

Evolutionary Computation and the Tinkerer's Evolving Toolbox

Philip G.K. Reiser
pgr94@aber.ac.uk

Centre for Intelligent Systems
University of Wales
Aberystwyth, Wales, UK

Abstract. In nature, variation mechanisms have evolved that permit increasingly rapid and complex adaptations to the environment. Similarly, it may be observed that evolutionary learning systems are adopting increasingly sophisticated variation mechanisms. In this paper, we draw parallels between the adaptation mechanisms in nature and those in evolutionary learning systems. Extrapolating this trend, we indicate an interesting new direction for future work on evolutionary learning systems.

1 Introduction

It has been argued, [9], that the complexity of life can be accounted for merely by a small number of stochastic processes. They are reproduction, mutation, competition, and selection.

Neo-Darwinism asserts that the history of the vast majority of life is fully accounted for by only a very few statistical processes acting on and within populations and species [18, p.39]. These processes are reproduction, mutation, competition, and selection. Reproduction is an obvious property of all life. But similarly as obvious, mutation is guaranteed in any system that continuously reproduces itself in a positively entropic universe. Competition and selection become the inescapable consequences of any expanding population constrained to a finite arena. Evolution is then the result of these fundamental interacting stochastic processes as they act on populations, generation after generation [23], [36, p.25] and others. ([9, p.37]).

It is through these elementary processes that the morphology, physiology and behaviour of an organism are adapted to the environment. However, the mechanisms that produce variation have not remained static. Instead, the variation mechanisms have also evolved so that increasingly complex adaptations occur. The resulting trend, the evolution of increasingly sophisticated adaptation mechanisms, has been referred to as the *tinkerer's evolving toolbox*, [27]. In this paper, a selection of variation mechanisms is considered and we investigate how these

mechanisms have (or in some cases have not) been modelled in evolutionary learning systems.

The mechanisms considered are (1) mutation (asexual reproduction); (2) genetic recombination (sexual reproduction); (3) nervous system; (4) symbolic reasoning; and (5) language. In actual fact, these variation mechanisms are cumulative, as summarised in Table 1, and so their effect is compounded.

Compound Variation Mechanism
mutation
mutation + recombination
mutation + recombination + nervous system
mutation + recombination + nervous system + reasoning
mutation + recombination + nervous system + reasoning + language

Table 1. The accumulation of variation mechanisms in evolutionary adaptation

In the next sections, we describe each mechanism in turn, and describe evolutionary learning systems that employ these mechanisms.

2 Mutation

The four basic processes identified above are sufficient to allow the genotype to be transformed to cause the organism’s phenotype to be highly adapted to its environment. However, the efficiency of adaptation at this basic level is very poor. Truly random mutation is a double-edged sword: as the mutation rate increases, the possible rate of adaptation also increases, but simultaneously, there is a higher risk of destructive mutations occurring.

Consequently, any organism that exhibits a slightly better means of adaptation than a uniform distribution of random mutations will propagate rapidly. Indeed, the processes reported in the biological literature are far more complex than random mutation. For instance, there is increasing evidence to suggest that there exist mechanisms encoded in the genotype that are able regulate the rate of mutation, [27, 31, 30].

Variation through mutation is modelled in Evolutionary Programming, [12, 11], which concerns the simulation of asexual reproduction. Algorithms are based upon the processes of reproduction, selection, competition, and mutation. Mutation is viewed as an operation that causes perturbations while “[maintaining] a behavioural link between each parent and its offspring”, [11]. These approaches have been successfully applied to real-world and inherently difficult problems, (e.g. the travelling salesman problem which is NP-hard, [10]).

3 Genetic Recombination

In asexual reproduction, the only changes to the genotype come about by chance mutations—offspring remain genetically similar, if not identical, to the parent. With the evolution of sexual reproduction, offspring genotypes arise not from a number of small changes to the genotype, but are composed of two potentially very different genotypes through genetic recombination. As a result the population diversity influences the degree of change between parent and offspring.

In nature, sexual reproduction suffers disadvantages such as finding a suitable mate, yet it is ubiquitous in higher order organisms. This suggests it is advantageous from the point of view of adaptation.

