

Evolutionary Learning of Mobile Robot Behaviors

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 describes a messy genetic algorithm for the automatic design of fuzzy logic controllers. The method is applied to adapt a wall following behavior of a mobile robot.

Keywords: messy genetic algorithms, fuzzy control, mobile robot, adaptation of behavior

1 Introduction

Designing controllers for mobile robots by hand becomes a difficult task as soon as the behaviour becomes more complex. In many applications the robot's environment changes with time in a way that is not predictable by the designer in advance. In addition, the information available about the environment is subject to imprecision, incompleteness and imperfection due to the limited perceptual quality of the sensors. These problems limit the utility of traditional model-based reasoning approaches for the design of intelligent robots.

Evolutionary computation provides an alternative design method that adapts the robot's behavior without requiring a precisely specified model of the world. Its adaptive power enables the robot to deal with changes in the environment and to acquire a robust behavior tolerating noisy and unreliable sensor information.

Evolutionary algorithms constitute a class of search and optimization methods guided by the principles of natural evolution. Evolutionary computation plays an important role in modeling and designing artificial-life systems [7]. Essentially, artificial life and machine learning understand evolutionary algorithms as an abstraction of natural evolution in that they adapt a system to its environment. "Evolutionary robotics" is concerned with the design of systems with lifelike properties by optimizing the robot's controller with respect to some objectives. The essential distinction of the proposed methods is the way in which the evolutionary algorithm represents the controller in the genotype. Dynamic recurrent neural nets, tree-structured programs [8], classifier systems [3] or fuzzy rules [1][6] have been employed to implement the control function.

Fuzzy systems employ a mode of approximate reasoning, which allows them to make decisions based on imprecise and incomplete information in a way similar to human beings. 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. Fuzzy control provides a flexible tool to model the relationship between input information and control output and is distinguished by its robustness with respect to noise and variation of system parameters.

Soft computing is concerned with the design of intelligent and robust systems, which exploit the tolerance for imprecision inherent in many real world problems. In order to achieve this objective, soft computing suggests a combination of fuzzy logic, neural networks, probabilistic reasoning and genetic algorithms. Soft computing advocates that the integration of these complementary methodologies, each of them adequate for its specific domain of problems, results in more powerful hybrid methods than using a single method exclusively.

Recently, numerous researchers have explored the integration of evolutionary algorithms with fuzzy systems[2] in so-called genetic fuzzy systems (GFS). The majority of publications are concerned with the automatic design or optimization of FLCs either by learning the fuzzy if-then rules or by tuning the fuzzy membership functions. Promising results have been achieved by employing evolutionary methods to develop rule based behaviors for mobile robots, given the task of prey following [1], light following [3] and collision avoidance [6]. A significant part of the engineering task of designing an intelligent robot is delegated to the evolutionary algorithm, which explores alternative behaviors and optimizes the controller's parameters.

2 Mobile Robot

The mobile robot is given the task of following a corridor and avoiding collisions with the walls. It perceives its environment by means of five ultrasonic sensors. Each of the sensors covers an angular range of approximately 50°. The probability of omitting an object increases

at the borders of the perceptual field. The orientation of the sensors on the robot is adjustable. An optimal perception of obstacles is achieved by a compromise between a sufficient overlap of the individual sensors and a wide total field of vision.

Compared to optical sensors, sonar distance information is much more afflicted with incompleteness and imprecision because of the physical characteristics of ultrasonic signals and varying reflection properties for different kinds of objects. Therefore, it is impossible to determine the exact environmental situation using the actual sensor data solely. As a result, controllers turned out to be unable to recognize deadends in a reliable way [6].

A two-layer feedforward neural network preprocesses the sensor data in order to guarantee a more reliable detection of obstacles. The network obtains the four previous distance measurements of the five sensors as input. If an object closer than $1.5m$ is detected by a sensor at time $t - n\Delta t$, $\{n = 0, 1, 2, 3\}$, the input is 1, otherwise it is 0. Additional input neurons contain the four previous steering angles of the robot, and the orientation of the sensors in relation to the robot's heading. The input is mapped to three output neurons, which specify, if an obstacle appears to the left, in front of or to the right of the robot. The network was trained with the backpropagation algorithm using a training set of 4000 input-output pairs generated by means of a simplified sensor model.

