Evolutionary Design of Interpretable Fuzzy Controllers*

Maciej HAPKE, Maciej KOMOSINSKI

Abstract. This paper presents an approach that allows to evolve fuzzy controllers that can be expressed as fuzzy rules in human-readable form and interpreted. For comparison, the evolution is also performed on simple neural controllers. The control task considered here is a balancing problem, where a construct made of articulated elastic elements is equipped with sensors and actuators. The goal of the construct is to keep the top heavy part from touching the ground. Evolved controllers are evaluated using computer simulation. Control systems process signals from tilt sensors to actuators fixed in the construct. During evolution, fuzzy controllers (including their fuzzy sets and rules) are reconfigured by genetic operators in order to maximize fitness of the control. The article compares evolvability of neural and fuzzy controllers and demonstrates how additional, comprehensible knowledge can be gained which explains the work of the fuzzy controller. The representation for the fuzzy control system, evolutionary operators, various evaluation functions, and the best evolved control systems are presented. A sample evolved fuzzy control system is analyzed in detail to explain its behavior.

Keywords: Fuzzy control, pendulum, evolution, optimization, simulation, interpretable rules, neural networks

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. Every year the list of applications of fuzzy logic control in various domains is becoming longer and longer [24].

Among many reasons of a growing number of applications, one should be emphasized here. Fuzzy logic allows for inclusion of expert knowledge in control systems.

*Maciej Hapke and Maciej Komosinski. Evolutionary design of interpretable fuzzy controllers. *Foundations of Computing and Decision Sciences*, 33(4):351–367, 2008.

The knowledge gained from experienced process operator can be easily represented in the form of fuzzy control rules where both premises and conclusions contain imprecise linguistic variables. A fuzzy control system defined in this way, after necessary refinement and tuning, is often good enough to substitute for a human. If the control system built on expert knowledge is worse than human control, it can be then enhanced by adding more fuzzy rules (e.g. rules discovered by learning systems). The knowledge acquired from data and represented by fuzzy rules can also explain the behavior of the system. This observation became a motivation for the research described in this paper.

Many authors reported efficiency of evolutionary algorithms for the optimization of fuzzy controllers [4, 9, 15, 23, 6, 7]. In these approaches, evolutionary algorithms (EAs) were used mostly for automatic design of fuzzy controllers with predefined structures, e.g. based on the standard MacVicar-Whelan [13] rule table. On the other hand, EAs turned out to be efficient in fuzzy rule based system optimization, where their task consists in tuning and/or learning fuzzy rules. While tuning is mainly concerned with optimization of membership functions, learning constitutes an automated design method for fuzzy rule sets, starting from scratch. There are various approaches to learning rules described in the literature [14]. One of the most extended trends in rule learning by EAs is the Pittsburgh approach [22] which is characterized by representing an entire rule set as a single chromosome. In this approach, a population consists of candidate fuzzy rule sets that are subject to the selection mechanism, and crossover and mutation evolutionary operators. The conclusion coming from the results reported in the literature is that the use of EA shortens design time, reduces design costs and results in near-optimal control.

Motivated by these results, we have chosen a control task to determine the ability of evolutionary processes to optimize a complete fuzzy control system, and additionally, to describe it in a way that is understandable to a human. A 3D simulation and optimization environment is used [11] to model both the physical structure and the control system, and to optimize control in order to maximize their fitness function. Since the control system is modeled as a fuzzy system (consisting of fuzzy control rules), the acquired knowledge can be potentially understood by a human. The aim of this work is therefore to employ evolutionary processes following the Pittsburgh approach to optimize Mamdani-type fuzzy logic controllers compared to artificial neural networks, and to use techniques to further explain the fuzzy system that was evolved without human intervention. Consequently, the evolved fuzzy control is not a "black box" and can be interpreted even though it was created automatically in simulated evolution.

The following sections introduce the simulation environment, describe the representation of a fuzzy control system, evolutionary operators and evaluation (fitness) functions. Then, experiments with evolved control are presented and results are concluded.

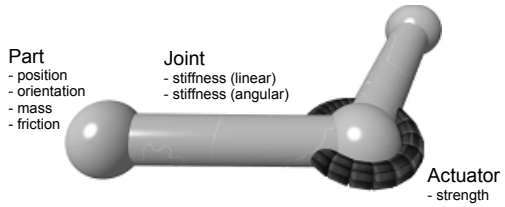

Figure 1: Elements of the physical structure of a simulated agent.

2 The simulation environment

For the experiments, we employ the Framsticks environment [11] which allows simulating and optimizing three-dimensional agent structures and their control systems. The software can be used to perform predefined experiments, but user-defined experiment definitions, fitness functions, and neuron types are also supported. A scripting language is provided for these purposes, so the simulator is suitable for testing various research hypotheses where fast 3D simulation and evolutionary optimization is required.