There are several classes of evolutionary algorithm that are based on genetic recombination. Genetic recombination, or *crossover*, allows good partial solutions, or *building blocks*, to be propagated to subsequent generations. As described by Goldberg’s building block hypothesis: “instead of building high-performance strings by trying every conceivable combination, [genetic recombination] constructs better and better strings from the best partial solutions of past samplings”, [15, p.41]. We briefly describe several basic models.

The Genetic Algorithm, [21, 15], is distinguished by a fixed-length binary string representation and a crossover operator that does not affect the length of the string. An example is illustrated in Figure 1(a) and some successful applications are described in [16].

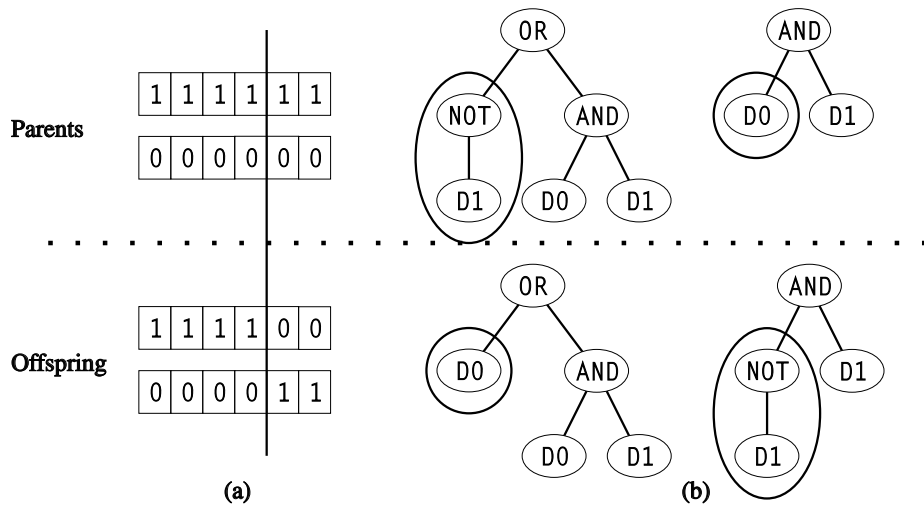


Fig. 1. (a) Crossover of fixed-length binary strings in genetic algorithms; (b) crossover of trees of Lisp S-expressions in genetic programming.

In Genetic Programming, [25], the structures undergoing adaptation are general, hierarchical computer programs of dynamically varying shape and size.

These are expressed as trees of Lisp S-expressions. New programs are primarily produced by moving branches from one tree and inserting them into another. This ensures that programs created by crossover are syntactically valid. Figure 1(b) shows an example of a crossover operation.

Evolutionary Programming, Genetic Algorithms and Genetic Programming are computational models that simulate the processes believed to occur in sexual and asexual reproduction. While many of the details of natural systems have been discarded, the learning algorithms are capable of solving a wide variety of problems, suggesting that some useful processes have been abstracted. However, in the models described so far the organisms, or agents, do not vary their behaviour with respect to their environment. Adaptation only occurs over one or more generations. In the next section, we advance along the evolutionary timescale to consider the mechanisms that enable an organism to adapt its behaviour.

4 Nervous Systems

The information reservoir of an organism comprises not solely of that held in the genotype, but also includes information stored in other ways. For example, information is stored in the firing properties and topology of nerve cells in the nervous system. Changes to nerve cell properties allow the organism's behaviour to modify and adapt during its lifetime.

Genetic adaptation and nervous system adaptation operate over different time scales. Changes to the organism's environment can only be responded to slowly, requiring of the order of generations. The nervous system, on the other hand, is a mechanism that enables rapid adaptive behaviour. Learning in the nervous system allows plasticity in the behaviour of an organism, while genetically-determined behaviour (or instinct) does not vary within the organism's lifetime and therefore tends to be brittle.

Evolutionary learning systems have been constructed that model the evolution of organisms with a nervous system. The nervous system is modelled by a neural network whose structure (or topology) and weights correspond to interconnections between neurons and their firing properties. Neural architectures are evolved by an evolutionary algorithm similar to those described in previous sections. The genotype may encode information about neural network topology, the functions computed by the neuron (e.g. threshold, sigmoid, etc.), and the connection weights, [2]. Over several generations, the population evolves towards genotypes that correspond to high-fitness neural networks. This relation between evolutionary and neural network algorithms is illustrated in Figure 2.

