BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES
Vol. 54, No. 4, 2006

Optimizing complex systems by intelligent evolution: the LEMd method and case study

R.S. MICHALSKI*

Machine Learning and Inference Laboratory, George Mason University, M.S.T.C8, Fairfax, VA, 22030, USA
Institute of Computer Science, Polish Academy of Sciences, 21 Ordona St., 01-237 Warszawa, Poland

Abstract. Most methods of evolutionary computation follow a Darwinian-type model that proceeds through random mutations or recombinations of the genetic material[1] and natural selection of individuals carried out according to the principle of the survival of the fittest. In such a model, the creation of new individuals is not guided by any reasoning process or "external mind", but rather by random or semi-random changes. Recently, a new, non-Darwinian approach to evolutionary computation has been proposed, called Learnable Evolution Model (LEM), in which the evolutionary process is guided by computational intelligence. In LEM, a new way of creating individuals is proposed, namely, by hypothesis formation and instantiation. In numerous experiments, LEM has consistently and significantly outperformed compared conventional Darwinian-type algorithms in terms of the evolution length (the number of fitness evaluations) in solving complex function optimization problems. Based on the LEM ideas, we developed a method, called LEMd, which is tailored to problems of optimizing very complex engineering systems. This article provides a brief description of LEMd and its application to the development of a specialized system, ISHED, for the optimization of evaporator designs in cooling systems. According to experts in cooling systems, ISHED-developed designs have matched or outperformed the best human designs. These results and those from the experimental testing of learnable evolution on problems with hundreds of variables suggest that LEMd may be an attractive new tool for optimizing very complex engineering systems.

Key words: engineering design optimization, evolutionary computation, function optimization, learnable evolution model, machine learning, genetic algorithms.

1. Introduction

A popular method for optimizing complex engineering designs is to apply evolutionary computation, because it can be implemented relatively easily and is applicable to a very wide range of problems. Most methods of evolutionary computation are based on the Darwinian model of evolution in which new individuals (in this case, designs) are generated by semi-random change operators, such as mutations and/or recombinations, and are selected for the next generation according to some variant of the survival of the fittest principle. The Darwinian-type of evolutionary computation is not guided by any "external mind" [1], and does not involve much domain knowledge. Although it has found many engineering applications (e.g., [2–6]), it is not very efficient, because it is basically an unguided trial-and -error search.

This paper briefly describes a new approach to engineering applications of evolutionary computation that is based on Learnable Evolution Model (LEM), a form of evolutionary process that is guided by computational intelligence, specifically, by machine learning [7]. In contrast to Darwinian-type evolutionary algorithms that model biological evolution, LEM attempts to model intellectual evolution – an evolution of human artifacts [7]. In such intellectual evolution, new generations of solutions (designs, systems, artifacts, etc.) are created through a reasoning process that involves an analysis of advantages and disadvantages of past solutions. This analysis is done by engineers or designers, and thus engages human intelligence.

In LEM, such an analysis is done by a machine learning program that looks at the high-fitness and low-fitness individuals (solutions), and applies a machine learning program to create a general hypothesis that differentiates between the two groups. A new population of solutions is created by instantiating the hypothesis in different ways. Thus, unlike Darwinian-type evolutionary algorithms that create new candidate solutions by random mutations and/or recombinations, the central new idea in LEM is to generate new solutions by hypothesis formation and instantiation.

In all experiments performed so far, LEM has significantly and consistently outperformed all tested Darwinian-type algorithms in terms of evolution length, measured by the number of fitness function evaluations needed to achieve a desired solution. Moreover, the LEM advantage has increased with the complexity of the problem, as measured by the number of variables to optimize [8]. These results suggest that LEM may be particularly useful for optimizing very complex designs, with hundreds of controllable variables.

The following sections briefly describe LEMd, a method that tailors LEM to problems of optimizing complex designs,

*e-mail: michalski@mli.gmu.edu

[1]The idea that small random genetic mutations are fully sufficient for producing innovation observed in nature has been recently questioned by some scientists. To resolve this problem in Darwinian model of evolution, a new theory was proposed, called "facilitated variation", which treats the individual organism not as a passive object of natural selection but an active player in evolution [1].

and then describes a specialized program, ISHED, that applies LEMd to the optimization of evaporators in cooling systems. The performance of ISHED is illustrated by an excerpt from a log of a program run.

2. An overview of the LEMd method

2.1. Underlying assumptions. The hypothesis generation and instantiation in LEM can be viewed as "intelligent operators" because they do not make purely random or semi-random design changes, but rather innovate designs using hypotheses created by machine learning. The application of these operators has proven to shorten the evolution length in every experiment that compared LEM with Darwinian-type of evolutionary computation [8]. It should be noted, however, that these operators are more computationally complex than conventional change operators, such as mutations and recombinations. Thus, their application in evolutionary computation creates a tradeoff between the evolution length and the complexity of computation.

