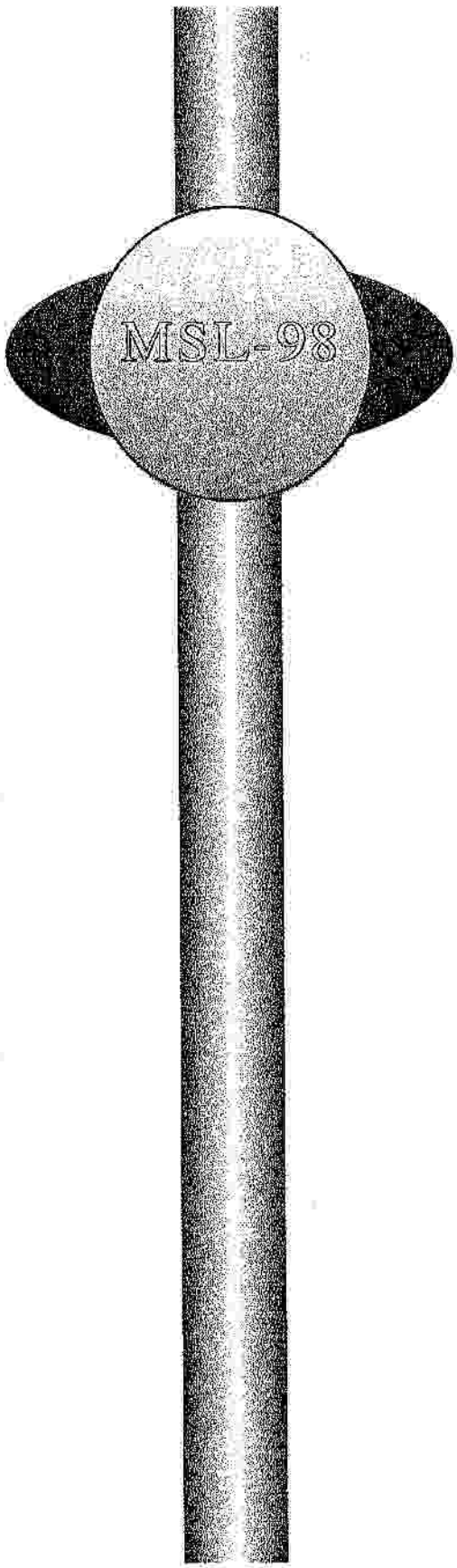

Proceedings of the Fourth International Workshop on

MULTISTRATEGY LEARNING

(MSL-98)

June 11-13, 1998
Desenzano del Garda, Italy

Edited by
Floriana Esposito
Ryszard S. Michalski
Lorenza Saitta

Organized by
Dipartimento di Informatica
Università di Torino
Torino, Italy

Dipartimento di Informatica
Università di Bari
Bari, Italy

Machine Learning and Inference Laboratory
George Mason University
Fairfax, VA, USA

Sponsored by
Italian Association for Artificial Intelligence

LEARNABLE EVOLUTION:
Combining Symbolic and Evolutionary Learning
Initial Results

Ryszard S. Michalski
George Mason University
Fairfax, VA
and
Institute of Computer Science, Polish Academy of Sciences
Warsaw, Poland
michalski @gmu.edu

Abstract

This paper introduces a new methodology for multistrategy learning, called *Learnable Evolution Model* (or LEM), which combines symbolic learning and evolutionary computation. The method employs a form of Lamarckian evolution , in which a symbolic learning process is used to determine "reasons" why certain individuals in a population are superior to others in performing designated tasks. These reasons, expressed in the form of symbolic descriptions, are used, in combination with the standard evolutionary operators, for creating a new generation of individuals in an evolutionary computation process. The method has been compared to two standard genetic algorithms in solving a range of function optimization problems. In the experiments, the proposed method has outperformed two standard genetic algorithms by a wide margin, frequently achieving a speed-up of two or three orders of magnitude.

