Towards an Adaptive Implementation of Genetic Algorithms

César Bravo
Escola Politécnica da Universidade de São Paulo (Brazil), Computer and Digital Systems Dept., São Paulo, Brazil, 05508-900
cesarabravop@hotmail.com

João J. Neto
Escola Politécnica da Universidade de São Paulo (Brazil), Computer and Digital Systems Dept., São Paulo, Brazil, 05508-900
joao.jose@poli.usp.br

Fabiana S. Santana
Escola Politécnica da Universidade de São Paulo (Brazil), Computer and Digital Systems Dept., Computer and Digital Systems Dept., São Paulo, Brazil, 05508-900
fabiana.santana@gmail.com

Antonio M. Saraiva
Escola Politécnica da Universidade de São Paulo (Brazil), Computer and Digital Systems Dept., São Paulo, Brazil, 05508-900
amsaraiv@usp.br

Abstract

Ecological niche modelling allows inferring the niche of a species according to the values of environmental variables and species presence points in a given geographic area; modelling algorithms proceeds correlating the species data with the raster environmental layers in a sampled area until a niche model were identified. An *adaptive decision table* is a decision table able to modify the rules during the execution time. As a proof of concept, we applied the proposed method to the Genetic Algorithm for Rule-set Production (GARP), which is one of the most applied modelling algorithms. One of the main problems is to identify which algorithm parameters are able to result in a better performance and to generate an adequate model for the species geographic distribution. In this case, the adaptive techniques may imply in a better evaluation of the context and, in the experiments executed during this research, it was possible to show that the new implementation does not affect the performance of the genetic algorithm. This approach may result in a methodology for adoption and software implementation of adaptive decision tables in order to solve problems related to the use of genetic algorithms. The methodology is presented in this paper.

Keywords: Adaptive decision table, ecological niche modelling, GARP, Genetic Algorithms, Adaptive techniques.

1 Introduction

This paper treats of two very strong trends, adaptivity and evolutionary programming, which are put together so as to propose an application related to environmental modeling. This approach is suitable for the practice of complex systems programming, featuring artificial intelligence capabilities, such as grammatical inference, computer learning and autonomous dynamically changeable behavior. This experience is based on the concepts of modelling provided by the *openModeller* framework [http://openmodeller.sourceforge.net/], which infers a probability distribution of a species, in a given geographical area, from data about the species occurrence and environmental variables in that area (input data).

The openModeller framework includes several alternative algorithms to do the inference and, among them, the GARP genetic algorithm is the best choice for these kind of experiments, because it is simultaneously one of the most sensitive to parameters changes and one of the most applied worldwide [6], [7]. GARP is a genetic algorithm which takes the input modelling data and, based on their values, produce a set of rules which allow to infer the presence of the given specie in another geographical area according to the values of the considered environmental variables.

An adaptive decision table is a decision table provided with a set of adaptive functions able to add or remove the rules on defining the subjacent decision table. Any genetic algorithm can be executed by an adaptive decision table [1] and, in order

to show a concept proof of this claim, the original implementation of the GARP algorithm was substituted by the AdapGARP, proposed in this paper, and some experiments were done in order to verify the impacts on the performance of the new algorithm related to original GARP. At last, the openModeller framework will be provided with a new version of the GARP algorithm, implementing adaptive the operators Crossover and Mutation, of a genetic algorithm definition, so as to offer a smart way to define parameters choices.

2 Adaptive Devices

In this work, the adaptive models follow a slightly modified version of the formulation defined in [2].

Non-adaptive devices are formal devices whose behavior is defined as a static set of rules. This kind of devices may have their operation enhanced by adding to them an adaptive layer that associates each device's rule to a set of adaptive actions. Adaptive actions specify the changes to be applied to the device's sets of rules by using primitive editing operators, which allow applying inspection, deletions and additions to the set of rules, defining the device. In another words, the adaptive layer performs all operations that are needed for dynamically modifying the set of rules, defining the operation of the device.