Because of this tradeoff, and because neither conventional nor machine learning-based operators can guarantee that the evolutionary process will always converge to the global optimum without backtracking, it is desirable to combine the two types of operators in a way that will lead to the maximum efficiency of the evolutionary design process. The above objective underlies the LEMd method for applying learnable evolution to the optimization of complex engineering designs.

To achieve this objective, LEMd integrates two modes of operation, Learning Mode and Probing Mode. Learning Mode creates new designs through hypothesis generation and instantiation, while Probing Mode creates them by applying expert-suggested design modification (DM) operators tailored to the specific design problem. Expert-suggested operators are used because applying random mutations and recombinations to complex systems will hardly produce improvements over any reasonable amount of time. DM operators represent experts' knowledge as to the types of system modifications that are meaningful and may plausibly improve the design.

2.2. Description of the LEMd method. An underlying assumption for applying LEMd, as for any evolutionary computation method, is that there exists a mechanism for evaluating the quality of candidate designs, for example, a simulator of the system being designed. Such an evaluation does not have to assign a specific quality value to a design, but must be able at least to rank the designs in the population of candidates.

A flowchart of the LEMd method is shown in Fig. 1. The first step of the method is to create an initial population of designs, which may include already existing designs, designs obtained through a previous execution of LEMd, randomly generated designs, or a combination of the above. The next step is to select a mode of operation, either Learning Mode or Probing Mode. The choice of the initial mode is not particularly important; it is made to best suit a given application domain.

Learning mode. In this mode, at each step of evolution, a population of designs is divided into three disjoint groups based on their fitness: H-group containing designs that score highest on the fitness evaluation function, L-group containing designs that score lowest, and the rest. The partition of the population into these groups can be done using different methods. One method, fitness-based, selects for the H-group designs with fitness above a given upper threshold, and for the L-group designs with fitness below a given lower threshold. The second method, population-based, selects high-scoring designs for the H-group so that they constitute a predefined percentage of the designs, and the low-scoring designs for the L-group so that they also constitute a predefined percentage of the designs [7]. The collection of designs from which H- and L-groups are selected can be the current population, or a union of the current and selected past populations. The latter is another distinguishing feature of LEMd, as compared to Darwinian-type evolutionary computation, as it allows a new population to be generated on the basis not only of the current population, but also of past populations.

The selected H-group and L-group are then supplied to a learning program that induces a general hypothesis distinguishing between these groups. For this purpose, LEMd employs an AQ-type rule learner [9, 10], which is particularly suitable for learnable evolution due to the use of attributional calculus as the hypothesis representation language [7,11]. The learned hypothesis is in the form of a set of attributional rules [11]. An attributional rule is a logical conjunction of attributional conditions.

For illustration, here are examples of such conditions with their interpretations:

[length = 2m]	(the length of the entity is 2 m)
[color = red or blue]	(the color is red or blue)
[weight = 5 . . . 8 kg]	(the weight is between 5 and 8 kg)
[width & height < 1m]	(both the width and the height are below 1m)
[width > height]	(the width is greater than height)
[count(X, EQ 2) = 3]	(the number of attributes from X that have value 2 is 3)
[climate: warm & humid]	(the climate is warm and humid)

The last example illustrates the use of a compound attribute "climate" whose values are taken from the Cartesian product of the domains of constituent attributes "temperature" and "humidity" [11]. As one can see from the examples above, attributional conditions have greater expressive power than the conditions typically used in decision rules, which are limited to the form «attribute-relation-value». They are, however, easy to understand and interpret in natural language. Because rules are conjunctions of such conditions, they are also easy to understand and interpret. Attributional rules can also be easily instantiated into examples (designs), especially in the case of rules with the first four types of conditions above. This feature is particularly important for LEMd, because in Learning Mode new designs are created by instantiating such rules.

Each attributional rule in a hypothesis that differentiates the H-group from the L-group describes a subset of the search space that likely contains an optimum. Because such a hypoth-

esis may consist of a number of rules, the LEMd method may simultaneously pursue different optima in a multimodal fitness function landscape.

After a hypothesis is learned, its rules are instantiated in different ways to create candidate designs that satisfy the given constrains. The newly created designs compete with each other in terms of their fitness with previously generated designs for inclusion in the new parent population. The fitness is determined by a design evaluation method, which will typically involve running a design simulator.