Introduction

Recent years have witnessed a significant progress in the development of monostrategy symbolic learning methods, and in scaling them up to cope with large datasets (e.g., Clark and Niblett, 1989; Cohen, 1995; Dietterich, 1997; T. Mitchell, 1997; Michalski, 1998). There has also been a significant progress in the area of evolutionary computation (e.g., Baeck, Fogel and Michalewicz, 1997; Koza, 1994). Because these two methodologies have complementary strengths and limitations, a question arises if their integration could not lead to a new powerful learning methodology. This question motivates research whose first results are presented in this paper.

All standard methods of evolutionary computation draw inspiration from principles of Darwinian evolution: the basic operators they employ are mutation, crossover (recombination), and selection of the fittest. These operators are simple and attractive because they can be applied without knowing a model of the problem domain (e.g., Holland, 1975; Michalewicz, 1996; M. Mitchell, 1996). Consequently, such evolutionary methods are very general, and have been applied to a wide range of problems (e.g., complex optimization problems, evolutionary programming, pattern recognition, engineering design, and evolvable hardware). The Darwinian-type evolution is, however, semi-blind: the mutation is a random modification of the current solution; the crossover is a semi-random recombination of two solutions; and the survival of the fittest is a form of parallel hill climbing. In this type of evolution, individuals cannot pass the lessons learned from their experience to the next generation. Consequently, computational processes based on Darwinian evolution are not very efficient. Low efficiency has been the major obstacle in the application of evolutionary computation to very complex problems.

The novel idea presented in this paper is to introduce symbolic learning to evolutionary computation in order to improve the way new generations of individuals are created. The proposed general learning methodology, called *Learnable Evolution Model* (briefly, LEM), is a form of Lamarckian[1] learning. LEM combines two processes—symbolic learning, concerned with determining and exploiting differences between sets of individuals (e.g., current solutions, the alternative descriptions characterizing training examples, etc.), and the evolutionary learning, concerned with an evolutionary improvement of the solutions, based on the results of symbolic learning.

In the presented method, the symbolic learning step employs the AQ15 learning program for generating hypotheses characterizing "best" individuals (solutions). The evolutionary learning step uses generated hypotheses, in combination with standard genetic operators, for creating a new generation of solutions. The process stops when the obtained solution is

[1] After Chevalier de Lamarck, the title of Jean Baptiste Pierre Antoine de Monet, French naturalist (1744-1829), who is the author of the theory that adaptive responses to environment cause structural changes capable of being inherited.

satisfactory, or the allocated computational resources are exhausted. The next section describes Lamarckian learning in more detail.

Learnable Evolution Model

The proposed evolutionary process, called *Learnable Evolution Model* (or LEM), is fundamentally different from the Darwinian type evolution, which underlies standard genetic algorithms. It is a form of a Lamarckian[1] type of evolution, in which the creation of new generations of individuals is guided by lessons learned from an analysis of the previous generation of individuals. Specifically, at selected steps of evolution, a symbolic learning system searches for "reasons" why some individuals in a population are superior to others in performing the given class of tasks. These reasons, expressed in the form of symbolic descriptions (e.g., rulesets), or other forms (e.g., neural nets), are then used to create a new generation of individuals that hopefully represent better solutions.

Basic steps of the LEM algorithm are as follows:

(1) Randomly, or according to certain prior rules reflecting domain knowledge, generate the starting population of solutions (in the case of symbolic learning, solutions would be concept descriptions).

(2) Execute the *genetic algorithm mode* (using standard selection, crossover and mutation operators), as long as the best solution in a sequence of *gen-length* iterations is better by the *gen-threshold* than the best solution found in previous generations.

(3) Execute the *symbolic learning* mode:

- Determine HIGH (high-performance) and LOW (low performance) solutions in the current population, on the basis of the value of the fitness function for a given task or problem.
- Apply a machine learning method for characterizing differences between HIGH and LOW solutions.
- Generate a new population of solutions by replacing not-HIGH individuals by those satisfying the learned description of HIGH solutions; the selection of new solutions among those satisfying the description is random or according to the predefined selection rules.
- Continue the process as long as the best solution in a sequence of *learn-length* iterations is better by the *learn-threshold* than the previously found best solution.

