

Incremental Tuning of Fuzzy Controllers by Means of an Evolution Strategy

Frank Hoffmann

Berkeley Initiative in Soft Computing (BISC)
Computer Science Division
University of California
Berkeley, CA 94720
Email: fhoffman@cs.berkeley.edu

ABSTRACT

This paper presents a novel evolutionary tuning method that adapts the knowledge base of a fuzzy logic controller. The evolution strategy adopted in this paper employs a variable size genome, which it adapts to the complexity of the control problem. The incremental learning scheme starts with a single control rule, which is refined through superposition of additional rules, variables and fuzzy sets in later generations. Additional regulator genes control the activation of the genome segments that represent the fuzzy rules and sets. The incremental development of the fuzzy knowledge base enables the evolution strategy to partition the controller design problem into smaller steps that become more feasible.

The proposed method is applied to adapt a wall-following behavior of a mobile robot. The evolutionary learning process takes place by a simulation of the robot, its sensors and the training environment. The adapted control behavior is subsequently evaluated on the physical robot situated in a real-world environment.

1 Introduction

Evolutionary algorithms constitute a class of search and optimization methods, which imitate the principles of natural evolution [5]. The major distinctions among *genetic algorithms*, *evolution strategies*, *evolutionary programming* and *genetic programming* are the genetic structures that undergo adaptation and the genetic operators that generate new candidate solutions. Despite these differences, their principal mode of operation is based on the same generic concepts, a population of competing candidate solutions, random combination and alteration of potentially useful structures to generate new solutions and a selection mechanism to increase the proportion of better solutions.

Fuzzy systems employ a mode of approximate reasoning, which allows them to make decisions based on imprecise and incomplete information. A fuzzy system offers the advantage of knowledge description by means of linguistic concepts without requiring the complexity and precision of mathematical or logical models.

Recently numerous researchers explored the integration of evolutionary algorithms with fuzzy logic in genetic

fuzzy systems [4][6][9][15]. The majority of publications are concerned with the automatic design or optimization of fuzzy logic controllers either by adapting the fuzzy membership functions or by learning the fuzzy if-then rules. The first method results in a self-tuning controller and is primarily useful in order to optimize the performance of an already existing controller. Our approach belongs to the second class of self-organizing processes that learn the appropriate relationship between control input and output starting without any previous knowledge.

2 Evolution Strategies

An evolutionary algorithm processes a population of genomes from one generation to the next. Each genome is decoded into a potential solution of the optimization problem that is evaluated by a scalar objective function. According to Darwin's principle, highly fit individuals are more likely to be selected to reproduce offspring to the next generation. Genetic operators such as recombination and mutation are applied to the parents in order

to generate new candidate solutions. As a result of this evolutionary cycle of selection, recombination and mutation more and more suitable solutions to the optimization problem emerge within the population.

RECHENBERG and SCHWEFEL developed evolution strategies in the 1960s [1], independently from the work of HOLLAND on genetic algorithms at the same time [8]. Both optimization methods employ a population of genetic structures, which evolves according to the rules of selection and genetic operators such as recombination and mutation.

A major difference between evolution strategies and genetic algorithms exists in the genetic representation of candidate solutions. The genotype in an evolution strategy consists of a vector of real numbers, whereas genetic algorithms usually process a population of binary strings. The genetic representation alone does not constitute a distinctive feature among both methods, e. g. some variants of genetic algorithms also employ floating-point representations [12].

In evolution strategies mutation is the main spring of evolutionary progress, whereas in genetic algorithms mutation only plays a minor role by preventing premature convergence of the population. The opposite holds for recombination. The original evolution strategies did not even employ recombination at all. In genetic algorithms, recombination in the form of genetic crossover plays a central role for the generation of new candidate solutions.

Evolution strategies are distinguished by self-adaptation of additional strategy parameters, which enables them to optimize the evolutionary process according to the structure of the fitness landscape. Due to these characteristics evolutionary strategies are preferable for problems in the domain of continuous optimization whereas genetic algorithms are an approved method for problems of discrete or combinatorial nature.

In an evolution strategy the genome of a single individual consists of two sets of real-valued variables. The vector (x_1, \dots, x_n) contains the object variables, which are tuned by recombination and mutation in order to optimize the objective function. The additional strategy parameters are stored in the vector $(\sigma_1, \dots, \sigma_n)$ that defines the step-size of mutations applied to the object variables. The mutation operator modifies the corresponding object variable x_i by adding normally distributed noise $N_i(0, \sigma_i^2)$ with variance σ_i^2 and zero mean.

$$x'_i = x_i + N_i(0, \sigma_i^2) \quad (1)$$

The strategy parameters σ_i do not remain constant. They undergo mutation

$$\sigma'_i = \sigma_i e^{\tau N(0,1)} \quad (2)$$

where $\tau \sim 1/\sqrt{2n}$ is a constant mutation step-size. The mutation of strategy parameters enables the evolution

strategy to adapt itself to the structure of the fitness landscape. At long sight, selection favors those strategy parameters, which are more likely to generate mutations to object variables that substantially improve their fitness. Due to the mechanism of self-adaptation, an exogenous control of step-sizes, utilized by standard mathematical optimization methods, becomes obsolete.

