Computing and Informatics, Vol. 28, 2009, 1001–1012, V 2009-Mar-2

THE LEM3 SYSTEM FOR MULTITYPE EVOLUTIONARY OPTIMIZATION

Janusz Wojtusiak

Machine Learning and Inference Laboratory
Department of Health Administration and Policy
George Mason University
4400 University Drive, MS 1J3, Fairfax, VA 22030, USA
e-mail: jwojtusi@gmu.edu

Revised manuscript received 9 December 2008

Abstract. LEM3 is the newest version of the learnable evolution model (LEM), a non-Darwinian evolutionary computation methodology that employs machine learning to guide evolutionary processes. Due to the deep integration of different modes of operation, several novel elements in its algorithm, and the use of the advanced machine learning system AQ21, the LEM3 system is a highly efficient and effective implementation of the methodology. LEM3 is particularly attractive for multitype optimization because it supports, and treats accordingly, different attribute types for describing candidate solutions in the population. These attribute types are nominal, ordinal, structured, cyclic, interval, and ratio. Application to optimization of parameters of a complex system illustrates multitype optimization problem.

Keywords: Evolutionary computation, learnable evolution model, machine learning, multitype optimization, representation space

1 INTRODUCTION

Research on non-Darwinian evolutionary computation is concerned with developing algorithms in which the creation of new candidate solutions in the population is guided by an "intelligent agent," rather than done merely by random or semi-random change operators, such as mutations and/or crossovers, employed in the "Darwinian-type" evolutionary methods [1]. The *learnable evolution model* (LEM)

employs machine learning to direct the evolutionary process [5]. Specifically, the method creates general hypotheses indicating *why* some candidate solutions perform better than others, and then instantiates these hypotheses to generate new candidate solutions.

Experiments with different implementations of LEM have demonstrated that it significantly and consistently speeds up the evolutionary process in terms of the evolution length, defined as the number of fitness evaluations needed to achieve the target solution, in comparison to Darwinian-type evolutionary algorithms. They also have indicated that the LEM advantage grows with the complexity of the problem, as measured by the number of variables to optimize. However, because LEM involves machine learning, its execution requires more computational time than execution of standard evolutionary operators such as mutation or recombination [5]. These findings indicate that LEM may be particularly attractive for solving very complex optimization problems in which the fitness evaluation is time-consuming or costly.

The LEM1 and LEM2 versions of the learnable evolution model employed the AQ15 [11] and AQ18 [3] rule learning programs, respectively, and included a relatively small subset of the LEM methodology. Their experimental testing produced very promising results on several benchmark problems. Domain-oriented LEM implementations, ISHED (Intelligent System for Heat Exchanger Design) and ISCOD (Intelligent System for Condenser Optimization and Design), were tailored to problems of optimizing heat exchanger designs, and also produced highly satisfactory results [7]. The latest version, LEM3, described in this paper, includes several significant improvements over the earlier ones. The methodology described in this paper is implemented in the LEM3 computer system. Another, independently developed, implementation of the learnable evolution model for multi-objective optimization, LEMMO, is based on rules generated from trees by the C4.5 learning program [9]. LEMMO was applied to a water quality optimization problem [2].

Other evolutionary computation methods that are the most similar to LEM include *cultural algorithms* (CA) [10] that in parallel to solutions evolve beliefs that constrain optimization, and *estimation of distribution algorithms* (EDA) [4] that use statistical learning to approximate the sub-space consisting of high-performing solutions. Despite their similarities, both CAs and EDAs are significantly different from LEM.

Many real world optimization problems are naturally described using different types of attributes. Some of these attributes may be numeric, and some may be symbolic with different internal structure (e.g. linearly ordered or representing a hierarchy), e.g. see optimization of parameters of complex systems in Section 4. These problems require ad-hoc definition of special encoding and genetic operators. Moreover, because of multitype character of the problem, neither very efficient numeric optimization methods, nor those used for symbolic problems (e.g. in combinatorial optimization) can be directly applied. This paper concentrates on describing features of the LEM3 system that make it particularly applicable to *multitype optimization* problems, by which we mean problems in which representation and fitness are defined using attributes of different types.

