M. Hapke, M. Komosinski, and D. Waclawski. Application of evolutionarily optimized fuzzy controllers for virtual robots. In *Proceedings of the 7th Joint Conference on Information Sciences*, pages 1605-1608, North Carolina, USA, September 2003. Association for Intelligent Machinery.

Application of Evolutionarily Optimized Fuzzy Controllers for Virtual Robots

Maciej Hapke, Maciej Komosinski, Dawid Wacławski

Poznań University of Technology
Institute of Computing Science
Piotrowo 3a, 60-969 Poznań, Poland

maciej.hapke@cs.put.poznan.pl

Abstract

This paper describes our work on a fuzzy controller for virtual robots in the Framsticks system. The fuzzy system controller processes signals from sensors to actuators. Its parameters are tuned during evolution by genetic operators to achieve success in simple optimization tasks. The representation for the fuzzy control system, evolutionary operators and an evaluation function are defined. The best evolved control systems are verified in agents within the Framsticks simulator [Komosinski and Ulatowski, 1997a], and these results are presented.

1. Introduction

Many successful implementations have exposed the power of fuzzy control. Nowadays, video cameras, air-conditions systems, car ABS, subway and many others systems are controlled by microcomputers using fuzzy logic.

The aim of this work was to verify the use of fuzzy logic controllers for controlling the behavior of virtual creatures in a specific 3D simulation and optimization environment – the Framsticks system [Komosinski and Ulatowski, 1997a].

In this approach fuzzy controllers are built during evolutionary optimization. The following sections introduce the Framsticks simulator, describe the representation of a fuzzy control system, evolutionary operators and an evaluation function. Then, two experiments with simple Framsticks agents ("walker" and "stand-up") are presented and their results are summarized.

2. Framsticks virtual world

Framsticks [Komosinski and Ulatowski, 1997a] is a system for simulation and optimization of three-dimensional agent structures and their control systems. It allows users to perform pre-defined experiments, but the system is open and allows creating user-defined experiment setups, fitness functions, and neurons. A scripting language is provided for these purposes. Therefore Framsticks is suitable for testing various research hypotheses where fast 3D simulation and evolutionary optimization is required.

The physical structure of Framsticks agents is made of parts (material points) and joints. The control system is made of neurons (including sensors and actuators) and their connections. In this work, only a subset of system features was used: physical structure was fixed (not optimized), and there was a homogeneous, fuzzy control system (ie. it was not composed of many heterogeneous neurons with changing topology).

Framsticks supports multiple genetic representations and operators. These representations range from simple and direct descriptions of agents to the ones encoding developmental process in genotypes [Komosinski and Rotaru-Varga, 2001]. In this work, the direct encoding was used, and special genetic operators were designed to provide mutation and crossing over of a fuzzy control system.

The Framsticks software is available for Linux and MS Windows, both as Graphical User Interface programs and as command line programs. Parts of the C++ source, mainly those concerning genetics (including the work described in this paper) are also available within the GDK (Genotype Developer Kit) [Komosinski and Ulatowski, 1997b].

3. Fuzzy controlled virtual creatures

Fuzzy systems applied in this work are based on the Mamdani model, i.e. both premises and conclusions of fuzzy rules are described by fuzzy variables. A very valuable feature of a fuzzy system is that one can infer a new conclusion from a new hypothesis. A fuzzy system is thus a kind of a function of many

variables, where input signals are first fuzzily evaluated by particular fuzzy rules, the results of firing rules are then aggregated, and the resulting fuzzy set is defuzzified [Michalewicz 2000; Ross 1995; Yager and Filev 1994].

The classical Mamdani model is composed of multiple input and single output variables. A multiple input – multiple output (MIMO) fuzzy system is a kind of a fuzzy rule-based system, in which premise and conclusion parts are compound of many inputs and outputs. It is a set of similar fuzzy systems with different conclusions.

In our approach, fuzzy sets are represented in a trapezoidal form, each fuzzy set is defined by four real numbers within the domain of [–1,1]. Figure 1 presents an example of fuzzy sets, which describe Framsticks' touch sensor.