While mutation is the major genetic operator in evolution strategies, recombination can be helpful to improve the search. In intermediate recombination an offspring inherits the average parameters of its parents. The discrete recombination resembles uniform crossover in genetic algorithms, where each single variable is randomly picked from either one of the parents. Normally, intermediate recombination is chosen for the strategy parameters, while discrete recombination is preferred for the object variables.

3 Fuzzy Control

A fuzzy logic controller is a knowledge-based system characterized by a set of rules, which model the relationship among control input and output. The reasoning process is defined by means of the employed aggregation operators, the fuzzy connectives and the inference method. The fuzzy knowledge base contains the definition of fuzzy sets stored in the fuzzy data base and a collection of fuzzy rules, which constitute the fuzzy rule base.

Fuzzy rules are defined by their antecedents and consequents, which relate an observed input state to a desired control action. Most fuzzy systems employ the inference method proposed by Mamdani in which the consequence parts are defined by fuzzy sets [11]. A Mamdani type fuzzy rule R_n has the form:

$$R_n \quad : \quad \text{if } X_1 \text{ is } A_{1n} \text{ and } \dots X_m \text{ is } A_{mn} \\ \text{then } Y \text{ is } B_n$$

The consequences B_n of the active rules R_n are aggregated into one fuzzy set for the output variable Y . The crisp control action is obtained through defuzzification, which calculates the centroid of the output fuzzy set.

Besides the more common fuzzy inference method proposed by Mamdani, Takagi, Sugeno and Kang introduced a modified inference scheme [14]. The first two parts of the fuzzy inference process, fuzzifying the inputs and applying the fuzzy operator are exactly the same. A Takagi-Sugeno-Kang (TSK) type controller employs different implication and aggregation methods than the standard Mamdani controller. Instead of using fuzzy sets the conclusion part of a rule, is a linear combination of the crisp inputs.

$$R_n \quad : \quad \text{if } X_1 \text{ is } A_{1n} \text{ and } \dots X_m \text{ is } A_{mn} \\ \text{then } y = c_{n0} + c_{n1}x_1 + \dots + c_{nm}x_m$$

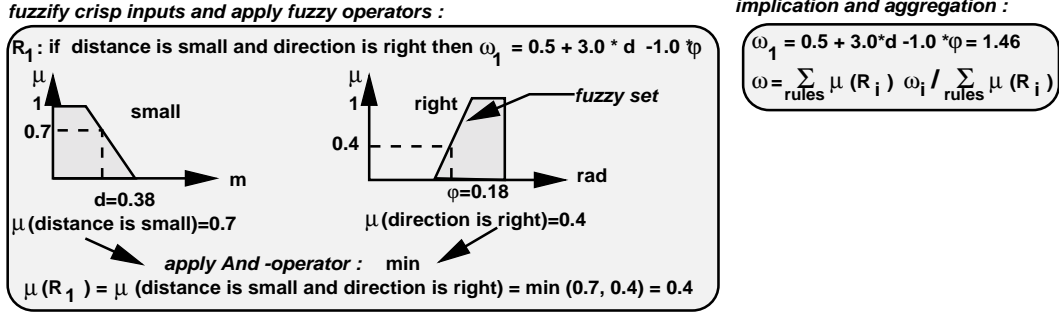


Figure 1 Steps in the TSK-fuzzy inference process : d =distance of robot to obstacle, ϕ =direction of obstacle, ω =change in robot heading

Since the consequence of a rule is crisp, the defuzzification step becomes obsolete in the TSK inference scheme. Instead, the control output is computed as the weighted average of the crisp rule outputs, which is computationally less expensive than calculating the center of gravity. Fig.1 shows the distinct steps

- fuzzification
- application of fuzzy operators
- implication from the antecedent to the consequent
- aggregation of rule outputs

involved in the interpretation of fuzzy if-then rules using an example taken from the mobile robot domain. The mobile robot is given the task to avoid an obstacle by changing its current heading during the next control step. The fuzzy controller computes the change in robot heading ω based on the sensor values distance to nearest obstacle d and relative direction of nearest obstacle ϕ . The fuzzification step evaluates the firing strength $\mu(R_1)$ of the rules antecedent terms *if distance is small and direction is right* by fuzzifying the crisp inputs d and ϕ and combining their degrees of membership $\mu(\text{distance is small})$ and $\mu(\text{direction is right})$ using the fuzzy AND-operator. The degree of support $\mu(R_i)$ of a fuzzy rule R_i determines the relative weight by which the consequent part contributes to the overall output. The crisp output value ω_i of rule R_i is computed by means of the linear control law that constitutes the consequent of a TSK fuzzy rule. The implication step weights the outputs ω_i by the rules firing strength $\mu(R_i)$. Finally the weighted outputs are aggregated into the overall controller output ω .

A TSK fuzzy inference system is extremely well suited to the task of smoothly interpolating gains across the input space. This method of interpolating multiple linear controllers for different operating points is known as *gain scheduling* in classical control system design. In addition, learning techniques are applicable to TSK fuzzy controllers, by adapting the continuous gain factors c_{ij} of the linear control rules. Sugeno's method

has the drawback that rules become less intuitive than in the Mamdani case, which makes it more difficult for a human expert to interpret the control behavior.

