# *Reports*

## *Machine Learning and Inference Laboratory*

**Progress Report on Learnable Evolution Model**

**Ryszard S. Michalski**
**Janusz Wojtusiak**
**Kenneth A. Kaufman**

**MLI 07-2**
**P 07-2**
**May 15, 2007**


George Mason University

# PROGRESS REPORT ON LEARNABLE EVOLUTION MODEL

Ryszard S. Michalski, Janusz Wojtusiak and Kenneth A. Kaufman
Machine Learning and Inference Laboratory
George Mason University

Fairfax, VA 22030-4444, USA

{michalski,jwojt,kaufman}@mli.gmu.edu
http://www.mli.gmu.edu

## Abstract

This report reviews recent research on Learnable Evolution Model (LEM), and presents selected results from its application to the optimization of complex functions and engineering designs. Among the most significant new contributions is a multi operator methodology for generating individuals (candidate solutions) and the employment of a more advanced learning program, AQ21, as the learning module. The new features have been implemented in the LEM3 program. To evaluate LEM3's performance, it was experimentally compared to other evolutionary computation programs, such as, EA--a conventional, Darwinian-type evolutionary computation program, CA--a cultural evolution algorithm, and EDA--an estimation of distribution algorithm_on selected function optimization problems. To determine the scalability of LEM3 and compared programs, the number of variables in the optimized functions was varied from 2 up to 1000. In every experiment, LEM3 outperformed the other programs in terms of the evolution length, sometimes more than an order of magnitude. Another recent research result is the development of early versions of two LEM-based systems, ISHED and ISCOD, for the optimization of heat exchangers evaporators and condensers, respectively. This work was done in collaboration with scientists from the National Institute of Science and Technology. In experimental testing, the systems produced designs that matched or were superior to human designs, particularly, in the cases of non-uniform air flows. This collaboration continues, and may ultimately produce systems that NIST will use to develop better designs of heat exchangers and have them implemented by the industry.

**Keywords:** Evolutionary Computation, Function Optimization, Learnable Evolution Model, Guided Evolutionary Computation

## Acknowledgments

This report is a significantly extended and modified version of a paper presented at the 18th IEEE International Conference on Tools with Artificial Intelligence, ICTAI'06 (Michalski, Wojtusiak and Kaufman, 2006)

This research has been conducted in the Machine Learning and Inference Laboratory at George Mason University. The Laboratory's research has been supported in part by the National Science Foundation under Grants No. IIS-0097476 and IIS-9906858, and in part by the UMBC/LUCITE #32 grant. The findings and opinions expressed here are those of the authors, and do not necessarily reflect those of the above sponsoring organizations.

## 1 INTRODUCTION

An attractive approach to solving very complex optimization problems is to employ evolutionary computation. In a conventional, Darwinian-type methods of such computation, innovation to the population of solutions is introduced through mutations and/or recombinations. Because these are semi-random operators, conventional evolutionary computation is a form of a trial and error search method, and thus not very efficient.

Another approach to evolutionary computation is to employ an "intelligent agent" to guide the process of introducing innovation. Such an approach has been implemented in Learnable Evolution Model (LEM) in which the role of an intelligent agent is performed by a machine learning program (Michalski, 2000). In LEM, innovation is introduced by a new type of operators—*hypothesis generation* and *hypothesis instantiation*—that apply a learning and reasoning process.

To generate new solutions, these operators exploit the differences between groups of high and low performance solutions. First, the hypothesis generation operator induces general rules delineating subareas in the space likely to contain the optimum, and then the hypothesis instantiation operator populates these subspaces with proposed new solutions. Multiple experiments have confirmed that an application of these operators can significantly shorten the *evolution length*, as measured by the number of fitness evaluations needed to achieve a desired solution.

Hypothesis generation and instantiation operators are, however, computationally more complex than conventional mutations and recombinations, or operators used in standard gradient methods of optimization, because they require an execution of non-trivial inductive and deductive inferences. This means that there is a trade-off between advantage of applying the new operators and computational simplicity of executing conventional operators. To take advantage of this trade-off, LEM integrates both types of operators—new and conventional ones—and tries to apply them in a way that maximizes the effectiveness of the optimization process.

