (9): An example of GA processes in our proposed classifier system in which rule 5 and rule 7 (which were selected due to their higher fitness values) are generating new consequent for rules 1,2.

This agent is used as a prototype simulator to simulate the control of light and temperature in a room with various conditions such as multiple occupancy, different levels of natural light and temperature and different times of the day and different human desires. There is a built in Economy behaviour that should switch the heat low and ventilation off after the room is vacated. There is also a Safety behaviour that prevents the heat going below a minimum safe level (e.g. zero degrees which would result in pipes freezing). Centrally, there is the Comfort behaviour for the occupant themselves which will be learnt using our patented fuzzy-genetic techniques.

The agent deals in a proactive way with the occupant interacting with him to see if the action was completed satisfactory or not, and whether it is required to decrease or increase levels in question. The agent was tried, in an emulated environment, under different conditions such as hot sunny days and cold and dark days. The agent using our patented techniques was able to find a satisfactory rule base for different users in an average of 22 trials. Our method also optimised the number of rules by using only the rules that are important to the room occupier. The system also has the ability to adapt later on to add or delete rules and modify the actions of the existing rules interactively to the room occupant's wishes. Table (1) shows the rules obtained by our method and the Mendel-Wang method, which is an offline rule extraction method that outperforms the ANFIS network. Note that our method had optimised the number of rules from an expected $3^4 = 81$ rule base to only 49 rules. Also a problem with supervised learning techniques, such as Mendel-Wang, are that they need a person to supply a set of desired values. Which would be difficult since the values are not intuitive. If the person wants the heating turned set to what might be the equivalent of 37 % and the illumination to 40 % (taken as percentages of their maximum output) then finding or deriving an appropriate value from the device setting for the purposes of generating a training rule would be difficult. This is why, although the rules extracted by our method are similar to the Mendel-Wang a rule, the difference is that in our case we interact with the person until he is satisfied. Another advantage our method is that our system can quickly adapt to any change, starting from the closest previous solution, while the supervised learning techniques are off-line and repeat the learning cycle from the beginning and requiring a full set of training data.

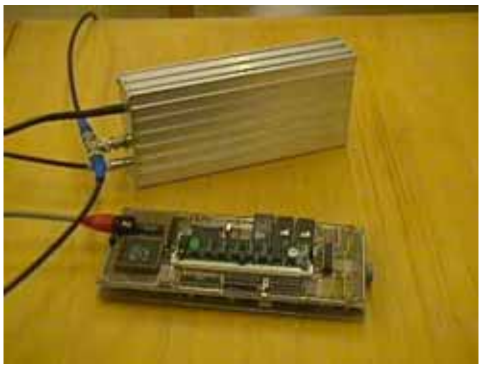

Figure (4): The IB agent

				Our Method		**Mendel-Wang Method**	
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 Low	Very Very Low	Norm
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 found by our method and by Mendel-Wang method.

6. Conclusion and Future Work

In this paper we have presented a novel online learning, adaptation and control algorithm based on a double hierarchical fuzzy genetic system. In this prototype IB agent this system is composed of three fixed behaviours - the Safety, Emergency and Economy behaviours and an adaptable rule set of Comfort behaviours that are adapted according to the occupants actual behaviour. The system interactively learnt an optimised rule base for the Comfort behaviour in a small set of interactions. The system produced similar rules to the rule base learnt by off-line supervised techniques but our system has many advantages such as the ability to adapt to changes in environmental conditions or the room occupant. It adapted in real time, on-line accommodating new data and rules, as they become known to the system. The system is memory based, possesses data, exemplar storage and retrieval capacities; leaning and improving through active interaction with the user and the environment. It includes parameters to represent short and long-term memory, age, forgetting, and analyses itself in terms of behaviour, error and success. Thus we would argue that our approach substantially meets the criteria for intelligent systems set out by researchers such as Kasabov & Steels and described earlier in this paper.

For our future work, we plan to move from an experimental bench based rig to IB equipped rooms with our intelligent, autonomous, agents and thereby extend the investigation to a richer set of environmental inputs which include a wider set of occupant behaviours. Also, we believe that an ideal IB agent should require little or no involvement from the occupant and thus another goals of our future work is further minimise the interactive aspects of the agent.

Acknowledgements

We are pleased to acknowledge the contribution from 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 Doling and Filiz Cayci with whom we have had many stimulating discussions on embedded-agent architectures.

References

[Bonarini 99] A. Bonarini, "Comparing Reinforcement Learning Algorithms Applied to Crisp and Fuzzy Learning Classifier systems", Proceedings of the Genetic and Evolutionary Computation Conference, pp. 52-60, Orlando, Florida, July, 1999.

[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 (CT'97), Japan 1997.

[Callaghan 2000] Callaghan V, Clarke, G, "*Buildings As Intelligent Autonomous Systems: A Model for Integrating Personal and Building Agents*", The 6th International Conference on Intelligent Autonomous Systems (IAS-6), Venice, Italy; July 25 - 27, 2000

[Coen 97] M.H.Coen, "Building Brains for Rooms: Designing Distributed Software Agents", Proc. Ninth Innovative Applications of AI Conf., AAAI Press, 1997.

[Davidsson 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.

[Hagras 99a] H.Hagras, Victor Callaghan, M.Colley, "A Fuzzy-Genetic Based Embedded-Agent Approach to Learning and Control in Agricultural Autonomous Vehicles", 1999 IEEE International Conference on Robotics and Automation, pp. 1005-1010, Detroit- U.S.A, May 1999.

[Hagras 99b] H.Hagras, Victor Callaghan, M.Colley, "Online Learning of Fuzzy Behaviours using Genetic Algorithms & Real-Time Interaction with the Environment", 1999 IEEE International Conference on Fuzzy Systems, Seoul-Korea, pp. 668-672, August 1999.

[Kasabov 98] N. Kasabov , "Introduction: Hybrid intelligent adaptive systems. International Journal of Intelligent Systems" Vol.6, pp.453-454,1998.

[Mozer 98] M. Mozer "The Neural Network House: An Environment That Adapts To Its Inhabitants". In Proc of American Association for Artificial Intelligence Spring Symposium on Intelligent Environments, pp. 110-114, 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", Proceedings of the 6th IEEE International Conference on Fuzzy Systems, Vol.1, pp. 573-578, Barcelona, Spain, 1997.
[Srinivas 96] M. Srinivas, L. Patnaik, "Adaptation in Genetic Algorithms", Genetic Algorithms for pattern recognition, Eds, S.Pal, P.Wang, CRC press, pp. 45-64, 1996.
[Sharples 99] S. Sharples, V. Callaghan, G. Clarke, "A Multi-Agent Architecture for Intelligent Building Sensing and Control" International Sensor Review Journal, May 1999.
[Steels 95] L. Steels, "When are robots Intelligent autonomous agents", Journal of Robotics and Autonomous Systems, Vol. 15, pp.3-9, 1995.
[Zhou 90] H. Zhou, " A Computational Model of Cumulative Learning", Machine learning Journal, pp. 383-406, 1990.