Both genetic algorithms and neural networks have independently demonstrated themselves useful in solving difficult real-world problems. Consequently, the evolution of neural networks has attracted attention in the hope of capturing the benefits of both GA and NNs. The next section, however, will describe a task for which this approach is poorly suited.

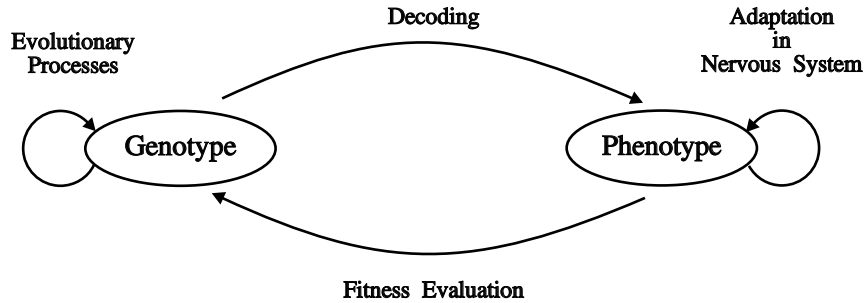


Fig. 2. Evolutionary design of neural networks (adapted from [2]). The genotype is decoded to determine properties of a neural network. The neural network, interacting with the environment, adapts to it; and the better adapted, the greater the probability that the genotype will contribute to the population of the next generation.

5 Symbolic Reasoning

In the 1920s, Wolfgang Köhler studied learning and problem solving in chimpanzees. In a typical experiment, Köhler placed a chimpanzee in an enclosure with a desirable piece of fruit, often a banana, out of reach. The animal had to use nearby objects (e.g. stack several boxes strewn around the enclosure) in order to obtain the fruit. Several interesting observations were made, [1]: (1) rather than being a gradual trial-and-error process, the solution was found suddenly; (2) once solved, repeating the problem resulted in few irrelevant moves; and (3) the animal was able to transfer what it had learned to novel situations.

One popular view is that the animal constructs an internal representation or model of the environment and the objects in it. The model might be a map of the environment or one or more abstract concepts. It is then possible to reason about this model rather than operate on the world itself.

Furthermore, in building a representation of the environment, objects are grouped into sets of properties or concepts. Treating different objects as members of the same concept with the same properties allows different environmental situations to be responded to in a uniform way, thus significantly reducing the complexity of the environment. Concepts may be combined to form propositions, and propositions may be combined to form further propositions. This combining process is commonly referred to as inference.

But how networks of neurons achieve this kind of complex behaviour is still poorly understood and remains a challenging problem for those trying to unravel these phenomena. It is perhaps significant that the evolution of complex reasoning from the nervous system took a very long time even by evolutionary timescales. Unless the process is guided in some way, it is unrealistic to expect to

evolve such complex inference mechanisms in neural networks with the limited time and computational resources available.

Several models of evolutionary learning have distanced themselves from their natural counterparts in the following ways.

1. The model of the neuron is discarded altogether and a representation amenable to symbolic manipulation employed instead. This avoids the problem of constructing complex neural networks.
2. The two-tiered representation (i.e. information stored in both genotype and the network of neurons) is replaced with a single representation.¹

The remainder of this section describes learning systems that make the above two assumptions in order to facilitate reasoning about the environment. Each approach uses a symbolic internal model which is constructed using evolutionary techniques, however, they vary along the following dimensions:

Procedural vs. declarative semantics. Evolving code of conventional programming languages using operators such as crossover and mutation will seldom produce syntactically correct programs, and even less frequently semantically correct ones, [6]. Examples of conventional languages used in evolutionary learning systems include machine language, [13], and state transitions of a finite state machine, [12], as well as high-level languages such as C and Pascal.

Conventional programming languages typically have procedural semantics, and are characterised by a changeable store. The commands of a program influence one another by means of variables held in storage. The relationship between two given commands can be completely understood only with reference to *all* the variables that they both access, and with reference to all the other commands that access these same variables, [33, p. 230]. Clearly, commands are dependent on their context, and operations that change context in an ad hoc manner, such as crossover for instance, will have a poorly understood effect on program meaning.

Representations with declarative semantics on the other hand do not rely on a changeable store and therefore the meaning of a program section is less dependent on its context. This suggests better suitability to manipulation and genetics-based operations. Examples of systems that employ languages with declarative semantics include Learning Classifier Systems, [20, 29], Genetic Programming, [25], and Genetic Logic Programming, [34, 35].