The rest of this paper is organized as follows. Section 2 briefly describes the LEM3 implementation of learnable evolution. Section 3 explains different operators for generating new candidate solutions in LEM3. Section 4 describes LEM3's Control Module that selects the operator to be applied at any given step. Newly generated individuals (solutions) are then selected for a new population. This process is described in Section 5. Section 6, briefly describes ISHED (version 2), a LEM-based a system specialized for the optimization of heat exchanger designs, and Section 7 reports selected results obtained by it. Section 8 relates LEM research to other methods of evolutionary optimization. The final Section 9 concludes the paper with suggestions of desirable directions of further research.

## 2 AN OVERVIEW OF LEM3

Figure 1 presents a flow diagram of LEM3. The process starts with a generation of an initial population of candidate solutions. This can be done in three different ways, by a random process, by loading an existing population from an external source, or by a combination of these two methods.

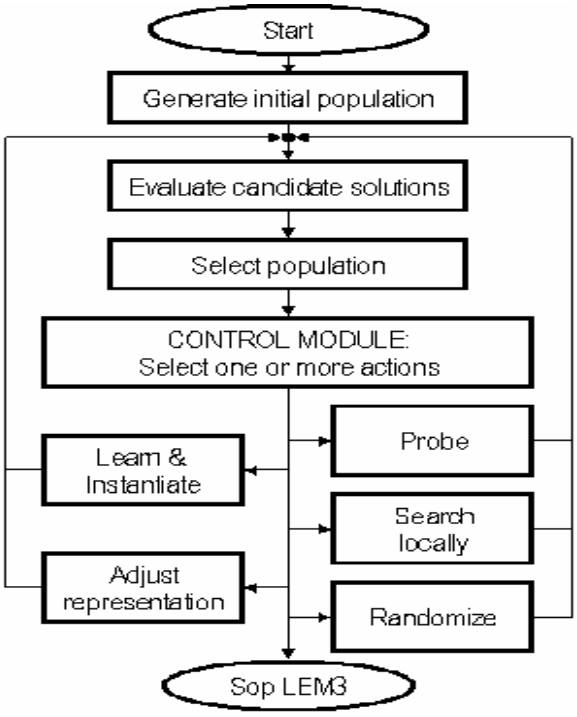


Figure 1: The LEM3 top level algorithm.

In the next step, candidate solutions in the population, either those in the initial population or those resulting from a previous run of program are evaluated according to a user-defined fitness function (a.k.a. objective function). Based on the results of the evaluation, a new population of solutions is created by one of the standard selection methods developed in the field of evolutionary computation. The current version of LEM3 implemented rank, tournament, and proportional methods of selection. Details on this step are presented in Section 3.5.

The subsequent steps perform the most elaborated part of LEM3, namely, introduce innovation to the current population. This is done in LEM3 in several ways, depending on what action or actions are selected by the Control Module. One important novelty of LEM3 is that it can execute different actions (alternatively called *modes of operation*) each employing a different type of innovation operators. Another novelty is that it can execute two or more actions in parallel. Possible actions include: *Learn and Instantiate*, *Probe*, *Search Locally*, *Adjust Representation*, and *Randomize*. These actions are described in detail in Section 3. Section 4 describes the method used for determining which action or actions to apply at any given step.

## 3 A DESCRIPTION OF LEM3 ACTIONS

### 3.1 Learn and Instantiate

The "Learn and Instantiate" action is the original and central component of the Learnable Evolution Model. This action creates new candidate solutions by performing three steps:

(1) Selecting a training set of solutions from a *precursor population* for the learning program,

(2) Learning a general hypothesis characterizing subspaces likely containing the optimum, and

(3) Instantiating the hypothesis in different ways to create new candidate solutions.

The precursor population is the current population, or a union of the current and some previous populations, specified by the *lookback parameter* (Michalski, 2000)[1]. Step (2) starts by determining the *training set*, which consists of a group of *high-performing* (*H-group*) and a group of *low-performing* (*L-group*) candidate solutions selected from the precursor population according to the fitness function. Details on methods of creating H- and L- groups are described in (Michalski, 2000) and (Wojtusiak and Michalski, 2005). The H- and L-group serve as positive and negative examples, respectively, for a learning program, which in LEM3 is AQ21. This program is the newest member in the AQ learning programs family (Wojtusiak, 2004a; Wojtusiak et al., 2006).