2 OVERVIEW OF THE LEM3 METHOD

This section presents an overview of the LEM3 method, whose algorithm is presented in Figure 1. LEM3 is the most recent version of the learnable evolution model. Its algorithm contains several components found in traditional evolutionary algorithms, such as generation of an initial population, selection of candidate solutions into a new population, and evaluation of candidate solutions. Other LEM3 components are concerned with guiding evolutionary computation through machine learning. This is done by selecting at each step of evolution the highest and lowest performing candidate solutions in the population, the H-group and L-group, respectively, and then employing the AQ21 program [15], the newest implementation of very successful AQ series of separate-and-conquer rule learning programs, to generate a hypothesis that differentiates between the two groups. The hypothesis is then instantiated in various ways to generate new candidate solutions.

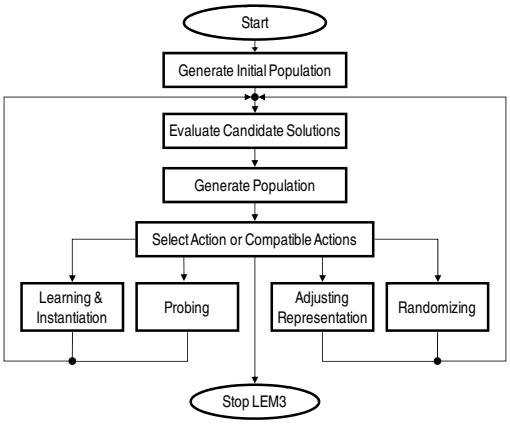

Fig. 1. Flowchart of the top level LEM3 algorithm

LEM3 works in different modes of operation. Each action (learning and instantiating, probing, adjusting representation, and randomizing) presented in Figure 1 represents one mode of operation that may consist of a number of different operations (e.g., in learning mode, three operations are executed, namely example selection, hypothesis creation, and hypothesis instantiation). The program can select one action (mode of operation), or several actions can be executed in parallel (e.g. learning mode and probing mode). The selection of actions is made based on the stage of evolution, diversity in the population, and user-specified parameters.

2.1 Learning Mode

In learning mode new candidate solutions are created through hypothesis formulation and instantiation. This is a three step process:

1. selection of example candidate solutions,
2. hypothesis formulation, and
3. hypothesis instantiation.

Based on the fitness of known candidate solutions, step 1) selects high-performing (H-group) and low-performing (L-group) candidate solutions from the population. These groups serve as training examples for the AQ21 learning program.

There are two methods of creating these groups [1]. The first one, *fitness-based selection*, selects solutions whose fitness values are within given thresholds for H- and L-groups. For example, if high and low fitness thresholds are both 25 %, then candidate solutions whose fitnesses are in the highest 25 % of the range and the lowest 25 % of the range are included in the H-group and L-group, respectively. The second method, *population-based selection*, selects a specified percentage of candidate solutions from the population for each group. For example, if both percentages are set to 30 %, then the 30 % of the candidate solutions with the highest fitness and the 30 % with the lowest fitness are included in the H- and L-group, respectively.

2.1.1 Hypothesis Formulation

The central component of the LEM3 algorithm is a learning module that induces general hypotheses for discriminating H-group from L-group candidate solutions. Such hypotheses explain *why* some candidate solutions perform better than others. In LEM3, this function is performed by AQ21, whose learning module generates hypotheses in the form of attributional rules [15]. Such a representation of hypotheses is beneficial to LEM because of the rules' high expressive power, the ease of instantiation of learned hypotheses, and the understandability of the hypotheses by experts.

AQ21, given positive and negative examples of a concept, induces a general concept description in the form of an attributional ruleset, a set of rules with the same consequent [6]. The simplest form of an attributional rule used in this study is:

$$CONSEQUENT \Leftarrow PREMISE \tag{1}$$

where CONSEQUENT and PREMISE are conjunctions of *attributional conditions* (a. k. a. *selectors*). A basic form of selectors is (2), where L is an attribute, R is a value, disjunction of values, or a range, and rel is a relation applicable to L and R. For simplicity this paper does not consider any advanced forms of selectors.

$$[L\ rel\ R] \tag{2}$$

An example of such a rule is in Figure 2. It is interpreted as follows: *a design is high-performing if its length is between 21 and 73 inches, shape is polygonal, and power usage is smaller than 25 KW.*

The AQ21 learning algorithm starts with focusing on one high-performing candidate solution, called a *seed*, and generates possible rules that cover the seed and

```
[Design = high-performing] <== [Color = red v chrome v black] &
                               [Length = 21..73 inches] &
                               [Shape = polygonal] & [Power usage < 25 KW]
```

Fig. 2. Example of attributional rule

maximize user-specified preference criteria (e.g. maximize the number of covered high-performing solutions and minimize the number of low-performing solutions). This operation is repeated for different seeds until the union of learned rules covers all high-performing candidate solutions included in H-group. Details of AQ learning and its different variants are available in the literature, e.g. [15].