¹ A multi-cellular organism stores an instance of the genotype in each cell. Therefore, it is physically very difficult for a multi-cellular creature to modify its genotype rapidly in response to changes in its environment. This difficulty is overcome with behaviour governed by a central organ: the nervous system. Evolutionary change occurs through the culmination of improvements. While this represents a good motivation for having two representations in a natural system, it is not an issue for computer architectures which can flexibly modify data.

Intermediate representation. Approaches may be classified on whether or not they try to fully exploit the implicit parallelism of the genetic algorithm. To draw on the genetic algorithm's implicit parallelism, a representation, typically a subset of first order logic, is associated with a string of elements taken from a binary alphabet. The binary strings are then mapped to a logical expression which is used to reason about the problem domain. For instance,

$$\begin{array}{c} A \quad \wedge \quad B \quad \rightarrow \quad C \\ \hline \boxed{1 \ 0} \quad \boxed{0 \ 0} \quad \boxed{1 \ 1} \end{array}$$

Evolving binary strings has the effect of discovering new rules. Examples include Learning Classifier Systems, [20, 29], and genetic concept learning, [7].

The alternative approach is to allow the genetic operators to manipulate a higher level representation directly without an underlying binary representation. Examples include Genetic Programming, [25], and Genetic Logic Programming, [34, 35].

There are arguments both for and against the use of an intermediate representation. As the alphabet size increases, the implicit parallelism (and thus search efficiency) reduces. However, problem domains rarely map naturally onto a binary coding. Furthermore, the choice of encoding is particularly important to ensure good building blocks can be formed. On the other hand, if no intermediate representation is used, then it is easier to handle variable length representations, and the resulting solutions are far more intelligible. However, the price of this is reduced implicit parallelism. The choice of whether to use an intermediate representation is not clear cut.

Operators. The majority of symbolic evolutionary approaches employ operators derived from the genetic algorithm. However, by modifying these operators it is possible to narrow down the search space. For instance, the crossover point may be restricted so that some invalid representations are eliminated.

Other operators are based on the semantics of first-order logic, e.g. generalisation and specialisation operations, that constrain this space in a controlled manner. Examples include [24, 32, 17, 14]. Furthermore, background knowledge supplied by a user can further restrict the search space. For example, operators that perform inductive inference operators to find solutions that are logically consistent and complete with this knowledge, [37, 28].

It is interesting to note that many evolutionary learning systems eliminate the intermediate neural network representation altogether and manipulate symbolic world models directly. The use of first-order logic as the representation language offers several advantages including comprehensible theories, logical inference, and permits a priori (background) knowledge to be exploited.

Several axes have been identified for classifying evolutionary approaches to learning symbolic representations. In the next section language and its role in evolutionary learning is considered.

6 Language

Recall from Section 4 that the genotype does not undergo changes within the lifetime of an organism, and that a secondary representation, embodied in the nervous system, permits more rapid adaptation. Consequently, the information accumulated *during* an organism's lifetime is not genetically propagated to subsequent generations. Instead, this learned behaviour is lost. If, however, an organism were capable of passing on this learned behaviour, and were capable of acquiring experience from another organism, it may well have a selective advantage over other organisms that do not have this capacity. Indeed, mechanisms *have* evolved that allow information acquired during the lifetime of an organism to be propagated.

One such mechanism is the Baldwin effect. In 1896, Baldwin [3] proposed that the ability of an individual to learn could guide the evolutionary process. Furthermore, abilities that initially require learning are eventually replaced by the evolution of genetically determined mechanisms that do not require learning. Consequently, there is a gradual transferral of learned behaviours into the genotype. However, this process is slow, requiring of the order of generations for the learned behaviour to become genetically determined.

Another mechanism that avoids the loss of information acquired during the lifetime of an organism is language. A language provides a common framework for expressing the combination of symbols to describe complex notions such as an organism's environment. It permits the communication of information in a highly structured way and thus allows the exchange of information about symbolic world models. Language therefore provides a mechanism to avoid losing learned information at the end of an organism's lifetime. This description is necessarily oversimplistic; however, our aim is to highlight its role as a mechanism for communicating symbolic world models.

