A Fuzzy-Immune Algorithm for Autonomous Robot Navigation

Hassan Abou Nasser[‡], **Gamal Selim**[‡], **Amr Badr**[†]

‡ AASTMT, Department of Computer Engineering & Science, †Cairo University, Faculty of Computers & Information

Corresponding Author:- ruaab@rusys.eg.net

ABSTRACT

Autonomous robot navigation can support humans in many applications. Autonomous navigation is a difficult problem due to the variability of the surrounding world. Fuzzy logic system (F.S.) has features that make it an adequate tool for addressing this problem. The fuzzy rules are collected according to the given application. In this paper, the parameters of the control rules are evolved using artificial immune system (A.I.S.) paradigm. The optimized model is used as an immune system to prevent the robot from illegal moves or collisions. The approach is tested over ten different simulated data and compared with traditional genetic algorithm (G.A). The results show that the proposed approach performs better with time limitation.

Keywords: Soft Computing, Autonomous Robot Navigation, Fuzzy Logic System (F.S.), Artificial Immune System (A.I.S.), Genetic Algorithm (G.A.), Intelligent Adaptative Control.

1 INTRODUCTION

One of the most important advances in computer science and electronics is the invention of autonomous mobile robotics. Autonomous mobile robots are physical systems that can navigate without human intervention between a starting point and a target point in unmodified environments; that is, in real-world environments that have not been specifically engineered for the robot (Ashlock, 2006)(Hermanu, 2004). Since real-world environments are usually full of obstacles and vagueness, building robust control techniques that reliably drive the robot safely to perform complex tasks within an acceptable time is the main challenge of today's autonomous robotics.

Current research on autonomous robot navigation (A.R.N.) is driven by the current gap between available technology and new application demands for robots in real-world environments that are full of imprecision. For example, current industrial robots lack flexibilities and autonomy (Driankov, 2001). Typically, these robots perform pre-programmed sequences of operations in highly constrained environments, and are not able to operate in new environments or even to face unpredicted situations. On the other hand, there is a clear emerging market for truly autonomous mobile robots (Saffiotti, 1997). For example, intelligent service robots for offices, hospitals, and factory floor; maintenance robots operating in hazardous or hardly accessible areas; domestic robots for cleaning or entertainment and so on, are all a possible applications.

The challenge of environmental uncertainties in designing autonomous mobile robots and other similar complex problems motivated computer scientists and engineers to develop technological paradigms to overcome these difficulties (Cernic, 1999). Some of these paradigms are classified as computational intelligence (C.I.) paradigms which are inspired from studies of natural systems. A definition for such techniques is that they comprise practical adaptation concepts, paradigms, algorithms and implementations that enable or facilitate appropriate actions (intelligent behaviour) in complex and changing environments. Soft computing (S.C.), a term coined by Lotfi Zadeh (Engelbrecht, 2003), is a different grouping of paradigms, which usually refers to the collective set of C.I. paradigms and probabilistic methods. C.I. paradigms include artificial neural networks (A.N.N.s), evolutionary computing (E.C.), swarm intelligence (S.I.), and fuzzy logic systems (F.S.s).

For the A.R.N. application in this paper, two main paradigms of C.I., namely F.S. and E.C. are considered. The basis for this application is a two-dimensional mobile robot simulator that is intended to aid developers implementing control and navigation logic (Hercock, 2003). This simulator, which is named Rossum's Playhouse (R.P.1) version 0.48, is designed by Gary Lucas (Lucas, 1999) in a very modular style and facilitates the transfer of developed code modules onto real hardware robot. It is selected as the basis for this application as it offers the essential features of a mobile robot simulator without excessive complexity. A number of developers are currently using the simulator to test and improve competition strategies for small robots (e.g., (Vassilis, 2001)).

This paper is organized as follows. Section 2 explores the fuzzy logic controller as a F.S. application in the A.R.N. problem. Genetic algorithms (G.A.s) are presented in section 3 as evolutionary computing algorithms (E.A.s) for evolving fuzzy rule base. Since immunology is now receiving more

attention and is slowly being realized as a new biologically inspired E.A. approach in C.I. paradigms (Ayara, 2002)(De Castro, 2002a)(De Castro, 2000a)(Gaspar, 1999)(Gonzalez, 2003), section 4 introduces the main A.I.S. concepts, while section 5 proposes a fuzzy-artificial immune algorithm for evolving a fuzzy rule base which is used to adaptively control simulated mobile robots. In section 6, the obtained results are compared with the results of G.A. techniques. Finally, conclusions are provided in section 7.

2 FUZZY-LOGIC CONTROLLERS

One highly successful C.I. theory that has emerged is F.S. set theory. F.S. is a system design technique that is based on how the brain deals with vagueness and uncertainty (Baturone, 2000). The key issue resolves around designing the required input and output rule sets, where domains are characterized by linguistic terms, rather than by numbers. F.S. has enjoyed wide popularity in engineering as an advanced control and A.I. technique (Hercock, 2003), particularly within the Japanese and Asian markets. However, it is only in the past two decades that it has been recognized in Europe and America as a useful A.I. system.

