Applying Learnable Evolution Model to Heat Exchanger Design

Kenneth A. Kaufman and Ryszard S. Michalski*
Machine Learning and Inference Laboratory
George Mason University
Fairfax, Virginia 22030-4444
{kaufman, michalski}@gmu.edu

* Also with the Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland

Abstract

A new approach to evolutionary computation, called *Learnable Evolution Model* (LEM), has been applied to the problem of optimizing tube structures of heat exchangers. In contrast to conventional Darwinian-type evolutionary computation algorithms that use various forms of mutation and/or recombination operators, LEM employs machine learning to guide the process of generating new individuals. A system, ISHED1, based on LEM, automatically searches for the highest capacity heat exchangers under given technical and environmental constraints. The results of experiments have been highly promising, often producing solutions exceeding the best human designs.

Introduction

This paper describes an application of a new approach to evolutionary computation, called *Learnable Evolution Model* (LEM), which employs machine learning to guide the process of generating new populations (Michalski 1998; 2000). LEM integrates two modes of operation, a Darwinian Evolution mode, which is based on traditional evolutionary computation methods (e.g., Holland 1975; Michalewicz 1996), and Machine Learning mode, which generates new individuals through a process of theory formation and instantiation. Specifically, Machine Learning mode generates hypotheses that characterize differences between groups of high performing and low performing individuals, and then instantiates these hypotheses to generate new individuals.

LEM has been applied to a range of function optimization problems (Michalski and Zhang 1999; Cervone 1999), and to the design of nonlinear filters (Coletti et al. 1999). In both applications, LEM significantly outperformed the evolutionary computation algorithms used in the experiments, sometimes speeding up the evolution process by two or more orders of magnitude in terms of the number of births (or generations).

This paper describes the application of LEM to the very complex practical problem of optimizing heat exchangers. The implemented program, ISHED1 (Intelligent System for Heat Exchanger Design) searches for the best arrangement of the evaporator tubes in the heat exchanger of an air conditioner. This is a very difficult problem because the search space is poorly structured and extremely large (there are about 10^{80} possible tube arrangements in a medium-sized heat exchanger). In order to avoid the cost of solving this problem for different operating conditions, manufacturers of air conditioning systems currently assume in their models average operating conditions (regarding the temperature, humidity, airflow, etc.). Since real conditions are often different from the assumed averages, such air conditioning systems tend to perform sub-optimally.

ISHED1 does not make such assumptions. It evolves toward structures that are best suited for any given technical constraints and environmental conditions. In the process of evolutionary design, it employs a heat exchanger simulator that serves as an evaluator of the proposed designs under the assumed conditions (Domanski 1989).

Problem Description

The problem of heat exchanger design is to seek a structure of tubes that provides the maximum heat transfer given technical and environmental constraints. These constraints include the size of the exchanger (the number of rows of tubes and the number of tubes per row), the refrigerant used, the outside air temperature and humidity, the flow of air through the heat exchanger, and others.

In an air conditioner, refrigerant flows through a loop. It is superheated and placed in contact with cooler outside air (within the condenser unit), where it transfers heat out and liquefies. Coming back to the evaporator, it comes into contact with the warmer interior air that is being pushed through the heat exchanger, thus cooling the air, and heating and evaporating the refrigerant.

The heat exchanger itself consists of an array of parallel tubes through which the refrigerant flows back and forth. A typical model is shown in Figure 1. In this figure, there are three rows of 16 tubes, with one inlet tube and two outlet tubes. The path from the inlet tube to the outlet tubes splits along the way. In general, there can be multiple inlet tubes, and the paths from each of them to the outlet tubes may or may not split. Individual tubes may be connected to each other in many different ways. The efficiency of the heat exchanger strongly depends on the order in which the tubes are connected.

