Intelligent Optimization via Learnable Evolution Model

Ryszard S. Michalski, Janusz Wojtusiak, and Kenneth A. Kaufman
Machine Learning and Inference Laboratory
George Mason University
{michalski, jwojt, kaufman}@mli.gmu.edu

Abstract

A new method for optimizing complex functions and systems is described that employs Learnable Evolution Model (LEM), a form of non-Darwinian evolutionary computation guided by machine learning. LEM's main novelties are operators for creating new individuals that include hypothesis generation, which learns rules indicating subareas in the search space likely containing the optimum, and hypothesis instantiation, which populates these subareas with new candidate solutions. LEM3, the newest and most advanced implementation of learnable evolution, is briefly described and experimentally compared with other evolutionary computation programs on selected function optimization problems. We also describe two specialized LEM-based systems for heat exchanger optimization.

1. Introduction

Research on intelligent optimization is concerned with developing algorithms in which the optimization process is guided by an "intelligent agent." In the Learnable Evolution Model (LEM), described in this paper, the role of the intelligent agent is played by a machine learning program.

LEM's evolutionary computation differs from conventional evolutionary computation in the way innovation is introduced into the process. In addition to conventional blind operators, such as mutation and recombination, LEM employs new types of innovation operators—*hypothesis generation* and *hypothesis instantiation*—that create new individuals by taking into consideration the properties of the current and possibly also past populations.

Specifically, the hypothesis generation operator hypothesizes general rules that differentiate between groups of high fitness and low fitness individuals (solutions). These rules delineate sub-areas in the search space likely to contain the optimum. Subsequently, the hypothesis instantiation operator populates these subspaces with proposed new solutions. Multiple experiments have confirmed that these operators can drastically shorten the *evolution length*, defined as the number of fitness evaluations needed to achieve a desired solution.

Hypothesis generation and instantiation operators are, however, computationally more complex than mutations and recombinations. Therefore, LEM integrates both types of operators—new and conventional ones—in a way that seeks to maximize the efficiency and effectiveness of the optimization process. It also employs other operators for introducing innovation.

2. Brief Description of LEM3

LEM3 starts by generating an initial population of candidate solutions. It can generate such solutions randomly, load a previously created population from an external source, or use a combination of the two methods. Solutions in the current population are evaluated according to a user-defined fitness function (or objective function). Based on the results of the evaluation, a new population of solutions is selected using one of the selection methods developed in the field of evolutionary computation, e.g. proportional selection, tournament selection, etc.

The next steps introduce innovation into the population in one of several ways, depending on what action or actions are selected by the LEM3 Control Module. These actions, also called *modes of operation*, are: *Learn and instantiate*, *Probe*, *Search locally*, *Adjust representation*, and *Randomize*. Figure 1 presents the top-level algorithm underlying LEM3.

3. LEM3 Actions

3.1. Learn and Instantiate

The *Learn and Instantiate* action is the most important novel component of the Learnable Evolution Model. This action creates candidate solutions in three steps: (1) select a training set for the learning program, (2) learn from the training set a general hypothesis that characterizes subareas likely to contain the optimum, and (3) instantiate the hypothesis to create new candidate solutions.

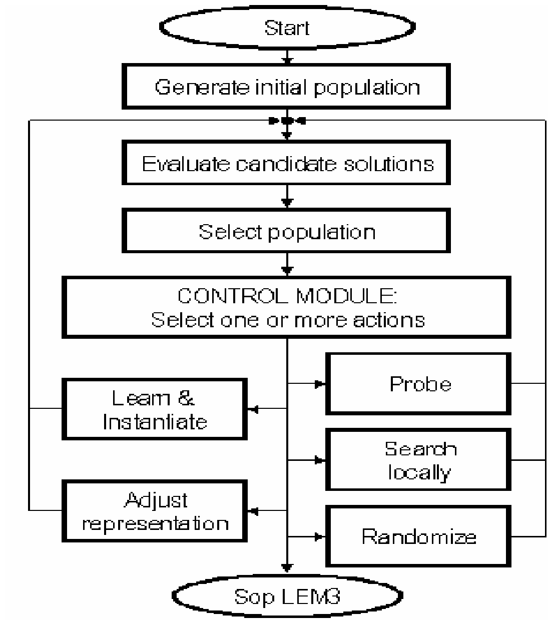

Figure 1. The LEM3 top level algorithm.

To determine a training set, Step (1) selects high-performing (H-group) and low-performing (L-group) candidate solutions from the current population (or from the current and past populations), according to the given fitness function. These solutions serve as positive and negative examples, respectively, for a learning program, which in LEM3 is AQ21, the newest implementation of the AQ learning family [12].

In principle, there is no restriction on what learning method can be used in LEM, provided that an effective hypothesis instantiation method is developed for the hypotheses it generates. The AQ-type learner is, however, particularly suitable for LEM, because it creates hypotheses that are easy to instantiate and easy to interpret.