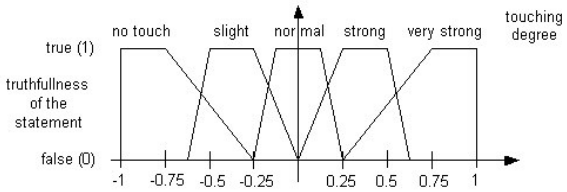

Figure 1. Example of fuzzy sets for a touch sensor

For example, let us consider a fuzzy rule-based system with two inputs (x0, x1), two outputs (y0, y1), two rules (R0, R1) and five fuzzy sets (F0 .. F4):

```
F0={-0.35; 0.05; 0.4; 0.65}
F1={-1; -0.8; -0.8; -0.35}
F2={0.2; 0.5; 0.7; 0.8}
F3={-0.65; -0.5; -0.3; 0.1}
F4={0.4; 1; 1; 1}
R0:  IF  x0  is  F0  AND  x1  is  F1
     THEN y0 is F5 AND y1 is F2
R1:  IF  x0  is  F2  AND  x1  is  F3
     THEN y0 is F0 AND y1 is F1
```

The fuzzy system works as follows:

- the fuzzification function evaluates a degree of membership of the input signal
- the inference uses the Mamdani implication
- the defuzzification function calculates the center of weight of the resulting fuzzy set

These three operations are implemented in a Framsticks neuron, named a Fuzzy neuron.

4. Fuzzy neuron in Framsticks

Fuzzy neuron is one of neurons used in Framsticks to control behavior of agents. It is not a neuron in the classical meaning of that word; the only common thing is that the fuzzy neuron gets information from the input and transforms it into the output. Inside the neuron, there is no sigmoid function, but a complete fuzzy rule-based system is encoded.

Inputs and outputs of the fuzzy-system neuron can be connected with any framstick neuron, muscle or sensor. It is desirable for inputs to be connected with sensors (like gyroscope, smell or touch, which get information from the environment) and for outputs to be connected with muscles (bending or rotating muscles, which control agent movement). An example of such use is shown on Figure 2.

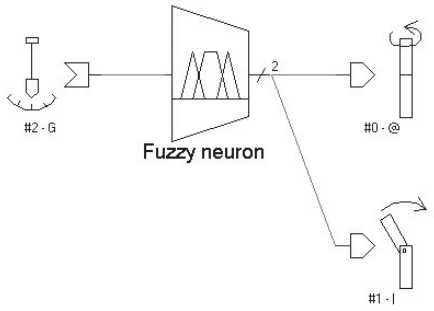

Figure 2. Example of a fuzzy Framsticks brain

5. Evolutionary operators

In our experiments, the physical structure of each agent is fixed during evolution. In another words, the task of evolution is to find the best fuzzy rule-based system (the best parameters of a fuzzy neuron) for a single instance of the body. While fuzzy neuron changes, the rest of an agent stays intact, and thus the number of fuzzy neuron inputs and outputs is constant too. The only values that evolve are fuzzy sets and rules.

5.1. Genetic encoding of the fuzzy rule-based system

Framsticks ***f0*** genetic format [Komosinski and Ulatowski, 1997a] encodes body and its control system, which is a fuzzy system neuron (see Figure 3 for an example). Special genetic operators (named ***f0-fuzzy***) for the fuzzy system neuron are provided to allow for its evolution.

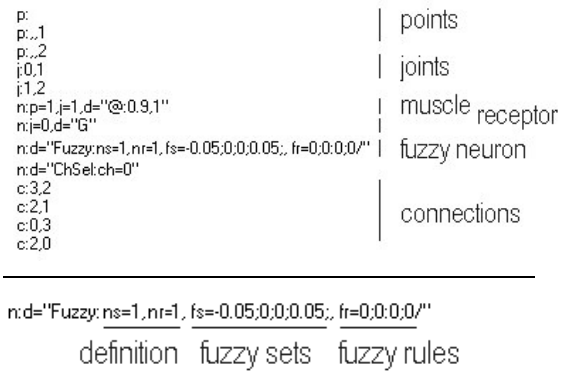

Figure 3. Example of the ***f0*** genotype. Bottom: the three sections in the fuzzy neuron description

The first section defines how many fuzzy sets and fuzzy rules are present in the fuzzy system (in total). These are integer values. The second section defines fuzzy sets, and the third section – fuzzy rules. A single fuzzy rule is defined by nonzero integer values. They represent the number of inputs in the premise part, the fuzzy set used for the fuzzification of the input, the number of outputs in the conclusion part and the fuzzy set used for the defuzzification of the output. Each fuzzy rule may have a different number of premises and conclusions.