2.1.2 Instantiation

Instantiation (step 3 of the learning mode) is a process of generating new candidate solutions that satisfy a given hypothesis. The instantiation process is the main way to create new candidate solutions in the learnable evolution model. The process of instantiating attributional rules starts with computing the numbers of candidate solutions to be generated from each rule (by default it is proportional to rules' strengths). Each rule is instantiated by randomly assigning values of attributes included in the rule. For attributes that are not included in the rule being instantiated, a value is taken from an existing candidate solution. The solution is selected randomly with probability proportional to its fitness value.

To illustrate the instantiation process, consider the following example. Suppose that candidate solutions are defined using attributes $Color$, $Length$, $Shape$, $Material$, $Tubes$, and $Power\ usage$. Suppose that AQ21 learned the rule presented in Figure 2. The following candidate solutions may be generated from the rule:

$$(Color = red, Length = 50, Shape = triangle, Tubes = 9, Power\ usage = 17\,KW)$$

$$(Color = black, Length = 27, Shape = square, Tubes = 8, Power\ usage = 24\,KW)$$

Note that *triangle* and *square* are children values of *polygonal* in the domain of the structured attribute $Shape$. The numbers of tubes 9 and 8 were selected from existing candidate solutions, because it is not included in the rule in Figure 2.

2.2 Probing, Randomizing, and Changing Representation

The probing mode executes Darwinian-type operators in order to generate new candidate solutions. The two operators implemented in LEM3 are single-point crossover and mutation. The reason for including probing mode operators is that mutation is helpful in maintaining diversity within population (without diversity LEM3's learning mode cannot be executed), and cross-over may be appropriate for searching for values "between" two existing candidate solutions. LEM3 is also equipped

with a randomization operator that either randomly creates candidate solutions, or restarts the evolution process.

Adjusting the representation space of solutions in LEM3 include removing irrelevant attributes, adjusting domains of attributes via discretization, and creating new attributes [13]. The AQ21 system used by LEM3 is equipped with a constructive induction module that automatically constructs new attributes and removes irrelevant ones. Discretization is done using *adaptive anchoring discretization*. It starts with a rough discretization and increases the precision in the most promising areas.

3 MULTITYPE OPTIMIZATION

The LEM3 system allows the user to easily define the representation space (and fitness function) using different types of attributes. Such a definition does not require designing an ad-hoc attribute encoding, as sometimes needed, for example, in genetic algorithms, but instead can be made in a more natural way, with the direct use of the attributes used in the problem definition. This feature extends LEM3's applicability to domains in which solutions are measured not only by numerical parameters, but also by a combination of qualitative and qualitative properties. The LEM3 system recognizes attribute types defined in attributional calculus and available in AQ learning [6, 8].

- *Nominal* attributes have unordered symbolic domains of possible values. For example an attribute $Color$ with domain $\{red, green, blue, black, yellow\}$ is nominal. The AQ21 learning module creates conditions involving nominal attributes in the form (2) where rel is "=", and R is a value or disjunction of values. To instantiate such conditions LEM3 randomly selects a value from R and assigns it to L in a newly created candidate solution. Because there is no distance measure defined between values of nominal attributes, the mutation operator randomly selects a value from the entire domain.
- *Ordinal* (a. k. a. rank) attributes have linearly ordered symbolic domains of possible values. For example an attribute Length with domain $\{very\ short, short, medium, long\}$ is ordinal. The AQ21 learning module creates conditions involving ordinal attributes in the form Equation (2) where rel is "=", and R is a value, disjunction of values, or a range. To instantiate such conditions LEM3 randomly selects a value from R and assigns it to L in a newly created candidate solution. The distance measure for ordinal attributes in LEM3 is defined as $d(v, w) = \#values\ between\ v\ and\ w + 1$. The mutation operator randomly selects a value from the domain of L with probability linearly decreasing from the original value from the parent solution.
- *Cyclic* attributes have ordered symbolic domains whose values form a cycle. For example, an attribute Day of week with domain $\{Mon, Tue, \ldots, Sun\}$ is cyclic. In both learning and probing modes cyclic attributes are treated in the same way as ordinal attributes, with using the shortest distance between values.