Recent work by several groups has shown that the qualitative nature of F.S. makes it a formal, useful and attractive tool, in terms of expressive power and flexibility, for constructing complex multilevel control structures and dealing with problems characterized by the pervasive presence of uncertainty (Baturone, 2000). F.S.s have been applied successfully to advanced control systems such as hydro-electrical power plants and robot navigation systems (Driankov, 2001)(Saffiotti, 1997). The design of fuzzy logic controllers was one of the largest application areas of F.S. set theory. Its strength lies in its ability to control a complex system, even if no precise mathematical model of the underlying processes is available. While F.S. is frequently described as computing with words rather than numbers, fuzzy logic control is described as control with sentences rather than equations (Driankov, 2001)(Saffiotti, 1997)(Baturone, 2000). Thus, instead of describing the control strategy in terms of differential equations, control is expressed as a set of linguistic rules. These linguistic rules are easier understood than systems of mathematical equations.

A fuzzy logic controller can be regarded as nonlinear static function that maps controller inputs onto controller outputs. A controller is used to control some system or plant. The system has a desired response that must be maintained under whatever inputs are received. The inputs to the system can, however, change the state of the system, which causes a change in response. The task of the controller is then to take corrective action by providing a set of inputs that ensures the desired response.

2.1 Components of Fuzzy Logic Controller

As illustrated in Figure 1, a fuzzy logic controller consists of four main components:

- Fuzzifier (Condition interface): The fuzzifier receives the actual outputs of the system, and transforms these non-fuzzy values into membership degrees to the corresponding fuzzy sets. In addition to the system outputs, the fuzzification of input values to the system also occurs via the condition interface.

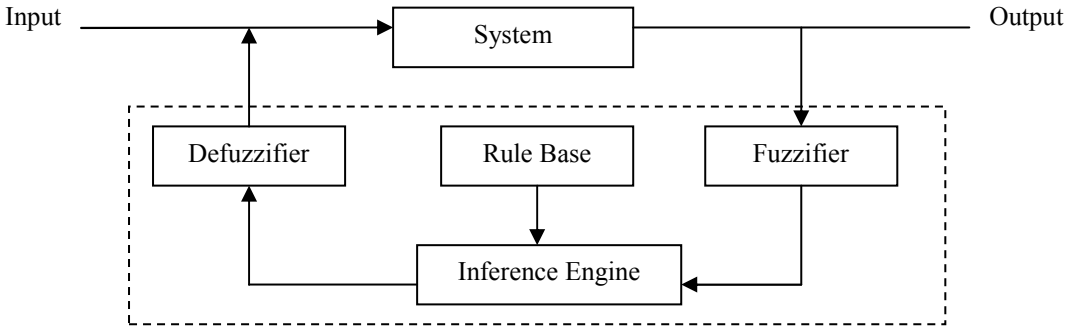

Figure 1: Fuzzy logic controller architecture

- Fuzzy rule base: The rule base, or knowledge base, contains the fuzzy rules that represent the knowledge and

experience of a human expert of the system.

- Inference engine (rule firing): The inference engine maps the fuzzified inputs (as received from the fuzzification process) to the rule base to produce a fuzzified output for each rule.
- Defuzzifier (Action interface): the defuzzifier converts the output of the fuzzy rules –i.e., outcome of inference engine- into a scalar or non-fuzzy value to be applied to the system.

In fact, a large number of design options exist based on these generic components. These are common features to all fuzzy-based control systems that may be classified in general as knowledge-based controllers. However, a number of additional features are required for an industrial control system, such as stability analysis of the final fuzzy rule set (Drainkov, 1996).

2.2 Mamdani Fuzzy Logic Controller

Mamdani and Assilian produced the first fuzzy logic controller in 1975. This type of fuzzy logic controller forms the basis for all types of linguistic fuzzy logic controllers and is derived directly from available heuristic control strategies mimicking the control knowledge of a human expert. While the system is statically described by rules (Engelbrecht, 2003), the output sets in Mamdani-type controllers are singletons (i.e., a single set), or combinations of singletons where the combinations are achieved through application of fuzzy set operators. Mamdani-type controllers follow the following simple steps:

1. Identify and name input linguistic variables and define their numerical ranges.
2. Identify and name output linguistic variables and define their numerical ranges.
3. Define a set of fuzzy membership functions for each of the input variables, as well the output variables.
4. Construct the rule base that represents the control strategy.
5. Perform fuzzification of input values.
6. Perform inferencing to determine firing strengths or weights of activated rules.
7. Defuzzify, using centroid of gravity, to determine the corresponding action to be executed.

Fuzzy logic controllers have the advantage of allowing two approaches to their construction. For applications where the human expert knows what action the system should take given particular input variable scenarios, the human expert can usually specify the system parameters, and they are static. This contrasts with A.N.N.s approaches, where the user is in no position to specify weights values, even if the appropriate system response is known. On the other hand, if the fuzzy system response is not known, the system parameters can be adapted or refined until the overall system response matches the desired behaviour through learning using E.A.s or A.N.N.s strategies. For the A.R.N. application, the second approach has been adopted proposing A.I.S. to adapt the fuzzy logic controller so that the F.S. performance fits the desired observed behaviour.

3 GENETIC ALGORITHMS (G.A.S)

Physics, Biology, Economy or Sociology often have to deal with the classical problem of optimization (Corne, 1999). Purely analytical methods widely proved their efficiency. Other methods, combining mathematical analysis and random search have appeared such as G.A.s. Not only do G.A.s provide an alternative method for solving problems, they consistently outperform other traditional methods in most of these problems (Coley, 2001). G.A.s are numerical optimization and adaptive heuristic general search algorithms inspired by both natural selection and natural genetics. The basic concepts used for designing G.A.s follow the principle of survival of the fittest, first laid down by Charles Darwin. They are used in computing to find true or approximate solutions to optimization and search problems where the algorithms are simple to understand and the required computer code easy to write (Engelbrecht, 2003)(Coley, 2001).