In principle, there is no restriction on which learning program is used in LEM, provided that an effective method has been developed for instantiating hypotheses induced by it. The AQ-type learner has shown to be highly suitable for LEM, because the classifiers it learns are both, easy to instantiate and more expressive than those learned by other programs, because they employ a more expressive representation language, namely *attributional calculus* (Michalski, 2004).

Specifically, classifiers learned by AQ21 are sets of *attributional rules*, whose simplest form is:

$$\text{CONSEQUENT} <= \text{PREMISE} \quad (1)$$

where CONSEQUENT and PREMISE are conjunctions of *attributional conditions* (a.k.a. *selectors*). An attributional condition defines a relation between an attribute, or a group of attributes, and the values satisfying that relation. Here is an example of an attributional rule:

```
[refrigerator-design = modern]
   <= [energy use = 130..150] & [style = french_door] &
        [surface material= aluminum v titanium] &
        [dimensions: height < 6 & width = 36 &
         depth = cabinet-depth v countertop-depth]
```

This rule states that a design is classified as modern, if its energy use is between 130 and 150 kW/year (units are defined in the attribute domain), its style is “French door”, its surface material is aluminum or titanium (these are values of the structured[2] attribute “material”), its dimensions are: the height is smaller than 6’, the width is equal 36”, and the depth is either cabinet depth or countertop-depth. The attribute “dimensions” is a *compound attribute* whose constituent attributes are *height*, *width*, and *depth*. As one can see, the rule is easy to interpret, which makes it possible for experts to develop an insight into the problem being optimized. For more information about attributional rules and compound attributes, see (Michalski, 2004).

For the purpose of understanding LEM, it is sufficient to know that its learning module learns a classifier that consists of attributional rules whose set-theoretical union is a generalization of the

[1] All references in this paper by one or more of the authors refer to papers downloadable from http://www.mli.gmu.edu/mpubs.html

[2] The domain of a structured attribute is a partially ordered set. The most common structured attribute is a hierarchical attribute whose domain is a hierarchy of concepts (Kaufman and Michalski, 1996).

H-group solutions, but does not include any solutions from the L-group. Rules in the classifier delineate segments of the search space that is worth to explore further, as there is a likelihood that one of them may contain the optimal solution. If there is more than one optimal solution, they may be located in different segments. By instantiating rules in the classifier in different ways (which is equivalent to sampling these segments), new candidate solutions are created. Details of this process are presented in Section 3. Because each rule can be separately instantiated, the generation of new solutions may be conducted in parallel. If there are several optimal solutions, the program may find them all or a subset of them simultaneously.

To instantiate a rule, for each condition of the rule, the program randomly assigns an attribute value that satisfies that condition. For attributes not included in the rule, the program selects a value that the attribute takes in a randomly selected individual from the H-group. Because rule conditions can usually be satisfied by several different values, many different individuals can be created by instantiating one rule. For details of this process, see e.g., (Wojtusiak and Michalski, 2005; 2006).

There are several modifications to the above basic instantiation algorithm, one of which is a flexible interpretation of selectors in a rule. For example, if a rule states that a design is high-performing if its energy use is between 130 and 150, its may advantageous to generate designs with energy use 129 or 151 as instances of the high-performing class, although their energy use does not strictly match the condition. A flexible interpretation of a selector assigns a degree of match to it that diminishes with the distance of the attribute value in an entity matched to the attribute value/s stated in the selector. This degree affects the probability of generating solutions with an attribute value outside of the strict range.

As mentioned earlier, learned rules are used to generate new solutions, not to match given solutions against the rules in order to classify them, as in classification problems. Therefore, a flexible interpretation of attributional rules in LEM3 is done differently than in classification problems. The method for flexibly instantiating rules, implemented in LEM3, generates *s*% individuals with attribute values strictly satisfying rule conditions, and *f*% individuals with attribute values whose probabilities linearly decrease with distance from the condition border. where, s% and f% are control parameters (s% = 100% – f%).. In experiments, such a flexible rule interpretation gave better results than strict interpretation for some problems, e.g., for optimizing the Rosenbrock function in which the solution is located on a narrow ridge and therefore may be missed by strictly interpreted rules.