4 Evolutionary Adaptation of the Fuzzy Knowledge Base

The essential part in fuzzy controller design is the acquisition of the knowledge base. Usually the fuzzy system designer observes or interviews the human operator of the plant in order to derive the linguistic variables and fuzzy control rules. A learning process aiming to achieve a desired system behavior can automate this knowledge acquisition step. When learning is applied to a rule-based controller, the objective becomes to meet user specified performance criteria of the system. In this context, learning a fuzzy knowledge base can be regarded as an optimization problem in which the set of possible fuzzy rules and fuzzy sets constitutes the search space.

4.1 Incremental Learning Scheme

In case of genetic fuzzy systems, an evolutionary algorithm processes a population of genomes corresponding to fuzzy knowledge bases. Each candidate solution represents a possible combination of fuzzy rules and fuzzy sets. The data set contains the definition of the input fuzzy sets described by parameterized membership functions. A vector of real-valued object variables forms the first segment of the genome, which encodes the intersection and the overlap among adjacent fuzzy sets as depicted in Fig.2. The second segment embodies the genes defining the fuzzy rule base. The object variables contain the gain factors in the consequence parts of the fuzzy rules. The decoding of object variables into fuzzy rules and sets is controlled by additional regulator genes, which determine the knowledge base structure.

For complex control problems with a large number of inputs and rules, the task of finding the optimal gain factors soon becomes a search for the proverbial "needle in the haystack". In case of two inputs the number of parameters remains tractable, but it grows rapidly with an increasing number of variables. In the follow-

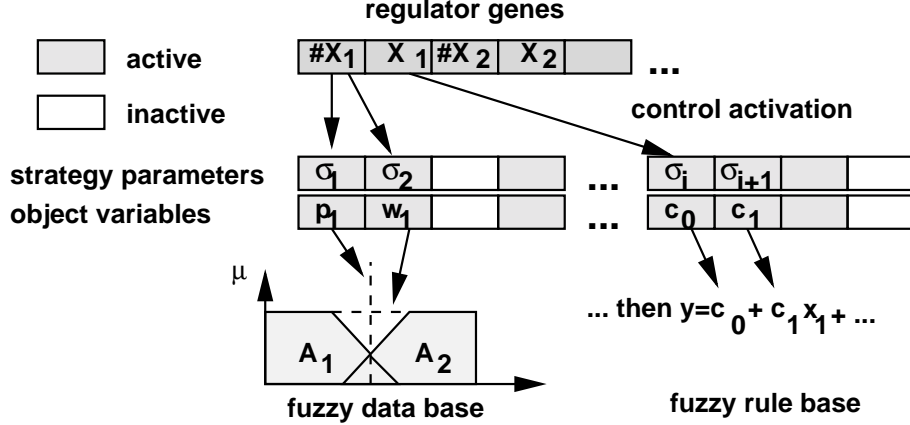


Figure 2 Genetic representation of the TSK fuzzy knowledge base

ing, we propose an incremental learning scheme for TSK type fuzzy control rules. The underlying idea is to partition the overall optimization problem into smaller steps that are more feasible. The evolution strategy first optimizes a fuzzy knowledge base of lower complexity and then refines it through superposition of additional rules, variables and fuzzy sets in later generations. The evolution strategy gradually increases the number of object variables in correspondence to additional parameters employed by the fuzzy knowledge base.

The most simple TSK fuzzy knowledge base contains a single rule R_1 covering the entire input space, which consequence part consists of a linear control law based on a single crisp input variable x_j :

$$R_1 : \quad \text{if } X_1 = A_{11} \text{ and } \dots X_m = A_{1m} \\ \text{then } y = c_{1j}x_j$$

The domain of input for a variable X_i is entirely represented by a single fuzzy set A_{1i} . This primitive knowledge base is encoded by a single object variable for the gain factor c_{1i} and its corresponding mutation step-size σ_{1i} . For the mobile robot example from the previous section this primitive rule might look like

$$R_1 : \text{if } \textit{distance} = \textit{any} \text{ and } \textit{direction} = \textit{any} \\ \text{then } \omega = c_{1\phi}\phi$$

where *any* represent fuzzy sets which membership functions acquire the constant value 1 over the entire input space. Choosing a negative gain factor $c_{1\phi}$ constitutes a simple control behavior that turns the robot away from the obstacle. In most scenarios a single fuzzy rule is not able to control the system in a satisfactory manner. In order to enable a more sophisticated control behavior the first fuzzy rule can be expanded in two ways.

1. Partition the domain of an input variable X_j into two partially overlapping fuzzy sets \tilde{A}_{1j} and \tilde{A}_{2j} , with the result that the knowledge base now contains two rules \tilde{R}_1, \tilde{R}_2 instead of one.

$$\tilde{R}_1 : \text{if } X_1 = A_{11} \text{ and } \dots X_j = \tilde{A}_{1j} \dots X_m = A_{1m} \\ \text{then } y = \tilde{c}_{1j}x_j$$

$$\tilde{R}_2 : \text{if } X_1 = A_{11} \text{ and } \dots X_j = \tilde{A}_{2j} \dots X_m = A_{1m} \\ \text{then } y = \tilde{c}_{2j}x_j$$

The genome is expanded in the data base segment as well as in the rule base segment, in order to reflect the increased number of fuzzy sets and rules. The data base segment incorporates a pair of object variables, which define the intersection point p_j and the overlap w_j among the two fuzzy sets \tilde{A}_{1j} and \tilde{A}_{2j} (see Fig.2).