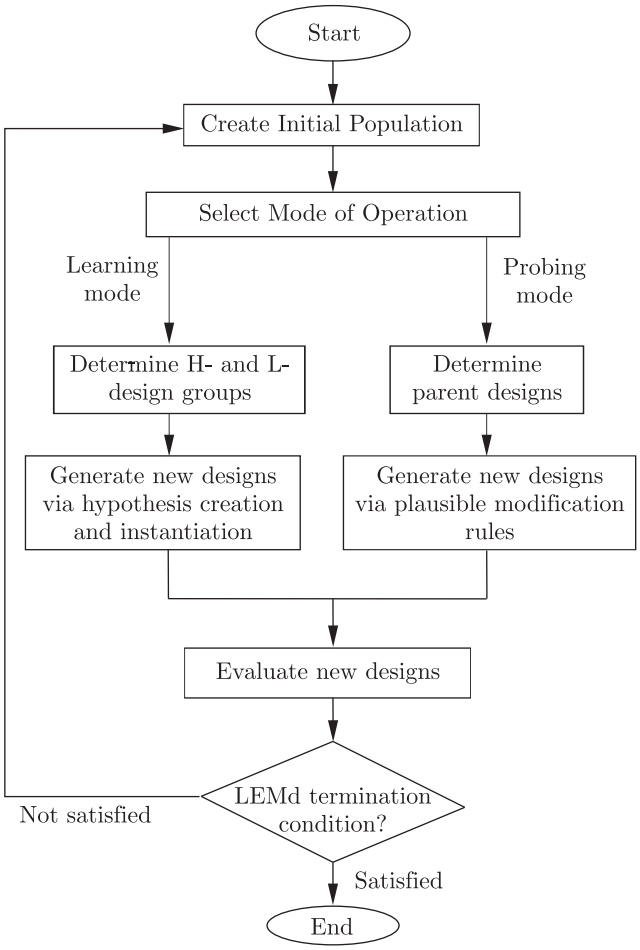

Fig. 1. A flowchart of the LEMd method

Probing mode. In this mode, LEMd applies problem-dependent design modification (DM) operators that make feasible and plausibly useful changes to the proposed designs, according to expert-provided domain knowledge (see Section 3.2). Before DM operators are applied to a population, the population is transformed into a new population by selecting each design with a probability proportional to its fitness (proportional selection). Thus, high fitness designs may be selected several times to be parent designs for the next generation, and some low fitness designs may not be selected at all.

Mode and LEMd termination criteria. Each mode of LEMd runs until a mode termination criterion is met, and then control switches to the other mode. The mode termination criterion is satisfied if there is insufficient improvement to the fitness of the best candidate design in the population or to the average fitness of the population as a whole after a given number of iterations. Parameters of the mode termination criterion have default values that can be changed by the user. LEMd terminates when the obtained best design is satisfactory or when it completes an assumed number of fitness evaluations or, more generally, exhausts the allocated computational resources.

The first step in applying LEMd to a specific design problem is to define a suitable representation for designs. Such a representation needs to be in the form of a list of attribute values. In LEMd, the attributes can be of different types, such as nominal (the domain is an unordered set, e.g., the type of material), structured (the domain is a hierarchically ordered set, e.g., a hierarchy of shapes), rank (the domain is discrete, finite and totally ordered, e.g., "size" discretized into small, medium and large), cyclic (like rank, but the domain is cyclically ordered; e.g., seasons of the year), interval (the domain is a measurement with an arbitrary zero, e.g. "temperature in Celsius degrees"), ratio (measurements for which "0" is not arbitrary, such as "weight"), and absolute (numerical counts of distinct elements). LEMd's capability to use different types of attributes in its evolutionary process allows one to tailor it to a wide range of applications, without need of defining ad-hoc encoding.

The next step is to specify and implement plausible design modification operators and constraints imposed on the acceptable designs. Subsequent steps are to create an initial, sufficiently diverse population of designs and to define termination criteria for both probing and learning modes of operation, and for the entire LEMd run.

Finally, a hypothesis instantiation algorithm must be developed that will produce designs satisfying design constraints. The process of instantiating the learned hypothesis, which is in the form of a set of attributional rules [11], proceeds as follows:

1. Rules are arranged in descending order of their significance, which is defined according to the problem at hand. The default measure of significance is the sum of the fitness values of the designs in the H-group that satisfy that rule. Each rule is instantiated in different ways to produce different designs. A specific design is created by assigning values to attributes in a rule that satisfy that rule and the design constraints. Attributes not present in the rule are assigned values according to some value-assigning method. The default method is to assign to such attributes values from a randomly selected H-group design. The number of designs produced from a rule is proportional to the rule's significance, but not smaller than i-min, a program parameter. This method is called a proportional instantiation.
2. The reason for the i-min parameter is to assure that even low significant rules will have a chance to produce an offspring. Because each rule delineates a subarea in the search space, i-min enforces parallel exploration in several subareas of the search space. This feature is particularly important for optimization problems with a multi-modal fitness landscape.