### 3.2 Probe Action

The probe action generates new individuals by *guided* Darwinian-type operators. These operators are akin to mutations and crossover but are designed to represent types of changes in the solutions that according to an expert may plausibly lead to their improvement, and satisfy constraints imposed on the attributes describing solutions by attribute types and domain sizes. In order to be applicable to a wide range of problems, these operators are defined to LEM3 by the user.

To make LEM3 applicable to wider range of problems, it allows the user to describe solutions in terms of several different types of attributes, such as nominal, structured (hierarchical), ordinal, cyclic, interval, ratio, absolute, and compound (Michalski and Wojtusiak, 2007). These types are

taken into consideration during both hypothesis generation and instantiation, and probing as they represent problem background knowledge that is used to guide these operators. For example, mutations of metric attributes (interval, ratio and absolute) involve making small modifications to their values within the scope of the attribute domain. Mutations of symbolic attributes (nominal, ordinal, cyclic and structured) are done appropriately for each attribute type. Nominal attributes are mutated by randomly taking another value from the attribute domain. Ordinal attributes are mutated by taking a neighboring value. Mutations of hierarchical attributes involve making small steps in climbing up and down the attribute hierarchies. For more details on probing operators implemented in LEM3, see (Wojtusiak and Michalski, 2005; 2006).

A crossover operator in probing action is done by randomly selecting two parent individuals from the population, and creating two new individuals by exchanging values of the first *k* attributes, where *k* is selected randomly. Results are accepted only when they do not contradict the constraints that reflecting relationships among different attributes and any other problem knowledge introduced to the program.

### 3.3 Search Locally

A local search employs user-defined methods. It is used when at least some solutions in the current populations are expected to be close to the global optimum and only in searching for optimal values of the metric attributes. Because local search methods have been studied for many years and are well-known, LEM3 has been designed to allow the user to attach an external program to run a method of the user's choice. A full implementation of this feature is still under development. Currently, the local search is executed by applying the user-defined method (an external program) to the best candidate solution whenever the "Probe" action is executed.

### 3.4 Adjust Representation Space

This action applies operators that modify the representation space of solutions in order to make it more suitable for a successful application of the learn and instantiate action. The representation space can be modified by such operators as modifying domains of metric variables (through different ways of discretizing them), removing variables considered irrelevant to the optimization problem, and/or creating new, more relevant variables as functions of the original variables.

So far, we developed of an operator that seeks an optimal discretization of metric attributes. It employs the method for *Adaptive Anchoring Discretization*, called ANCHOR (Michalski and Cervone, 2001) that discretizes continuous attributes with a granularity size dynamically increasing in the subranges of the variable domain that appear to require such an increase. The method starts with an initial, very rough discretization of the variable domain. Once it starts converging toward a possible solution, the precision of metric attributes is increased in the subranges of the entire domain suggested by the best known individuals. Conditions under which LEM3 invokes ANCHOR are specified in Section 3.6.

### 3.5 Select Population

The survival-of-the-fittest principle that underlies Darwinian-type algorithms is applied using one of the selection methods developed in the field of evolutionary computation. LEM3 implements several methods: the rank selection (that selects solutions that have the highest rank,

as determined by the fitness function), probabilistic selection (a.k.a. proportional or roulette-wheel selection that selects solutions with probability proportional to their fitness), and tournament selection (that selects solutions that "win" when compared with other randomly drawn solutions). Note that these selection methods are based on the fitness (a measure of quality) of individual solutions, and do not take into consideration other factors, such as the need to maintain diversity of population, that is, to keeping representative solutions from different parts of the space.

### 3.6 Action Selection Module

The Action Selection Module uses an *Action Profiling Function* (APF) to control which actions are applied at any given step of the computation. Initially, by default, the Control Module selects the "Learn and Instantiate" action. If an unsatisfactory progress is observed after a number of iterations defined by the *learn-probe* and *learn-threshold* parameters, the program switches to the "Probe" action. The *learn-probe* parameter defines the minimum number of iterations for which the Learn and Instantiate action is to be performed, even if the progress is unsatisfactory. The *learn-threshold* parameter specifies the minimal improvement of the fitness value of the best individual in the population in order to evaluate progress as satisfactory. After applying the Probe operator, LEM3 attempts "Learn and Instantiate" again.