(4) Switch to (2), and repeat the process. Continue switching between (2) and (3) until the *termination condition* is met (the generated solution is satisfactory, or the allocated computational resources are exhausted).

In the above, *gen-length*, *gen-threshold*, *learn-length*, *learn-threshold*, and parameters for determining the HIGH and LOW solutions are determined by analytical considerations, or experimentally, according to the given problem domain.

The LEM methodology can, in principle, employ any of the existing genetic/evolutionary algorithms in step (2), and any learning method in step (3), which is able to generate discriminant descriptions of the solutions (Michalski, 1983).

Initial Experiments

In the experiments presented below, we used the LEM1 system, which is the first, rudimentary implementation of the LEM algorithm. In LEM1, the symbolic learning step employs the AQ-15 rule learning system, which has the ability to generate discriminant descriptions, and uses the VL1 concept representation language (Wnek et al., 1995). These experiments concerned problems of function optimization. The two problems used here have been selected from a problem set developed by De Jong (1975) for testing genetic algorithms. To compare LEM against standard genetic algorithms we used algorithms GA1 and GA2 described in De Jong (1998).

Problem A: Find the maximum of a non-differentiable function (Figure 1):

$$f_3(x_i) = \sum_{1}^{5} \text{integer}(x_i), \qquad -5.12 \le x_i \le 5.12$$

Maximum: 25; Minimum: -30

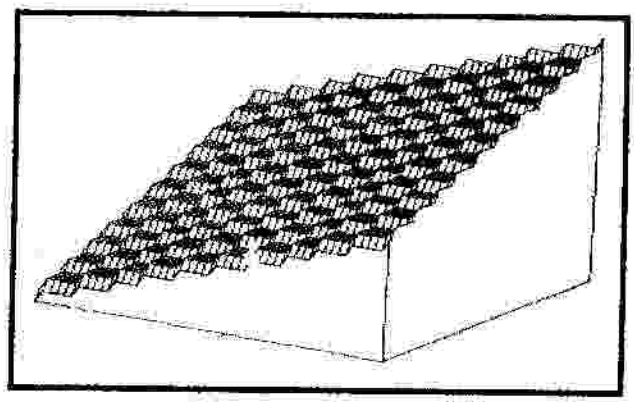

Figure 1. Inverted two-dimensional graph of the function used in Problem A (Reprinted with the permission of K. De Jong).

Results are presented in Figure 2. As shown in the figure, LEM1 has significantly outperformed GA1 and GA2. It found the global maximum after about 60 generations, while GA1's and GA2's solutions were far from the maximum even after 500 generations.