The physical structure of an agent is made of parts (material points) and elastic joints (see Fig. 1). The control system can be composed of many types of neurons, sensors and actuators, and their weighted connections [1]. In this work, only a subset of simulation features is used – the physical structure of agents is fixed (not optimized).

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 [10]. In this work, the direct encoding is employed, and special genetic operators are designed to provide mutation and crossing over of a fuzzy control system.

3 Fuzzy control model

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 [17]. A valuable feature of a fuzzy system is that one can infer a new conclusion from a new hypothesis. This allows the fuzzy system to be 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 [16, 21, 26].

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 compounded of many inputs and outputs. Therefore 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

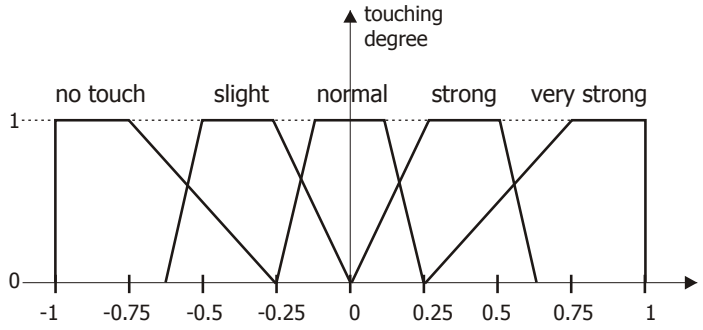

Figure 2: Example of fuzzy sets for the touch sensor.

being defined by four real numbers within the normalized domain of $[-1, 1]$. Fig. 2 presents an example of fuzzy sets, which correspond to the simulated touch sensor.

The example of a fuzzy rule-based system with two inputs (x0, x1), two outputs (y0, y1), two rules (R0, R1) and five fuzzy sets (F0 .. F4) can be described as follows:

```
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
```

In the considered approach, the classical Mamdani model of the fuzzy system is used. It is composed of three parts with the following functionality:

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

These three operations were implemented within the simulation environment. They constitute a single control unit that gets information from inputs and transforms it into outputs, using a fuzzy rule-based system. Inputs and outputs of this unit can be connected with any other unit or neuron, actuator or sensor. It is desirable for inputs to be connected with sensors (like gyroscope, touch, or smell, which acquire information from the environment) and for outputs to be connected with actuators (bending or rotating "muscles", which control movement). An example of such use is shown on Fig. 3.

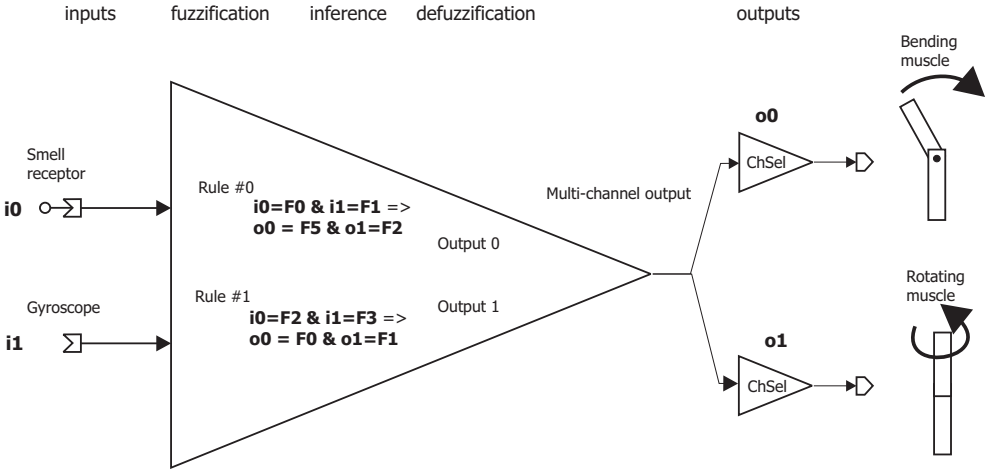

Figure 3: Example of a fuzzy unit within the control system.

4 Evolutionary algorithm

In our experiments, the physical structure of each agent is fixed during evolution. In another words, the goal of evolution is to find the best control system for a single instance of the body shape. If the control system is a neural network, then the network topology and connection weights are evolved. If it is a fuzzy rule-based system, then the fuzzy neuron parameters are evolved. While the fuzzy unit changes, the rest of an agent stays intact, and thus the number of fuzzy unit inputs and outputs is constant too. The only values that evolve are fuzzy sets and rules.