The version of G.A. outlined below borrows heavily from Goldberg (Welstead, 1994). The steps in the algorithm are straightforward and not difficult to describe:

0. Initialize the algorithm. Randomly initialize each individual chromosome in the population of size N (N must be even), each of same

length (m bits) and compute each individual's fitness.

1. Select N/2 pairs of individuals for crossover. The probability that an individual will be selected for crossover is proportional to its fitness.
2. Perform crossover operation on N/2 pairs selected in step 1 at some randomly selected point along each chromosome. Randomly mutate bits with a small probability during this operation.
3. Compute fitness of all individuals in new population.
4. Optional: Select N fittest individuals from combined population of size 2N consisting of old and new populations pooled together.
5. Rescale fitness of population.
6. Determine maximum fitness of individuals in population. If |max fitness-optimum fitness| < tolerance then STOP. Otherwise, go to step 1.

Designing a suitable G.A. for a real-world task is a nontrivial exercise, almost an art. The interactions between representation, recombination, mutation, and selection are a complex balance between exploitation and exploration. This point, in addition to G.A.'s very high consumption of computational resources, has been the source of active research to discover alternative schemes and evolutionary strategies (Gaspar, 1999).

4 ARTIFICIAL IMMUNE SYSTEMS (A.I.S.S)

Similar to the way the nervous system inspired the development of A.N.N., the natural vertebrate immune system has now led to the emergence of a new computational paradigm for C.I. community named A.I.S. (Gonzalez, 2003)(De Castro, 2002b)(De Castro, 2002c). A.I.S. is defined as an adaptive system inspired by theoretical immunology and observed immune functions, components, principles and models in order to build novel C.I. tools to solve complex computational or engineering problems in a vast range of domain areas. A number of successful A.I.S. implementations have been applied in pattern recognition, optimization and many others (De Castro, 2002a)(Corne, 1999)(McCoy, 1997). However, even in the most complex A.I.S.s only a fraction of the functionality of the biological immune system is exploited. Typically, the antibody-antigen interaction coupled with rapid (or somatic) hypermutation, form the basis for many A.I.S. applications (De Castro, 2000a).

Antigenic recognition is the first pre-requisite for the immune system to be activated and to mount an immune response. The recognition has to satisfy some criteria. First, the cell receptor recognizes an antigen with a certain affinity, and a binding between the receptor and the antigen occurs with strength proportional to this affinity. If the affinity is greater than a given threshold, which is called the affinity threshold, then the immune system is activated. The nature of the antigen, type of recognizing cell, and the recognition site also influence the outcome of an encounter between an antigen and a cell receptor.

One can identify two major groups of immune cells: B (bone) cells and T (thyroid) cells. The B-cells originate in the bone marrow and enter the lymphatic network. The function of a B-cell is to detect foreign antigens and assemble antibody proteins marking the antigen for deletion by other immune system products. Each B-cell can only produce one particular antibody (A.b.). If a B-cell encounters a nonself antigen with sufficient affinity, it is excited with the aid of a helper T-cell, so it rapidly proliferates and differentiates into memory cells -that are more sensitive to the triggering antigen than the original B-cell with long life spans to provide a faster response in the future - and effector (or plasma) cells that also produce antibodies specific to the triggering antigen; a process named clonal selection (or clonal expansion) (De Castro, 2000a). In contrast, if a B-cell recognizes a self-antigen, it might result in suppression, as opposed by the immune network theory (De Castro, 2002a)(De Castro, 2001).

In the thymus, T-cells mature into two classes: helper and killer. During this maturation, all T-cells that recognize self-antigens are excluded from the repertoire of T-cells; a process termed negative selection. The helper T-cells aid the growth of B and T cells and produce a variety of proteins useful in the immune system. The killer T-cells attack cells of the body that become infected. The antigens of a sick cell are typically different from a healthy one, so the T-cells can recognize and eliminate the infected cell.

4.1 Clonal Selection

In the literature, some authors (Forrest, 1993) have argued that a G.A. without crossover is a reasonable model of clonal selection. However, the standard G.A. does not account for important properties such as affinity proportional reproduction and mutation. Other authors (De Castro, 2000b) proposed a clonal selection algorithm, named CLONALG to fulfill these basic processes involved in clonal selection. This algorithm was initially proposed to perform pattern recognition and then adapted to solve multi-modal optimization tasks (Gonzalez, 2003).

Given a set of patterns to be recognized (P), the basic steps of the CLONALG algorithm are as follow:

0. Randomly initialize a repertoire of antibodies (M).

1. For each pattern of P, present it to the repertoire M and determine its affinity with each antibody of the repertoire M.
2. Select N1 of the best highest affinity antibodies of M and generate copies of these antibodies proportionally to their affinity with the antigen.
3. Mutate all these copies with a rate inversely proportional to their affinity with the input pattern.
4. Add these mutated antibodies to the repertoire M and reselect N2 of these maturated antibodies to be kept as memories of the system.
5. Repeat step 1 to 4 until a certain criterion is met.