The left part of Fig. 3 depicts the segmentation of the input variable d into two fuzzy sets *small* and *large* in the mobile robot example. As a result the fuzzy rule base contains two modified rules

$$\tilde{R}_1 : \text{if } \textit{distance} = \textit{small} \text{ and } \textit{direction} = \textit{any} \\ \text{then } \omega = \tilde{c}_{1\phi}\phi \\ \tilde{R}_2 : \text{if } \textit{distance} = \textit{large} \text{ and } \textit{direction} = \textit{any} \\ \text{then } \omega = \tilde{c}_{2\phi}\phi$$

that can differentiate among nearby and remote obstacles. Choosing a smaller gain factor $\tilde{c}_{2\phi}$ for *large* distances results in a smoother reactive obstacle avoidance manoeuvre.

The intersection point p_d separating the two new fuzzy sets *small* and *large* is located in the center of the original fuzzy set *any*. The initial overlap w_d amounts to one third of the support of the individual fuzzy sets. The evolution strategy is able to shift and to dilate the fuzzy sets *small*, *large* through mutation of the parameters p_d and w_d . The original gene for the gain factor $c_{1\phi}$ is duplicated in the rule base segment, which afterwards contains two genes associated to the new gain factors $\tilde{c}_{1\phi}$ and $\tilde{c}_{2\phi}$.

2. Adding a linear term for a second input variable x_j to the consequence part of the rule R_1

$$\tilde{R}_1 : \text{if } X_1 = A_{11} \text{ and } \dots X_m = A_{1m} \\ \text{then } y = c_{1j}x_j + c_{1j}x_j$$

with the result that the control law becomes a linear function of two inputs.

In the mobile example, the modified rule

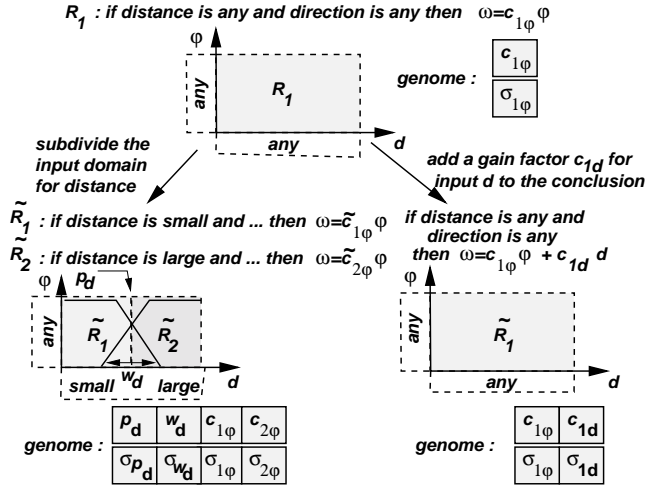


Figure 3 Left : Expanding the initial fuzzy rule by (a) subdividing an input domain (b) adding a gain factor to the consequent. Right : Constructional selection mechanism to regulate genome development

$$\tilde{R}_1 : \text{if distance} = \text{any and direction} = \text{any} \\ \text{then } \omega = c_{1\phi}\phi + c_{1d}d$$

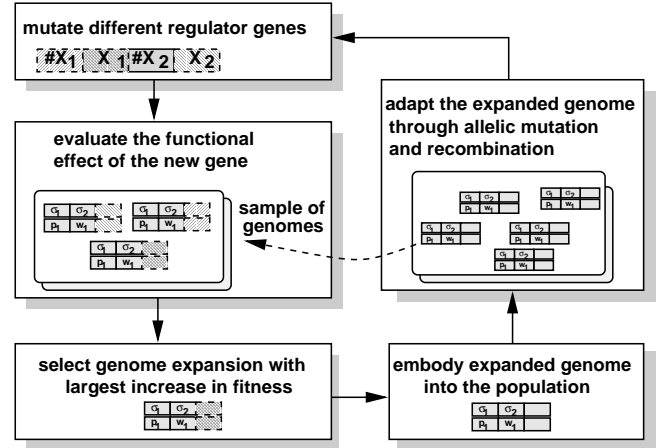
acquires an additional linear term $c_{1d}d$ in the consequent part as depicted in Fig.3. The genome is extended by a second object variable for the additional gain factor c_{1d} plus its corresponding strategy parameter σ_{1d} . In order to guarantee the congruence of the new rule \tilde{R}_1 with its evolutionary ancestor R_1 , the new parameter c_{1d} is initialized with a zero value.

The quality of the obstacle avoidance behavior can not be improved by the new term $c_{1d}d$ alone since the rule antecedent does not discriminate among different directions of the obstacle location. Assume the input variable ϕ was subject to the partitioning step described in the previous paragraph resulting in two rules

$$\tilde{R}_1 : \text{if distance} = \text{any and direction} = \text{left} \\ \text{then } \omega = c_{1\phi}\phi + c_{1d}d \\ \tilde{R}_2 : \text{if distance} = \text{any and direction} = \text{right} \\ \text{then } \omega = c_{2\phi}\phi + c_{2d}d$$

that trigger for different obstacle locations. Choosing a negative gain factor c_{1d} for obstacles to the left and a positive gain factor c_{2d} for obstacles to the right results in a much more refined control behavior.

