Speeding Up Evolution through Learning: LEM

Ryszard Michalski*, Guido Cervone and Kenneth Kaufman

Machine Learning and Inference Laboratory
George Mason University, Fairfax, Virginia

Phone: (703) 993-1558 or 764-9142 Email: michalski@gmu.edu
Fax: (703) 993-3729 or (703) 764-9142

Also, Institute of Computer Science
Polish Academy of Sciences, Warsaw, Poland

Abstract

This paper reports briefly on the development of a new approach to evolutionary computation, called the *Learnable Evolution Model* or *LEM*. In contrast to conventional Darwinian-type evolutionary algorithms that employ mutation and/or recombination, LEM employs machine learning to generate new populations. At each step of evolution, LEM determines hypotheses explaining why certain individuals in the population are superior to others in performing the designated class of tasks. These hypotheses are then instantiated to create a next generation. In the testing studies described here, we compared a program implementing LEM with selected evolutionary computation algorithms on a range optimization problems and a filter design problem. In these studies, LEM significantly outperformed the evolutionary computation algorithms, sometimes speeding up the evolution by two or more orders of magnitude in the number of evolutionary steps (births). LEM was also applied to a real-world problem of designing optimized heat exchangers. The resulting designs matched or outperformed the best human designs.

1 Introduction

Recent years have witnessed significant progress in the development of machine learning methods and in scaling them up to cope with large datasets (e.g., Cohen, 1995; Dietterich, 1997; Mitchell T., 1997; Michalski, 2000b). There has also been significant progress in the area of evolutionary computation (e.g., Baeck, Fogel and Michalewicz, 1997; Koza, 1994; Banzhaf, 1999; Michalewicz et al., 1999). As symbolic learning and evolutionary computation have complementary capabilities and strengths, a question arises as to whether they can be integrated in a way that will lead to a new, more powerful model of evolutionary computation. This paper presents some of first results from the efforts toward such a goal.

Specifically, we briefly describe the *Learnable Evolution Model*, LEM, which employs machine learning to guide evolutionary computation and report a few results from its testing on selected problems.

2 What is Learnable Evolution Model (LEM)?

The central engine of evolution in LEM is *Machine Learning* mode which generates hypotheses about differences between high fitness and low fitness individuals, and then uses these hypotheses to generate new individuals. New individuals are thus generated not by a semi-blind process of mutation and/or recombination, as in conventional evolutionary algorithms, but rather by a deliberate process of inference. LEM can be viewed as a form of genetic engineering. Below is a simplified form of LEM:

1. *Generate a population.*
2. *Execute Machine Learning mode:*

   2a Derive a training set: Select from a population (either the current one or a union of the current and selected past populations) a *high performance group*, or briefly *H-group*, and *Low performance group*, or briefly *L-group*, according to the fitness function.

   2b Create a hypothesis: Apply a machine learning method to create a description of the H-group that differentiates it from the L-group.

   2c Generate a new population: Instantiate the hypothesis in different ways to generate new individuals, and combine them with those in the H-group. Create a new population from the resulting set by some form of selection operation.

   2d Go to step 2a, and continue repeating Machine Learning mode until the *Machine Learning mode termination condition* is met. When this termination condition is met, take one of the following actions:

   A. If the *LEM termination condition* is met, go to step 5.
   B. Repeat the process from step 1. This is called a *start-over* operation.
   C. Go to step 3.

3. *Execute Darwinian Evolution mode:*

   Apply some form of mutation, crossover (optionally) and selection operators to generate a new population. Continue this mode until the *Darwinian Evolution mode termination condition* is met.

4. *Alternate:*

   Go to step 2, and then continue alternating between step 2 and step 3 until the *LEM termination condition* is met (e.g., the generated solution is satisfactory, or the allocated computational resources are exhausted), in which case the control goes to step 5.

5. *End:*

   The best individual or individuals obtained are the result of evolution.

If LEM executes repeatedly only step 2, or steps 1 and 2, then it is called uniLEM; otherwise, it is called duoLEM. The full description of the LEM algorithm includes some additional features and a few control parameters (Michalski, 2000). The LEM methodology can employ, in

principle, in step 2 any machine learning method that can generate discriminant descriptions (Michalski, 1983), and in step 3 any existing evolutionary algorithm.