- *Structured* attributes have partially ordered symbolic domains whose values form a hierarchy. For example, an attribute Shape with domain {*rectangular*, *triangular*, *oval*, *square*, *rectangle*, *right triangle*, *acute triangle*, *circle*, *ellipse*} is structured when values form a hierarchy. The instantiation operators randomly select a value that is a leaf in an appropriate subtree.
- *Interval* attributes have numerical domains for which linear transformations are defined. For example, Temperature (F) is an interval attribute because it is meaningful to say "the temperature inside is 20 degrees higher than the temperature outside."
- *Ratio* attributes have numerical domains for which ratio transformation are meaningful, and value 0 is defined. For example, Width (inches) is a ratio attribute because it is meaningful to say "object A is twice as wide as object B." Note that the attribute Temperature (F) is not ratio, because the 0 point is not well defined – it does not make sense to say that "the temperature inside is twice as high as the temperature outside."

Additionally LEM3 recognizes discretized interval and discretized ratio attribute types whose semantics are the same as for the interval and ratio types, but values are abstracted into their discrete representations [8].

4 APPLICATION TO OPTIMIZATION OF COMPLEX SYSTEMS

Previous experimental evaluation of the LEM3 system on a set of benchmark optimization problems indicated its advantage when compared to a simple Darwinian-type real coded evolutionary algorithm equipped with mutation and cross-over. The advantage of LEM3 over the compared method grows with the number of attributes (tested up to 1000 continuous attributes). The same study compared LEM3 with published results on *estimation of distribution algorithms* (EDAs), and *cultural algorithms* (CAs) also indicating LEM3's advantage [14].

The following sections describe an example application of LEM3 to a multitype opitmization problem. Because candidate solutions are described using multitype attributes, many evolutionary computation methods, whose primary area of application is to numerical problems, cannot be directly applied. On the other hand, LEM3 that uses different types of attributes does not need any special encoding of the representation.

4.1 Problem Definition

In order to evaluate the performance of LEM3 on a multitype optimization, it has been applied to finding optimal setting of a complex system. The AQ21 program is controlled by several parameters. Moreover, different applications may require different settings of parameters. For large datasets consisting of hundreds or thousands of examples, the learning process takes a considerable amount of time, sometimes

in order of hours. The parameters controlling AQ21 are both numerical and symbolic. There are also several constraints which define impossible combinations of parameters or combinations that don't make sense. The representation space spans all possible combinations of AQ21's parameters chosen to be optimized. Each of AQ21's parameters defines one attribute in the representation, and possible values of the parameter constitute a domain of the corresponding attribute. The complete list of attributes AQ21 parameters is described in [12, 13]. The space consists of 24 attributes (12 nominal, 10 ratio, and 2 absolute). Assuming that initially the numeric attributes are discretized by adaptive anchoring discretization into 10 ranges, the total size of the space is about 2.6×10^{21}. Assuming that a single execution of AQ21 takes in average only one second the exhaustive search that checks all possible combinations of parameters would take over 8×10^{11} years. Thus, finding optimal AQ21's parameters requires using an efficient optimization method.

The fitness function used here is based on the accuracy and complexity of the learned hypotheses. To compute accuracy and complexity, it is needed to execute AQ21 and analyze its results. In addition to *predictive accuracy* (defined as the ratio of the number of correctly classified examples to the total number of testing examples) AQ21 returns a measure of *precision* of the given answer [12], to reflect imprecise answers (e.g. an example is classified to more than one class). Similarly to the predictive accuracy, the precision is given by a number varying from 0 % (all answers are fully imprecise) to 100 % (all answers are precise).

In natural induction, simplicity of learned knowledge is equally important to its accuracy. AQ21 learning module computes complexity, CX, of learned hypotheses by assigning a weight to each operation using (3) as proposed in the attributional calculus by Michalski [6, 12]. Complexity of an exception is multiplied by two.

$$\begin{aligned} CX \;=\; & 4 \times \#conjunctions + 10 \times \#disjunctions + 2 \times \#internal\ disjunctions \\ & + 2 \times \#ranges + \#equal + \#less\ or\ greater + 2 \times \#not\ equal \end{aligned} \tag{3}$$

Finally, the formula predictive accuracy, precision and complexity can be combined in one fitness function (4), where ICX is the complexity of the input data.