Following the expansion of the genome, the evolution strategy reshapes the control surface through subsequent mutations applied to the newly acquired genes. The evolution strategy adapts the additional parameters for a few generations before the next expansion of the fuzzy knowledge base is applied. Each expansion step increases the complexity of the knowledge base, either by adding a new input variable to the consequence part of all fuzzy rules or by partitioning an input domain into fuzzy sets with higher resolution. The proper initialization of the additional object variables guarantees that



the more complex fuzzy knowledge base resembles its parent with a more elementary structure.

4.2 Regulation of Genome Development

So far, we have only addressed the issue of possible gene additions, having different functional effects on the fuzzy knowledge base structure. A constructional selection mechanism is needed that decides which of the multiple possible expansions results in optimal evolvability, or in other words most likely promises the production of fitter offsprings. The idea is to reject gene additions that have a neutral or even negative effect on the performance and to embody those genes permanently that explore novel degrees of freedom in a way that increases the fitness.

For every expansion cycle there are multiple ways to extend the fuzzy knowledge base. The first choice is whether to add a linear input term to the rule consequences or to refine the granulation of the input space by adding a new fuzzy set. The second decision concerns which of the input variables to add to the rule output, respectively which input variable to refine. If an input variable already contains multiple fuzzy sets, one also has to select the set to be partitioned. It is reasonable first to incorporate those input variables that are most relevant to the control problem. Based on the knowledge about the relevant control variables, the designer generates a schedule, which determines the order of variables and fuzzy sets by which the evolution strategy expands the knowledge base.

Instead of a predefined schedule of expansion, a constructional selection mechanism regulates the augmentation of the fuzzy knowledge base. An additional set of regulator genes governs the activation of the actual genome containing the object variables as shown in Fig.2.

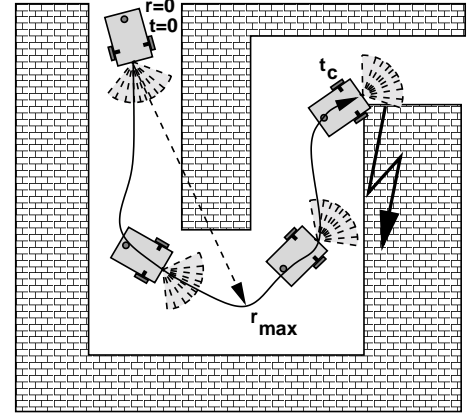
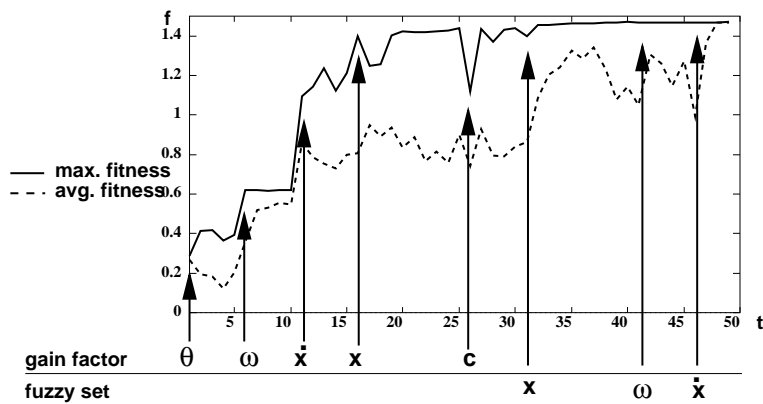


Figure 4 Left : Average and maximum fitness measures vs. time for the cart-pole balancing problem. The arrows indicate the generations at which the genome acquires new gain factors or partitions the input fuzzy sets. Right : Fitness evaluation of controllers, t_c : time until collision, r_{max} : maximum distance from starting point

Each regulator gene corresponds to a specific input variable and is composed of two integers. The first value defines the number of fuzzy sets per input variable, which in return determines the overall number fuzzy rules. The second binary value indicates whether the consequent parts of fuzzy rules incorporate a linear term related to this variable or not. Therefore, regulator genes define the way in which a genotype is mapped into fuzzy membership functions, gain factors and fuzzy rules. They are subject to discrete mutations, which in consequence alter the genotype-phenotype map and therefore effect the structure of the fuzzy knowledge base. Mutation of regulator genes operates in a way that gradually increases the complexity of the controller.

A constructional selection mechanism evaluates the effect of a genome expansion on the fitness of future offspring as depicted in the right part of Fig.3. A small random set of already adapted individuals acquires the new gene initialized with a random allelic value. The fitness increase of constructional mutations is compared with improvements resulting from mere allelic mutations. A new gene that produces a significant rise in fitness in at least one test case is considered a candidate for genome expansion. The evaluation of constructional mutations is repeated for a variety of gene additions having different functional effects. Finally, the genome permanently embodies the gene addition that demonstrated the largest fitness increase. The expanded genome is adapted through allelic mutations and recombinations allowing the population to converge to a new suboptimal fitness peak before the next constructional growth takes place.

In the following we illustrate the structural genome development in the light of the cart-pole balancing control problem. A pole is pivoted to a cart moving on a horizontal track. The controller can exert a limited force on the cart either in the left or right direction. The pole

angle θ , angular velocity ω , cart position x and velocity \dot{x} define the state of the system.