3. The total number of new designs generated is Population-size - elite-size, where elite-size is a parameter controlling the size of the elite group, a collection of the best designs found so far that is automatically accepted into the new parent population.

New designs created by either Learning or Probing Mode are evaluated to determine their fitness. This evaluation can be done by running a design simulator, by a subjective judgment of an expert or group of experts, or by some other method. Subsequently, the LEMd termination condition is tested, and if satisfied, the program ends; otherwise, the population of designs with their fitness values is passed to the "Select Mode of Operation" module to start the next iteration.

If the control passes to Probing Mode, a new parent population is assembled from the best past and newly generated designs using one of the selection methods developed in the field of evolutionary computation (e.g., proportional selection, tournament, etc.).

3. An example of LEMd application: optimizing evaporator designs

3.1. Problem description. This section briefly describes an application of LEMd to the development of a specialized system, ISHED, for optimizing evaporators in cooling systems. This is a complex engineering problem of great practical importance because of the widespread use of evaporators in air conditioners and refrigerators. This study has been conducted in consultation with experts on heat exchanger design, dr. Piotr Domanski and his collaborators from the National Institute of Standards and Technology (NIST).

To explain the problem, let us briefly describe how an evaporator works in an air conditioning unit [12]. The refrigerant flows in a loop through an air conditioner. It enters, in cool liquid form, an array of parallel tubes that constitutes an evaporator. When it comes into contact with the warmer interior air that is being pushed through the evaporator, it heats up and evaporates, but the air cools down. Figure 2 shows an example of an evaporator with tubes arranged into 16 columns and 3 rows.

The configuration of the tubes through which refrigerant flows affects both the temperature of the refrigerant when it reaches any given tube, and the air temperature after the air passes over the tube. The amount of heat transfer the air conditioner provides depends on the heat transfer of its evaporator's tubes and the efficiency of its condenser in transferring heat from the refrigerant to the outside air. It also depends on the distribution of the velocities of the air passing through the evaporator (see the graph in Fig. 2).

The problem of optimizing an evaporator lies in how to connect the tubes and how to direct the flow of the refrigerant through them so that for the given technical and environmental conditions, the capacity of the evaporator will be maximized. The capacity is measured by the amount of heat transfer (in kW) per unit of time. Technical conditions include the size of the evaporator (the number of tubes and their length and diameter), material of the tubes, and the type of refrigerant. Environmental conditions include the inside and outside temperatures and the airflow distribution. The optimization space is defined by the variables defining how tubes are connected in the evaporator between inlets (input tubes) and outlets (output tubes). In a moderate-sized evaporator, such as the one in Fig. 2, each of the 48 tubes can in theory receive its refrigerant from any of the other 47 tubes, or from the flow of refrigerant into the evaporator. Thus, a naive search of the design space would have to consider 48^{48} potential designs. The vast majority of these designs are infeasible because of the problem constraints, but even if it were possible to search only through potentially feasible designs, the search space would still be orders of magnitude too large for any exhaustive search.

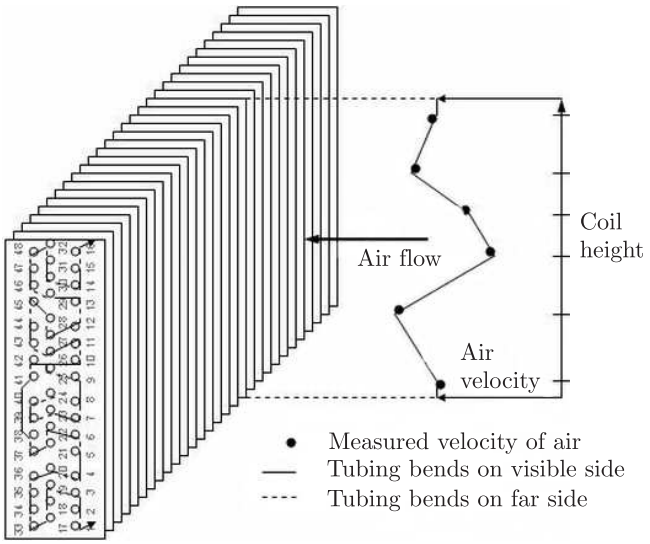

Fig. 2. An example of a 16×3 evaporator

In selecting a path for the refrigerant to flow through an evaporator of a given size, real-world constraints limit the set of feasible designs. Based on discussions with a domain expert, several constraints were built into the ISHED program.

3.2. The ISHED system. As mentioned earlier, ISHED applies LEMd to the specific problem of optimizing evaporators in air-conditioning units. The initial population is of designs may be partially specified by an expert, and the rest is generated randomly. The proportion of designs to be generated by the random process with specific numbers of inlets and outlets is specified within the ISHED program based on expert advice, and is dependent on the total number of tubes in the evaporator being optimized (smaller evaporators usually function better with fewer inlets and outlets).