3 LEM as Progressive Partitioning of the Search Space

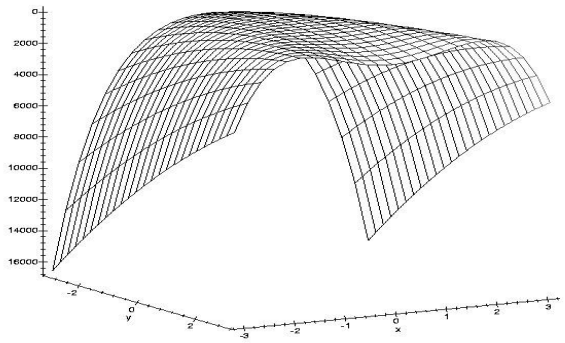
The search conducted in Machine Learning mode can be interpreted as a progressive partitioning of the search space. An H-group description hypothesizes a region or regions that likely contain the optimal individual. Each subsequent H-group description hypothesizes a new, typically more specialized, partition of the search space. Due to this effect, the LEM evolution process may converge to the optimum (local or global) much more rapidly than Darwinian-type evolutionary algorithms. Since partitioning is guided by inductive inference, this process may miss the area with the global optimum. In such cases, LEM executes a start-over operation or temporarily switches to the Darwinian Evolution mode.

The next sections briefly describe a selection of results from preliminary studies (Michalski, 1998; Michalski and Zhang, 1999; Cervone and Michalski, 2000; Kaufman and Michalski, 2000).

4 Pilot Study: Optimization Problems

This study applied rudimentary implementations of LEM (LEM-1 and LEM-2) and several evolutionary computation algorithms to problems of optimizing five functions f_1, f_2, f_3, f_4, and f_5, described in (De Jong, 1975), which have been used by researchers for testing evolutionary algorithms. For the sake of space, we present here only a small selection of results. Other results followed the same pattern as those presented here (Michalski, 1999; Cervone and Michalski, 2000).

Problem 1: Find the minimum of function f_2 of 5 variables bound between –10.1 and 10.1 (an inverted 2D graph of f_2 is presented in Figure 1).



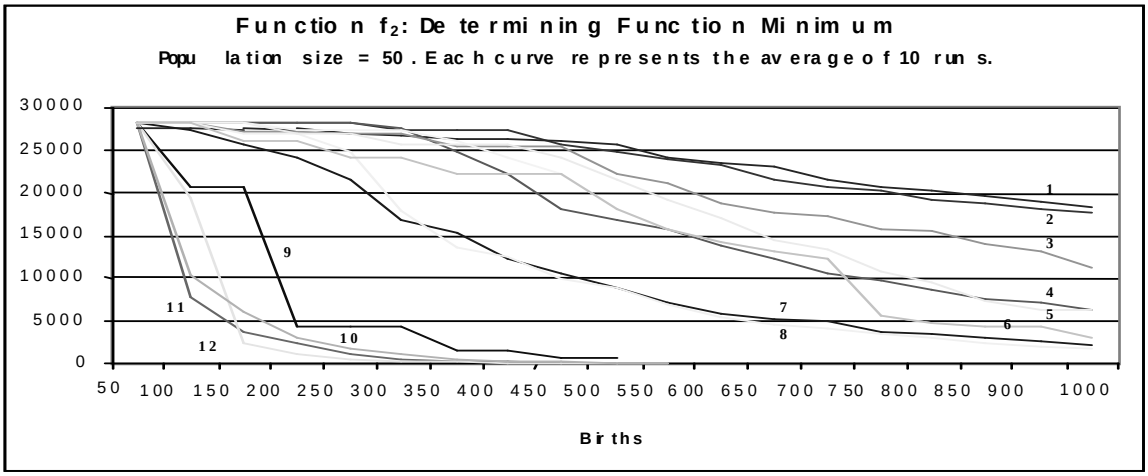
$$f_2(\mathbf{x}) = \sum_{i=0}^{5}(100 \cdot (x_{i+1} - x_i)^2 + (x_i - 1)^2)$$

This function represents a complex minimization problem because it has a very narrow ridge and variables are interdependent. The tip of the ridge is very sharp, and runs along a parabola. Algorithms that are not able to discover good directions underperform in this problem. Although the function looks symmetric, it is not.

Figure 1. Inverted 2D graph of function f_2.