The task is to balance the pole in a vertical position, while keeping the cart between left and right boundaries and finally return it to the origin. According to these two objectives the score f of a controller depends on the ending time t_{end} of a trial and the mean square deviation $\int x^2(t)dt$ of the cart position from the center.

$$f = t_{end}/5 \quad : \quad t_{end} < 1$$

$$= t_{end}/5 + (1 - 1/4) \int_{t=1}^{t=5} x^2(t)dt \quad : \quad t_{end} \geq 1$$

Fig.4 depicts the evolution of the maximum and average fitness of the population for a design run over fifty generations. Constructional genome extensions are evaluated every fifth generation indicated by the vertical arrows along the time axis. The constructional selection scheme first incorporates the pole angle θ as the most relevant input variable, since preventing the pole from tipping over constitutes the primary control objective. In subsequent expansions the genome acquires gain factors for the angular velocity ω , the cart velocity \dot{x} , the cart position x and a constant offset c . Finally, the domains for the input variables x , ω and \dot{x} are partitioned into two fuzzy sets each.

The correlations among structural expansions and increases in fitness in Fig. 4 indicate that the evolution strategy explores novel functional features of the fuzzy rule base to refine the control behavior. The constructional selection mechanism incorporates new genes in order of their relevance to the control task. Therefore, incremental genome growth provides a suitable means to gradually adapt a control behavior by subsequently refining the already adapted fuzzy rules. The incremental design approach provides an alternative for genetic fuzzy system in cases where the complexity of the control prob-

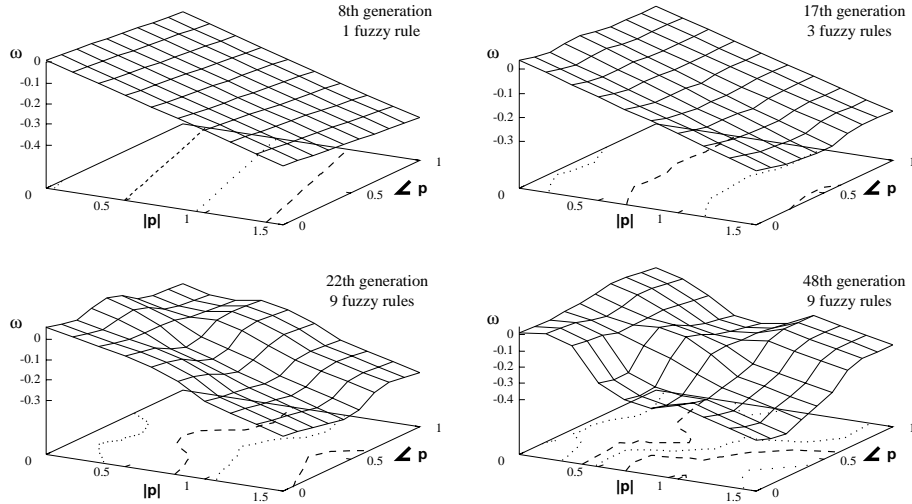


Figure 5 Change in heading ω as a function of $|\vec{P}|$ and $\angle \vec{P}$ after 8 generations, 1 fuzzy rule (upper left), 17 generations, 3 fuzzy rules (upper right), 22 generations, 9 fuzzy rules (lower left), 48 generations, 9 fuzzy rules (lower right)

lem does not allow the evolutionary algorithm to adapt the entire fuzzy knowledge at one step.

5 Mobile Robot

During the last few years, several researchers have proposed evolutionary computation for the adaptation of robotic behaviors [3]. This new field of *evolutionary robotics* is concerned with the design of intelligent systems with life-like properties by means of simulated evolution.

An essential distinction among the approaches in evolutionary robotics is in the genetic structure that undergoes adaptation. Neural network controllers [3], genetic fuzzy systems [2][7][16] and genetic programming [13] were proposed as learning techniques for the control of real robots.

We utilized the evolutionary learning method proposed in the previous section to adapt a wall-following behavior of a mobile robot that is implemented by means of fuzzy control rules. The evolutionary design process takes place by a simulation of the robot, its sensors and the training environment. The simulation allows for noisy actuators, imprecise distance measurements, changing reflection properties of objects and failures of sensors to detect the sonar echo signal.

5.1 Fitness Function

The desired wall-following behavior is specified by a scalar fitness function describing the performance of the controller. Each controller is tested for a training set of environments, which differ in the geometry and dimension of walls and obstacles. The robot is started in every environment at various initial positions and headings. A single run either ends when the robot collides with a wall or if a maximum number of control steps is exceeded.

The performance of the robot is comprised into a

scalar fitness value based on two conflicting objectives. The primary objective is to avoid any collision with walls or obstacles blocking the robot's path. Therefore, the fitness value of a controller is proportional to the time it guides the robot without a collision. This sole criterion is not sufficient, since a robot that simply turns on the spot without moving forward easily achieves this task. In order to elude the evolution of this trivial, but inadequate control strategy, the fitness function includes a second component that rewards the robot for moving along a straight path. A simulated run stops either if the robot collides with an obstacle or it operates for a maximum trial time T_{max} without having a collision. At the end of a trial the controller is rewarded with the period of time t_c that elapsed before the collision. In order to encourage exploration of the environment the controller is also rewarded with the maximum distance r_{max} of the described trajectory to the starting point as depicted in Fig.4. The fitness f for a single run is the product of the time t_c and the distance r_{max} .

$$f = t_c * r_{max} \quad (3)$$

The overall fitness of a controller is computed as the average fitness in all runs.