5 PROPOSED FUZZY-IMMUNE ROBOT NAVIGATION ALGORITHM

5.1 Fuzzy Logic Control

Part of the mobile robot autonomy problem is how to merge the low-level numeric processing involved in reactive systems with a knowledge representation of the tasks a user wishes to specify to the robot. A F.S. has features that can bridge the gap between low-level behaviours and the necessary reflective task-oriented processes acting on the robot in real-world problems, where the information is often incomplete, imprecise, vague, unreliable or uncertain, alleviating the symbol grounding problem faced by classical AI. By intelligently managing the interactions of multiple primitive behaviours, a fuzzy logic controller raises the competence level of an autonomous robot, which can greatly reduce the cognitive workload on a robots planning system. An example is the ability to recover from situations that commonly trap purely reactive systems, such as box canyons. In particular, an agent must be able to suppress behaviours that are no longer producing a useful response, over a timeframe, which is context-dependent.

A F.S. has a number of parameters, such as the fuzzy sets used for input and output variables, the membership functions that define the fuzzy sets, and the structure and entries in the fuzzy associative memory (F.A.M.) matrix (or fuzzy rules) which determine the action of the F.S.. Some F.S.s also include parameters that assign a weight to each fuzzy rule, or FAM matrix entry, indicating its relative importance in the overall system output. All of these parameters are candidates for adaptation with an optimization algorithm. While it is possible to design a system that adapts all of these parameters, this is a daunting task for the code developer. Common sense can provide pretty good estimates for fuzzy sets and membership functions to be associated with each variable. The F.A.M. matrix entries have the most influence in determining system output, so adapting these will be focused on.

In order to create an adaptive data processing system, F.S. effectively embeds expert knowledge into a set of flexible overlapping rules. The use of a fuzzy rule base allows a direct linguistic description of the particular relationships the developer requires between any given behaviours. It is therefore provides the tools to develop software products that model human reasoning (also referred to as approximate reasoning). Fuzzy rules are normally taking the form:

If (x is A) and (y is B) then (z is C) (1)

for x, y, z as linguistic variables representing inputs and outputs of the fuzzy logic controller, and A, B and C are the terms of the variables, in the universes of discourse X, Y, and Z. For example, the rule base may specify "*If robot NEAR to obstacle THEN apply LARGE negative feedback to navigate to goal behaviour",* where the objective of the robot is to focus on its current response while maintaining some awareness of its final goal. This represents a form of "attention span" that allows the mixing of short-and medium-term goals, in a continuous manner. An example of this is when the robot enters a potential well while navigating toward a target. As the robot approaches an obstacle, the importance of avoiding it increases due to the utility-response generated by the fuzzy rule base which turns down the utility of moving toward the target. So, by applying a small bias to one of the rotate behaviours, the emergent response for the robot is to follow the perimeter of the obstacle until a free path toward the target is found then it returns to navigate in that direction.

As stated, a general-purpose F.S. works by encoding an expert's knowledge into a set of rules, which are smoothly interpolated, and the resultant is defuzzified to give crisp actuation output. Each rule is specified as either a triangular, trapezoid, logistic or some other functions (e.g., bell shape), and assigned to some range of input variable. It is the task of the human expert of the domain to define the function which captures the characteristics of the fuzzy set.

In this research, trapezoid functions were selected since these membership functions are computationally efficient in real-time control systems (Hercock, 2003). In A.R.N. application design, 4 input variables (robot position, robot velocity, robot angle, and angle velocity) and one output variable (force applied to move robot wheels) have been identified. Simplicity of design has been imposed, so the same 5 fuzzy sets for all inputs and output variables have been incorporated ["Positive Large" (P.L.), "Positive Medium" (P.M.), "Zero" (ZE.), "Negative Medium"

(N.M.), and "Negative Large" (N.L.)] with symmetry and 25% overlap degree criteria (Drainkov, 1996) (Kosko, 1992). A range of 5 output rules is a common choice for a fuzzy logic control application as this provides adequate resolution without excessive computational cost. Fuzzy sets are physically defined as N.L. (from 0.0 to 0.3), N.M. (from 0.25 to 0.5), ZE. (from 0.4 to 0.6), P.M. (from 0.55 to 0.8), PL (from 0.75 to 1.0) but simplified as 0.0, 0.25, 0.5, 0.75 and 1.0 respectively. Moreover, 50 fuzzy rules with equal weighting have been used.

An alternative method to store rules in a multidimensional array of fixed dimension can be achieved by allowing each rule to carry with it index information that specifies its location in a virtual multidimensional array. In other words, higher-dimensional spaces are separated into submatrices of two dimensions. With this approach, no need to allocate the full array. Only the actually used entries in the array will be stored. Moreover, this approach will allow a variable number of inputs and flexible F.A.M. matrix structuring where the rule entries are stored consecutively and the order in which they are stored is not significant. For a low cost in storing the index information using this approach, huge benefits in flexibility and ease of implementation will be obtained. In addition to having the ability to handle F.A.M. matrices of arbitrary dimension (up to a preset maximum), the added bonus of being able to handle multiple lower-dimensional F.A.M. matrices simultaneously can be achieved. Finally, this approach results in a very compact implementation of the defuzzification system.