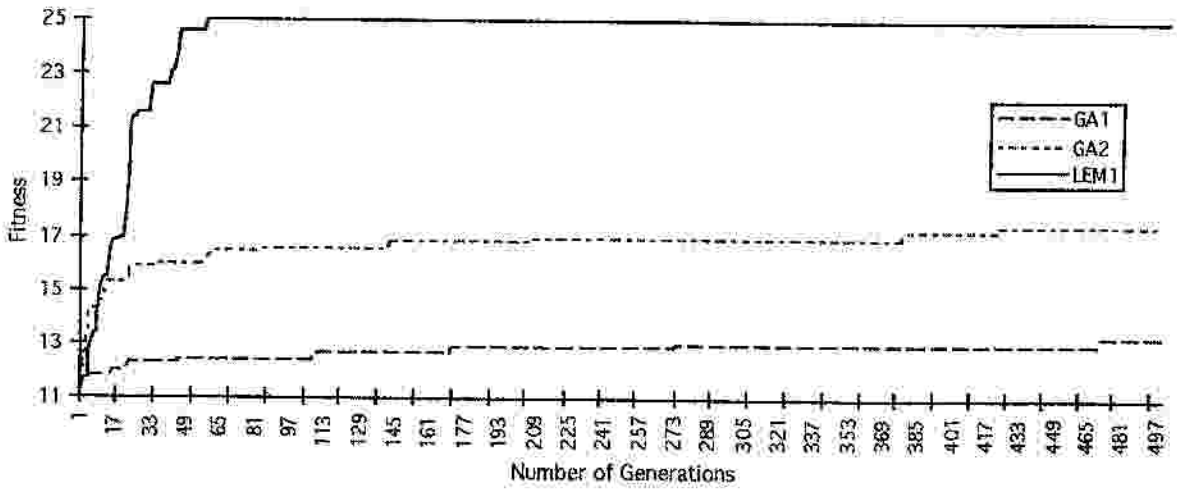

Figure 2. The evolution process within 500 generations (10000 births) for function.

	Relative-Distance-to-Target	Best-So-Far Fitness Value
GA1	47.2%	13.2
GA2	30.4%	17.4
LEM 1	0.0%	25.0 (global max)

Table 1. The Relative-Distance-to-Target after 500 generations (10000 births) for Problem A.

δ	0.0%	1.0%	2.0%	3.0%	4.0%	5.0%	6.0%	7.0%	8.0%
GA1	UNS	UNS	UNS	UNS	UNS	UNS	UNS	UNS	UNS
GA2	UNS	UNS	UNS	UNS	UNS	UNS	UNS	UNS	UNS
LEM 1	58	58	45	45	45	44	44	44	42

"UNS" means unsuccessful when reaching 10,000 generation (200000 births)

Table 2. The δ-close numbers for different δ values and different algorithms.

In order to characterize the relative performance of the tested algorithms, we introduced a measure, called "Relative-Distance-to-Target," defined as the ratio of the difference between the target (here, the function maximum) and the result obtained by an algorithm, to the target value, expressed in percentage, after a given number of generations (here, 500).

Table I presents the Relative-Distance-to-Target for all three algorithms. To evaluate the performance of the

algorithms in another way, we also determined the *δ-close number*, defined as the number of generations in the evolutionary process after which the Relative-Distance-to-Target of the solution produced by an algorithm reaches a given value, δ.

Table 2 presents δ-close numbers for different values of δ and different algorithms. To achieve δ=0.0$, LEM needed 58 generations, while GA1 and GA2 were not δ-close at 10,000th generation even for δ= 8.0%. By dividing the δ-close number for GA1 and GA2 by the δ-close number for LEM1, we estimated the LEM1's

	LEM Speed-up for Different δ								
δ	0.0%	1.0%	2.0%	3.0%	4.0%	5.0%	6.0%	7.0%	8.0%
GA1/LEM1	>>172	>>172	>>222	>>222	>>222	>>227	>>227	>>227	>>238
GA2/LEM1	>>172	>>172	>>222	>>222	>>222	>>227	>>227	>>227	>>238

* GA1 and GA2 solutions have not become δ-close to the maximum within 10,000 generations
">> N" means that if the solution was δ-close at 10 000th generation, the speedup would be N.

Table 3. LEM1's speed-up over GA1 and GA2 for different δ values.

evolution "speed-up" over GA1 and GA2, respectively. A speed-up of 10 means that LEM1 reaches the given δ-close fitness function value using 10 times fewer evolution generations than the algorithm with which it is being compared.

The word "speed-up" may be a little misleading here, because each generation in LEM involves more complex operations than in GAs, as it requires a symbolic learning step. On the other hand, such a step could be made very fast by implementing it in a specialized hardware.

Table 3 presents LEM 1's speed-ups over GA1 and GA2 for different values of δ. One can see, for example, that for δ equal 1%, the LEM1's speed-up over GA1 and GA2 was at least 172. For δ equal 5%, the speed-up was at least 227 (since GA1 and GA2 have not reached maximum at 10,000th generation).