A steady-state evolutionary process was used for optimization [5]. It is different from a popular generational process in that individuals are processed one by one. Thus the steady-state evolution is performed in these steps:

- Choose the genetic operation: mutation or crossing over
  - For mutation: select a genotype from the gene pool and mutate it
  - For crossing over: select two genotypes and cross them over
- Build an agent from the resulting genotype, simulate and evaluate it
- Add the evaluated genotype to the gene pool
- If the gene pool size exceeds the maximum size, delete a genotype

A number of test runs were performed in order to choose the best parameters for the evolutionary process. The final settings were as follows: the selection mechanism was tournament, and the gene pool size was 400. Genotypes were deleted randomly. Additionally, we introduced a random noise to the initial state of neural units to gain

n:d="Fuzzy:**ns**=4, **nr**=2,
fs=-0.1647;-0.1526;-0.0087;0.0631;
-1;-0.8774;-0.7725;-0.6767;
0.0087;0.2308;0.3585;0.4806;
0.011;0.1664;0.2362;0.2718;,
fr=0;3;1;0;2;0:0;2;3;1;2;1;1;3/
2;0;0;2;1;2:3;1;2;0;1;2;0;0/"

Figure 4: Example of the Fuzzy system encoding within the *f0* genotype.

more robust behaviors. Thus agent behaviors could be non-deterministic, and we needed to evaluate them many times and average their performance. Therefore the "cloning" operation accompanied the mutation and crossing over operations. Cloning produces an agent that is based on the selected genotype that is unchanged (so it must have been already evaluated). The proportion of cloning:mutation:crossover operations was 20:64:16. The optimization is stopped when there is no best-fitness improvement for 5000 genotype evaluations.

4.1 Genetic encoding of the fuzzy rule based system

The basic genetic format in Framsticks [10] is named *f0*, and it directly encodes body and its control system, which may be a fuzzy system unit (see Fig. 4 for an example). Special genetic operators for the fuzzy system unit are provided to allow for its evolution.

The **ns** and **nr** fields define how many fuzzy sets and fuzzy rules are present in the fuzzy system (in total). The **fs** field defines fuzzy sets, and the **fr** field describes 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.

4.2 Fuzzy system crossing over

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

1. The operators cut and exchange genetic information based on a fixed number of cross-over points (one-, two-, or more) [Fogel 2000].

2. Some information taken from the parents is averaged.

The general definition of this method is presented in [5, 19]. 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 [3], where it is called the *partially complementary* method.

The information inherited from the parent fuzzy systems are both 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 number of the descendant's fuzzy rules is drawn as a random number from between the numbers of parent fuzzy rules. 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).

4.3 Fuzzy system mutation

A number of dedicated mutation types are used during optimization.

- *Add new fuzzy set* 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).
- *Remove random, existing fuzzy set.*
- *Add new fuzzy rule* creates a rule with random numbers of inputs and outputs, and random number of fuzzy sets assigned to each input and output.
- *Remove random, existing fuzzy rule.*
- *Add new input or output* and *Remove random, existing input or output*: the rule to be modified is selected randomly, then random input/output is added/removed.

5 Experiments and results

5.1 Experimental setup

To verify whether evolutionary optimization is capable of producing successful, interpretable control systems automatically, we designed two types of balancing constructs. The considered balancing problem is in some aspects similar to the inverted pendulum problem, but it is much more difficult. We consider an autonomous, active, elastic construct that is capable of acting and changing its shape based on its own evolved logic. Therefore the means to achieve success are different than in the classical inverted pendulum problem [8, 12] where the control is external to the fixed pendulum construct. In our experiments, the base (point of support) is usually not being moved (although it is not fixed to the ground), but the construct itself is active and can bend within two vertical, perpendicular planes.

Tilt sensors and actuators are fixed in the construct, and sensors provide information for the control system which controls the actuators. The sensors generate signals

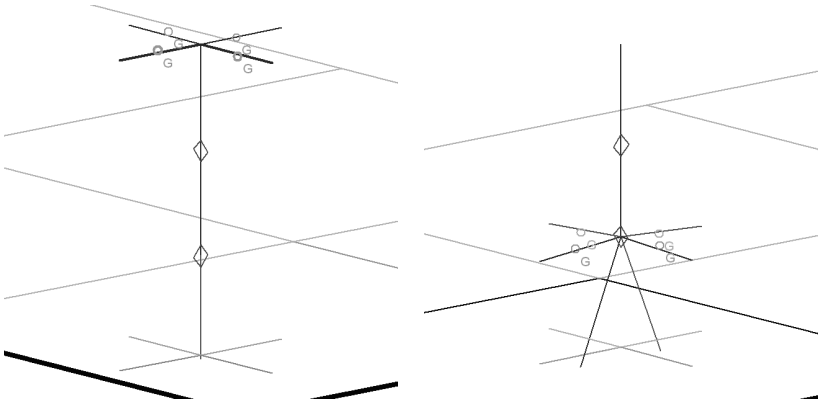

Figure 5: Two kinds of balancing constructs. The "G" symbols denote tilt sensors, and squares illustrate bending actuators.

from the $[-1, 1]$ range depending on the spatial orientation of the joint they are located in, relative to the horizontal ground plane. The two considered construction models (with one and two points of support) are shown on Fig. 5.