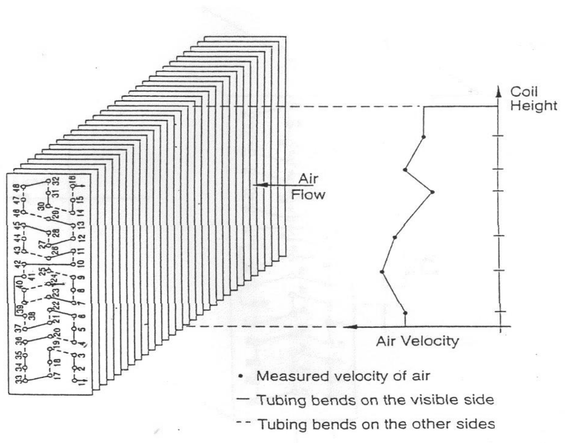
Coil Height
Air Flow
Air Velocity
• Measured velocity of air
— Tubing bends on the visible side
-- Tubing bends on the other sides

Figure 1: An Architecture of Evaporator Circuitry: A sample16 x 3 configuration.

While the refrigerant flows through the tubes, air is forced through the unit, whose velocity/volume profile may be as illustrated in the figure. The air first comes into contact with and is cooled by the refrigerant in the first depth row, then in the subsequent rows.

The amount of cooling the air conditioner provides is the aggregate of the heat transfer provided by each of its tubes. Each tube's transfer is a function of the temperature and volume per unit time of both the air and the refrigerant coming into contact at that tube. Different orderings of the tubes affect the temperature and pressure of the refrigerant passing through each tube. The results of prior air/refrigerant interactions affect their temperatures at later interactions. Additionally, the refrigerant loses pressure (and velocity) while passing through the bends between tubes; it thus helps in maximizing heat transfer if adjoining tubes are physically close to each other.

In short, the goal of ISHED1 is to determine how to order the flow through the tubes such that heat transfer is maximized for the given constraints. Note that the number of depth rows and the number of tubes per row are mutable, and ISHED1 can handle different heat exchanger sizes so long as there are equal numbers of tubes in each row, the number of depth rows does not exceed 5, and the total number of tubes does not exceed 130.

ISHED1 is able to apply background knowledge reflecting the nature of the problem in order to constrain the search to plausible architectures. A user-defined parameter (or its default value) imposes limitations on the lengths of most tube bends. ISHED1 also enforces six real-world constraints on architectures, ranked from suggested to essential. The program rejects structures that violate a required constraint, and only under special circumstances (namely when designing a more compliant architecture is very difficult) generates structures that violate the most lenient constraints.

Two of the six constraints state that inlet tubes should not, and that tubes from which exit tubes receive their refrigerant should, be located next to exit tubes. These constraints allude to the fact that while through most of its travels through the heat exchanger the refrigerant is a mixture of liquid and gaseous coolant, and thus at a temperature close to its evaporation point at its current pressure, the refrigerant in the exit tube is all gas, and as such is warmed rapidly to a higher temperature by the exchange of heat (as opposed to the heat being used in a phase shift). There is some noticeable conduction of heat between the exit tubes and their immediate neighbors; this is minimized when the refrigerant in those neighboring tubes is also close to leaving the heat exchanger system. Similarly another constraint, the constraint that exit tubes should be in the first depth row, is based on the fact that the overall cooling will be most effective when this warmest refrigerant encounters some of the warmest air, and the coolest refrigerant meets already cooled air.

A constraint limiting splits in refrigerant paths is based on the unacceptable drops in refrigerant pressure that will occur if a single path undergoes multiple splits. Another constraint requiring inlets and outlets to be on the same side of the heat exchanger manifold is based on the structural requirements of the air conditioning unit, as is one that forbids looping in the refrigerant path.

Overview of ISHED1

The goal of ISHED1 is to apply the recently developed Learnable Evolution Model (Michalski 2000) to assist an expert in designing optimal heat exchanger architectures under given operating conditions. ISHED1 works in conjunction with two other major systems, a simulator, EVAP5, that evaluates the performance of given heat exchanger architectures (Domanski 1989) and a general purpose AQ-type inductive learning system (Michalski 1983, 2000; Kaufman and Michalski, 1998) that is employed in the Symbolic Learning Module of ISHED1.