The behavior of the mobile robot is implemented by means of fuzzy control rules. The fuzzy controller receives input from eight crisp variables, the five distances measured by the sensors and the three outputs of the neural net. These three variables contain information on the environmental situation. Their essential role is to define additional constraints on the antecedents, in a way that certain rules become active only in a specific context. Instead of only using the current sensor data, the crisp input values of the distance variables are calculated as the minimum distances measured during the last time steps. The number of previous measurements taken into account is a compromise between completeness and topicality of information.

3 Evolutionary Algorithm

Evolutionary algorithms process a population of competing candidate solutions from one generation to the next. Each individual is represented by a set of parameters called a genotype. The algorithm evaluates the quality of an individual in regard to the optimization task by means of a scalar fitness function. According to Darwin's principle, highly fit individuals obtain a better chance to reproduce offspring to the next generation. Genetic operators such as crossover and mutation are applied to the selected parents in order to generate

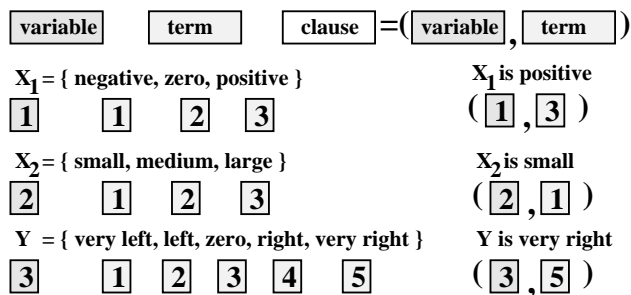


Figure 1: Coding of fuzzy clauses

new candidate solutions. In the course of time, more and more suitable solutions to the optimization problem emerge within the population.

3.1 Messy Coding Scheme

The choice of an appropriate genetic representation of candidate solutions plays a crucial role in the design of an evolutionary algorithm. In a conventional coding scheme for fuzzy rulebases the chromosome contains one output term for every possible combination of antecedents. The number of rules and thereby the size of the chromosome increase rapidly with the number of input variables, with the result that the evolutionary optimization becomes less feasible.

This paper proposes a messy coding scheme for fuzzy rules and rulebase in order to maintain the efficiency, robustness and comprehensiveness of the fuzzy controller. The coding is able to bound the size and complexity of the rulebase, especially for a larger number of input variables. A fuzzy rulebase is constituted by a smaller number of fuzzy rules, which correspondingly encompass larger regions of input space. The task of designing the fuzzy controller remains tractable for the genetic algorithm, since it benefits from the more compact genetic representation.

Our coding imitates the genetic representation used in messy genetic algorithms [4]. The function of a gene is part of its coding instead of depending on its position within the chromosome. This property allows chromosomes of variable length in which genes are arranged in any order. Changing the order of the genes, enables the genetic algorithm to generate building blocks more efficiently by minimizing the defining length of highly fit schemata.

First of all, the linguistic variables and terms are numbered consecutively without distinguishing between input and output variables. Fuzzy clauses, represented by a gene of two integers, constitute the basic elements of our coding scheme. The first integer value defines the fuzzy variable, the second value specifies the associated linguistic term. In the example depicted in fig. 1 the gene (1,3) encodes the fuzzy clause $X_1 \text{ is positive}$, in which the integer 1 defines X_1 as the variable, and the

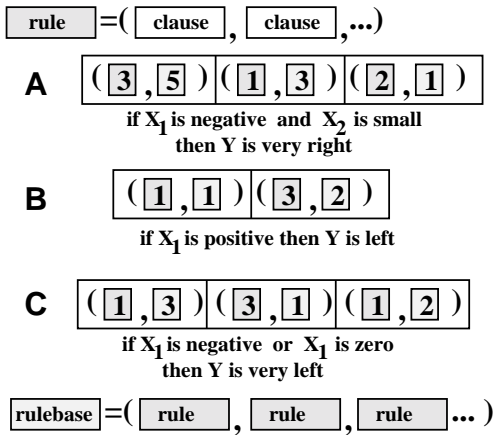


Figure 2: Coding of fuzzy rules and rulebase

value 3 determines the linguistic term *positive*.