Using a F.A.M. representation (Kosko, 1992), the firing strength or weight for the i^{th} F.A.M. entry is calculated using the minimum rule:

$$w_i = \min\{\mu_A(x_i), \mu_B(y_i)\} \quad (2)$$

where x ∈ A, y ∈ B and they represent the input dimensions of the F.A.M. matrix. If each output fuzzy membership set is defined as:

$$\mu_C(z_i) = \text{output fuzzy membership set} \quad (3)$$

where z ∈ C output 'universe of discourse' and C is composed of a finite set of discrete values, then the total defuzzified response for n output membership sets is calculated as:

$$output = \frac{\sum_{i=1}^{n}(w_i \times \mu_C(z_i))}{\sum_{i=1}^{n} w_i} \quad (4)$$

where $(w_i \times \mu_C(x_i))$ is the height of the clipped set of output variable C, and n equals the number of active rules at a given time. This is generally referred to as "height defuzzification" (or clipped center of gravity). In real-time control, this method is computationally efficient compared to other methods such as the "fuzzy centroid" or "center of gravity" defuzzification scheme, in which the output is a combination of centroids for each overlapping fuzzy membership function. In addition, this method generates a unique fuzzy centroid and utilizes better use of the information in the output distribution. However, by taking the simplified case of assigning a singular discrete value to each output membership set, the computational requirements can be reduced in exchange for a small sacrifice in resolution of the output fuzzy action surface. In other words, the fuzzy membership function assigns the value 1 to a single point and 0 to all other points. Alternatively, the output universe of discourse can be considered as it consists of a finite number of discrete values. Most F.S. implementations use a similar assumption. This does not mean that the F.S. output is limited to a finite number of values (Welstead, 1994).

5.2 Evolved Fuzzy Logic System

The design of the parameters and rule structure of a fuzzy logic controller for a particular real-world application can be considered as time consuming and knowledge-intensive due to the rapid increase in the possible combinations of rules, along with the number of input and output variable dimensions or number of sets per variable. Moreover, a F.S. normally contains no learning ability in itself. Therefore it is a suitable domain to apply E.A. or A.N.N. techniques to find an optimum set of rules and hence form a hybrid system (Gonzalez, 2003)(Nasaroui, 2002). In this problem, A.I.S. paradigm is proposed to evolve the control parameters of the fuzzy logic controller that is designed for a simulated robot.

The problem addressed in this work is how to automate the process of allocating the output rules for a fuzzy logic robot control algorithm. The encoding scheme for A.I.S. paradigm here takes each FAM matrix entry (or output rule) and assigns it an integer value in the range 5-9 for the output sets N.L., N.M., ZE., P.M., and P.L. respectively. These indices are then strung together to form a single fixed length array of 50 integer elements that constitutes the antibody where the repertoire used consist of 44 antibodies, see Figure 2. Similarly, the integer values in the range 0-4 are used to index the input sets in fuzzy rules. Each complete F.A.M. matrix (or antibody) is then

evaluated against the affinity function and the normal processes of affinity maturation and clonal selection are applied.

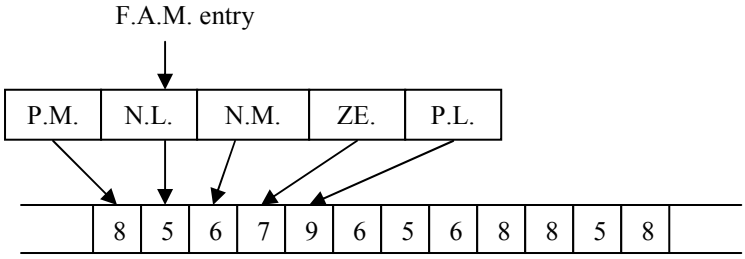

Figure 2: An A.I.S. antibody showing the encoding for a F.A.M. matrix using an integer representation

The affinity function in this application is clearly how well the robot manages to complete its specified tasks, namely maneuver around the environment and navigate toward a target t1 in Figure 3. A simple affinity function $f(t)$ can be defined in this case as the distance covered by the robot, measured by the linear range of its target sensor, and a measure of the time taken to perform the task (normalized for 1.0 = max affinity):

$$f(t) = 1 - \left[\frac{r}{mr} + \frac{t}{mt} \right] \quad (5)$$

where r is the minimum final distance between the robot and the target, mr is the maximum range to target, t is the time spent in a run, and mt is the maximum time available for a run. A range of possible affinity functions could be applied to this problem with equal effect, using other metrics of the robot performance.

A set of termination criteria for the robot are needed. Hence, if any of the robot collision sensors fire, then the current run is terminated and the robot is returned to the starting point to begin a new run, with a new antibody. Similarly, if the robot's obstacle sensors return a value below some threshold value or a time-out criterion is reached, then the run ends. The sequence of operations required for these tasks is as the following (see Figure 4):