In this experiment, LEM-2 and a conventional evolutionary computation algorithm, EV, were run with different values of parameters to test the sensitivity of the methods to the parameter values. EV employs a uniform selection for the parent population, and binary tournament for the selection of the survivors. Each parent is cloned and then mutated using uniform mutation. All

new individuals compete with a random selected parent. Results from experiments are presented in Figure 2.



1 – EV .1 with crossover	4 - EV .3 without crossover	7 – EV .5 without crossover	10 – LEM .1 .3 (uniLEM)
2 - EV .1 without crossover	5 - EV .7 with crossover	8 – EV .7 without crossover	11 – LEM .3 .3 (uniLEM)
3 - EV .5 with crossover	6 – EV .9 with crossover	9 – LEM .1 .1 (uniLEM)	12 – LEM .3 .1 (uniLEM)

Figure 2. Results from applying LEM-2 and EV to the problem of minimizing f_2.

Each curve in Figure 2 represents the average of 10 runs, each starting with a different initial population (which was the same for LEM-2 and EV). Curves 1, 3, 5 and 6 represent runs with a uniform crossover; the remaining runs of EV did not use crossover. All runs used elitism. The initial population was 100 and kept constant in all the experiments. In Figure 3, the value after EV represents the mutation rate, and a pair of values after LEM represents the high (HT) and low (LT) thresholds, respectively.

In this experiment LEM-2 was relatively insensitive to its basic parameters, HT and LH (a similar behavior was observed in other experiments too, including cases when the number of variables was increased to 100). LEM-2 found the global minimum in almost all cases. EV was quite sensitive to its parameters and was unable to find the global minimum within the limit of 1000 births for any of the parameter values that were tried.

Problem 2. Find the minimum of function f_3 of 100 variables (an inverted 2D graph of f_3 is presented in Figure 3).

$$f_3(x_1, x_2, x_3, ..., x_{100}) = \sum_{i=1}^{100} \text{integer}(x_i) \quad -5.12 \le x_i \le 5.12$$

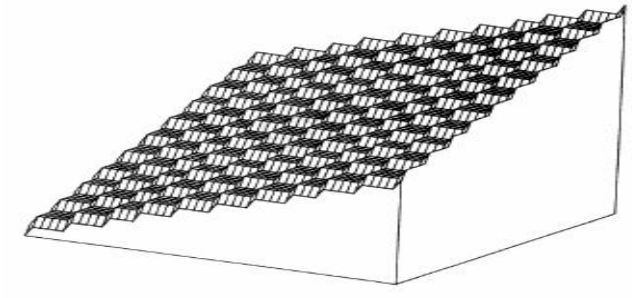
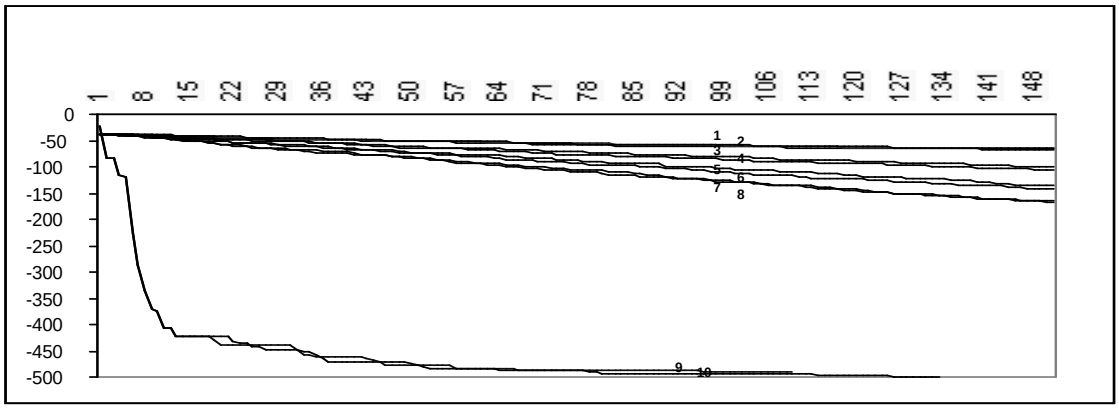


Figure 3. Inverted two-dimensional graph of f_3.
(Reprinted with permission of Kenneth De Jong)

In this experiment, the number of variables in function f_3 was increased from the original 5 to 100 in order to test LEM's scalability. Results from applying LEM-2 and three evolutionary computation methods EV, EP and ES with different mutation rates are presented in Figure 4.