Such improved devices are called *adaptive devices* and the original devices from which they are obtained are said to be their corresponding *subjacent non-adaptive devices*. There is no restriction in the nature of subjacent devices, so one may obtain adaptive devices from virtually any kind of abstraction defined as set of rules, e.g. automata, grammars, decision tables, etc.

3 Mapping Genetic Algorithms into Adaptive Decision Tables

For the rest of this work, the subjacent abstraction will be the *decision table*, and the corresponding adaptive device will be named as *adaptive decision tables*.

A *decision table* ([1], [3], [4] and [5]) is a table encoding a set of rules in columns. The first column designates a number of *Condition Rows*, on top of the table, and a set of *Actions Rows*, on the bottom of the table. The set of rules starts at an initial configuration, and the *decision table* operates checking the valid conditions against the values defined in the column rules. When a condition is found to be true for a given rule, then all the marked actions for this rule are executed.

		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Condition rows	$\mathbf{c_1}$										
	$\mathbf{c_2}$										
	…										
	$\mathbf{c_n}$										
Actions rows	$\mathbf{a_1}$										
	$\mathbf{a_2}$										
	…										
	$\mathbf{z_m}$										

Table 1 Decision Table

An adaptive version of this abstraction may be obtained by adding to an existing non adaptive table a number of further lines which encode, for each column representing a single rule, the (parametric) calls to *adaptive functions* associated to the execution of that particular rule. Whenever a rule in the table is applied, the associated adaptive functions are invoked, and their corresponding collateral effects change the current set of rules, according to the adaptive operators performed by the adaptive function.

The strategy adopted for this work consists of encoding decision-taking rules as rules of an *adaptive decision table*, and the changes in behavior will be modelling as adaptive actions, over the rules encoded in the decision table. In order to be useful, the resulting *adaptive decision table* must replicate the behavior of the original genetic algorithm.

Once the programming of the mapping law is determined, any genetic algorithm-based program may be converted to an equivalent adaptive transition table-driven version. In this work, both version were compared regarding the number of iterations as a function of the fitness function precision.

3.1 Adaptive decision table format and operation

As we already pointed out, to create an *adaptive decision table* from a decision table, it is necessary to add some *adaptive functions* to the definition of the *decision table*. As defined in [2], these *adaptive functions* are placed in a set of rows under the *Actions Rows* of the *decision table*. The *Table 2* shows the general setup of an *adaptive decision table*.

							D_1	D_2	...	D_h	R_1	R_2	...	R_h
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**h+4**				
Condition rows	c_1													
	c_2													
	…													
	c_n													
Actions	a_1													
	a_2													
	…													
	z_m													
Adaptive Functions	ba_1													
	ba_2													
	…													
	ba_f													

Table 2 Adaptive Decision Table Format

To operate an adaptive decision table, first the status of the system is checked against the combinations of conditions stated in each of the rules encoded in the table. If no rule matches the current status, then no action is executed. Otherwise, if a single rule matches the current status, then a deterministic choice is identified. So, the matching rule is selected and is applied. Finally, if more than one rule matches the current status, then there is a non-deterministic situation and, consequently, all those rules need to be applied in parallel. From the practical point of view, the parallelism may be simulated, e.g. by some exhaustive backtracking strategy, and this is the reason that implicates in the requirement of the performance analysis of the new algorithm. The selected rule is then applied by executing the set of all actions indicated with a boolean value *True* in the cells of the rule corresponding to action rows. Once the selected rule has been applied, the decision table is ready to be used again.

4 The *AdaptGARP* Algorithm

The *AdaptGARP* algorithm describes an adaptive implementation of the GARP algorithm [6], [7], and its *Crossover* and *Mutation* operators, considering the version implemented in the openModeller framework. In order to do this, the rule-set will describe an individual, at a specific step of the genetic algorithm, in a format suitable for specifying these adaptive versions of the classical genetic operators.

Adaptive Crossover and *Adaptive Mutation* will act as rules manipulators over other rules stated in this new format, and they are invoked by the main algorithm – indeed, the genetic schemata – as subroutines, replacing the non-adaptive versions. After describing suitable data structure for use with adaptive genetic operators, Adaptive Crossover and Adaptive Mutation are described. For the rest of this work, these operators will be named as *AdaptCrossover* and *AdaptMutation*.