Specifically, AQ21 produces hypotheses in the form of sets of *attributional rules* [7] that are generalizations of conventional decision rules. The simplest form of an attributional rule is:

CONSEQUENT <= PREMISE

where CONSEQUENT and PREMISE are conjunctions of *attributional conditions* (a.k.a selectors), whose basic form relates an attribute to a set of attribute values.

Attributional rules delineate subsets of the search space that can be separately instantiated. If the search space has several equivalent optima, different rules may delineate subspaces that contain these optima. Because rules are easy to interpret and understand, it is possible for experts to develop insights into the problem being optimized.

The instantiation process (Step 3) generates new individuals using attributional rules in the hypothesis. For each condition in each rule, the program randomly assigns an attribute value or values that satisfy 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 conditions in a rule can be usually satisfied by a number of different values, many different individuals can be created by instantiating one rule. For more details on this process, see [11].

There are several modifications to the above basic instantiation algorithm, one of which is a flexible interpretation of selectors. For example, if a rule states that a design is high performing if its length is between 12 and 70 feet, it is plausible that a design with length 71 feet may also perform well, although it does not strictly match the condition.

A simple method for flexible interpretation, implemented in LEM3, is to generate *s*% individuals strictly satisfying the rule conditions (*s* is a parameter), and (100-s)% individuals with values whose probabilities linearly decrease with distance from the condition border (e.g., *s*=95%).

3.2. Probe and Search Locally Actions

The *probe* action executes conventional "Darwinian-type" operators such as mutation and crossover. The crossover creates two new individuals from two parents by exchanging their values of the first k attributes, where k is randomly chosen. Because the representation of variables depends on their type, the crossover and mutation operators need to support all attribute types available in LEM3. This is done by performing these operations using semantic information about the attribute types, not necessarily their internal representation (as, for example, in genetic algorithms that operate on bits).

The *search locally* operator is applied to improve the current solutions by searching their neighborhoods. For example, for continuous attributes, the program may apply a gradient method. The LEM3 design allows users to attach an external program to run a local search method. This feature is currently under development in the program.

3.3. Adjust Representation Action

The *Adjust representation* action modifies the representation space of solutions to make it more suitable for hypothesis generation, using operators such as adjusting domains of variables, removing variables irrelevant to the optimization problem, and creating new, more relevant variables. Currently, only the operator that changes the discretization level of attributes has been implemented in LEM3. Specifically, LEM3 employs *adaptive anchoring discretization* that splits dynamically selected ranges or subranges of continuous attributes into more discrete units. The method starts with a very rough discretization, and then increases the precision of numeric attributes in the most promising intervals, as indicated by the best individuals.

4. Action Selection Module

The Action Selection Module (a.k.a. Control Module) uses an *Action Profiling Function* (APF) to control which actions are applied at a given step of evolution. Initially, by default, the Control Module selects the "Learn and Instantiate" action. When after a number of iterations, insufficient progress is observed, LEM3 applies the "Probe" action to increase diversity of solutions in the population. If after several repetitions there is still no progress, representation of individuals is adjusted. Then, if after a number of repetitions the program still does not make progress, it may mean that an optimum has been reached, local or global. If the termination condition is not satisfied, evolution is restarted by employing the "Randomize" action.

An important novelty of LEM3 is that it allows a parallel execution of these actions. 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 mutation, and 5 by random generation. Numbers of individuals created by different actions can be adjusted based on their performance.

5. Heat Exchanger Optimization

Using the LEM methodology, we developed two specialized systems, for heat exchanger optimization, ISHED for optimizing evaporators and ISCOD for optimizing condensers [2]. In both systems, the objective is to optimize the connections among the heat exchanger tubes, so that maximum heat transfer can occur.

The heat exchangers are subject to a variety of constraints, resulting in a very large number of feasible designs, scattered throughout intractably large representation spaces.

ISHED and ISCOD are equipped with learning and probing operators tailored to the given application. Experiments consistently showed that both systems are able to create designs that adapt to varying environmental conditions and technical constraints. They are able to evolve designs that perform on a par with, or better than the best human designs, according to experts. In problems with highly uneven airflows, the ISHED designs were evaluated by experts as superior to the best human designs.

6. Experimental Results from LEM3

LEM3 was applied to selected benchmark problems of optimizing Rastrigin, Griewangk and Rosenbrock functions with numbers of variables ranging between 2 and 1000. For comparison, EA, a conventional Darwinian-type program [3], was applied to the same problems. We also compared LEM3's results with published results by Estimation of Distribution Algorithms (EDAs), and a Cultural Algorithms (CAs) applied to these problems.

The results of 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 number of fitness evaluations required by EA to those required by LEM3 to achieve the same result.