A chromosome encoding a fuzzy rule, is formed by a set of basic genes as shown in fig. 2. Notice, that a chromosome can contain an arbitrary number of genes arranged in any order. As a consequence, the coding scheme has to handle overspecification, in case genes with different linguistic terms for the same variable occur more than once in the chromosome. The chromosome *C* in fig. 2 has two contradicting genes (1,2) and (1,3) for the input variable X_1 . Both corresponding clauses X_1 is zero and X_1 is positive are comprised in the fuzzy rule by combining them with a fuzzy OR operation.

In a messy coding scheme the dual problem to overspecification is underspecification, which arises whenever the chromosome lacks one or more genes. The chromosome *B* in fig. 2 contains no gene with an integer 2 in the first position. In this situation, the coding scheme considers X_2 as a do-not-care term for which the fuzzy clause is omitted in the antecedent. The initialization and the recombination of chromosomes are slightly modified in order to guarantee, that each chromosome contains at least one gene for the output variable.

Finally, a set of these chromosomes for fuzzy rules builds up the coding of one individual representing an entire rulebase. Overspecification of the rulebase causes no problem, since the output of a fuzzy controller is an interpolation of the control actions suggested by the firing rules. Underspecification occurs, if none of the rules match the actual control input. In this case, a rule generation process is put into operation, which inserts a new chromosome matching the current input and acquires a randomly chosen control action.

In addition to a modified coding scheme, messy genetic algorithms also employ different genetic operators for recombination [4]. The usual crossover operator is replaced by cut and splice operations. Cut and splice operators are applied on the level of rules, as well as on the level of the rulebase. A matching procedure compares the chromosomes of two parent rules, so that

only clauses of rules bearing a minimal resemblance are recombined. On the level of rulebase recombination, the offspring's chromosome is afterwards completed with some of the parent rules that became lost during cut and splice. In a messy coding scheme chromosomes can grow or shrink in size. This enables the genetic algorithm to adapt the number of rules and the number of predicates in the antecedent to the complexity of the control problem. The messy coding scheme as well as the cut and splice operators are described in more detail in [5].

3.2 Evolutionary Strategy

An evolutionary strategy is run in parallel to the messy genetic algorithm designing the rulebase in order to optimize additional real valued parameters of the controller and the sensors. Each candidate solution further embodies scaling factors for input and output variables, the orientation of the sensors and the number of preceding sensor measurements taken into account.

The membership functions, which characterize the underlying fuzzy sets of linguistic terms are defined in advance, in a way that reflects the designer's perception with regard to the dimensions of variables, e. g. *near* distance or *sharp left* steering. Therefore, the designer is able to comprehend and to interpret the rulebase as automatically generated by the genetic algorithm. On the other hand, an inappropriate definition of membership functions may result in a suboptimal control behavior. A reasonable compromise between the clarity of fuzzy rules on one hand and optimal approximation to the control task on the other hand can be achieved, by introducing two linear scaling factors for input and output. The first one scales the sensor input of all distance variables in common, while the second one plays the role of an output gain applied to the steering angle. The meaning of linguistic terms is preserved, while at the same time the evolutionary strategy is able to adjust the sensitivity in regard to close objects and the magnitude of the control action.

Another real valued parameter determines the angle between the sensor axes. The evolutionary strategy adjusts this parameter in order to achieve an optimal compromise between a sufficient overlap of the sensory fields and a wide range of total perception.

Finally, a fourth parameter serves as some kind of memory time, which defines how many preceding sensor measurements are comprised within the distance variable. A longer memory time improves the reliability of distance information, but on the other hand reduces the topicality of the data.

3.3 Fitness Evaluation

The evaluation of the control behavior by means of a scalar fitness function depends on the objectives and constraints of the robotic application. There are only

a few examples [1][8], in which the real robot is used to evaluate the controller. Often, online performance evaluation is too time consuming or even becomes impossible in case a poor controller may cause severe damage to the robot. Therefore, most of the approaches employ a quantitative simulation model of the robot and its environment.