It has been formally shown that communication in multi-agent search can significantly improve search performance over a non-cooperative search, [22, 19]. As learning can be formulated as a search, [26], agent communication may be perceived as a mechanism for accelerating learning. The genetic algorithm has been analysed as a cooperative search process, [4]. The population is viewed as a set of cooperating agents, and a generation is one of many repeated encounters among agents. During crossover, agents communicate parts of their solution (or schemata) to other agents. Test cases were presented that suggest that the genetic algorithm may at times yield a behaviour similar to a cooperative search, [4]. This suggests that crossover can accelerate search because of its role as a mechanism for communication between agents.

There exist learning systems that model agent communication in the form of recombination, as described in Section 3. Yet the literature reports of no evolutionary learning system that employs language or structured communication. A major conclusion therefore drawn is the prediction that introducing structure communication in an evolving population of agents will yield accelerated learning.

7 Conclusion

Natural evolution may be described in terms of the processes of reproduction, mutation, competition, and selection, where mutation refers to the class of mechanisms that introduce variation. The variation mechanism need not be an arbitrary operation, but may perform a complex transformation. Furthermore, there is evidence to suggest that the variation mechanisms themselves are also subject to evolutionary pressure. In natural learning systems, this trend is referred to as the “tinkerer’s evolving toolbox”.

In this paper, it is proposed that evolutionary learning algorithms may also be characterised by the four processes listed above and therefore share this same framework.

Algorithms are classified according to the complexity of their variation mechanisms. The types of variation, summarised in Table 1, range from chance mutation to complex semantic transformations. It is shown that strong parallels exist between the variation mechanisms of natural evolution and those implemented in evolutionary learning algorithms.

The comparison also reveals that undue significance may be attributed to the mutation and crossover mechanisms as more powerful transformations are possible. Evolutionary learning models that employ a complex representation, such as first order logic, still fit within this framework — and also permit powerful transformations that observe semantic properties.

Finally, it is argued that agent communication serves to accelerate learning. While crossover may be viewed as a form of agent communication, it is unstructured as the information exchanged is chosen arbitrarily. Structured communication between agents therefore represents a promising area worthy of attention.

Acknowledgements

The author would like to thank John Hunt and the anonymous referees for their valuable comments. This work is supported by the University of Wales, Aberystwyth.

References

1. R. L. Atkinson, R.G. Atkinson, E.E. Smith, and D.J. Bem. *Introduction to Psychology*. Harcourt Brace College Publishers, 1993. 11th edition.
2. Karthik Balakrishnan and Vasant Honovar. Evolutionary design of neural architectures – a preliminary guide to the literature. Technical Report CS TR#95-01, Artificial Intelligence Group, Iowa State University, January 1995.
3. J. M. Baldwin. A new factor in evolution. *American Naturalist*, 30:441–451, 1896.
4. Helen G. Cobb. Is the genetic algorithm a cooperative learner? In *Proceedings of the Workshop on the Foundations of Genetic Algorithms and Classifier Systems*, pages 277–296. Morgan Kaufmann, July 1992.