In both cases, the basic optimization task was to evolve a control system capable of keeping the top part of the construct from touching the ground for as long as possible. The single-support-point model could use its two actuators to bend its base in two planes. The second model could bend its top joints in two planes to balance and stabilize. Note that if the pendulum is not moving its base point, then it is hardly possible to keep it balanced once it starts falling and its center of mass is shifted off-base (balancing would require the construct to immediately rotate one of its parts, while actuators cannot act instantly). The only way to move the base point that supports the weight of the entire construct is to jump – which is extremely unlikely to happen (actuators have limited power) and extremely difficult to control (note the pendulum is elastic). We are going to see how evolutionary process will cope with this nearly impossible task.

In the experiments, the agents had a fixed body structure (parts, joints), and the number and placement of sensors and actuators were also fixed. The control system was optimized. It was either a fuzzy system (with a variable number of fuzzy sets and fuzzy rules) or a neural network (with a fixed or evolved topology).

5.2 Neural network and fuzzy controllers

Two kinds of experiments were performed with the artificial neural network controllers [2, 20]. Firstly, a few network topologies were designed (these are shown on Fig. 6), and for these topologies, only connection weight values were optimized. Note that recurrent connections existed in one topology. Secondly, the evolutionary process

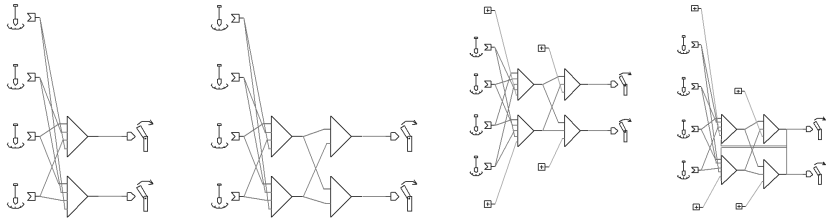

Figure 6: Pre-designed neural network topologies used in the balancing control task. Units shown as "+" provide a constant signal value, 1. Tilt sensors are shown on the left side, and bending actuators are shown on the right side of each diagram.

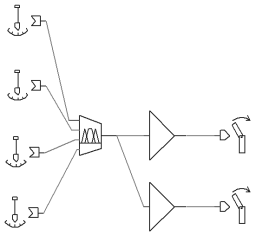

Figure 7: The fuzzy system control architecture.

was given a chance to modify the network topology as well (i.e. add and remove neurons and connections). In this case, neural networks could have any complex topology with multiple recurrent connections.

The general structure of the fuzzy control system is shown on Fig. 7. Four tilt sensors provide signals for the fuzzy system, which outputs two control signals for the bending actuators (on the picture, this is a single two-channel connection). The two triangle units are required to select a single signal channel from the two-channel fuzzy system output.

Additionally, a number of experiments have been performed with more complex topologies of neural networks, other locations of sensors in the pendulum (i.e. on the vertical shaft), differentiating neurons introduced for sensory inputs, and different rates of simulation of the control system (versus physical simulation). These modifications did not yield considerable changes in evolved behaviors so the simple setups are described below.

5.3 The need to explain fuzzy systems

To investigate the meaning of evolved control systems, a mapping was needed from the evolved rules and fuzzy sets to the variables understood by an expert, in or-

der to present the rules in a linguistic form. A *FuzzyExplain* application has been implemented to investigate the knowledge existing in evolved control systems. The mapping from evolved fuzzy data to the variables understood by an expert is based on the comparison of evolved fuzzy sets and linguistic variables defined by an expert. Accordingly, the evolved fuzzy sets are then substituted with the most similar variables previously defined by an expert. Thus the mapping allows presenting the fuzzy neuron parameters as natural language sentences.

The sample explanation of the fuzzy neuron parameters is shown below. Assume the fuzzy neuron is described as follows:

```
n:d="Fuzzy:ns=5, nr=4,
fs=0.0123;0.182;0.4877;0.4958;-0.9654;-0.8361;-0.644;
-0.4643;-1;-1;-0.9719;-0.852;0.3982;0.6898;0.8066;
1;0.8152;0.9052;1;1;,
fr=1;1;3;1;0;2:1;4;0;1/2;0;3;0:0;0/1;2:0;1/2;3:0;0/"
```

Then, an expert defines linguistic variables that cover the most characteristic parts of the variable domains. Assume that three disjunctive fuzzy sets are defined: `upright` (-1; -1; -1; 0), `leveled` (-1; 0; 0; 1) and `upside_down` (0; 1; 1; 1) on each input variable domain (input variables correspond to the tilt sensor signals). Moreover, three disjunctive fuzzy sets: `right` (-1; -1; -1; 0), `none` (-1; 0; 0; 1) and `left` (0; 1; 1; 1) are defined on the domains of the two output variables (bending actuators). This sample mapping results in the following set of four linguistic rules:

```
1. s2=upright and s4=upright and s1=upright => bend1=right and bend0=left
2. s3=upside_down and s4=upside_down        => bend0=right
3. s2=upright                               => bend0=left
4. s3=upside_down                           => bend0=right
```

This approach is used in Sect. 5.5 to explain behaviors of evolved fuzzy control systems.