1 - EP 100, 10, 0.1; 2 - EV 100, 10, 0.1; 3 - EP 100, 10, 0.5; 4 - ES 100, 0.5;
5 - EP 100, 10, 0.9; 6 - ES 100, 0.9; 7 - EV 100, 10, 0.5; 8 - EV 100, 10, 0.9;
9 – uniLEM2 (AQ18 ; HFT & LFT 30%) 10 - duoLEM2 (AQ18 + EV 100,10, 0.5)

Figure 4. An evolutionary process for finding the minimum of f_3 with 100 variables using EP, EV and ES evolutionary computation methods and LEM-2 in uniLEM and duoLEM versions.

The horizontal axis in Figure 4 represents the number of births in thousands, and the vertical axis represents the function value. In the caption of Figure 4, EP p, c, m; EV p, c, m; and ES p, m stand, respectively for "Evolutionary Program," "Evolutionary Algorithm," and "Evolutionary Strategy," where p denotes the number of parents, c denotes the number of children. and m is the mutation rate. LEM-2 was run in uniLEM mode (i.e., using only Machine Learning mode), and in duoLEM mode (i.e., using both Machine Learning and Darwinian Evolution). As seen in Figure 4, LEM-2 dramatically outperformed the tested evolutionary algorithms. Both variants of LEM (uniLEM and duoLEM) performed similarly (graphs 9 and 10 in Figure 4).

The results presented above represent only a sample of experiments. In all pilot experiments, the tested implementations of LEM consistently outperformed selected evolutionary computation algorithms in terms of the number of evolutionary steps, sometimes by a wide margin. These experiments were repeated many times and results consistently indicated a similar pattern of performance: LEM required a significantly smaller number of evolutionary steps to reach the solution and exhibited a relatively low sensitivity to the initial population and its parameters (for details, see Michalski, 2000a; Michalski and Zhang 1999; Cervone and Michalski, 2000; Cervone, Coletti, and Latiner, 2000).

5 An Exploratory Application to Heat Exchanger Design

To test LEM on a complex engineering design problem, we started a collaboration with the National Institute of Standards and Technology (NIST). In this research, we applied LEM to problems of designing highly efficient heat exchangers. The problem is how to design heat exchangers that have maximal capacity for any given set of environmental and technical constraints. Such constraints include the outside air temperature and humidity, the flow of air through the heat exchanger, the number of rows of tubes and the number of tubes per row in the exchanger, the refrigerant used, etc.

Below is a highly abbreviated illustration of one of our initial experiments. A population consists of a set of heat exchanger design structures. Each structure is represented by a vector of numbers characterizing the connections between tubes. The vector is associated with capacity of that design, determined by a simulator.

Exchanger Size: 16 x 3
Population Size: 15 Generations: 40
Operator Persistence: 5(# of unsuccessful trials of a structure modifying operator)
Mode Persistence: Dar-probe=2 and Learn=probe=1.

Initial population:
Structure #0.3: 17 1 2 3 4 5 6 7 8 9 12 13 29 15 31 I 18 33 20 36 22 38 24 40 26 42
11 2 7 45 14 47 16 34 35 19 37 21 39 23 41 25 43 44 28 46 30 48 32: 5.5376
Structure #0.8: 17 1 20 3 4 22 6 24 8 26 10 28 27 15 16 32 33 2 18 19 5 38 7 40 9 42
11 44 13 46 30 48 34 35 36 I 21 37 23 39 25 41 27 43 29 45 31 47: Capacity =
5.2099
and 13 others

Selected Members: 3, 2, 3, 7, 9, 3, 9, 3, 6, 9, 9, 6, 8, 1, 7
Operations: NS(23, 39), SWAP(8), SWAP(28), SWAP(19), SWAP(1), SWAP(27),
SWAP(40), SWAP(43), SWAP(15), SWAP(25), SWAP(7), SWAP(36),
SWAP(29), SWAP(25), SWAP(1)

Below is one of the structures created by the application of a SM operator in Machine Learning algorithm mode (by swapping the two tubes following tube 29 in Structure #0.8)

Generation 1:

Structure #1.13: 17 1 20 3 4 22 6 24 8 26 10 28 27 15 16 32 33 2 18 19 5 38 7 40 9 42
11 4 13 45 30 48 34 35 36 I 21 37 23 39 25 41 27 43 46 29 31 47:
Capacity=5.2093
(15 structures in a population)

Selected Members: 6, 15, 11, 3, 13, 1, 10, 6, 12, 10, 5, 4, 13, 1, 3
. . . . . .
Generation 5: Machine Learning mode
A learned rule:
Rule:
[x1.x2.x3.x4.x5.x6.x7.x8.x9.x11.x12.x13.x14.x15.x17.x18.x19.x20.x21.x22.x23.x24.
x25.
x26.x27.x28.x29.x30.x31.x32.x33.x34.x35.x36.x37.x38.x39.x40.x41.x42.x43.x44.x45.
x46.
x47.x48=regular] & [x10=outlet] & [x16=inlet] (t:7, u:7,
q:1)

An example of a generated structure:
Structure #5.1: 17 1 2 3 4 5 6 7 8 9 12 29 45 30 31 I 18 33 20 36 22 38 24 40 26 42
11 27 13 15 47 48 34 35 19 37 21 39 23 41 25 43 44 28 46 14 32 16:
Capacity=5.5377

Below are examples of structures from the 20th generation:

....
Generation 21: Machine Learning mode

Structure #21.7: 18 1 4 2 6 3 5 7 8 9 12 13 45 15 31 I 33 17 35 36 22 39 24 40 42 25
11 44 30 46 32 47 34 19 20 37 21 23 44 41 26 43 28 27 29 14 48 16: 4.1702
Structure #21.15 2 18 4 1 6 3 5 7 8 9 12 13 45 15 31 I 33 17 35 36 22 39 24 40 42 25
11 44 30 46 32 47 34 19 20 37 21 23 38 41 26 43 28 27 29 14 48 16: 5.5387
and 13 others

Selected Members: 11, 4, 4, 13, 15, 10, 12, 13, 15, 15, 12, 2, 3, 5, 10.

```
.....
Generation 40:

Structure #40.15: 33 17 2 41 4 5 6 9 7 8 12 29 46 45 47 I 1 34 20 36 22 38 24 3 42 43
       44 27 13 15 32 16 18 11 19 37 21 32 23 25 40 26 28 35 30 14 48 31:
       Capacity=6.3686
 . . .
```

As shown above, the initial designs evolved to better designs (as measured by the capacity). In these experiments, LEM explored many different architectures, reaching heat capacities comparable to the best human designs in the case of uniform flow. In the case non-uniform flow, LEM designs have been judged by collaborating experts as superior to best human designs. For details on this work, see (Kaufman and Michalski, 2000a).

6 Relation to Other Research

The proposed LEM methodology is to our knowledge an original development. In searching through the literature, we found several papers describing efforts to apply machine learning to evolutionary computation, but they are substantially different from LEM. Work has also been done on the application of evolutionary computation to machine learning. A brief review of some of this work is below.

Sebag and Schoenauer (1994) applied machine learning to adaptively control the crossover operation in genetic algorithms (implementing their own version of AQ-type learning). In their system, a machine learning method develops rules characterizing relevant crossovers. These rules are then used to bias the selection of crossover operators. Sebag, Schoneauer and Ravise (1997a) used inductive learning to determine mutation step-size in evolutionary parameter optimization. Ravise and Sabag (1996) described a method for using rules to prevent new generations from repeating past errors. In a follow-up work, Sebag, Schoenauer and Ravise (1997) proposed keeping track of past evolution failures by using templates of unfit individuals, called "virtual losers." An evolution operator, called "flee-mutation," aims at creating individuals different from the virtual losers. Grefenstette (1991) developed a genetic learning system, SAMUEL, that implements a form of Lamarckian evolution. The system was designed for sequential decision making in a multi-agent environment. A strategy, in the form of *if-then* control rules, is applied to a given world state and certain actions are performed. This strategy is then modified either directly, based on the interaction with the environment, or indirectly by changing the rules' strength within the strategy. The changes in a strategy are passed to its offspring. This is a Lamarckian-type process that takes into consideration the performance of a single individual when evolving new individuals. Reynolds (1994) proposed cultural algorithms, which are dual inheritance systems that provide a cooperation between a cultural and population-based levels of evolution. The cultural level is represented by a belief space that contains global knowledge in the form of beliefs. The beliefs constrain the way in which individuals are modified by genetic operators. Cultural algorithms relate to LEM (developed independently) in that they utilize top individuals in the population, but do it differently, namely they vote for the beliefs to be accepted into the belief space (e.g., Rychtyckyj and Reynolds, 1999).