The structure of an evaporator is represented by a sequence of integer values. Each value indicates the number of the tube that is the source of each tube's refrigerant (tubes are numbered left-to-right, starting with the first row). For the attributional representation used by the learning program, the type of a tube is encoded in the sequence in the following way:

1. If the tube in the kth position is an inlet, it is indicated by symbol I.

2. Other types of tubes are indicated by the number of occurrences of the value k in the string. If k doesn't appear at all, the tube is an outlet. If k appears twice, the tube is a split location. If k appears exactly once, the tube is a regular interior one.

For example, consider the following structure:
<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>

In this structure, the tube 17 provides refrigerant for tube 1; tube 1 provides refrigerant for tube 2, etc. Tube 16 is the inlet ("I" in the 16th position). Tube 10 is the outlet (there are no 10s in the string). There are no splits in the structure (because all other numbers between 1 and 48 appear exactly once). All other tubes are regular ones.

In experiments, the initial population of designs (defining structures of evaporators) was usually a combination of existing designs and random designs. The quality of designs, measured by their capacity, is determined by a evaporator simulator developed at the National Institute of Standards and Technology [13]. The capacity of a design indicates its energy efficiency.

Running the simulator is the most time-consuming part of the evolutionary design process. A single evaluation may take on the order of a few seconds, which is a relatively long time, given that a typical optimization process may take tens of thousands or more evaluations. Therefore, the evolutionary speedup provided by learnable evolution [8] is particularly advantageous for this application.

For Probing Mode, the following design modification operators have been developed on the basis of an expert's advice:

– FORK adds a fork in a path.
– BREAK breaks a path at a given point, creating a new inlet and a new outlet location at the tubes following and preceding the break, respectively.
– COMBINE combines two separate unforked paths into a single, forked one.
– INSERT combines two separate unforked paths into a single, unforked one.
– MOVE-FORK moves a fork in a path up or down to an earlier or later point on the path.
– SWAP interchanges the position of two successive tubes in the refrigerant path.
– INTERCROSS, given break points on two separate refrigerant paths, exchanges the portions of the paths following the breakpoints; this operator is analogous to a genetic crossover.
– NEW-CONNECTION that assigns a new source to the operand tube.

To illustrate an operator, Fig. 3 presents graphically the function of the COMBINE operator.

In Learning Mode, given a population of designs and their evaluations, ISHED determines the H-group and the L- group. This is done using a fitness-based method that arranges design fitnesses from the highest value to the lowest (the range of values is called the fitness range), and selects a given percentage of the top of the fitness range (specified by a program parameter) for the H-group, and a given percentage of the bottom of the fitness range (defined by a parameter) for the L-group [7].

In ISHED, the H- and L-groups are passed as positive and negative examples to the AQ19 rule learning program [9]. The program outputs a general hypothesis distinguishing between these groups that is in the form of a set of attributional rules.

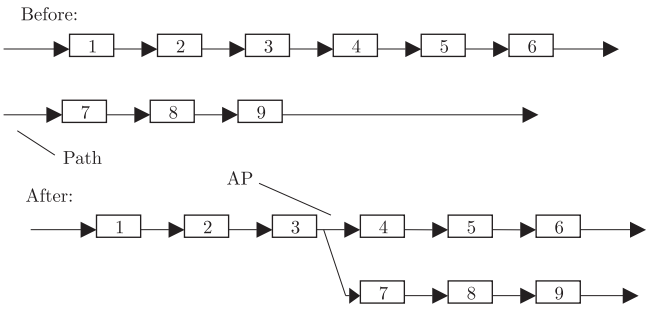

Fig. 3. The ISHED COMBINE(7,4) DM operator

Each rule in the hypothesis is then instantiated to a set of specific designs by randomly assigning to each attribute different values satisfying the rule. The attributes not present in the rule are assigned values according to the default LEMd method (values from a randomly selected H-group design). The number of designs created from each rule is determined by proportional instantiation (the number of created designs is proportional to the rule significance). The total number of designs generated by hypothesis instantiation is the assumed population size minus the size of the H-group, because the designs in latter group are considered elite designs, and automatically included as candidates for the new population.

The final step creates feasible designs that satisfy the constraints on a heat exchanger design. This process takes into consideration the geometry of the inlet and outlet points. The need for satisfying the constraints may cause a decrease of diversity in the instantiated designs, which in turn may decrease Learning Mode performance. When this happens, ISHED switches to Probing Mode.