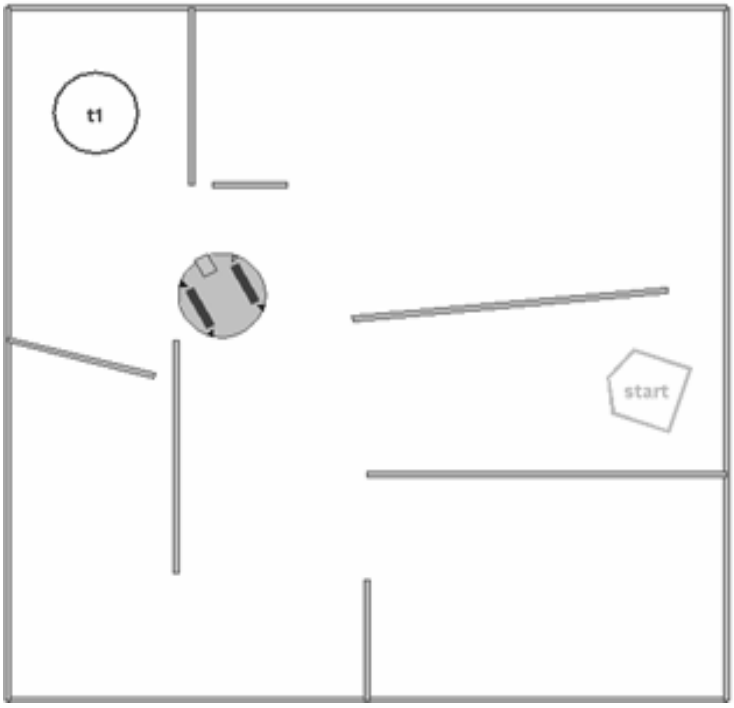

Figure 3: Example of G.U.I. interface to robot simulator

0. Initialize simulation, robot, and A.I.S. algorithm.
1. Run through A.I.S. repertoire of N antibodies, and assign each antibody to a F.A.M.
2. Start a run of the robot simulator system, using the F.A.M. to modulate the strength of the interactions between the conflicting

obstacle avoidance and target navigation behaviours.

3. If robot fails to complete task, assign an affinity value to the current antibody based on the affinity function.
4. Repeat from step 1, for M runs or until a specified affinity level is reached.

repertoire/population size, five experiments are conducted. In each of these five experiments, the system is allowed to run for 100 generations. Two different parameters are recorded for each generation, namely: best affinity/fitness and average affinity/fitness. The average of these two parameters are then computed for the five experiments conducted,

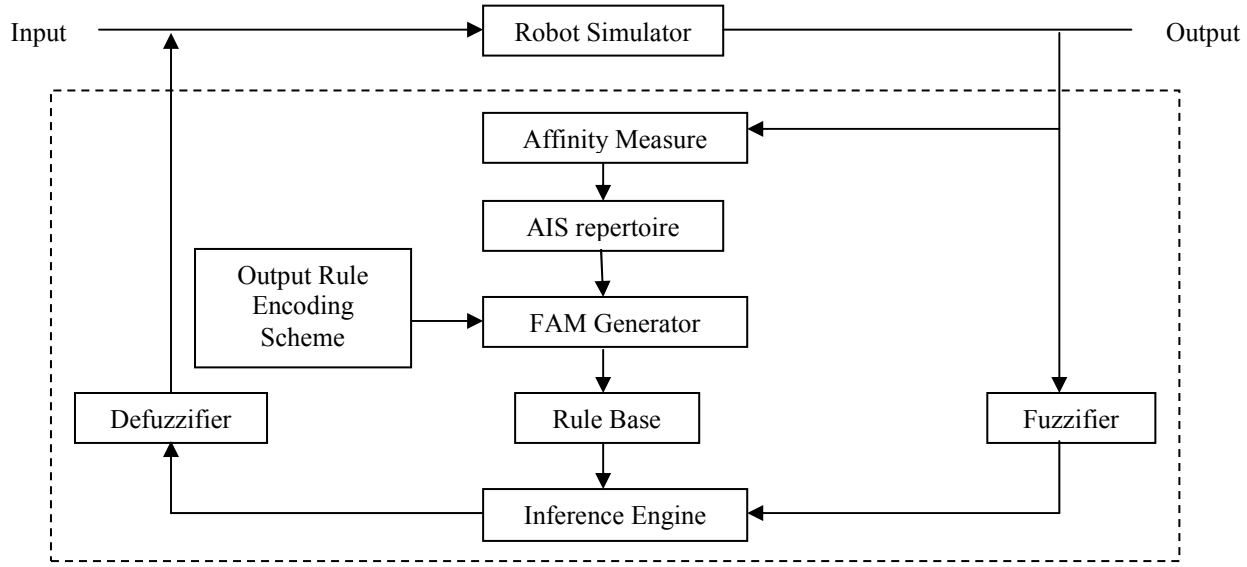

Figure 4: Evolved fuzzy logic controller architecture using AIS paradigm

6 SIMULATION & RESULTS

The following results were generated from a series of trial runs on the robot simulator mentioned above with a 1.6 GHz system, running Windows XP SP2. The trial runs involved using the proposed A.I.S. algorithm against the G.A. to evolve a set of fuzzy output rules for the fuzzy logic controller. The performance of the algorithms developed is measured using two different repertoire/population sizes. In the first set of experiments, a repertoire/population size of 44 antibodies/chromosomes is used. In the second set of experiments, a repertoire/population size of 24 antibodies/chromosomes is used. For each resulting in 100 values for each parameter. A time series of these two averaged parameters is then plotted using the number of generations for time axis. Specifically, the average affinity is computed for the five experiments that were conducted using the A.I.S. paradigm and the average fitness is computed for the experiments conducted using the G.A.

Figure 5 shows the results when a repertoire/population size of 44 is used. As can be seen the best fuzzy rules evolved using A.I.S. paradigm (i.e., best affinity curve) are evolved faster than the best fuzzy rules evolved using G.A. (i.e., best fitness curve). The reason is that the A.I.S. paradigm