Following the LEM methodology, ISHED1 integrates two evolutionary strategies: *Darwinian evolutionary learning* and *Symbolic evolutionary learning*. Figure 2 presents a general diagram of the implemented ISHED1 system. Darwinian evolutionary learning is performed by the Evolutionary Learning Module, and Symbolic evolutionary learning is performed by the Symbolic Learning Module, which implements AQ-type inductive learning and a hypothesis instantiation module.

The Control Module takes the current population of candidate heat exchanger designs and determines which evolutionary strategy to apply. The selected strategy operates on the population, generates the subsequent population, and passes it to the simulator for evaluation of the individual structures. These structures and their evaluations are returned to the Control Module for the next generation (iteration).

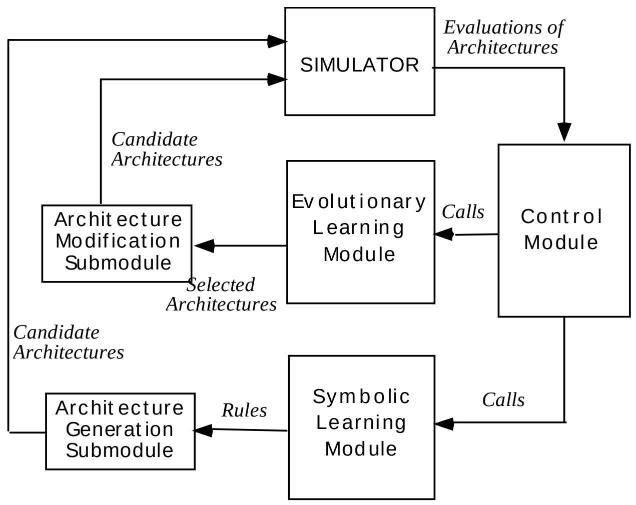

Figure 2: A general functional architecture of ISHED1.

Two related parameters guide the Control Module in determining which strategy to apply. Basically, ISHED1 applies Darwinian evolution until the population is no longer improving. It then switches to symbolic learning until similarly the performance (both in terms of the best individual and the population overall) has plateaued, continuing to alternate modes based on performance. Experiments have indicated that the application of a new strategy is often sufficient to remove the evolutionary process from the point where it has stalled.

Because of the nature of the problem and the feasible ways of internally representing heat exchanger structures, both evolutionary modules required problem-specific customization from previous LEM applications (e.g., Coletti et al. 1999; Michalski and Zhang 1999; Cervone and Michalski 2000). Traditional genetic operators would be, for the most part, unworkable in this domain, so eight analogous domain-specific *structure modifying* (SM) operators were implemented, such as swapping the positions of two adjacent tubes in the flow, or moving the source of a tube's refrigerant further upstream, in the process creating a split path. The SM operators change the characteristics of the candidate exchangers in ways that will most likely lead to *admissible* new structures, that is, structures satisfying the given physical and environmental constraints, as described above. A selected operator is tried repeatedly with different operands in order to generate a feasible structure, until it either succeeds or "times out" (based on limits specified in the user's control parameters), in which case another operator, hopefully more applicable, will be tried.

The second strategy, based on symbolic learning, examines the characteristics of both well- and poorly-performing designs, and automatically creates hypotheses (in the form of attributional rules) that characterize the better-performing architectures. These hypotheses are then applied to generate a new population of designs.

Due to the complexity of the domain, the learning program uses an abstract, rather than precise, specification of the different structures. Consequently, the learned rules also refer to abstract designs. These abstract rules allow the system to instantiate them in many different ways to produce specific designs. The rule instantiation process must, however, follow the previously mentioned constraints on designs. Generating heat exchanger architectures satisfying these constraints from a specification of inlet, outlet and split tubes is a computationally complex problem. To simplify it, ISHED1 usually generates only one architecture from a given specification of the key tubes.

To summarize, given instructions characterizing the environment for the sought heat exchanger design, an initial population of designs (either specified by the user, randomly generated, or a combination of the two), and parameters for the evolutionary process, ISHED1 evolves populations of designs using the above two strategies for a specified number of generations. At the end, it produces a report stating the best designs (architectures) found and their estimated capacity, as determined by the simulator. The ISHED1 control module determines when to apply each of the two evolutionary strategies (see Figure 2).