5.4 Fitness functions and optimization results

The design of a good fitness function turned out to be a nontrivial task. A few functions were created and their support for evolvability was verified for both neural and fuzzy controllers. The most interesting fitness functions are described below. Note that we consider an optimization process where the fitness value is taken into account only after a full period of simulation of an agent, not during simulation (as it would be the case e.g. in reinforcement learning). The figures show the first type of the balancing construct (see Fig. 5, left picture), because it is simpler to depict. However, the results were similar for both types of models.

The most obvious, maximized fitness function reflected directly the time from the beginning of agent simulation until the moment when any of the body points (except the lowest base point) touches the ground. The evolved neural network controllers

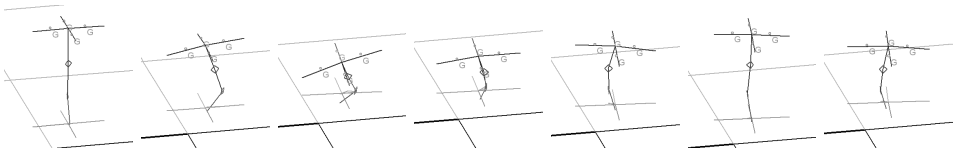

Figure 8: The balancing behavior generated by the evolved neural network controller.

(described in Sect. 5.2) were able to keep the construct in the upright position for a period of time, usually involving three or four cycles of bending and balancing. The more complex neural network topology (see Fig. 6), the better were the results. Best results were obtained for the network where both weights and topology were optimized (see Fig. 8). The first moments of simulation were the hardest to keep straight, because the elastic base was virtually compressed by the heavy top. The actuators have limited power, so they may be unable to bend against the pressing weight in spite of the control signal.

Evolutionary optimization of the fuzzy system failed for this fitness function. In an initial population where all (random) fuzzy systems were equally unfit, it was impossible to find (using described mutation and crossing over operators) any better fuzzy system than those which did nothing and let the construct collapse. Some fuzzy systems would bend actuators, but it would not cause improvement of the fitness function value. The fitness landscape generated by the fuzzy system representation and operators was initially so flat that there was no obvious direction for improvement to be found by random mutations.

Therefore we needed to consider other fitness functions, which incorporated some straightforward knowledge about the expected behavior, and produced easier fitness landscapes. The text below focuses on the fuzzy control system evolution. For these experiments, the simulation lasted a fixed amount of time, and the fitness was a sum of partial values computed each five simulation steps.

First, we estimated angles of the four horizontal sticks with the tilt ("G") sensor to make the control system keep these sticks horizontally. This was intended to help the construct from falling. We also added a vertical position fitness component to make the construct "stretch" and stand vertically. However, this simple function was tricked by an evolved fuzzy control system which made the construct balance upside-down, i.e. with the heavy "head" on the ground. This position was also highly rewarded according to the fitness function used. Such behavior of the construct exposed the *aliasing* problems [25], where a set of identical sensory signals is acquired for *different* positions of the body. As the construct is equipped with tilt sensors, it is not possible to judge (based only on the sensory input) whether the body is inverted or not, because only the relative angle to the horizontal plane is known.

To provide more freedom in balancing behaviors, the fitness function was changed to allow for non-horizontal positions of the four "G" sticks as long as their vertical position was adequately high. Additionally, a penalty was introduced for moments when any of the body points (except the base) was near the ground. This fitness

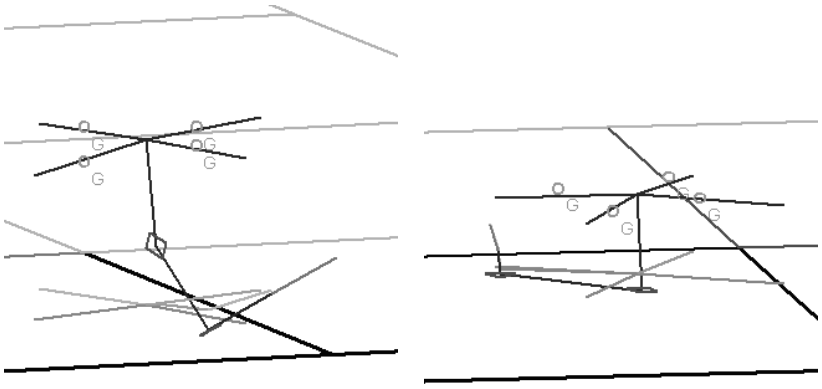

Figure 9: Behaviors generated by the evolved fuzzy controllers and various fitness functions. See text for details.

function produced interesting behaviors, taking advantage of the fact that the construct was active. It bends the actuators so rapidly as to jump or push back from the ground to minimize the time spent near the ground (Fig. 9, left picture). After falling down it tries to straighten as soon as possible, but is able to stand only for a short period of time.