Table1. Average speedup of LEM3 over EA in optimizing Rosenbrock, Griewangk and Rastrigin functions on numbers of variables ranging from 100 to 1000. Parameter δ was set to 0.1 and 0.01, and the table presents the average speedup.

No of variables	**100**	**200**	**300**	**400**	**500**
Speedup LEM3/EA	10.7	15	16.8	17.8	17.2
No of variables	**600**	**700**	**800**	**900**	**1000**
Speedup LEM3/EA	16.7	19	16.6	17.2	18

As one can see, the speedup of LEM3 over EA ranges between 10 and 18, and has a tendency to increase with the number of function variables. The stopping criterion for EA and LEM3 was finding solutions with a defined improvement of the fitness value over the starting population, called a δ-close solution [11]. When δ=0.1, the best solution at the end of the process must be at least 10 times closer to the optimum than the best solution in the starting population. Each experiment was repeated 10 times with different starting populations, which were the same for both programs. LEM3 used default parameters, without tuning to these particular functions.

A comparison of LEM3 results with the best results from the Cultural Algorithm program on the optimization of Rastrigin, Griewangk, and Rosenbrock functions that were reported in [10] for 5, 3, and 2 variables indicated a significant speedup by LEM3. LEM3 required on average 728 times fewer fitness evaluations on the Rastrigin function, 53 times fewer on the Griewangk function, and 243 times fewer on the Rosenbrock function. 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.

Comparing LEM3's results with the best results from the EDA implementations on Griewangk and Rosenbrock functions of 10 and 50 variables reported in [1] also shows LEM3's advantage. LEM3 required on average 142 and 66 times fewer fitness evaluations for optimizing the Griewangk and Rosenbrock functions, 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.

7. Related research

The LEM3 program follows earlier implementations, LEM2 and LEM1, that used earlier versions of AQ learning. An implementation of Learnable Evolution Model for multi-objective optimization, LEMMO, developed by another research group, is based on rules derived from decision trees learned by the C4.5 program, and was applied successfully to a water quality optimization problem [5].

The evolutionary methods that seem to be close in spirit to LEM are cultural algorithms [10] that perform an optimization 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.

Estimation of Distribution Algorithms (EDAs) use statistical inference, usually Bayesian or Gaussian networks, to estimate distributions of high-performing individuals selected from one population [1], [8]. LEM 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 during the learning process itself, through learning significance-based descriptions, while EDA uses it solely for selecting individuals.

Another form of non-Darwinian method is performed by memetic algorithms that combine conventional evolutionary computations with local search methods [4], [9]. A local search mode is included as one of the possible actions in LEM3.

8. Conclusion

LEM3 is the most recent and most advanced implementation of the Learnable Evolution Model. In some aspects, the algorithms implemented in LEM3 go beyond the methodology described in [6], for example, novel aspects include the introduction of the Action Profiling Function and new instantiation algorithms.

LEM3 is highly scalable, and experiments have confirmed that LEM3 can serve as a powerful optimization system for problems with hundreds or more controllable variables. In every experiment we conducted, it outperformed other evolutionary computation methods in terms of evolution length, sometimes more than an order of magnitude. LEM3 is also particularly suitable for optimization problems with many different types of attributes (a feature that was not demonstrated in the presented experiments).

Acknowledgments

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.

References

[1] Bengoextea, E., Miquelez, T., Larranaga, P., and Lozano, J.A., "Experimental Results in Function Optimization with EDAs in Continuous Domain", In P. Larranaga and J.A. Lozano (Eds.) *Estimation of Distribution Algorithms*, 2002.

[2] 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, 2004.

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

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

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

[6] Michalski, R.S., "LEARNABLE EVOLUTION MODEL: Evolutionary Processes Guided by Machine Learning", *Machine Learning*, 38, 2000.

[7] Michalski, R.S., "ATTRIBUTIONAL CALCULUS: A Logic and Representation Language for Natural Induction", *Reports of the Machine Learning and Inference Laboratory*, MLI 04-2, 2004.

[8] Miquelez, T., Bengoetxea, E., and Larranaga, P., "Evolutionary Computation Based on Bayesian Classifiers", *International Journal of Applied Mathematics and Computer Science*, 14, 2004.

[9] Moscato, P., "On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms", *Caltech Concurrent Comput. Program*, C3P Report 826, 1989.

[10] 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.

[11] Wojtusiak, J. and Michalski, R. S., "The LEM3 System for Non-Darwinian Evolutionary Computation and Its Testing on Complex Function Optimization Problems", *Proceedings of Genetic and Evolutionary Computation Conf.*, GECCO, 2006.

[12] Wojtusiak, J., Michalski, R.S., Kaufman, K.A., and Pietrzykowski, J., "The AQ21 Natural Induction Program for Pattern Discovery: Initial Version and Ins Novel Features", *Proceedings of The 18th IEEE International Conference in Tools with Artificial Intelligence*, ICTAI 06', 2006.