As to the application of evolutionary computation to machine learning, most research has concerned the improvement of propositional concept learning. An indirect form of such application is to evolve the "best" subset of attributes from a collection of original attributes in

order to improve concept learning (Forsburg, 1976; Vafaie and De Jong, 1992). Another form concerns an improvement of the learning method itself (e.g., Vafaie and De Jong, 1991; De Jong, Spears, and Gordon, 1993; Giordana and Neri, 1995; Greene and Smith 1993; Janikow, 1993; Hekanaho, 1997). There have also been efforts to use genetic algorithms to evolve a population of biases for a machine learning algorithm. For example, Turney (1995) applied a genetic algorithm to evolve weights assigned to attributes in the C4.5 program in order to derive the minimum cost decision tree. Evolutionary algorithms have also been applied to improve relational learning, e.g. (Augier, Venturini and Kodratoff, 1995; Hekanaho, 1998).

7 Conclusion

Although the results obtained in the pilot studies are highly encouraging, there are many unanswered questions and desirable directions for further research. These include a systematic theoretical and practical investigation of the methodology, determining best methods for implementing its steps, and a determination of the type of tasks for which it will likely be successful. One emerging pattern in performance of LEM in comparison with the evolutionary computation methods is that LEM typically needs far fewer generations (or births) to reach the solution. This makes LEM particularly attractive in problem domains with a high cost of determining the fitness function. Another pattern is that LEM appears to be relatively insensitive to the choice of initial population and to the variation of its basic parameters, HT and LT (within a range).

Acknowledgments

This research was conducted in the Machine Learning and Inference Laboratory at George Mason University. The Laboratory's research related to the work presented in the paper has been supported in part by the National Science Foundation under Grants No. IIS-9904078 and IRI-9510644, in part by the Defense Advanced Research Projects Agency under Grant No. F49620-95-1-0462 administered by the Air Force Office of Scientific Research, and in part by the Office of Naval Research under Grant No. N00014-91-J-1351. This research was also supported in part by International Intelligent Systems, Inc.

References

Augier, S., Venturini, G. and Kodratoff, Y., "Learning First Order Logic Rules with a Genetic Algorithm," *Proceedings of the First*^{t} *International Conference on Knowledge Discovery and Data Mining*, pp. 21-26, Montreal, Canada, AAAI Press, 1995.

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

Banzhaf, W., Daida, J., Eiben, A.E., Garzon, M.H., Honavar, V., Jakiela, M. and Smith, R.E. (eds.), *Proceedings of the 1999 Genetic and Evolutionary Computation Conference* (GECCO), Orlando, Florida, July 13-17, 1999.

Cervone, G., Coletti, M. and Latiner, R., *ECC^{++}: A Generic C++ Library for Evolutionary Computation, Reports of the Machine Learning and Inference Laboratory*, George Mason University, Fairfax, VA, 2000 (in preparation).

Cervone and Michalski, "Design and Experiments with the LEM2 Implementation of the Learnable Evolution Model," *Reports of the Machine Learning and Inference Laboratory,* George Mason University, 2000 (to appear).

Coletti, M., Lash, T., Mandsager C., Michalski, R.S., and Moustafa, R., "Comparing Performance of the Learnable Evolution Model and Genetic Algorithms on Problems in Digital Signal Filter Design." *Proceedings of Genetic and Evolutionary Computation Conference* (GECCO), Orlando, Florida, July 14-17, 1999 (an extended version was published in *Reports of the Machine.Learning and Inference Laboratory*, MLI 99-5, 1999)

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, Ann Arbor, 1975.

De Jong, K. A., Spears, W. M., and Gordon, F. D., "Using Genetic Algorithms for Concept Learning," Machine Learning, 13, pp. 161-188, 1993.

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

Forsburg, S., "AQPLUS: "An Adaptive Random Search Method for Selecting a Best Set of Attributes from a Large Collection of Candidates," *Internal Technical Report*, Department of Computer Science, University of Illinois, Urbana, 1976.