Finally, the fitness function was modified to keep the plane defined by the four horizontal sticks parallel to the horizontal surface. The vertical position fitness component was retained. This was successful, because the evolved fuzzy systems were able to keep the heavy head from touching the ground. The construct balanced using one segment, the two other bent on the ground plane (Fig. 9, right picture) which formed a stable base for the top, balancing part. Moreover, the resulting configuration was robust to manual displacement and joggling of the top part, and the construct was able to continuously maintain the top from touching the ground plane.

The results of the experiments show that evolution of both neural and fuzzy controllers for the considered task was successful. It turned out that neural networks were easier to evolve than fuzzy systems (with respect to reconfiguration operators that were used). More complex neural networks performed better. Fuzzy control systems required more knowledge-rich fitness function to proceed with evolutionary optimization, but they have an advantage – control with fuzzy rules can be explained and interpreted.

5.5 Understanding evolved fuzzy rules

A large number of performed experiments yielded a set of interesting individuals. In this paragraph, we present a detailed analysis of the evolved fuzzy rules, explaining

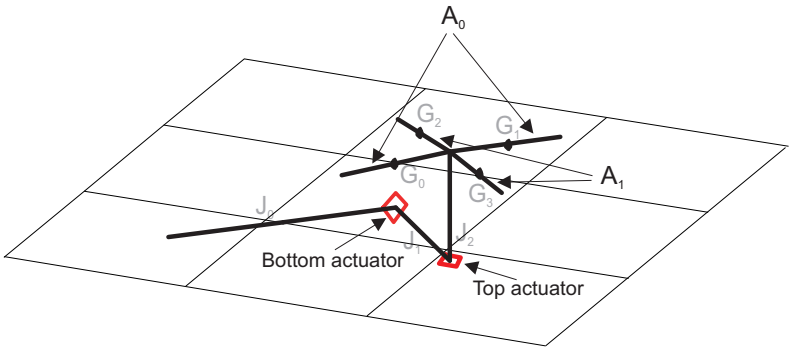

Figure 10: The construct body (shown in a bent position).

behaviors based on these rules. Fig. 10 introduces names for parts of the construct.

The base of the single-support-point construct is composed of three joints (J_0, J_1, J_2) equipped with two actuators (bottom and top) working in two planes. The top part of the construct is composed of four perpendicular sticks each equipped with a single tilt sensor (G_0, ..., G_3). Sticks with G_0 and G_1 form an arm A_0, and sticks with G_2 and G_3 form an arm A_1. A_0 and A_1 form a cross that is called a "head".

For analysis, a fuzzy system is selected that performed best for one of the fitness functions, as described in Sect. 5.4. The evolved behavior is characterized by the following observations:

- it reaches a stable position (shown on Fig. 10) very quickly,
- in the stable position, J_0 and J_1 lie down on the ground, while J_2 stands upright supporting the head in the horizontal position ,
- after the construct is manually thrown off balance, it reaches the stability quite quickly and the behavior strategies depend on the side it has been pushed to:
  - if it has been pushed along its bottom joint (J_0), the actuators are bent only slightly,
  - if it has been pushed crosswise to the J_0, it makes sudden moves and after a few cycles it usually reaches the stable position,
  - if it is thrown upside down, the fuzzy system is unable to make it stand straight.

After making these observations, two questions regarding the construct behavior arise: what makes it stand still and what makes it come back so quickly to the quasi-vertical stable position. To answer these questions, we perform a detailed analysis of the evolved fuzzy rule system.

Since the optimization experiments considered only behaviors, not the complexity of the control systems, the typical evolved fuzzy systems had many fuzzy sets and

fuzzy rules. Before attempting to analyze the control system, it was reasonable to try to simplify it. This was achieved by performing an additional, short optimization process with disabled genetic operators of fuzzy set add and fuzzy rule add. Modifications and deletions of sets and rules were allowed. Thus we were able to radically decrease the number of fuzzy sets and rules – the number of rules was reduced from twenty to five without deteriorating fitness.