LEM3 counts how many times the "Probe" action was applied after "Learn and Instantiate" and failed. If this number reaches *mutation-probe*, control switches to the "Adjust Representation" action. The *representation-probe* parameter defines the maximum number of times the representation is adjusted before switching to the "Randomize" action, which randomly generates new individuals.

One way to apply above actions is to execute them in the sequence listed above. After the Learn and Instantiate action stops improving the population, the Probe operator applies mutation to introduce diversity. This is particularly important when the population becomes uniform, and it is not possible to determine different H- and L-groups.

If the Probe action does not lead to a sufficient improvement after a defined number of repetitions, the next action is to increase the representation precision, which is done by discretizing selected ranges of values into smaller units according to the ANCHOR method. Again, if this action does not bring sufficient improvement after a certain number of steps, the Control Module executes a start-over action that generates a number of solutions randomly and introduces them into the population. This step seeks to explore parts of the search space that may have been previously missed.

Another way to execute the above actions is to apply some of them in parallel. This is an important novelty of LEM3. For example, the program may generate 100 individuals in each generation, 80 of which are created by learning and instantiation, 10 by applying crossover, 5 by applying mutation, and 5 by random generation. Numbers of individuals created by different actions can be adjusted based on the program's performance. This is the newest feature of LEM3 that is being currently tested, and obtained results will be presented in another report.

## 4 APPLYING LEM3 TO FUNCTION OPTIMIZATION

To test the performance and scalability of LEM3, it was applied to selected benchmark problems that involve optimizing Rastrigin, Griewangk and Rosenbrock functions with numbers of variables ranging between 2 and 1000. Results were compared with those obtained by applying EA, a conventional Darwinian-type program, implemented using Evolutionary Objects Library, to the same problems. We also compared LEM3 results with the published results by Estimation of Distribution Algorithms (EDAs), and a Cultural Algorithm (CAs).

Results from comparing LEM3 with EA are presented in Table 1. The relative performance of LEM3 and EA is measured by the speedup LEM3/EA, defined as the ratio of the evolution length of EA to the evolution length of LEM3 needed to achieve the same result. The evolutionary length is the number of fitness evaluations required by a program to reach a desired result. The speedup LEM/EA thus states how many times the number of fitness of evaluations done by EA is greater than the number of fitness evaluations done by LEM3. The stopping criterion for EA and LEM3 was finding a *δ-close solution*, that is, a solution that is better than the best solution in the starting population by a factor $1/\delta$, where $\delta$ is a user-defined parameter (Wojtusiak and Michalski, 2005; 2006). For example, if $\delta$=0.1, the best solution in the final population must be at least 10 times better that the best solution in the original population.

**Table 1:** Average speedups of LEM3 over EA in optimizing the Rosenbrock, Griewangk and Rastrigin functions with the number of variables ranging from 100 to 1000 for $\delta$=0.1 and $\delta$=0.01.

| **Number of variables** | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 | 1000 |
|---|---|---|---|---|---|---|---|---|---|---|
| **Speedup LEM3/EA** | 10.7 | 15 | 16.8 | 17.8 | 17.2 | 16.7 | 19 | 16.6 | 17.2 | 18 |

The table presents the speedup averaged for the three functions and two different values of $\delta$. Each experiment was repeated 10 times with a different starting population, which was the same for LEM3 and EA.

As one can see, the speedup of LEM3 over EA ranged between 10 and 18, and has a tendency to increase with the number of function variables. There was not a single case when speedup was 1 or smaller than one. It should be noted that LEM3 was executed with default parameters, without tuning it to these particular functions.

A comparison of LEM3 results with the best results from the Cultural Algorithm program (CA) was done by applying LEM3 to optimize the same functions and with the same number of variables as CA, as reported in (Reynolds and Zhu, 2001); specifically, to the optimization of the Rastrigin, Griewangk, and Rosenbrock functions of 5, 3, and 2 variables. For these numbers of variables, LEM3 required on the average times fewer fitness evaluations on the Rastrigin function, 53 times fewer fitness evaluations on the Griewangk function, and 243 times fewer fitness evaluations on the Rosenbrock function. Details are presented in Table 2. The stopping criterion for LEM3 was finding an individual with fitness at least as good as reported for the CA. Each experiment was repeated 40 times, and the above numbers are averages.