For practical reasons, the simulation of the robot is based on some idealistic assumptions about its dynamics, environment and sensors, but still bears sufficient resemblance to the real world, e. g. it models incomplete and noisy distance information on the obstacles. The tolerance of fuzzy controllers in respect to noise and imperfection is exploited, in that the evolutionary algorithm utilizes a simplified model for fitness evaluation. Previous experiments with the robot demonstrated that the controllers adapted in the simulation achieve an identical performance in the physical reality[6].

During the fitness evaluation, the fuzzy controllers are tested on the simulated robot, which is placed in several different kinds of corridors. They are formed by deadends, straight segments and branchings at a right angle leading to an exit either located to the right or to the left. In order to find its way out of the corridor, the robot has to distinguish among these different types of segments. A simulated run of the robot either stops if it collides with a wall or if a maximum evaluation time is exceeded. The fitness obtained by the controller for a single run is proportional to the time the robot travels without having a collision. The most simple obstacle avoidance behavior is to turn the robot on the spot. In order to elude the evolution of this trivial, but inadequate control strategy, the fitness is multiplied by the distance the robot covered from the starting point.

Fig. 3 shows a simulation of the robot behaviour adapted by means of the proposed method. The robot is initially heading towards the deadend. The controller is able to recognize the deadend, turn the robot and follow the walls of the corridor leading to the exit.

4 Conclusions

This paper presented an evolutionary design method for mobile robot behaviors implemented by means of fuzzy rules. The proposed messy genetic algorithm enables the formation of compact and thereby more efficient rule-bases. At the same time, an evolutionary strategy optimizes additional real valued parameters of the sensors and the controller. The robot's perception of the environment is improved by preprocessing the raw sensor data by a neural network.

Acknowledgements

This work was funded by the MURI (ARO DAAH04-96-1-0341) and BISC program of UC Berkeley.

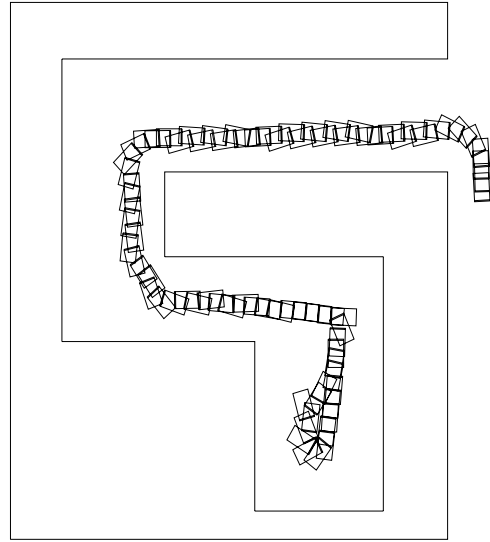


Figure 3: Simulated run of the robot with the evolutionary designed fuzzy controller

References

- [1] A. Bonarini, F. Basso, "Learning behaviors implemented as Fuzzy Logic Controllers for Autonomous Agents", *2. Online Workshop on Evolutionary Computation*, (1996).
- [2] O. Cordon, F. Herrera, "A General Study on Genetic Fuzzy Systems", *Genetic Algorithms in Engineering and Computer Science*, pp. 33-57, (1995).
- [3] M. Dorigo, U. Schnepf, "Genetics-Based Machine Learning and Behaviour-Based Robotics: A New Synthesis", *IEEE Trans. on SMC*, vol. 23, no. 1, pp. 141-154, (1993).
- [4] D. E. Goldberg, B. Korb, K. Deb, "Messy Genetic Algorithms Motivation, Analysis, and First Results", *Complex Systems*, vol. 3, pp. 493-530, (1989).
- [5] F. Hoffmann, G. Pfister, "Learning of a Fuzzy Control Rule Base Using Messy Genetic Algorithms", *Genetic Algorithms and Soft Computing*, (1996).
- [6] F. Hoffmann, O. Malki, G. Pfister, "Evolutionary Algorithms for Learning of Mobile Robot Controllers", *Proc. EUFIT'96*, pp. 1105-1109, (1996).
- [7] M. Mitchell, S. Forrest, "Genetic Algorithms and Artificial Life", *Artificial Life 1 (3)*, pp. 267-289, (1994).
- [8] P. Nordin, W. Banzhaf, "Genetic Programming Controlling a Miniature Robot", *AAAI Fall Symposium on Genetic Programming*, (1995).