5.2. Crossover algorithm

The crossing over used in this work is a hybrid of two methods:

1. The operators cut and exchange the genetic information based on a fixed number of cross-over points (one-, two-, or more) [Fogel 2000; Davis 1991].
2. Some information taken from the parents is averaged.

The general definition of this method is presented in [Goldberg 1989; Michalewicz and Fogel 2000]. In this paper, the length of the parent genotype parts is averaged. Then individual genes are taken randomly from both parents. Some genes are copied from one parent without changes. A very similar method is presented by Casillas [Casillas et al. 2000]. It is called a *partially complementary* method.

The information inherited from the parent fuzzy systems are fuzzy sets and rules. Crossing over operations are performed in such a way that all the fuzzy sets (from both parents) are copied, and the identical ones are used only once. The way fuzzy rules evolve is more complex, and for the limited space of the paper only general information is provided.

The number of the descendant's fuzzy rules is drawn according to the following formula:

$$rulesNr3 = \min\{rulesNr1; rulesNr2\} + random(|rulesNr1 - rulesNr2| + 1)$$

where *rulesNr* is the number of rules for parents (*rulesNr*1, *rulesNr*2) or a descendant (*rulesNr*3). So the number of descendant rules is a random value in the range:

$$\min\{rulesNr1; rulesNr2\} \leq rulesNr3 \leq \max\{rulesNr1; rulesNr2\}$$

The same procedures are used to draw the numbers of inputs and outputs in the new fuzzy rule (using the numbers of inputs and outputs in the parent rules, respectively).

5.3. Mutation

The system uses many kinds of mutations.

- *New fuzzy set add* creates a new fuzzy set with random values. It is also necessary to add a new fuzzy rule, which uses the newly created fuzzy set (to avoid unused fuzzy sets).
- *Removal of existing random fuzzy set*.
- *New fuzzy rule add* creates a rule with random inputs and outputs number, and random fuzzy sets numbers assigned to each input and output.
- *Removal of existing fuzzy rule* is possible when there is at least one rule in the system.
- *Adding new input or output* and *Removing existing input or output*: rule number to modify is selected randomly, then random input/output is added/removed.

6. Experiments and results

6.1. The walker agent

The walker agent is constructed of four legs with touch sensors at the end of each leg. The legs are connected to the rest of the body by rotating muscles, thus each leg can move. The agent's brain consists of a fuzzy neuron with four inputs (touch sensors) and four outputs (rotation muscles). The fitness goal is the agent's velocity, so the task of the evolution is to find a fuzzy rule-based system which enables the creature to move as fast as possible.

The results of the evolution are quite interesting. The walker is moving by means of jumping. Its hind legs rotate back and to the default position periodically, and so do the front legs, as shown in Figure 4.

During the evolution, it appeared that the information obtained from three touch receptors is enough to know the agent's position – one sensor signal is unnecessary. The walker agent turns slightly left, as shown on Figure 4, because the rules and the fuzzy sets are not tuned perfectly – it is very possible that the system found a good local optimum which is hard to leave. It is very difficult to tune the fuzzy system manually. Every little change in the fuzzy sets definition or in the fuzzy rules causes changes in the agent behavior. This affects signals taken from the inputs (touch sensors), so the rules' premise part must be changed too.

6.2. The Stand-up agent

This testing agent has a stable base, and is put into the simulation environment upside-down. Two upper arms are a bit longer, so when it is created, it falls down. There are two long sticks, joined with a muscle, which can bend in two dimensions, to make it possible to get up. At four base sticks, there are two gyroscopes (which make it possible to find out

agent's vertical position) and two touch sensors (to find out if its base touches the ground). There is also a feedback from muscles, because it could be helpful for the agent to know its muscle state – whether it is bent or not and how much.