Each fuzzy system has four inputs and two outputs. Input signals `s0`, `s1`, `s2`, `s3` come from four sensors. Based on their values, the fuzzy system sends two outputs signals for actuators: `bend_bottom` and `bend_top`. Input and output fuzzy variables are defined in the normalized domain $[-1, 1]$. Input linguistic variables `upright`, `leveled` and `upside_down` are defined as follows: (-1, -1, -1, 0), (-1, 0, 0, 1) and (0, 1, 1, 1), while the outputs characterizing bending directions are expressed by linguistic variables `right` (-1, -1, -1, 0), `none` (-1, 0, 0, 1) and `left` (0, 1, 1, 1). The fuzzy system is therefore rendered as:

```
1. s2=leveled and s0=leveled     => bend_bottom=left and bend_top=left
2. s3=leveled and s1=upside_down => bend_top=left
3. s1=upright                    => bend_bottom=left and bend_top=left
4. s3=upside_down                => bend_bottom=right and bend_top=left
5. s1=upside_down                => bend_bottom=left and bend_top=none
```

The first interesting observation is that the pairs of sensor signals (`s0`, `s1`) and (`s2`, `s3`) never come together in a single premise of the rule. It is because the simplifying optimization process discovered a property of the construct shape: the signals from these tilt sensor pairs are almost the same. This is the consequence of placing sensors (G_0, G_1) and (G_2, G_3) on the same arms, respectively.

The first control rule refers to the situation where the construct head is parallel to the ground. In response, both actuators are bent left, which corresponds to the stable position as shown on Figure 10. Thus the stable position is guaranteed by the position of J_0 – even in situations when the A_0 slants, which is detected by G_1 (`upright` or `upside_down`) in three rules: #2, #3 and #5. The rule #2 is active when the tilt sensor G_3 is leveled, and G_1 (placed on the stick parallel to J_0) is turned upside down. The rule conclusion controls only the top actuator, bending it left. The rule #5 is similar to #2, and it evolved to reinforce #2. The rule #3 results in bending both actuators left. The construct behavior for the cases described above involves only slight moves, leading to the stable position very quickly.

Rule #4 is crucial and explains the most spectacular behavior in cases when the head (its A_1 arm) is bent crosswise to the bottom part J_0. One could expect this to be the most difficult situation, but the output `bend_bottom=right` results in a sudden move that in most cases returns the construct to the stable position. If not, then the rapid move is repeated. This rule has been also tested in extreme cases. If the body is excessively moved from side to side and it reaches very difficult positions, then rapid moves caused by the rule #4 may deteriorate the construct orientation, and finally it may unfortunately fall upside down (with the head on the ground). This position is also stable. The explanation of the lack of a special rule for this situation is quite simple. During evolution, the initial simulation position of the construct was always

the same, so it never had a chance to reach extreme positions that we evoked during testing. Thus the fuzzy system was never evaluated nor optimized in such specific situations.

The evolution of fuzzy control systems required fitness functions that more explicitly expressed desired goals. However, the analysis of the evolved controllers confirms additional value of the fuzzy rules knowledge. The fuzzy rules obtained during experiments, when referenced to the construct structure, are plain and easily understandable to a human.

6 Conclusions

This work concerned evolving neural and fuzzy controllers for balancing active constructs. Evolutionary optimization experiments were performed to compare these controllers in terms of their support for evolvability, fitness of obtained solutions, their quality, and ability to understand knowledge existing in the optimized control systems. With the representation and operators described in this paper, neural networks were easier to optimize both in terms of weights and their topology, than the fuzzy system. However, where the fuzzy system optimization succeeded, both neural and fuzzy controllers produced similar behaviors of the construct, and similar fitness values.

The qualitative difference between the neural and fuzzy controllers is the verified ability to extract knowledge from the latter. The obtained rules are easy to understand, and can clearly describe and explain the behavior of the construct, thus ensuring a human expert that the control system is known and correct. A few fuzzy rules can explain characteristic behaviors, e.g. vertical position of the construct and the nontrivial ways it achieves stability. Such knowledge may be of great help for the designer of the physical structure of an active agent, because it enables cooperation between manual constructing of designs and optimization of the corresponding fuzzy control systems. This feature is important for robotics [18] and automated design of control systems and behavior scenarios which need to be understood, not only evaluated and highly rated.

The simulation environment can be used to coevolve both the mechanics (placement and connections of physical parts of its design, as well as the placement of sensors and actuators) and the control system at the same time. Such a simultaneous coevolution of body design and the associated control system may discover even better solutions for various problems.

An interesting direction of work is to formalize the presented approach and automate the corresponding procedure (evolving, simplifying, explaining) so that interpretable fuzzy controllers can be created evolutionarily without much work for each individual control task.

Acknowledgement

This work has been supported by the Ministry of Science and Higher Education, grant no. N N519 3505 33.