Problem B: Find maximum of a function of a large number of continuous variables (30) and with added Gaussian noise (Figure 3).

$$f_4(x_i) = \sum_{1}^{30} i\, x_i^4 + \text{Gauss}(0,1), \qquad -1.28 \le x_i \le 1.28$$

Maximum: approximately 1248.225. Minimum: 0.

This function was chosen in our tests because finding its optimum is not easy by standard methods. The optimization results (finding the maximum) using GA1, GA1 and LEM1 are presented in Figure 4.

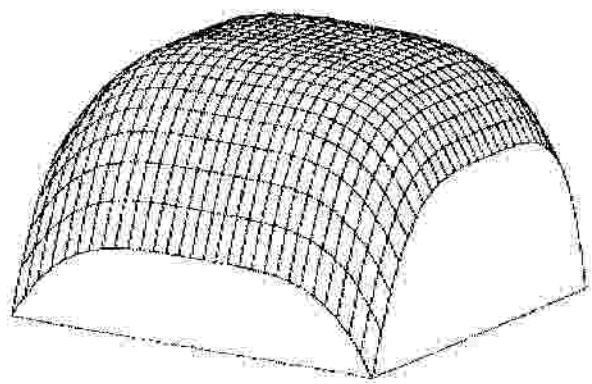

Figure 3. Inverted two-dimensional graph of the function used in Problem B. (Reprinted with permission of K. De Jong).

As one can see, in this case LEM1 dramatically outperformed GA1 and GA2. It found a near-maximum solution already after 150th generation, while GA1 and GA2 were still over 44% away from the maximum after 500th generation. Table 4 presents the number of generations after which the fitness value becomes d-close to the maximum. Table 5 shows the speed-up of LEM1 over GA1 and GA2 for different values of **d**.

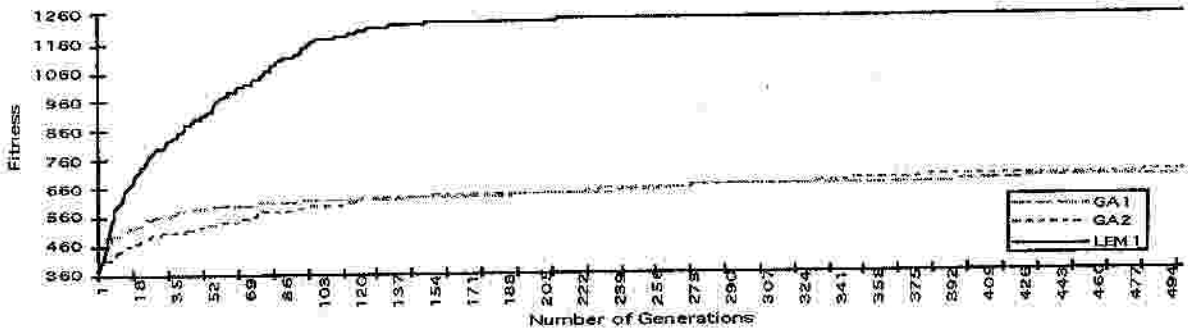

Figure 4. The evolution process within 500 generations (10000 births) for Problem B.

δ	1.0%	2.0%	3.0%	4.0%	5.0%	6.0%	7.0%	8.0
GA1	UNS	UNS	UNS	UNS	UNS	UNS	UNS	UNS
GA2	UNS	UNS	UNS	UNS	UNS	UNS	UNS	UNS
LEM 1	491	151	132	120	116	110	98	97

UNS means unsuccessful when reaching the 10000th generation (200000 births).

Table 4. The number of generations after which the fitness value becomes δ-close to the maximum.

	LEM Speed-up Ratio for Different δ							
δ	1.0%	2.0%	3.0%	4.0%	5.0%	6.0%	7.0%	8.0%
GA1/LEM	>>20.37	>>66.23	>>75.76	>>83.33	>>86.21	>>90.91	>>102.0	>>103.1
GA2/LEM	>>20.37	>>66.23	>>75.76	>>83.33	>>86.21	>>90.91	>>102.0	>>103.1

* GA1 and GA2 solutions have not become δ-close to the maximum within 10 000 generations;
">> N" means that if they were δ-close at 10 000th generation, the speedup would be N.