**Table 2**: Comparison of LEM3 with CA on the Rastrigin, Griewangk, and Rosenbrock functions (reproduced from Wojtusiak and Michalski, 2006).

| Function and # variables | Method | Best fitness Value | Evolution Length | LEM3/CA Speedup |
|---|---|---|---|---|
| Rastrigin 5 variables | LEM3 | 0 | 687 | **~728** |
| | CA | 5.4532e-05 | ~500,000 | |
| Griewangk 3 variables | LEM3 | 0 | 1,521 | **~53** |
| | CA | 1.0E-10 | ~79,900 | |
| Rosenbrock 2 variables | LEM3 | 0 | 219 | **~243** |
| | CA | 1.0e-10 | ~53,200 | |

Comparing LEM3's results with the best results from the several EDA implementations on Griewangk and Rosenbrock functions of 10 and 50 variables reported in (Bengeoxtea et al., 2002) also indicated its significant advantage. Specifically, LEM3 required on the average 142 and 66 times fewer fitness evaluations for optimizing the Griewangk and Rosenbrock function, respectively. The LEM3 stopping criterion was finding a solution with fitness at least as good as the one found by the EDA program. Each experiment was repeated 10 times, and reported numbers are averages. The averages for functions of 10 and 50 variables are reported in Table 3.

As Table 3 shows, greatest speedup was achieved in optimizing the Griewangk function of 10 variables, which was about 231. Note also that while LEM3 found the optimum (0), EDA result was very close (0.0511), but not the exact optimum.

Table 3: Comparison of LEM3 with EDA on the Rastrigin, Griewangk, and Rosenbrock functions (reproduced from Wojtusiak and Michalski, 2006).

| Function and # variables | Method | Best fitness Value | Evolution Length | LEM3/EDA Speedup |
|---|---|---|---|---|
| Griewangk 10 variables. | LEM3 | 0 | 1,305 | **~ 231** |
| | EDA | 0.051166 | 301,850 | |
| Griewangk 50 variables | LEM3 | 0 | 4,005 | **~ 54** |
| | EDA | 8.7673E-6 | 216,292 | |
| Rosenbrock 10 variables | LEM3 | 1.2 | 1,389 | **~ 118** |
| | EDA | 8.6807 | 164,519 | |
| Rosenbrock 50 variables | LEM3 | 46.74 | 7,875 | **~ 15** |
| | EDS | 48.8234 | 275,663 | |

## 5 LEM-BASED SYSTEMS FOR OPTIMIZING HEAT EXCHANGERS

Because LEM shortens the evolution length, this suggests that it may be particularly suitable for solving optimization problems in which fitness evaluation is time consuming or costly. Problems of optimizing complex engineering designs are of this type.

Using LEM methodology, we developed two specialized systems for optimizing designs of engineering systems, one, ISHED, for optimizing evaporators in heat exchangers (Kaufman and Michalski, 2000; Domanski et al., 2004; Michalski and Kaufman, 2006), and the other, ISCOD, for optimizing condensers. The evaluation of such design requires running a complex simulator and is time consuming. Heat exchangers are subject to a variety of physical and environmental

constraints, resulting in a very large number of different feasible designs, scattered throughout intractably large representation spaces. In both systems, the objective is to arrange the connections among the tubes that maximize the heat transfer. This problem is very important, because due to the ubiquity of heat exchangers in a modern society, improving efficiency of heat exchangers can bring significant economic, as well as environmental benefits.

ISHED and ISCOD were equipped with two modes of operation: learning and probing. When after a specified number of trials one of the modes makes insufficient progress, the program switches to the other mode. The learning operator learns rules expressed in terms of attributes that abstracted the heat exchanger design, and returns a hypothesis specifying parts of the abstracted representation space. The program then instantiates the rules, linking the tubes in the heat exchanger in ways that given the domain knowledge are plausibly feasible. More recent versions of the programs have enhanced the flexibility by which such an instantiation is made, so that the same rule can now generate more distinct heat exchangers.

Probing action utilizes eight operators akin to mutation and crossover, but tailored to the heat exchanger optimization domain. One operator, for example, swaps the position of two adjacent tubes in a refrigerant path, while another operator moves a fork point in a path up or down the path.

Experiments have consistently shown that both systems are able to adapt to varying environmental conditions, and evolve heat exchanger designs that perform on a par with, or better than the best human designs. In problems with highly uneven airflows, the ISHED designs were evaluated by experts as superior to the best human designs.

