ICGA-93

Automatic Generation of Programs for Crawling and Walking

Graham F. Spencer
Computer Science Dept.
Stanford University
Stanford, CA 94309

Abstract

This paper describes efforts to automatically generate programs that enable a simulated six-legged insect to walk. Genetic programming was used to evolve programs that controlled the motion of the insect's legs. Given a minimum amount of *a priori* knowledge about the task of locomotion, each of 28 separate trials evolved programs capable of controlling the legs for the purpose of walking.

1 THE PROBLEM

We followed Beer's physical model as described in *Intelligence as Adaptive Behavior* (Beer 1990). This model features an artificial insect with six legs, each of which may be up or down. The insect can raise or lower each leg, and can apply a force to move the leg.

2 THE APPROACH

We used genetic programming (Koza 1992) to generate programs to control the artificial insect. Our terminals were random constants and environmental variables. We included standard arithmetic functions (+, −, ×, /) in our terminal set. However, an expression composed of only standard arithmetic functions evaluates to a single output number. To control a six-legged insect we needed at least 12 outputs: 6 outputs to control the force applied to the legs, and 6 outputs to control whether the legs are raised or lowered. We solved this difficulty by using side-effecting functions. Our `Set-Leg` functions evaluated their single argument and returned this value as their own value, but their side-effect was to set the force or state of the leg. Note that the `Set-Leg` functions are the only outputs of the program, and the terminals are the only inputs; the actual value of the expression itself is ignored.

3 RESULTS

We ran three different experiments, and in each experiment it became progressively more challenging for the insect to walk. In our first experiment, we made two simplifications to encourage oscillatory motion in the legs: we included a `Clock` terminal, and we automatically reversed the force on legs when they were raised. In our second experiment, we removed the first simplification and provided feedback from the legs instead of a clock value. In our last experiment, we removed the second assumption. For programs to walk in this environment, the emergence of oscillation in the legs was *required*.

We ran a total of 28 trials (8 each in experiments 1 and 2, 12 in experiment 3). In each of the trials, programs evolved that were capable of efficient locomotion. Regardless of the conditions, each trial seemed to follow a similar pattern of evolution: the early best-of-generation individuals moved one or two of their legs through one cycle of motion and then fell. In later generations, individuals developed oscillatory motion in a few legs, and dragged the rest for balance. After about 50 generations, the oscillatory motion had propogated (through crossover) to all the legs, and the individuals were moving quickly and efficiently. Despite the similarities in the progress of evolution, the best strategies showed astonishing variety: some programs moved in long strides, some moved by leaping, and some moved with short hops.

Acknowledgements

I am grateful to John Koza for his invaluable advice in the conception and formulation of this paper.

References

Beer, R. D. (1990). *Intelligence as Adaptive Behavior.* New York: Academic Press.

Koza, J. (1992). *Genetic Programming.* Cambridge: MIT Press.

Using Genetic Algorithms in Structuring a Fuzzy Rulebase

Chuen-Tsai Sun
Department of Computer and Information Science
National Chiao Tung University, Hsinchu, Taiwan

Jyh-Shing Jang
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley

Fuzzy rulebase identification is the task of describing a complicated system with fuzzy if-then rules. In recent years researchers have been making efforts in automating this process. Neural networks, for instance, were used to optimize the system performance by adjusting system parameters such as the membership functions of fuzzy rules and the combination logics of the rules. From experience, however, we found that although individual parameters could be fine-tuned through adaptation, an optimal system configuration was still hard to achieve. The determination of the number of rules and the partition of the feature space remains a design art. In this study, we found that genetic algorithms (GAs) could play an important role in structuring a fuzzy rulebase.

Structure identification is a critical problem for modeling. A rulebase structure can be viewed as a partition in the multi-dimensional feature space. Existing neuro-fuzzy systems usually partition feature space into *adaptive fuzzy grids*. Two problems exist in this scheme. First, the number of linguistic terms for each input variable is predetermined and is highly heuristic. Second, the learning complexity suffers an exponential explosion as the number of inputs increases.

To cope with these problems, we adopted a flexible partitioning, the *fuzzy k-d trees*, for structure identification. A k-d tree results from a series of *Guillotine cuts*. There are various strategies to decide which dimension to cut and where to cut it at each step. With the learning ability of adaptive networks in mind, we do not need to find a perfect partition since our goal is just to find a *satisfiable* initial state for the adaptive network to tune. However, we do need to explore multiple partitions to avoid the major disadvantage of standard gradient descent methods, i.e., their inability to go uphill to get over barriers and find global optimal points. Consequently, we propose a method of employing genetic algorithms.

For feature space partitioning, we define the format of a chromosome as:

$$[C_1 C_2 \cdots C_{n-1}],$$

where each C_i encodes the information for a Guillotine cut in the feature space and n is the total number of rules. Each C_i contains two pieces of information: on which dimension to cut a subspace and at which point to cut it. Given a chromosome, we can decode it to get a partition and then evaluate it by the performance of that partition. Under this definition of the chromosome, all new structures resulting from crossover are legal partitions for our modeling scheme.

The standard mutation operator, which varies a single Guillotine cut randomly, results in a structure slightly different from the parent structure. However, since this kind of local variations can be taken care of by gradient descent methods, such as the adaptive networks we are using, we define two application-independent mutation operators which are used for exploring new structures, too. The first one is to switch two cuts in the partition sequence, with the switch position determined by a random number. The second is to rotate the cuts, with the shift length determined randomly.

To test the suitability of the proposed model in the real world, we applied it to a medical diagnosis problem. There were 145 samples in the data set; we used 116 of them as training data and the other 29 for testing. The population size was 100. Each member in the population corresponded to a feature space partition. We then constructed an adaptive network based on the partition and started the gradient descent learning process. At the end of the fifth training epoch, we took the number of mis-classified data items as the performance indicator. Based on the evaluation, the genetic algorithm selected the partitions that fit better and proceeded to the next generation. After applying the modeled classifier to the checking data, we got two near-misses and one mis-classified case. The result is better than the cluster analysis performed originally by the physicians.