$$Fitness(X) = Acc(X) \times Prec(X) \times (100 \times (ICX - CX(X))/ICX)^{0.5} \tag{4}$$

4.2 Medical Datasets

In this study AQ21's parameters are optimized to achieve the best performance on three medical datasets. The datasets are available form the author upon request. These datasets represent different types of learning problems, with different numbers of classes, examples, attributes, etc. The first dataset, called here *metabolic syndrome*, consists of measurements of different parameters aggregated over different groups of patients. It consists of 20 examples drawn from 4 classes that repre-

sent three diseases from the metabolic syndrome spectrum, and a healthy status. This dataset has been collected from articles published in medical journals such as *Hepatology*, *Obesity Research*, *International Journal of Obesity* and some others.

The second dataset, called here *vitality score*, consists of measurements of different parameters of a group of patients and their vitality scores. The vitality score is a measure of patients' performance computed from answers to the SF-36 form (e.g., see `http://www.sf-36.org`). The dataset consists of 43 patients, 11 input attributes selected based on expert's decision about their relevance to the problem. All of the selected attributes are numeric. The continuous output attribute representing the vitality score has been discretized into three classes.

The third dataset, called here *lifestyles*, consists of lifestyles and diseases of nonsmoking males, aged 50–65 recorded by the American Cancer Society. It contains 73 553 records of responses of patients to questions regarding their lifestyles and diseases. Each patient is described in terms of 32 attributes: 7 lifestyle attributes (2 Boolean, 2 numeric, and 3 ordinal), and 25 Boolean attributes representing diseases. From the original set of examples, 200 examples were randomly chosen for the presented experiments. The problem considered here is to determine rules for recognizing prostate cysts based on lifestyles and presence of other diseases.

4.3 Experimental Setup

The presented experiment have been performed for each of the three datasets in the following steps:

1. Execute AQ21 with default parameters. Calculate predictive accuracy, precision, and complexity of learned hypotheses averaged in 5-fold cross validation, and use (4) to calculate the fitness value.
2. Execute LEM3 to find the best settings of AQ21's parameters. The best parameters are those for which the average value of the fitness function (4) on 5-fold cross validation is the highest.
3. Repeat the second step 10 times, and report average results.
4. Compare results obtained by AQ21 with default parameters with those obtained after LEM3's optimization.

In the presented experiments LEM3 was executed with the following parameters: population size 100, the number of candidate solutions created in each generation 30, 30 % of candidate solutions from population assigned to H-group, 30 % of candidate solutions from population assigned to L-group (population-based selection), and enabled automatic improvement of representation space. The program was set to stop after the total of 150 generations.

4.4 Results

When applied to the metabolic syndrome dataset, AQ21 with default parameters gives 75 % predictive accuracy, with precision 100 % and complexity 12. These values correspond to the fitness function value 74 624.06. After LEM3 optimization of parameters on the same dataset AQ21 achieved 95 % predictive accuracy, 100 % precision and complexity $CX = 9$, giving the fitness value 94 523.81. This is a significant improvement (about 27 % of fitness value gain). An interesting result is that the best results were found with enabled automatic improvement of representation spaces by constructive induction in LEM3.

When applied to the vitality score dataset, AQ21 with default parameters achieved predictive accuracy 40 %, precision 100 %, and complexity $CX = 65$. These values correspond to the fitness function value 37 309.52. After optimization the program achieved predictive accuracy 80 %, precision 100 %, and complexity 33.8, giving the fitness value 77 149.21. This is over two times improvement in the fitness value.

When applied to the lifestyles dataset, AQ21 with default parameters achieved predictive accuracy 57 %, precision 99.02 %, and complexity $CX = 610.4$. These values correspond to the fitness function value 55 012.27. After optimization, the program achieved predictive accuracy 82 %, precision 97.12 %, and complexity $CX = 59.40$, giving the fitness value 79 639.4. This is about 45 % in the fitness value improvement.

The above results show that LEM3 can be indeed applied to multitype optimization problems. Note that the goal of this research was not to compare LEM3's performance with other methods on well known problems (such a comparison was made before [14]), but to show its applicability to problems on which other methods cannot be directly applied. Other details of the above experiment needed are presented in [13].

5 CONCLUSIONS

The presented LEM3 system is the newest and the most advanced version of the learnable evolution model. In many aspects, the algorithms implemented in LEM3 go beyond the original LEM methodology described in [5]. For example, novel aspects include the possibility of applying different operators in parallel or sequentially, new instantiation methods, multitype optimization, and automatic improvement of representation spaces. LEM3 showed high scalability that could not be achieved with previous implementations.