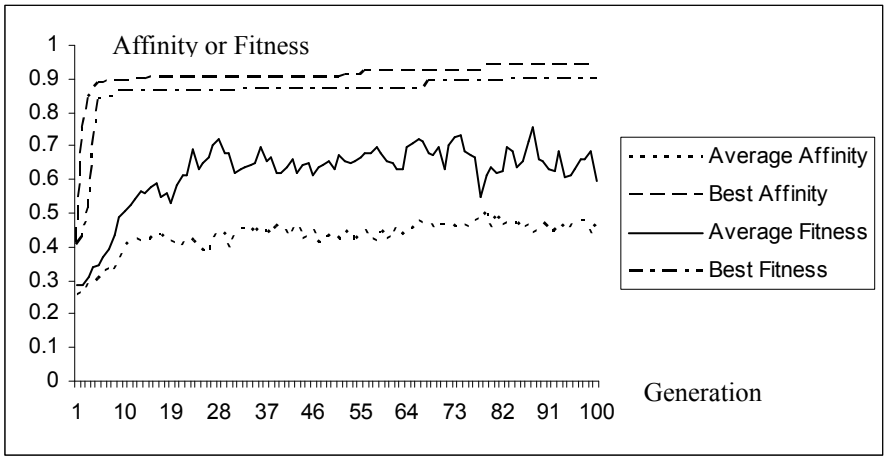

Figure 5: Example of average and best affinity/fitness data for a repertoire/population of 44 antibodies/chromosomes

keeps a memory for maturated antibodies in order to give faster response whereas G.A. does not. However, the same figure shows that the average fitness of the G.A. is better than the average affinity of the A.I.S. paradigm. This is due to the fact that the A.I.S. paradigm applies rapid hypermutation inversely proportional to antibody affinity (30% for the worst affinity) whereas G.A. applies small fixed mutation rate (6% is used) for all chromosomes in population.

The effect of having a too small repertoire/population of antibodies/chromosomes is shown in Figure 6, which may fail to provide sufficient diversity in antibodies/chromosomes for the best fuzzy rules to be evolved quickly. It requires significantly more generations -with a repertoire of 24 antibodies or a population of 24 chromosomes- for the robot to achieve the same required task.

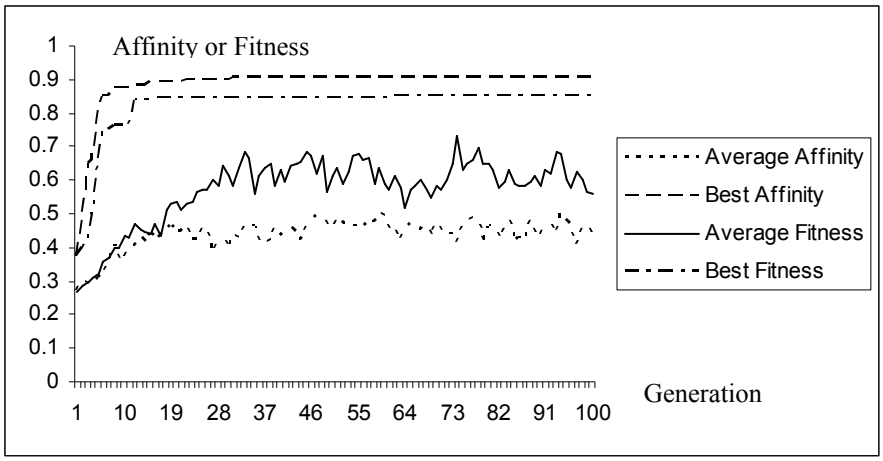

Figure 6: Example of average and best affinity/fitness data for a repertoire/population of 24 antibodies/chromosomes

As can be seen in Figure 5, the best affinity reached a value of 0.88 and the best fitness reached the value of 0.83 after five generations. For the same number of generations, Figure 6 shows that best affinity reached a value of 0.78 and the best fitness reached the value of 0.62. This clearly indicates that a larger repertoire/population size enables the algorithms to converge to the final solution faster than when a smaller population size is used. It can also be seen that at the end of the experiments, after 100 generations, the final value of the average affinity/fitness and best affinity/fitness is closer to the optimal in experiments that used 44 antibodies/chromosomes than the experiments that used 24 antibodies/chromosomes.

Figures 5 and 6 show that the average affinity curve of A.I.S. paradigm is almost the same for the two repertoire sizes used, whereas the average fitness curve of G.A. in Figure 5 converges faster than the average fitness curve in Figure 6. The reason is that the A.I.S. paradigm incorporates more exploitation than exploration whereas G.A. compromise between them.

7 CONCLUSIONS

F.S. has been shown to provide a powerful and flexible mechanism for processing numeric information using linguistic rules and offers significant advantages over conventional control methods. As can be seen from the results of the evolved fuzzy logic controller, A.N.N. or E.A. techniques are usually used along with F.S. to enable automatic parameter design in a complex control application.

Several experiments have been conducted with different repertoire/population sizes. Small repertoire/population size does not provide sufficient diversity in antibodies/chromosomes for the optimum fuzzy rules to be evolved quickly. However, using A.I.S. paradigm has been shown to outperform G.A. on the short run, whereas G.A. outperforms A.I.S. on the long run. The reason is that A.I.S incorporates more exploitation over exploration for the universe of discourse, whereas G.A. compromises between them.

REFERENCES

Ashlock, D.A., Manikas, T.W., and Ashenayi, K. (2006), "Evolving a Diverse Collection of Robot Path Planning Problems", in *Proc. IEEE Congress on Evolutionary Computation (CEC2006)*: 1837-1844.

Ayara, M., Timmis, J., De Lemos, L., De Castro, R., and Duncan, R. (2002), September, "Negative selection: How to generate detectors", in *1st International Conference on Artificial Immune Systems*, (J. Timmis and P. Bentley, eds.), UNIVERSITY OF KENT AT CANTERBURY PRINTING UNIT, University of Kent at Canterbury: 89-98.