5. W.H.E. Davies and P. Edwards. The communication of inductive inferences. In *Lecture Notes in Artificial Intelligence (1221): Distributed Artificial Intelligence Meets Machine Learning: Learning in Multi-Agent Environments*, pages 223–241. Springer Verlag, Berlin, 1997.
6. Kenneth De Jong. On using genetic algorithms to search program spaces. In John J. Grefenstette, editor, *Genetic Algorithms and their Applications: Proceedings of the second international conference on Genetic Algorithms*, pages 210–216, George Mason University, July 1987. Lawrence Erlbaum Associates.
7. Kenneth A. DeJong, William M. Spears, and Diana F. Gordon. Using genetic algorithms for concept learning. *Machine Learning*, 13:161–188, 1993.
8. Tim Finin, Rich Fritzon, Don McKay, and Robin McEntire. KQML – A language and protocol for knowledge and information exchange. In *Proceedings of the 13th International Workshop on Distributed Artificial Intelligence*, pages 126–136, Seattle, WA, July 1994.
9. David B. Fogel. *Evolutionary Computation: Towards a New Philosophy of Machine Intelligence*. IEEE Press, New York, 1995.
10. D.B. Fogel. Applying evolutionary programming to selected travelling salesman problems. *Cybernetics and Systems*, 63:111–114, 1993.
11. D.B. Fogel. Evolutionary programming: an introduction and some current directions. *Statistics and Computing*, 4:113–129, 1994.
12. Lawrence J. Fogel, Alvin J. Owens, and Michael J. Walsh. *Artificial Intelligence Through Simulated Evolution*. John Wiley and Sons, Inc., New York, 1966.
13. R.M. Friedberg. A learning machine: Part I. *IBM Journal of Research*, 2:2–13, 1958.
14. A. Giordana and F. Neri. Search-intensive concept induction. *Evolutionary Computation*, 3(4):375–416, 1995.
15. D. E. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley, Reading, MA, 1989.
16. David E. Goldberg. Genetic and evolutionary algorithms come of age. *Communications of the ACM*, Vol. 37:113–119, March 1994.
17. David Perry Greene and Stephen F. Smith. Competition-based induction of decision models from examples. *Machine Learning*, 13:229–257, 1993.
18. A. Hoffman. *Arguments on Evolution: A Paleontologist's Perspective*. Allen and Unwin, London, 1989.
19. Tad Hogg and Bernardo A. Huberman. Better than the rest: The power of cooperation. In L. Nadel and D. Stein, editors, *SFI 1992 Lectures in Complex Systems*, pages 163–184. Addison-Wesley, 1993.
20. J. H. Holland. Escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems. In T. Mitchell, R. Michalski, and J. Carbonell, editors, *Machine Learning, Volume 2*, chapter 20, pages 593–623. Morgan Kaufmann, San Mateo, CA, 1986.
21. John H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, 1975.
22. Bernardo A. Huberman. The performance of cooperative processes. *Physica D*, 42:38–47, 1990.
23. J. S. Huxley. The evolutionary process. In J. Huxley, A.C. Hardy, and E.B. Ford, editors, *Evolution as a Process*, pages 9–33. Collier Books, New York, 1963.
24. Cezary Z. Janikow. A knowledge-intensive genetic algorithm for supervised learning. *Machine Learning*, 13:189–228, 1993.

25. John R. Koza. *Genetic Programming: On the Programming of Computers by Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.
26. T.M. Mitchell. Generalization as search. *Artificial Intelligence*, 18(2), March 1982.
27. E. Richard Moxon and David S. Thaler. The tinkerer's evolving toolbox. *Nature*, 387:659–662, 12 June 1997.
28. Philip Reiser. *EVILL1*: a learning system to evolve logical theories. In *Proc. Workshop on Logic Programming and Multi-Agent Systems (International Conference on Logic Programming)*, pages 28–34, July 1997.
29. Stephen F. Smith. *A Learning System Based on Genetic Adaptive Algorithms*. PhD thesis, University of Pittsburgh, 1980.
30. Paul D. Sniegowski, Philip J. Gerrish, and Richard E. Lenski. Evolution of high mutation rates in experimental populations of *E. coli*. *Nature*, 387:703–705, 12 June 1997.
31. F. Taddei, M. Radman, J. Maynard-Smith, B. Toupance, P.H. Gouyon, and B. Godelle. Role of mutator alleles in adaptive mutation. *Nature*, 387:700–702, 12 June 1997.
32. Gilles Venturini. SIA: a supervised inductive algorithm with genetic search for learning attributes based concepts. In *Proceedings of the European Conference on Machine Learning*, pages 280–296. Springer Verlag, 1993.
33. David A. Watt. *Programming Language Concepts*. Prentice Hall International, Hertfordshire, UK, 1990.
34. Man Leung Wong and Kwong Sak Leung. Inductive logic programming using genetic algorithms. In J.W. Brahan and G.E. Lasker, editors, *Advances in Artificial Intelligence – Theory and Application II*, pages 119–124, 1994.
35. Man Leung Wong and Kwong Sak Leung. The genetic logic programming system. *IEEE Expert Magazine: Intelligent Systems and their Applications*, 10(2):68–76, October 1995.
36. D.E. Wooldridge. *The Mechanical Man: The Physical Basis of Intelligent Life*. McGraw-Hill, New York, 1968.
37. K. Yamamoto, S. Naito, and M. Itoh. Inductive logic programming based on genetic algorithm. *Algorithms, Concurrency and Knowledge*, pages 254–268, 1995.