System Operation

Control Parameters

The first step of the ISHED1 operation is to read a file that defines the control parameters for the program and the characteristics of the desired architecture. These parameters, which override defaults when read, allow users latitude in controlling the system run. They can be grouped into the following clusters (some individual parameters are alluded to in multiple clusters):

- Parameters defining the characteristics of a heat exchanger: its size and its shape
- Parameters defining the characteristics of the initial population: its size, any user-specified first generation individuals, and the nature of the individuals randomly specified by the system
- Parameters defining the length of the evolutionary process
- Parameters describing the airflow through the heat exchanger, defined by a list of locations in the cross-section of the exchanger, and the velocities of the air at these locations. The velocities in other locations are linearly interpolated.

- Run control parameters, including the persistence of the Darwinian and symbolic learning modes, parameters for guiding Darwinian mode operation and symbolic mode architecture generation, and the level of detail to be presented in the output file.

Defining the Initial Set of Structures

ISHED1 allows a user to define an initial set of heat exchanger architectures. If the user does not define them, the system generates the initial set randomly. The user may define one or more initial architectures, and specify the number of copies of each architecture to be generated to fill the initial population. User-specified architectures may be based either on previous ISHED1 runs, or draw upon the user's knowledge of the problem, so as to test hypotheses or to try to improve upon industry models.

It is also possible that the user may define only a portion of the initial population, in which case the system randomly generates the remaining designs. The random generation process creates different types of architectures in proportions defined a priori in the program. These proportions are determined through estimations of the form promising architectures are likely to take, based on the number of tubes in the target heat exchanger.

System Control

The control module starts in Darwinian evolution mode using an elitist strategy (i.e., the best performing architecture so far *always* gets passed to the next generation as the first element of the new population). Elitism also is used in Symbolic learning mode; there the best architecture explored by the program so far *and* all architectures that were in the “good” class for rule learning propagate directly to the next generation. Elitism has proven to be an important feature of evolutionary computation processes and is used in many algorithms.

In Darwinian mode, the structure modifying operator selected for application to a given structure is chosen probabilistically, based on the topology of the heat exchanger structure (the number of inlets, outlets and splits). The probabilities are based on an estimate of how likely an application of this operator is likely to result in a favorable change. Typically, the less catastrophic operators and those that will maintain architectures with three inlets or fewer are favored.

Darwinian Evolution Module

A generation in the Darwinian Evolution Module follows the same three-step pattern that is followed by traditional genetic algorithms: a step in which individuals are probabilistically selected to be the basis for the next generation, with selection probability proportional to their evaluated fitness; a step in which the selected individuals are modified by various operators; and finally a step in which the members of this new population are evaluated.

To guide the modification of structures in the evolutionary design process in harmony with the heat exchanger design constraints, we developed and implemented eight Structure Modifying (SM) Operators for ISHED1, each of which makes a change in the heat exchanger design on which it is operating.

The system probabilistically selects an operator to apply, based on the topology of the heat exchanger design being operated on (the data representation maps out the architecture precisely by encoding a vector of each tube’s refrigerant source). It will search for a feasible application of the operator, trying it with several different sets of operands if necessary. If it seems that the operator will not lead to a feasible change in the structure, another operator is tried; if the system is unable to admissibly apply an SM operator after a large number of iterations, the program has an escape clause, so as not to get stuck in an exit-less loop. In such a rare case, ISHED1 will apply a null operator to the structure.

Symbolic Learning Module

When the Symbolic Learning module is applied, members of the current population are divided into three classes based on their cooling capacity. If all individuals have identical performance, evolution in this mode is not possible, and Darwinian evolution will take place instead.

The range from best performance level in the population to the worst is examined. Individuals with performance in the top HFT% of this range are placed in the "good" class. Similarly, individuals in the bottom LFT% are placed in the "bad" class, where HFT and LFT (high and low fitness thresholds) are program parameters (in ISHED1 HFT=LFT=25%). Other individuals are placed in the "indifferent" class.