To increase the diversity of a population of designs in a tightly constrained domain, given a set of inlet and outlet points, ISHED determines the physical relationships among these points in the design (i.e., which are to the left of, to the right of, above, below, in the same column, or in the same row as others), and selects an instantiation strategy accordingly. Generally, a divide-and-conquer method is used, in which the evaporator is divided into regions based on the locations of the key tubes.

Both evolutionary modes use elitist strategy, that is, keep track of the best-performing designs. In Probing Mode, the elite-size best performing individuals thus far automatically become the basis for the first members of the new population. In Learning Mode, as mentioned earlier, all members of the H-group comprise the elite group that is always included in the next generation's population.

Exchanger Size: 16 x 3
Population Size: 15 Generations: 40
Mode Persistence: Probing-persistence=2 Learning-persistence =1
Operator Persistence: 5 **(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: **Capacity = 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** **(2)**
and 13 others
Selected Members: 3, 2, 3, 7, 9, 3, 9, ... **(3)**
Operations: NS(23, 39), SWAP(8), SWAP(28), ..., SWAP(29), SWAP(25), SWAP(1) **(4)**

Below is one of the structures created by the application of a DM operator in Probing 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** **(5)**
and 14 others.
Selected Members: 6, 15, 11, 3, 13, 1, ...
. . . . . .
Probing Mode runs for the next 4 generations, and then Learning Mode executed:
Generation 5: Learning mode
Learned 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) **(6)**
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** **(7)**
. . . . . .
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: **Capacity=5.5387** **(8)**
and 14 others
Selected Members: 11, 4, 4, 13, 15, 10, 12, 13, 15, 15, 12, 2, 3, 5, 10.
. . . . . .
ISHED continues to evolve structures, and finally 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** **(9)**

Fig. 4. An excerpt from the log of an ISHED run

3.3. An example of an ISHED design optimization process. Testing experiments with ISHED were performed with different settings of design parameters that define the type of refrigerant, evaporator shape and size, and airflow patterns (uniform or non-uniform). Industrially available air conditioning systems typically perform very efficiently as long as the airflow is fairly uniform. When the airflow is not uniform, their efficiency drops off significantly. The side of the unit over which more air flows has a heavier cooling burden, so to achieve the best performance it needs to carry more and colder refrigerant. This factor has not been usually taken into consideration by manufacturers of evaporators; therefore, in the case of non-uniform flow, commercial evaporators tend not to be very efficient.

The initial experiments with ISHED concerned designing evaporators of common sizes under the assumption of a fairly uniform airflow pattern. In these experiments, ISHED produced designs that were comparable to the industry standard. When ISHED worked for many generations in Probing Mode, some of its designs would tend to have chaotic intertube connections. This was because the simulator did not sufficiently reflect the detrimental effect such connections have.

The above problem was partially alleviated by tightening restrictions on the length of permissible tube-to-tube connec-

tions, and by using existing tools for smoothing some of the connections without significantly decreasing the evaporator's capacity estimated by the simulator [12]. To illustrate ISHED's performance, Figure 4 presents an abbreviated log of one run.

This run was executed in a verbose mode, which means that its log details every design tested, every operator applied, and every hypothesis learned. The figure only shows a very small sample of the full output that even in this small experiment would take up many pages. Numbers in parentheses on the right hand side are pointers to the explanations provided in the text below. In addition to these explanations, some comments to the log were added in italics.

Each structure shown in Fig. 4 is accompanied by its capacity estimated by the simulator. The first part of the log, denoted by (1), provides a summary of the ISHED parameters used in the given run. In this run, ISHED was creating evaporator designs consisting of 3 rows of 16 tubes. Over 50 generations, it evolved a population of 15 designs.

The Mode Persistence parameters, Learning-persistence and Probing-Persistence instructed ISHED to switch from Probing to Learning Mode after two consecutive generations failed to improve the best design in the population, and to switch back to Probing Mode after one Learning Mode generation did not make such an improvement. The Operator Persistence parameter was set to instruct ISHED to apply a specific design modification operator in Probing Mode up to 5 times (with different randomly selected operands) before giving up on this operator, and choosing another design modification operator.

The log shows designs in the initial population, which was generated randomly in this case, and the capacity of the designs estimated by the simulator. At the step denoted by (2), two of the 15 generated individuals are shown, specifically, the third and eighth design in the population, whose capacities were determined to be 5.5376 and 5.2009, respectively. From this population, the seeds for the next generation, with the exception of the first one, were selected probabilistically, based upon the capacities of the individuals of the current population. The first design was included in the elite, because it was the best design obtained so far.