Table 5. LEM1's speed-up over GA1 and GA2 for different values of δ.

Discussion of the Experiments

The results presented here concern only two problems from among five that constitute the testing set described in (De Jong, 1975). Actually, we applied the LEM1 program to all five problems in the De Jong's set, and experimented not only with finding the function maximum, but also with finding the function minimum. In all these experiments LEM1 outperformed GA1 and GA2, sometimes by a very wide margin (Michalski and Zhang, 1998).

In the case of problem A (concerned with maximizing a non-differentiable function; function f_5 in De Jong's set), and problem B (concerned with maximizing a function of 30 variables with noise; the function f_4 in De Jong's set), LEM 1 found the solution in the number of generations at least two orders of magnitude smaller than the number of generations needed by GA1 and GA2. The GA1 and GA2 algorithms could not find the solution even after 10,000 generations, while LEM1 found it in about 50 generations for problem A, and in about 200 generations for problem B.

Relation to Other Work

The proposed methodology for Lamarckian learning is an original development, and, to the author's knowledge, has not been proposed elsewhere. The work by Grefenstette (1991), which also uses the term "Lamarckian learning," describes a very different approach. It presents a genetic learning system SAMUEL, designed for sequential decision problems. SAMUEL's "Lamarckian feature" is a localized operation which makes modifications of an individual in a population. The learning methodology, proposed here, uses a symbolic learning system to determine general patterns characterizing a *set* of well-performing individuals, and then employs these patterns to improve the new generation. Thus, the proposed Lamarckian learning is a global process utilizing lessons from the "experience" of a group of individuals.

Another work related to combining genetic algorithms and symbolic learning was done by Vafaie and De Jong (1991), who used a standard genetic algorithm for

improving rules produced by an inductive learning system (AQ).

Conclusions

The preliminary experiments with the LEM1 system have demonstrated a significant promise of the proposed LEM methodology as a basis for developing a new type of evolutionary learning systems.

There are many unanswered questions and desirable research directions regarding the proposed Lamarckian learning methodology. These include a systematic theoretical and practical investigation of this methodology, and a determination of the type of tasks for which it will likely be successful. The initial system, LEM1, could be improved by employing a more advanced symbolic learning method (e.g., AQ18 rather than AQ15; Michalski, 1998; Bloedorn et al. 1998; Bloedorn and Michalski, 1998), and/or applying a more advanced method for generating populations individuals using the learned rules.

One of the major characteristics of the proposed LEM methodology is that it requires a symbolic learning system able to learn discriminant descriptions of groups of individuals in a population. If individuals are represented by attribute value vectors or Horn clauses, then standard attributional learning or inductive logic programming methods can be applied. If descriptions are more complex, e.g., are hierarchies of operators representing a computer program, as in evolutionary programming (Koza, 1994), then one needs to develop a learning program able to work with such representations. The AQ methodology can cope with such problems in principle, but a new representation of the input data, generated descriptions, and appropriate generalization operators would have to be developed.

Summarizing, the proposed methodology represents a novel and little understood way of integrating symbolic learning with evolutionary computation. The preliminary results provide a strong justification for conducting further research in this direction.

Acknowledgments

The author expresses his gratitude to Qi Zhang for implementing the evolutionary computation part of the LEM1 system and performing experiments described in the paper. The initial impetus and inspiration for this work is due to research on evolvable hardware conducted by Hugo De Garis (e.g., 1996). Thanks go also to Ken Kaufman for his comments and proofreading of this paper.

This research was conducted in the Machine Learning and Inference Laboratory at George Mason University, and was supported in part by the National Science Foundation under grants IRI-9510644 and DMI-9496192, in part by the Office of Naval Research under grant N00014-91-J-1351, and in part by the Advanced Research Projects Agency under grant No. N00014-91-J-1854 administered by the Office of Naval Research.