This paper showed that LEM3's advantage goes beyond optimization of continuous functions. It can be easily applied to multitype optimization, in which the representation space is defined using attributes of different types. The discussed attribute types are nominal, ordinal, cyclic, structured, interval, and ratio, and correspond to types of properties used by people. Application of LEM3 to optimization

of parameters of complex systems, exemplified by AQ21, illustrated its applicability to this type of problems.

Acknowledgment

The author expresses his gratitude to Dr. Ryszard S. Michalski, the creator of LEM, for his guidance during creation of the presented methodology and implementation of the LEM3 system. Dr. James Gentle provided valuable comments on this paper. The author also thanks anonymous reviewers whose comments helped to improve important aspects of this paper.

The presented research has been supported in part by the National Science Foundation Grants No. IIS 9906858, IIS 0097476 and CBET 0742487, and in part by the UMBC/LUCITE #32 grant. The findings and opinions expressed here are those of the author, and do not necessarily reflect those of the sponsoring organizations.

REFERENCES

[1] Holland, J. H.: Adaptation in Natural Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. University of Michigan Press, 1975.

[2] Jourdan, L.—Corne, D.—Savic, D.—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.

[3] Kaufman, K.—Michalski, R. S.: The AQ18 System for Machine Learning and Data Mining System: An Implementation and User's Guide. Reports of the Machine Learning and Inference Laboratory, MLI 00-3, George Mason University, Fairfax, VA, 2000.

[4] Larranaga, P.—Lozano, J. (Eds.): Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Kluwer Academic Publishers, 2002.

[5] Michalski, R. S.: Learnable Evolution Model: Evolutionary Processes Guided by Machine Learning. Machine Learning, 2000, 38, pp. 9–40.

[6] 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.

[7] Michalski, R. S.—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, Vol. 21, 2006, No. 12.

[8] Michalski, R. S.—Wojtusiak, J.: Semantic and Syntactic Attribute Types in AQ Learning. Reports of the Machine Learning and Inference Laboratory, MLI 07-1, George Mason University, Fairfax, VA, 2007.

[9] Quinlan, J. R.: C4.5: Programs for Machine Learning, Morgan Kaufmann, 1993.

[10] Reynolds, R. G.–Zhu, S.: Knowledge-Based Function Optimization Using Fuzzy Cultural Algorithms with Evolutionary Programming. IEEE Transactions on Systems, Man, and Cybernetics, Vol. 31, 2001, pp. 1–18.

[11] Wnek, J.—Kaufman, K.—Bloedorn, E.—Michalski, R. S.: Inductive Learning System AQ15c: The Method and User's Guide. Reports of the Machine Learning and Inference Laboratory, MLI 95-4, George Mason University, Fairfax, VA, March 1995.

[12] Wojtusiak, J.: AQ21 User's Guide. Reports of the Machine Learning and Inference Laboratory, MLI 04-3, George Mason University, Fairfax, VA, September, 2004.

[13] Wojtusiak, J.: Handling Constrained Optimization Problems and Using Constructive Induction to Improve Representation Spaces in Learnable Evolution Model. Ph. D. Dissertation, College of Science, Reports of the Machine Learning and Inference Laboratory, MLI 07-3, George Mason University, Fairfax, VA, November, 2007.

[14] Wojtusiak, J.—Michalski, R. S.: The LEM3 Implementation of Learnable Evolution Model and Its Testing on Complex Function Optimization Problems. Proceedings of the GECCO 2006 Conference, Seattle, WA, July 8–12, 2006.

[15] Wojtusiak, J.—Michalski, R. S.—Kaufman, K.—Pietrzykowski, J.: The AQ21 Natural Induction Program for Pattern Discovery: Initial Version and its Novel Features. Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence, Washington D. C., November 13–15, 2006.

Janusz Wojtusiak is an Assistant Professor in the Department of Health Administration and Policy and the Director of the Machine Learning and Inference Laboratory at George Mason University. He received his M. Sc. degree in computer science from Jagiellonian University (with honors), and his Ph. D. in computational sciences and informatics, concentration in computational intelligence and knowledge mining, from George Mason University. His research interests include human-oriented machine learning, knowledge mining, non-Darwinian evolutionary computation, and health informatics.