## 6 RELATED RESEARCH

The LEM3 program follows earlier implementations, LEM2 and LEM1, that used earlier versions of AQ learning programs and had fewer features. An implementation of Learnable Evolution Model for multi-objective optimization, LEMMO, developed by another research group, is described in (Jourdan et al., 2005). LEMMO is based on rules derived from decision trees learned by the C4.5 program, and was applied to a water quality optimization problem. The decision tree representation of the hypotheses is more limited than the attributional rule representation used in LEM3.

The evolutionary methods that seem to be closest in spirit to LEM are cultural algorithms (e.g. Reynolds and Zhu, 2001) that perform a constrained optimization process in which constraints are learned during the evolutionary computation. The constraints, called beliefs, reside in a belief space that is updated during the evolution process. Individuals that are stored in an optimization space are modified so that they satisfy the beliefs. The belief space is being built based on statistical information about individuals, which usually consists of intervals containing the fittest individuals.

Estimation of Distribution Algorithms (EDAs) use statistical inference, usually Bayesian or Gaussian networks, to estimate distributions of high-performing individuals selected from one population (Larranaga and Lozano, 2002). LEM significantly differs from EDAs in that it seeks rules for distinguishing between high- and low-performing individuals, and employs symbolic learning, rather than statistical. It also uses the fitness function not only for selecting individuals

for learning, but also during the learning process itself, while EDA uses it solely for selecting individuals. LEM does this by learning significance-based descriptions.

Another form of non-Darwinian evolutionary computation is performed by *memetic algorithms* that combine conventional evolutionary computations with local search methods (Hart, Krasnogor and Smith, 2004). LEM3 takes an advantage of this idea by including local search as one of its actions.

## 7 CONCLUSION

This paper reported recent results from research on Learnable Evolution Model (LEM). The most important result is the development of LEM3, the most complete and advanced implementation of the model so far. It includes some elements that go beyond the features described in (Michalski, 2000), such as the introduction of the Action Profiling Function and new instantiation algorithms. LEM3 is more advanced than LEM2 also due to the employment AQ21, the recent and most advanced version of AQ learning. In experimental applications of LEM3 to complex function optimization problems (with up to 1000 continuous variables) it outperformed EA, a standard Darwinian-type method. Comparisons with published results on Estimation of Distribution Algorithms and Cultural Algorithms also show the superiority of LEM3 in terms of evolution length.

LEM3 is highly scalable in comparison to the previous implementations. Extensive experiments have confirmed that LEM3 can serve as a powerful optimization system and that it outperforms other evolutionary computation systems in terms of evolution length. It was also applied to problems in which in addition to numeric attributes solutions are described in terms of different types of attributes, such as nominal, structured, and ordinal. The usefulness of this feature is not demonstrated in this paper, because in the first stage we wanted to compare LEM3 to other evolutionary computation programs, but these programs do not have this feature. There are, however, applications in which this feature may be very useful, such as the optimization of very complex engineering systems.

Early results from applying LEM-based ISHED and ISCOD systems to optimizing heat exchangers were evaluated by experts from the National Institute for Standards and Technology as superior to human designs in the cases of non-uniform air flows.

Summarizing, presented results confirm those published in previous papers that guided evolutionary computation represented by the LEM approach can be highly advantageous for very complex optimization problems in which the fitness evaluation is time-consuming or costly. It is especially recommended for problems for which standard evolutionary computation methods require long evolutionary processes. Our current research concerns several unresolved aspects of the LEM methodology, such as its computational complexity, convergence speed for different types of functions, and the areas of applicability to which it is the most suitable.

## REFERENCES


Bengoextea, E., Miquelez, T., Larranaga, P., and Lozano, J.A., "Experimental Results in Function Optimization with EDAs in Continuous Domain," in Larranaga, P. and Lozano J.A., *Estimation of Distribution Algorithms*, Kluwer Academic Publishers, 2002.

Domanski, P.A., Yashar, D., Kaufman K. and Michalski R.S., "An Optimized Design of Finned-Tube Evaporators Using the Learnable Evolution Model," *International Journal of Heating, Ventilating, Air-Conditioning and Refrigerating Research*, 10, pp 201-211, April, 2004.