At (3), the log shows that the first seven of the fifteen designs in Generation 1 would be built from individuals 0.3, 0.2, 0.3, 0.7, 0.9, 0.3, 0.9. As shown at (4), each seed for the new population then had a design modifying operator applied to it as follows: Individual 1.1 was created by applying operator NS(23,39) (change the source of refrigerant for tube 23 from whatever it was to tube 39) to individual 0.3; individual 1.2 was created by applying operator SWAP(8) (swap the positions of the two tubes preceding tube 8) to individual 0.2; etc. It is shown at (5) that individual 1.13, generated by applying operator SWAP(29) to design 0.8, had a capacity of 5.2093.

After progress had stalled, ISHED switched to Learning Mode, and discovered a rule (6) that indicated a pattern in which high-performing designs consisted of an outlet at position 10, an inlet at position 16, and interior tubes at all other positions. The learned rules were instantiated to become members of the new population, such as Structure 5.1 (7).

The run continued in this way, and at the halfway point, there was little progress in evolving better designs (8). But by the run's end (9), a significant leap in design quality has occurred. While the best design in the initial population had a capacity of 5.5376 kW, the best design after 40 generations in running ISHED had a capacity of 6.3686 kW, which represents a 15% improvement. This result illustrates the ISHED capability for improving evaporator designs.

3.4. Summary of the experiments. Experiments with ISHED have proven its ability to improve the initial designs, sometimes very significantly. With few preprogrammed assumptions, the program was able to evolve designs consistent with the given technical and environmental conditions and constraints. When faced with uniform airflows, designs roughly symmetric were favored, in the case of non-uniform airflow, the designs were less symmetric to compensate for such a flow. As a result, ISHED designs for non-uniform flows were evaluated by experts as superior to the best human designs.

4. Related research

To the author's best knowledge, LEMd represents a novel methodology for optimizing complex engineering systems through evolutionary computation. It is based on Learnable Evolution Model (LEM) that attempts to model evolution guided by an "intelligent mind" rather than evolution proceeding randomly or semi-randomly. Somewhat in sprit to LEMd are cultural evolution algorithms that execute a process of dual inheritance (e.g. [14,15]). Unlike LEMd, cultural evolution works at two levels, a "micro-evolutionary level" that involves individuals described by traits and modified by conventional evolutionary operators, and a "macro-evolutionary" level, in which individuals generate "mappa" representing generalized beliefs that are used to modify the performance of individuals in the population. LEMd is significantly different from cultural evolution algorithms as it integrates two modes of evoluationary computation, learning and probing, and its learning mode is different because it uses both high-performing and low-performing individuals. It also employs a completely different method of learning and representing the results.

In its general objective, somewhat related to LEMd is a "generic" evolutionary design system GADES [3,4] that aims to serve as a method for solving diverse design problems. The system employs a conventional Darwinian evolutionary method, but allows the user to specify the nature of the genetic operators to apply and the representation of individuals in the population. LEMd also has such capabilities, but in addition includes an entirely new form of generating individuals provided by Learning Mode and by integrating it with Probing Mode.

Another approach that extends the traditional Darwinian approach has been implemented in GADO [16], which also aims at optimizing complex engineering systems. It differs from conventional genetic algorithms by using five different crossover operators, three of which, crossover, double line crossover and guided crossover, were introduced in GADO. However, unlike LEMd, GADO does not have the ability to

guide evolution by general hypotheses created by a learning system.

Another extension of conventional Darwinian-type algorithms is represented by memetic algorithms (e.g., [17]). These are global-local hybrid algorithms that combine conventional evolutionary algorithms with local search methods to improve solutions. They have been applied to many real-world problems and have been the subject of intensive studies. They differ significantly from LEMd in that they do not apply learning methods to guide evolutionary computation.

5. Conclusions

The presented LEMd method is based on the Learnable Evolution Model, a non-Darwinian form of evolutionary computation. Unlike most methods of evolutionary computation that draw inspiration from natural evolution (the evolution of living organisms), LEM attempts to model intellectual evolution that has governed the development of human artifacts, such as automobiles, computers, airplanes, etc. In such evolution, the generation of new populations of artifacts results from the analysis of the advantages and disadvantages of past populations by a human mind.

The evolutionary process performed by LEMd is called "intelligent" because it is guided by computational intelligence, partially by machine learning and partially by expert domain knowledge. Experiments with ISHED applying LEMd to the problem of optimizing evaporators in heat exchangers indicate that it offers a new and effective tool for optimizing very complex systems when neither conventional optimization methods nor Darwinian-type evolutionary algorithms are very effective.

ISHED represents an initial implementation of the LEMd method for the task of optimizing evaporators in heat exchangers, and has shown promising results. This first implementation suffers, however, from several limitations. It is not capable of an orderly instantiation of all possible design specifications consisting of three or more inlets or outlets. The complexity of the instantiation process makes it currently difficult to inject a desirable amount of diversity into the process, although a new version of ISHED under development indicates strong improvement in both these areas. Experiments revealed also an ISHED weakness regarding an occasional lack of evolutionary advancement in seeded populations. There are several ways to work around the problem, such as making short, sequential runs of ISHED that build upon each other's results. Further research will explore more thoroughly the nature of the experienced problems and their possible solutions.