The AQ18 rule learning program (Kaufman and Michalski, 2000) generates a set of rules that distinguish "good" architectures from "bad", based on abstractions of the specific architectures. These rules are then instantiated in a different ways to generate new architectures. During consecutive generations, rules are used in the context of their predecessors, so as to further focus the concept of design optimality.

Each new generation consists of the best architecture discovered so far, architectures of the "good" class, and new architectures generated by instantiating the learned rules. If there are enough open slots in the population, each rule is applied at least twice, by rotating among them, ordered by the number of training examples satisfying each rule (called t-weights, which are indicators of the rule's strength). Finally, any other individuals are generated based on rules chosen probabilistically, with rules’ probabilities proportional to their t-weights.

Experiments

During the course of ISHED1 development, many experiments with the system were conducted. The initial experiments concentrated on a well-known problem, using a common heat exchanger size and a fairly uniform airflow pattern. In these experiments, ISHED1 designs were comparable to the industry standard. One concern in some of these designs was that after many generations of Darwinian evolution, the designs would become chaotic in terms of their inter-tube connections. Nonetheless, using available tools, an engineer can smooth the connections, hopefully at little cost to the capacity of the exchanger.

In later experiments, the refrigerant was changed, and the airflow pattern was defined as highly non-uniform. Under such conditions, industry-standard heat exchangers do not perform well. The best ISHED1-produced architectures conformed intuitively to expectations of what a successful architecture in a non-uniform airflow should look like, and indeed performed far better than the currently used expert-designed structures.

Subsequent experiments varied the size and shape of the heat exchanger -- between 2 and 4 depth rows, with between 40 and 90 total tubes. Similar results were observed. During the final stage of development, we began experimenting with pre-specified members of initial populations. These results were to some degree mixed. When a very large portion of the initial population was pre-specified with known good architectures, further improvement could often be found. To some degree, the pre-specification is analogous to an initial symbolic learning step using the prior background knowledge; as a result, ISHED1 begins with a solid population.

But when fewer individuals were used to seed the initial population, improvement was hard to come by. While further experimentation is needed to determine if this is a regular occurrence, and if so its cause, it is possible that a level of imbalance is reached in the population that hinders both the establishment of large numbers of seeded examples and their kin for improvement, and the blossoming of promising, but relatively weak, randomly generated individuals. It is also possible that system parameters then need to be adjusted from default values.

While it is not possible to include an entire run log from even a single experiment due to the space limitation, a small sample of the ISHED1 output may be useful to provide a flavor of the program's operation. Such an excerpt, with some annotations added for readability (in italics), is shown in Figure 3. Parameters t, u and q associated with a rule characterize the rule's quality (Kaufman and Michalski, 2000).

In general, these experiments served to confirm the ability of ISHED1 to generate improved designs, and to adapt to different environmental situations. Thus, it has proven its potential to be a powerful tool for automating the heat exchanger design processes.

```
Exchanger Size: 16 x 3
Population Size: 15   Generations: 40
Operator Persistence: 5
Mode Persistence: GA-probe=2 SL-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, ...
Operations: NS(23, 39), SWAP(8), SWAP(28), ...,
          SWAP(29), SWAP(25), SWAP(1)
```

Below is one of the structures created by the application of a SM operator in Darwinian 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
and 14 others.

Selected Members:  6, 15, 11, 3, 13, 1, ...
. . . . . .
```

The program soon shifts into Symbolic Learning Mode:

```
Generation 5: Learning mode
Learned rule:
    [x1.x2.x3.x4.x5.x6.x7.x8.x9.x11.x12.x13.x14.x
    15.x17.x18.x19.x20.x21.x22.x23.x24.x25.x26.x2
    7.x28.x29.x30.x31.x32.x33.x34.x35.x36.x37.x38
    .x39.x40.x41.x42.x43.x44.x45.x46.x47.x48=regu
    lar] & [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 is a structure from the 21st generation:

```
Generation 21: Learning mode
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 14 others

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

Finally, ISHED1 achieves:

```
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
```

Figure 3: An excerpt from the log of an ISHED1 run.

Conclusion

A method and system ISHED1 was described that assists engineers in optimizing heat exchanger designs. The method is based on the Learnable Evolution Model, which uses machine learning to guide evolutionary computation.

Among the areas for potential improvement of ISHED1 are several aspects of the Symbolic Learning module. One of them concerns the rule instantiation process, which is currently fixed and may constrict the diversity of the population generated. It is also not clear how well the representation space we chose reflects the realities of the design task. An interesting topic for future development is the integration of constructive induction into the symbolic learning engine in order to find the best fit between the representation space and the problem at hand.

Experiments with ISHED1 have demonstrated that it is capable of generating designs equal or superior to the best human designs, particularly in cases of non-uniform airflow. It thus provides a powerful new tool for assisting engineers in designing heat exchangers based on the synergistic application of Darwinian evolution and symbolic learning from examples. It is believed that the described methodology can also be applied to other problems in engineering design.

ISHED1 was developed in collaboration with the National Institute of Standards and Technology (NIST), which is currently conducting extensive experiments with ISHED1 and introducing it to the air-conditioning manufacturing industry (Domanski 2000) for use in their design processes.

Acknowledgments

The authors express their deep gratitude to Dr. Piotr Domanski from the National Institute of Standards and Technology for introducing to them the problem of designing optimal heat exchangers, for his excellent collaboration on the project, and for providing the heat exchanger simulator used in ISHED1. They also thank the National Institute of Standards and Technology and International Intelligent Systems, Inc. for their support of this project. The LEM methodology and the basic machine learning algorithms used were developed in the Machine Learning and Inference Laboratory at George Mason University, which is supported by the National Science Foundation under grants IIS-9904078 and IRI-9510644.

References

Cervone, G. 1999. An Experimental Application of the Learnable Evolution Model to Selected Optimization Problems. Master’s Thesis. *Reports of the Machine Learning and Inference Laboratory*, MLI 99-8, George Mason University, Fairfax, VA.

Cervone, G. and Michalski, R.S. 2000. Design and Experiments: LEM2 Implementation of the Learnable Evolution Model. *Reports of the Machine Learning and Inference Laboratory*, MLI 00-2, George Mason University, Fairfax, VA.

Coletti, M., Lash, T., Mandsager, C., Michalski, R.S., and Moustafa, R. 1999. Comparing Performance of the Learnable Evolution Model and Genetic Algorithms on Problems in Digital Signal Filter Design. *Proceedings of the 1999 Genetic and Evolutionary Computation Conference (GECCO)*.

Domanski, P.A. 1989. EVSIM - An Evaporator Simulation Model Accounting for Refrigerant and One Dimensional Air Distribution. NISTIR 89-4133.

Domanski, P.A. 2000. Evaporator Model with A Visual Interface. Presentation at *Winter Meeting of the American Society of Heating, Refrigeration and Air-conditioning Engineers (ASHRAE)*, Dallas, TX.

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

Kaufman, K.A. and Michalski, R.S. 2000. The AQ18 System for Machine Learning: User’s Guide. *Reports of the Machine Learning and Inference Laboratory*, MLI 00-3, George Mason University, Fairfax, VA.

Michalewicz, Z. 1996. *Genetic Algorithms+Data Structures = Evolutionary Programs*. Springer Verlag, 3rd edition.

Michalski, R.S. 1983. A Theory and Methodology of Inductive Learning. In Michalski, R.S. Carbonell, J. and Mitchell,T., eds., *Machine Learning: An Artificial Intelligence Approach*. Palo Alto: TIOGA Publishing Co., 83-134.

Michalski, R.S. 1998. Learnable Evolution: Combining Symbolic and Evolutionary Learning. *Proceedings of the Fourth International Workshop on Multistrategy Learning (MSL'98)*, 14-20.

Michalski, R.S. 2000. LEARNABLE EVOLUTION MODEL: Evolutionary Processes Guided by Machine Learning. *Machine Learning* 38, 9-40.

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