5.2 Perception of the Environment

Ultrasonic sensors provide the robot with low-quality information about its environment. The concept of *general perception* introduced by BRAUNSTINGL ET AL [2] attempts to model the environment indirectly through a representation of the perception itself rather than reconstructing its geometry in a straightforward way. The sensory perception is represented by a two component vector, which comprises the distance information provided by the five sensors. The direction of the perception vector vaguely indicates the location of close walls and

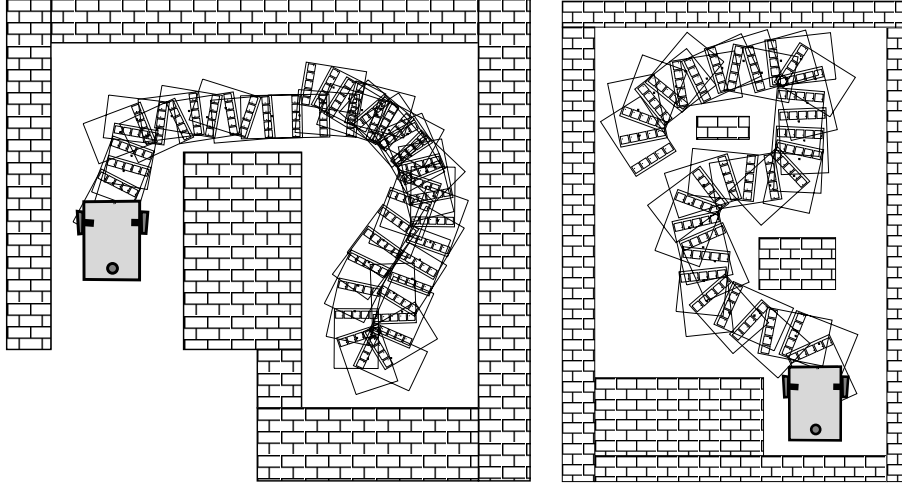


Figure 6 Runs of the mobile robot in a real world environment

obstacles. Its magnitude reflects the proximity of the nearby objects. A perception vector $\vec{p}_i(t)$ is calculated for every ultrasonic sensor i . Its direction coincides with the sensor axis while its length is inverse by proportional to the distance $d_i(t) \in [0, d_{max}]$ as measured by sensor i :

$$|\vec{p}_i(t)| = \frac{d_{max} - d_i(t)}{d_{max}} \quad (4)$$

where $d_{max} = 2m$ specifies the upper range for a dependable detection of objects. The overall perception vector $\vec{P}(t)$ equals the sum of the individual perceptions $\vec{p}_i(t)$.

The robot reacts to the perceived input by changing its direction of motion during the next time step. Based on the magnitude $|\vec{P}|$ and the direction $\angle\vec{P}$ of the perception vector, a fuzzy controller computes a desired rate of change in heading $\omega(t)$, which is converted into the angular velocities of the driven wheels.

Fig.5 depicts how the evolution strategy shapes the control surface by refining the input space and increasing the number of linear control rules. In the beginning, the object variables contain only one set of gain factors (c_{P_D}, c_{P_M}) and offset c_0 representing a single fuzzy rule

$$\begin{aligned} &\text{if } P_D = \text{any} \text{ and } P_M = \text{any} \\ &\text{then } \omega = c_0 + c_{P_D} |\vec{P}| + c_{P_M} \angle\vec{P} \end{aligned}$$

which covers the entire input space (see Fig.5 upper left). After ten generations the input variable $\angle\vec{P}$ is partitioned into three fuzzy sets $P_{D_1}, P_{D_2}, P_{D_3}$ corresponding to three fuzzy rules. The evolution strategy utilizes the additional object variables to further optimize the control surface, which starts to reveal some nonlinearities (see Fig.5 upper right). After another ten generations, the second input variable P_M corresponding to $|\vec{P}|$ is also refined to three fuzzy sets, with the result that the total number of fuzzy rules increases to nine (Fig.5 lower left). Finally, the control surface converges to the solution shown in the lower right graph of Fig.5, which repre-

sents the best controller found by the evolution strategy after fifty generations.

Compared to an evolution strategy that immediately starts with the complete set of object variables, the incremental method converges faster to the optimal solution. Taking the whole optimization task in hand bears the risk of premature convergence to brittle solutions over-adapted to coincidental features of the training situations. The incremental learning scheme avoids this risk by gradually increasing the number of parameters that the evolution utilizes for the optimization of the controller. An intermediate, suboptimal controller of limited complexity provides a beneficial point of departure to the design of more complex successors in future generations. Natural evolution widely employed the same concept of increasing the size of the DNA in order to create new species of higher complexity.

5.3 Experiments with the Mobile Robot

The evolution strategy employed a population of twenty controllers, which was evolved over fifty generations. The control function of the best individual is depicted in the lower right graph of Fig. 5. The evasive action becomes stronger with increasing magnitude and decreasing direction of the perception vector, reflecting that close objects located in the current direction of motion are most dangerous to the robot.