References

[1] Andrew Adamatzky. Software review: Framsticks. *Kybernetes: The International Journal of Systems & Cybernetics*, 29(9/10):1344–1351, 2000.

[2] C.W. Anderson. Learning to control an inverted pendulum using neural networks. *Control Systems Magazine, IEEE*, 9(3):31–37, 1989.

[3] J. Casillas, O. Cordon, F. Herrera, and M. J. Del Jesus. Genetic tuning of fuzzy rule-based systems integrating linguistic hedges. In *Proc. Joint 9th IFSA World Congress and 20th NAFIPS International Conference*, volume 3, pages 1570–1574, 25–28 July 2001.

[4] M. G. Cooper and J. J. Vidal. Genetic design of fuzzy controllers: the cart and jointed-pole problem. In *Proc. Third IEEE Conference on Fuzzy Systems IEEE World Congress on Computational Intelligence*, pages 1332–1337, 26–29 June 1994.

[5] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Co., 1989.

[6] F. Herrera and M. Lozano. Adaptation of genetic algorithm parameters based on fuzzy logic controllers. *Genetic Algorithms and Soft Computing*, 125, 1996.

[7] J. Žižka. Learning control rules for takagi-sugeno fuzzy controllers using genetic algorithms. *Proceedings of the Fourth European Congress on Intelligent Techniques and Soft Computing*, 2:960–964, 1996.

[8] J.S.R. Jang. Self-learning fuzzy controllers based on temporal backpropagation. *IEEE Transactions on Neural Networks*, 3(5):714–723, 1992.

[9] C.Z. Janikow. A genetic algorithm for learning fuzzy controllers. *Proceedings of the 1994 ACM symposium on Applied computing*, pages 232–236, 1994.

[10] Maciej Komosinski and Adam Rotaru-Varga. Comparison of different genotype encodings for simulated 3D agents. *Artificial Life Journal*, 7(4):395–418, Fall 2001.

[11] Maciej Komosinski and Szymon Ulatowski. Framsticks: Creating and understanding complexity of life. In Maciej Komosinski and Andrew Adamatzky, editors, *Artificial Life Models in Software*, chapter 5. Springer, New York, second edition, 2009.

[12] M.A. Lee and H. Takagi. Dynamic control of genetic algorithms using fuzzy logic techniques. *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 76–83, 1993.

[13] P.J. MacVicar-Whelan. Fuzzy sets for man-machine interaction. *Int. J. Man-Machine Studies*, 8:687–697, 1976.

[14] L. Magdalena, O. Cordon, F. Gomide, F. Herrera, and F. Hoffmann. Ten years of genetic fuzzy systems: current framework and new trends. *Fuzzy Sets and Systems*, 141(1):5–31, 2004.

[15] L. Magdalena and F. Monasterio. Evolutionary-based learning applied to fuzzy controllers. *Fuzzy Systems, 1995. International Joint Conference of the Fourth IEEE International Conference on Fuzzy Systems and The Second International Fuzzy Engineering Symposium*, 3, 1995.

[16] E. H. Mamdani. Advances in the linguistic synthesis of fuzzy controllers. *International Journal of Man-Machine Studies*, 8(6):669–678, 1976.

[17] E.H. Mamdani and S. Assilian. Application of fuzzy algorithms for control of simple dynamic plant. *IEE*, 121(12):1585–1588, 1974.

[18] Chris Melhuish, Andrew Adamatzky, and Brett A. Kennedy. Biologically inspired robots. In Yoseph Bar-Cohen, editor, *Smart Structures and Materials 2001: Electroactive Polymer Actuators and Devices*, volume 4329, pages 16–27, Newport Beach, CA, USA, 2001. SPIE.

[19] Z. Michalewicz and D.B. Fogel. *How to Solve It: Modern Heuristics*. Springer Verlag, 2000.

[20] C. Moraga and K.H. Temme. Functional equivalence between S-neural networks and fuzzy models. *Technologies for Constructing Intelligent Systems*, 2002.

[21] Timothy J. Ross. *Fuzzy Logic with Engineering Applications*. John Wiley and Sons, 2004.

[22] S.F. Smith. *A learning system based on genetic adaptive algorithms*. PhD thesis, Department of Computer Science, University of Pittsburgh, 1980.

[23] Andrea G. B. Tettamanzi. An evolutionary algorithm for fuzzy controller synthesis and optimization. In *In Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pages 22–25, 1995.

[24] H.B. Verbruggen and R. Babuška. *Fuzzy Logic Control: Advances in Applications*. World Scientific, 1999.

[25] S.D. Whitehead and D.H. Ballard. Active perception and reinforcement learning. *Neural Computation*, 2(4):409–419, 1990.

[26] R.R. Yager and D.P. Filev. *Foundations of fuzzy control*. Wiley, New York, 1994.

Received September, 2007