References

Baeck, T., Fogel, D.B., and Michalewicz, Z., (Eds.), *Handbook of Evolutionary Computation*, Oxford University Press, 1997.

Bloedorn, E., Kaufman, K., Michalski, R.S., and Zhang, Q., "An Implementation and User's Guide of the AQ18 Learning and Data Mining Environment," *Reports of the Machine Learning and Inference Laboratory*, George Mason University, 1998 (to appear).

Bloedorn, E. and Michalski, R.S., "Data-Driven Constructive Induction: A Methodology and Its Applications", Special issue on Feature Transformation and Subset Selection, *IEEE Intelligent Systems* Huan Liu and Hiroshi Motoda (Eds.), March-April 1998.

Clark, P. and Niblett, R., "The CN2 Induction Algorithm," *Machine Learning*, No.3, 1989.

Cohen, W.W., Fast Effective Rule Induction, *Proceedings of the Twelfth International Conference on Machine Learning*, 1995.

de Garis, H., "Evolvable Hardware Workshop Report," *Technical Report, Reports of the ATR Human Information Processing Research Laboratories*, Evolutionary Systems Department, Kyoto, Japan, 1996.

De Jong, K.A., "An Analysis of the Behavior of a Class of Genetic Adaptive Systems", Ph.D. thesis, Department of Computer and Communication Sciences, University of Michigan, An Arbor, 1975.

De Jong, K.A., *Evolutionary Computation: Theory and Practice*, MIT Press, 1998 (to appear).

Dietterich, T.G., Machine-Learning Research: Four Current Directions, *AI Magazine*, Vol. 18, No.4, 1997.

Grefenstette, J. "Lamarckian Learning in Multi-agent Environment," *Proceedings of the Fourth International Conference on Genetic Algorithms*, R. Belew and L. Booker (Eds.), San Mateo, GA: Morgan Kaufmann, pp. 303-310, 1991.

Holland, J., "Adaptation in Natural and Artificial Systems", Ann Arbor: The University of Michigan Press, 1975.

Koza, J.R., *Genetic Programming II: Automatic Discovery of Reusable Programs*, The MIT Press, 1994.

Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs*, Springer Verlag, third edition, 1996.

Michalski, R.S., "A Theory and Methodology of Inductive Learning," *Artificial Intelligence*, Vol 20, No. 2, pp. 111-161, 1983.

Michalski, R.S., "Inferential Theory of Learning: Developing Foundations for Multistrategy Learning," in *Machine Learning: A Multistrategy Approach*, Vol. IV, R.S. Michalski and G. Tecuci (Eds.), Morgan Kaufmann, San Mateo, CA, 1994.

Michalski, R.S., "The AQ18 Symbolic Learning and Data Mining Environment: Theory and Methodology," *Reports of Machine Learning and Inference Laboratory*, George Mason University, 1998 (to appear).

Michalski R.S. and Zhang, Q., "Lamarckian Evolution Model: Integrating Symbolic Learning and Evolutionary Computation," *Reports of the Machine Learning and Inference Laboratory*, George Mason University, 1998 (to appear).

Mitchell, M. *An Introduction to Genetic Algorithms*, Cambridge, MA, MIT Press, 1996.

Mitchell, T. M., Does Machine Learning Really Work, *AI Magazine*, Vol. 18, No.3, 1997.

Vafaie, H. and De Jong, K.A., "Improving the Performance of a Rule Induction System Using Genetic Algorithms," *Proceedings of the First International Workshop on Multistrategy Learning*, MSL-91, Harpers Ferry, WV, November 7-9, 1991.

Wnek, J., Kaufman, K., Bloedorn, E. and Michalski, R.S., "Inductive Learning System AQ15c: The Method and User's Guide," *Reports of Machine Learning and Inference Laboratory*, MLI 95-4, George Mason University, Fairfax, VA, March 1995.