Giordana A. and Neri, F., "Search-intensive Concept Induction," *Evolutionary Computation*, 3(4), pp.375-416, 1995.

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, CA: Morgan Kaufmann, pp. 303-310, 1991.

Greene D. P. and Smith, S.F., "Competition-based Induction of Decision Models from Examples," *Machine Learning*, 13, pp. 229-257, 1993.

Hekanaho, J, "GA-based Rule Enhancement Concept Learning," *Proceedings of the Third^{d} International Conference on Knowledge Discovery and Data Mining*, pp. 183-186, Newport Beach, CA, AAAI Press, 1997

Hekanaho, J., "DOGMA: A GA-based Relational Learner,: *TUCS Technical Reports Series,* Report No. 168, March 1998.

Janikow, C. Z., "A Knowledge-intensive Genetic Algorithm for Supervised Learning," *Machine Learning*, 13, pp. 189-228, 1993.

Kaufman K. and Michalski, R.S., ISHED-1: Applying the LEM Methodology to Heat Exchanger Design, *Reports of the Machine Learning and Inference Laboratory*, George Mason University, 2000 (to appear).

Kaufman, K. and Michalski, R.S., "The AQ18 Environment for Natural Induction, Machine Learning and Data Mining: A User's Guide," *Reports of the Machine Learning and Inference Laboratory*, George Mason University, 2000 (to appear).

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.

Michalewicz, Z., Schoenauer, M.. Yao, X., and Zazala, A. (eds.), *Proceedings of the Congress on Evolutionary Computation*, Washington, DC, July 6-9, 1999.

Michalski, R.S., "A Theory and Methodology of Inductive Learning," *Artificial Intelligence*, 20(2), pp. 111-161, 1983.

Michalski, R.S., "Learnable Evolution: Combining Symbolic and Evolutionary Learning," *Proceedings of the Fourth International Workshop on Multistrategy Learning*, Desenzano del Garda, Italy, pp. 14-20, June, 1998.

Michalski, R.S., "LEARNABLE EVOLUTION MODEL: Evolutionary Processes Guided by Machine Learning," *Machine Learning*, 38(1-2), January/February, 2000.

Michalski, R.S., "NATURAL INDUCTION: Theory, Methodology, and Applications to Machine Learning, Data Mining and Knowledge Discovery," *Reports of the Machine Learning and Inference Laboratory*, George Mason University, 2000 (to appear).

Michalski R.S. and Zhang, Q., "Initial Experiments with the LEM1 Learnable Evolutionary Model: An Application to Function Optimization and Evolvable Hardware," *Reports of the Machine Learning and Inference Laboratory*, George Mason University, MLI 99-4, 1999.

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

Mitchell, T. M., "Does Machine Learning Really Work," *AI Magazine*, 18(3), 1997.

Ravise, C. and Sebag, M., "An Advanced Evolution Should Not Repeat Its Past Errors," *Proceedings of the 13th International Conference on Machine Learning*, L. Saitta (ed.), pp. 400-408, 1996.

Reynolds, R.G., "An Introduction to Cultural Algorithms," *Proceedings of the 3rd Annual Conference on Evolutionary Programming*, Selbak, A.V. Fogel L.J. (eds.), River Edge, NJ World Scientific Publishing, pp 131-139, 1994.

Rychtyckyj, N. and Reynolds, R.G., "Using Cultural Algorithms to Improve Performance in Semantic Networks," *Proceedings of the Congress on Evolutionary Computation*, Washington, DC, pp. 1651-1663, 1999.

Sebag, M. and Schoenauer, M., "Controlling Crossover Through Inductive Learning," in Davidor, Y., Schwefel, H.P. and Manner, R. (eds.), *Proceedings of the Third Conference on Parallel Problem Solving from Nature*, Springer-Verlag, LNVS 866, pp. 209-218, 1994.

Sebag, M., Schoenauer M., and Ravise C., "Inductive Learning of Multation Step-size in Evolutionary Parameter Optimization," *Proceedings of the Eighth Annual Conference on Evolutionary Programming*, LNCS 1213, pp. 247-261, Indianapolis, April 1997.

Sebag, M., Shoenauer, M., and Ravise, C., "Toward Civilized Evolution: Developing Inhibitions," *Proceedings of the Seventh International Conference on Genetic Algorithms*, pp.291-298, 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.