Baturone, I., Barriga, A., Sanchez, S. Solano, Jimenez, C. J. Fernandez, and Lopez, D. R. (2000), *Microelectronic design of fuzzy logic-based systems*, CRC INC. Press, Boca Raton, FL, USA.

Cernic, S., Jezierski, E., Britos, P., Rossi, B. and García Martínez, R. (1999). "Genetic Algorithms Applied to Robot Navigation Controller Optimization", in *Proceedings of the International Conference on Intelligent Systems and Control*, Páginas: 230-234.

Coley, D. A. (2001), *An Introduction to Genetic Algorithms for Scientists and Engineers*, WORLD SCIENTIFIC PUBLISHING CO. PTE. LTD, Singapore.

Corne, D., Dorigo, M., Glover, F., Dasgupta, D., Moscato, P., Poli, R., and Price, K. V. (eds.) (1999), in *New ideas in optimization*, McGRAW-HILL LTD., Maidenhead, UK, England.

De Castro, L. N. and Zuben, F. J. V. (2000a), August, "The clonal selection algorithm with engineering applications", in *Artificial Immune Systems*, Nevada, USA: 36-39.

De Castro, L. and Timmis, J. (2002b), January, "Artificial Immune Systems: A Novel Paradigm to Pattern Recognition", in *Artificial Neural Networks in Pattern Recognition*, (L. A. J Corchado and C. Fyfe, eds.), University of Paisley: 67-84.

De Castro, L. N. and Timmis, J. (2002c), *Artificial Immune Systems: A New Computational Intelligence Approach*, SPRINGER-VERLAG, London.

De Castro, L. N. and Von Zuben, F. J. (2001), "aiNet: An Artificial Immune Network for Data Analysis", (full version, pre-print), in *Data Mining: A Heuristic Approach*, (H. A. Abbass, R. A. Sarker, and C. S. Newton, eds.), IDEA GROUP PUBLISHING, USA: 231-259.

De Castro, L. N. and Timmis, J. (2002a), September, "Hierarchy and convergence of immune networks: Basic ideas and preliminary results", in *Proceedings of the 1st International Conference on Artificial Immune Systems (ICARIS)*, (J. Timmis and P. J. Bentley, ed.), UNIVERSITY OF KENT AT CANTERBURY PRINTING UNIT, University of Kent at Canterbury: 231-240.

De Castro, L. N. and Von Zuben, F. J. (2000b), "The Clonal Selection Algorithm with Engineering Applications", in *GECCO'00 – Workshop Proceedings*, 36-37.

Drainkov, D., Hellendoorn, H., and Reinfrank, M. (1996), "*An Introduction to Fuzzy Control*", SPRINGER-VERLAG, Berlin.

Driankov, D. and Saffiotti, A. (eds.) (2001), Fuzzy Logic Techniques for Autonomous Vehicle Navigation, PHYSICA-VERLAG, New York.

Engelbrecht, A. P. (2003). *Computational Intelligence*, JOHN WILEY & SONS. LTD., Chichester, England.

Forrest, S., Javornik, B., Smith, R. E. & Perelson, A. S. (1993), "Using Genetic Algorithms to Explore Pattern Recognition in the Immune System", in *Evolutionary Computation*, 1(3):191-211.

Gaspar, A. and Collard, P. (1999), June-September, "From GAs to artificial immune systems: Improving adaptation in time dependent optimization", in *Proceedings of the Congress on Evolutionary Computation*, (P. J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzala, eds.), IEEE Press, Mayflower Hotel, Washington D.C., USA: 1859-1866

Gonzalez, F. A. (2003), "A study of artificial immune systems applied to anomaly detection", PhD thesis, Major Professor-Dipankar Dasgupta.

Hercock, R. Ghanea (2003), *Applied Evolutionary Algorithms in Java*. SPRINGER-VERLAG LTD., London.

Hermanu, A., Manikas, T.W., Ashenayi, K., and Wainwright, R.L. (2004), "Autonomous Robot Navigation Using a Genetic Algorithm with an Efficient Genotype Structure", in *Intelligent Engineering Systems Through Artificial Neural Networks: Smart Engineering Systems Design: Neural Networks, Fuzzy Logic, Evolutionary Programming, Complex Systems and Artificial Life*, (C.H. Dagli, et al., ed.), ASME Press, New York: 319-324.

Kosko B. (1992), *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*, PRINTICE HALL, USA.

Lucas, Gary (1999). "RP1: The Rossum's Playhouse Mobile-Robot Simulator", sourceforge, Trinity College, 2006, http://rossum.sourceforge.net/sim.html

McCoy, D. F. and Devaralan, V. (1997), "Artificial Immune Systems and Aerial Image

Segmentation", in *Proceedings of the SMC'97*: 867-872.

Nasaroui, O., Gonzalez, F., Dasgupta, D. (2002), "The fuzzy artificial immune system: motivations, basic concepts, and application to clustering and Web profiling", in *Proceedings of the IEEE International Conference on Fuzzy Systems*, Honolulu, HI, USA: 711-716.

Saffiotti, A. (1997). "The uses of fuzzy logic for autonomous robot navigation: a catalogue raisonn'e". *Soft Computing Research journal*, 1(4): 180-197.

Vassilis, V. (2001), "Robot Localization and Map Construction Using Sonar Data", http://rossum.sourceforge.net

Welstead, S. T. (1994), *Neural Network and Fuzzy Logic Applications in C/C++*, JOHN WILEY & SONS INC., USA.