The LEMd method is at an early stage of development, and poses many interesting new research problems. They include a theoretical and experimental investigation of the method, testing it in different application domains, and the development of efficient methods for applying LEMd to optimization problems with very complex constraints and dynamic landscapes.

Despite its limitations, LEMd offers a new method for optimizing complex engineering systems with a high number of controllable discrete and/or continuous parameters (on the order of hundreds or more), and is particularly attractive when design evaluation is costly and/or time consuming. Because of its generality, LEMd can be applied to a wide range of design optimization problems.

Acknowledgments. This paper is a significantly extended version of the invited presentation at the 12th International Workshop of the European Group for Intelligent Computing in Engineering, Radziejowice, Poland, June 17-19, 2005, organized by Martina Schnellenbach-Held from the University of Duisburg-Essen in Germany, and Adam Borkowski from the Institute of Fundamental Problems of Technology of the Polish Academy of Sciences in Warsaw, Poland.

The author expresses his gratitude to Ken Kaufman for his collaboration on the development of the LEMd method and the implementation and experiments with the ISHED system. He also provided useful comments on this paper. The author is also grateful to Janusz Wojtusiak for his review of the paper and research on the development of LEM3, the newest implementation of the learnable evolution model.

Thanks go also to Piotr Domanski from the National Institute of Standards and Technology for introducing the author and his collaborators to the area of optimization of heat exchangers and for providing a simulator for evaluating evaporator designs used in ISHED.

Theoretical research that provided a basis for the development of LEMd has been done 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.

References

[1] M.W. Kirschner and J.C. Gerhart, *The Plausibility of Life: Resolving Darwinin's Dilemma*, Yale University Press, 2005.

[2] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, London, 1996.

[3] P. Bentley, "From coffee tables to hospitals: generic evolutionary design", in *Evolutionary Design by Computers*, pp. 405–423, ed. P. Bentley, Menlo Park, CA: Morgan Kaufmann, 1999.

[4] P. Bentley, *Evolutionary Design by Computers*, Menlo Park, CA: Morgan Kaufmann, 1999.

[5] A. Oyama, "Multidisciplinary optimization of transonic wing design based on evolutionary algorithms coupled with CFD solver", *Eur. Congress on Computational Methods in Applied Sciences and Engineering*, 2000.

[6] P. Bentley and D. Corne, *Creative Evolutionary Systems*, Morgan Kaufmann, 2002.

[7] R.S. Michalski, "Learnable evolution model: evolutionary processes guided by machine learning", *Machine Learning* 38, 9–40 (2000).

[8] J. Wojtusiak and R.S. Michalski, "The LEM3 implementation of learnable evolution model and its testing on complex function optimization problems", *Proc. Genetic and Evolutionary Computation Conference, GECCO 2006* PO6–7, (2006).

[9] R.S. Michalski, and K.A.Kaufman, "The AQ19 system for machine learning and pattern discovery: a general description and user's guide", *Reports of the Machine Learning and Inference*

Laboratory MLI 01-2, George Mason University, Fairfax, VA, 2001.

[10] J. Wojtusiak, "AQ21 user's guide," *Reports of the Machine Learning and Inference Laboratory* MLI 04-3, George Mason University, Fairfax, VA, 2004.

[11] R.S. Michalski, "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, 2004.

[12] P.A. Domanski, D. Yashar, K. Kaufman, and R.S. Michalski, "An optimized design of finned-tube evaporators using the learnable evolution model", *Int. Journal on Heating, Ventilating, Air-Conditioning and Refrigerating Research* 10, 201–211 (2004).

[13] P. Domanski, "EVSIM-an evaporator simulation model accounting for refrigerant and one dimensional air distribution", *NISTIR*, 89–4133 (1989).

[14] R.G. Reynolds, "An introduction to cultural algorithms, *Proc. 3rd Annual Conf. on Evolutionary Programming*, 131–139 (1994).

[15] S. Saleem and R. Reynolds, "Cultural algorithms in dynamic environments", *Proc. Congress on Evolutionary Computation*, 1513–1520 (2000).

[16] K. Rasheed, "GADO: A genetic algorithm for continuous design optimization", *Technical Report DCS-TR-352*, Department of Computer Science, Rutgers University, New Brunswick, NJ, 1998.

[17] W.E. Hart, N. Krasnogor, and J.E. Smith, *Recent Advances in Memetic Algorithms*, Springer-Verlag, Berlin, 2005.