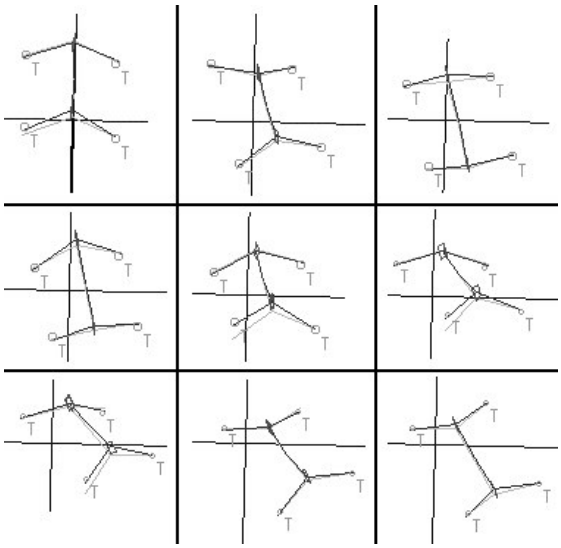

Figure 4. The "walker" agent – jumps of the body

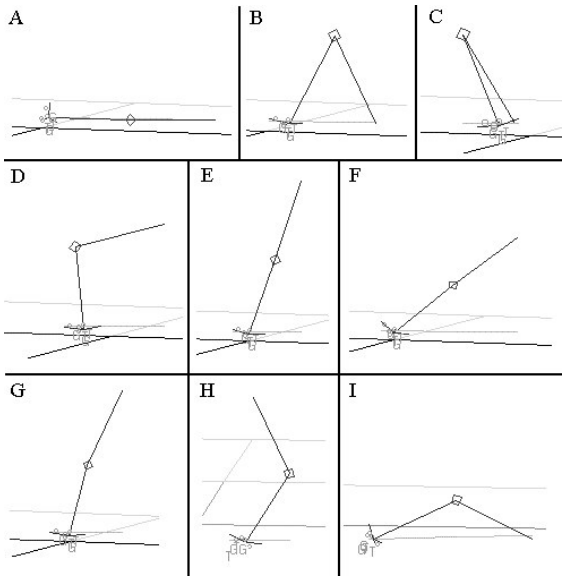

Figure 5. Behavior of the "stand-up" agent

When the simulation starts, the body lies down on the ground. The task of the rule-based system (with five inputs and one output) is to force the body to achieve the possibly tallest height. The first interesting evolution result was a rule-based system that made the agent get up. There were different genotypes, which behaved similarly. Some bent and erected its arm in sequence, some stood still by making an angle with both arms. Sometimes they stood stable quite straight – but with an arm bent, as shown on Figure 5 B.

It was much more difficult to evolve rules that would provide arms straightening, because of gravity – when an agent tried to do it, falling down was unavoidable.

Cyclic actions are presented on figure 5 A-F: an agent gets up (A, B), maximally bends its muscle (C), and suddenly straightens it (D, E) to provide better height, but then falls down (F, A). For such behaviors, the fitness (average vertical position of the agent center during the simulation period) was about 0.26 (where the stick length is equal to 1).

7. Conclusions

The work on implementation of a fuzzy system control in the Framsticks environment was successful. We designed special genetic operators for such a control system and validated their usefulness. The fitness of evolved behaviors was comparable to those achieved with heterogeneous neural network controllers.

There are many ways for further development of the fuzzy neuron. Trapezium membership function can be replaced with some other – triangular, gauss etc. Different methods of defuzzification procedure can be selected.

Interesting results could be achieved if the number of inputs and outputs of the fuzzy neuron could be changed, so that different bodies could be evolved and crossed over. Such co-evolution of both body and a fuzzy rule-based control system may bring efficient solutions for various tasks.

Acknowledgement

This work has been supported by the Foundation for Polish Science, from subsidy no. 11/2001.

References

Adamatzky A. (2000) Software review: Framsticks, *Kybernetes: The International Journal of Systems & Cybernetics*, **29** (9/10), 1344-1351.

Goldberg D.E. (1989) Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Publishing Co.

Komosinski M. and Ulatowski S. (1997a) Framsticks Web Site, http://www.frams.alife.pl

Komosinski M. and Ulatowski S. (1997b) Framsticks Developers Center, http://www.frams.alife.pl/dev

Komosinski M. and Rotaru-Varga A. (2001) Comparison of different genotype encodings for simulated 3D agents, *Artificial Life Journal*, **7** (4, Fall), 395-418, MIT Press.

Michalewicz Z. and Fogel D. B. (2000) How to Solve It: Modern Heuristics. Springer-Verlag Berlin Heidelberg

Ross T. J. (1995) Fuzzy Logic with Engineering Applications. McGraw-Hill, Inc.

Yager R.R. and Filev D.P. (1994) *Foundations of fuzzy control*. Wiley, New York.