Fig.6 shows two runs of the best controller implemented on the physical robot. In the left figure, the robot starts facing north, traverses the corridor and turns around when perceiving the dead-end and finally follows the corridor to the exit. For reasons of clarity, the final part of the robot's path to the exit is not plotted anymore. The right figure shows an environment with isolated obstacles avoided by the robot. Notice that both environments are different from the training situations in which the reactive navigation behavior was adapted.

The experiments demonstrate that evolving in simulation is a means to generate robotic behaviors that exhibit sufficient generality to be transferable to real robots.

6 Summary

This paper presented an evolution strategy for the adaptation of TSK-type fuzzy logic controllers. It proposed an incremental learning scheme, which gradually increases the complexity of the fuzzy knowledge base in the course of the evolutionary process. The expansion of fuzzy rules and fuzzy sets either follows a predefined schedule specified by a human designer or is controlled by the evolutionary algorithm through additional regulator genes. In the latter case, a constructional selection mechanism compares the functional effects of different newly incorporated genes. Genome expansions that significantly improve the control performance become permanently embodied in the knowledge base structure.

The proposed design method was applied to learn a local navigation behavior of a mobile robot. Experiments with the physical mobile robot demonstrate, that an evolutionary algorithm can in principle design a robust control system that is able to cope with the uncertainty and imprecision inherent to real-world situations. The ultimate criterion for the success of evolutionary robotics is whether its techniques are able to learn more complex behaviors that can not be easily designed by a human expert.

Acknowledgment

I thank Oliver Malki from the University of Kiel for carrying out the experiments with the real robot. This work was funded by a research grant (Ho 1790/2-1) from the Deutsche Forschungsgemeinschaft and supported by the Army Research Office under grants DAAH 04-96-1-0341.

References

- [1] Th. Bäck, H. P. Schwefel, "Evolution Strategies I: Variants and their computational implementation", "Evolution Strategies II: Theoretical aspects and implementation", *Genetic Algorithms in Engineering and Computer Science*, Ed. G. Winter et al, pp. 111-126, pp. 127-140, Wiley & Sons, (1995).
- [2] R. Brauningl, J. Mujika, J. P. Uribe, "A Wall Following Robot With a Fuzzy Logic Controller Optimized by a Genetic Algorithm", *Proc. FUZZ-IEEE'95*, vol. V, pp.77-82, Yokohama, (1995)
- [3] D. Cliff, I. Harvey, P. Husbands, "Explorations in Evolutionary Robotics, *Adaptive Behavior*, vol. 2, p.73-100, (1993).
- [4] O. Cordon, F. Herrera, "A General Study on Genetic Fuzzy Systems", *Genetic Algorithms in Engineering and Computer Science*, pp. 33-57, John Wiley & Sons, (1995).
- [5] D. E. Goldberg, *Genetic Algorithms in Search, Optimisation and Machine Learning*, Reading Massachusetts: Addison-Wesley, (1989).
- [6] F. Hoffmann, G. Pfister, "Learning of a Fuzzy Control Rule Base Using Messy Genetic Algorithms", *Genetic Algorithms and Soft Computing*, Ed. F. Herrera, J. L. Verdegay, Physica-Verlag, p. 279-305, (1996).
- [7] F. Hoffmann, G. Pfister, "Evolutionary Design of a Fuzzy Control Rule Base for a Mobile Robot", *Int. Journal of Approximate Reasoning*, vol. 17, no. 4, (1997).
- [8] J. H. Holland, *Adaptation in Natural and Artificial Systems*, MIT Press, Cambridge, MA, (1992).
- [9] C. L. Karr, "Design of a Cart-Pole Balancing Fuzzy Logic Controller using a Genetic Algorithm" *SPIE Conf. on Applications of Artificial Intelligence*, Bellingham, WA, (1991).
- [10] J. R. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs*, MIT Press, Cambridge MA, (1994).
- [11] E. H. Mamdani, "Application of fuzzy algorithms for control of simple dynamic plant", *Proceedings of the Institution of Electrical Engineers*, vol. 121, no. 12, pp. 1585-8, (1974).
- [12] Heinz Mühlenbein, Dirk Schlierkamp-Voosen, "The science of breeding and its application to the breeder genetic algorithm BGA", *Evolutionary Computation*, 1(4): 335-360, (1994).
- [13] P. Nordin, W. Banzhaf, "An On-Line Method to Evolve Behavior and to Control a Miniature Robot in Real Time with Genetic Programming" *Adaptive Behaviour*, 5 (2), pp. 107-140, (1997).
- [14] T. Takagi, M. Sugeno, "Fuzzy identification of systems and its applications to modelling and control", *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-15, no.1, pp.116-32, (1985).
- [15] H. Takagi, M. Lee, "Integrating Design Stages of Fuzzy Systems using Genetic Algorithms", *Proc. of the Seond IEEE Int. Conf. on Fuzzy Systems*, pp. 612-617, (1993).
- [16] T. Yoshikawa, T. Furuhashi, Y. Uchikawa, "Knowledge acquisition of fuzzy control rules for mobile robots using DNA coding method and pseudo-bacterial GA", *Simulated Evolution and Learning, SEAL '96*, Taejon, South Korea, pp. 126-35, (1997).