Evolutionary Objects Library, downloadable from the website: http://eodev.sourceforge.net

Hart, W.E., Krasnogor, N. and Smith, J.E. (eds.), *Recent Advances in Memetic Algorithms*, Springer, 2004.

Jourdan, L., Corne, D., Savic, D. and Walters, G., "Preliminary Investigation of the 'Learnable Evolution Model' for Faster/Better Multiobjective Water Systems Design," *Proceedings of The Third International Conference on Evolutionary Multi-Criterion Optimization*, EMO'05, 2005.

Kaufman, K. and Michalski, R.S., "A Method for Reasoning with Structured and Continuous Attributes in the INLEN-2 Multistrategy Knowledge Discovery System," *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining* (KDD-96), Portland, OR, pp. 232-237, August, 1996.

Kaufman, K. and Michalski, R.S., "Applying Learnable Evolution Model to Heat Exchanger Design," *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000) and Twelfth Annual Conference on Innovative Applications of Artificial Intelligence (IAAI-2000)*, Austin, TX, pp. 1014-1019, 2000.

Larrañaga, P. and Lozano, J. (eds.), *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*, Kluwer Academic Publishers, 2002.

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

Michalski, R.S., "ATTRIBUTIONAL CALCULUS: A Logic and Representation Language for Natural Induction," *Reports of the Machine Learning and Inference Laboratory*, MLI 04-2, George Mason University, Fairfax, VA, April, 2004.

Michalski, R.S. and Cervone, G., "Adaptive Anchoring Discretization for Learnable Evolution Model," *Reports of the Machine Learning and Inference Laboratory*, MLI 01-3, George Mason University, Fairfax, VA, 2001.

Michalski, R.S., Wojtusiak, J. and Kaufman, K., "Intelligent Optimization via Learnable Evolution Model," *Proceedings of The 18th IEEE International Conference on Tools with Artificial Intelligence*, Washington D.C., November 13-15, 2006.

Michalski, R.S. and Kaufman, K., "Intelligent Evolutionary Design: A New Approach to Optimizing Complex Engineering Systems and its Application to Designing Heat Exchangers," *International Journal of Intelligent Systems*, 21, 2006.

Reynolds, R.G. and Zhu, S., "Knowledge-Based Function Optimization Using Fuzzy Cultural Algorithms with Evolutionary Programming," *IEEE Transactions on Systems, Man, and Cybernetics*, 31, 2001.

Wojtusiak, J., "AQ21 User's Guide," *Reports of the Machine Learning and Inference Laboratory*, George Mason University, MLI 04-3, Fairfax, VA, 2004a.

Wojtusiak, J., "The LEM3 Implementation of Learnable Evolution Model: User's Guide," *Reports of the Machine Learning and Inference Laboratory*, George Mason University, MLI 04-5, Fairfax, VA, 2004b.

Wojtusiak, J. and Michalski, R.S., "The LEM3 System for Non-Darwinian Evolutionary Computation and Its Application to Complex Function Optimization," *Reports of the Machine Learning and Inference Laboratory*, MLI 05-2, George Mason University, Fairfax, VA, October, 2005.

Wojtusiak, J., Michalski, R.S., Kaufman, K. and Pietrzykowski, J., "Multitype Pattern Discovery Via AQ21: A Brief Description of the Method and Its Novel Features," *Reports of the Machine Learning and Inference Laboratory*, MLI 06-2, George Mason University, Fairfax, VA, June, 2006.

Wojtusiak, J. and Michalski, R.S., "The LEM3 Implementation of Learnable Evolution Model and Its Testing on Complex Function Optimization Problems," *Proceedings of Genetic and Evolutionary Computation Conference*, GECCO 2006, Seattle, WA, July 8-12, 2006.

A publication of the *Machine Learning and Inference Laboratory*
George Mason University
Fairfax, VA 22030-4444 U.S.A.
http://www.mli.gmu.edu

Editor: R. S. Michalski
Assistant Editor: Janusz Wojtusiak

The *Machine Learning and Inference (MLI) Laboratory Reports* are an official publication of the Machine Learning and Inference Laboratory, which has been published continuously since 1971 by R.S. Michalski's research group (until 1987, while the group was at the University of Illinois, they were called ISG (Intelligent Systems Group) Reports, or were part of the Department of Computer Science Reports).