4.1 Data structures

First of all, data structures must be designed so as to apply the general technique described in [2].

The *Crossover* genetic operator takes two individuals and interchanges genetic information between them, at the genetic level, producing a new individual; in the rules of the *GARP* algorithm, this handling of information operates on the limits of the intervals, defined within the rules.

Such technique requires that the rules of the non-adaptive device be encoded as a decision table. Then, it is necessary to design suitable adaptive functions for manipulating the rules encoded in the decision table. These adaptive functions are encoded as part of the adaptive decision table, which results in a compact representation of the genetic operators, as part of the adaptive decision table and representing the dynamic change of the rule-set.

In order to implement the *GARP* algorithm's rules, the format defined in [2] is applied. As an example, consider *h* interval rules, associated to *k* environmental variables, as depicted in Table 3.

	R_1	R_2	R_3		R_h
$x_1 \geq$	A_{11}	A_{21}	A_{31}	...	A_{h1}
$x_1 \leq$	B_{11}	B_{21}	B_{31}	...	B_{h1}
$x_2 \geq$	A_{12}	A_{22}	A_{32}	...	A_{h2}

$x_2 \leq$	B_{12}	B_{22}	B_{32}	...	B_{h2}
...	...	...	...	...	...
$x_k \geq$	A_{1k}	A_{2k}	A_{3k}	...	A_{hk}
$x_k \leq$	B_{1k}	B_{2k}	B_{3k}	...	B_{hk}

Table 3. Format for GARP chromosomes

Assume that interval rules are *presence* rules. In GARP, the usual codification of this kind of rule is:

$$\text{IF } x_1 \in [A_{11}, B_{11}] \text{ AND } x_2 \in [A_{12}, B_{12}] \text{ AND } \ldots x_k \in [A_{1k}, B_{1k}] \text{ THEN PRESENCE}$$

The meaning of a rule *Rj*, where $1 \leq j \leq h$, is that if each environmental variable x_i, where $1 \leq i \leq k$, then the species is present in the associated localization.

This represents the main data structure for the adaptive genetic operators described in the following subsections. The number of columns of the *Table 3* will change as a side-effect of *AdaptCrossover's* operator execution. The *AdaptMutation* operator will operate over some rule (column) in the table, by changing the limits of the intervals.

Some decisions must be taken on how to handle the rules. Indeed, any rule is referred by its index on the table, e.g. its column's index. Since in the rule-based adaptive device, the number of rules may change, the set of rules to be considered is a dynamic array, which maintain the sequential numbering of its columns, avoiding to renumber the columns in the table.

To specify the adaptive functions, auxiliary functions are applied, that are described below:

(a) $U_x(y) = \begin{cases} 0 & ; \text{ if } x < y; \\ 1 & ; \text{ if } \geq y \end{cases}$

(b) The *rand(n)* function, which produces a pseudo random integer number between 1 and n, and the *rand(a, b)* function, which produces a pseudo random integer number between *a* and *b*, when *a* e *b* are integers, and produces a pseudo random float number between *a* and *b*, when *a* and *b* are float.

4.2 Adaptive Implementation of Crossover Operator

The Crossover genetic operator takes two individuals and interchanges genetic information between them, at the genetic level, producing a new individual; in the rules of the *GARP* algorithm, this handling of information operates on the limits of the intervals, defined within the rules.

The Table 4 specifies the adaptive function *AdaptCrossover* in the format of the decision tables.

The cell on the first row of the first column is used to identify the function's name, leaving unused any other cells on that row; the second column, from second to fifth rows, are used to specify the parameters of *AdapCrossover*; the parameters *i* and *j* are integer numbers representing the indexes of the rules to be crossed; parameters *p* and *q* are integer numbers that specify the sections of the intervals which are inherited by the new individual, resulting from the crossover between rule *i* and rule *j*. The remaining rows specifies adaptive conditions, according to adaptive functions.

The adaptive function adds a new row to the table of the rules, which is expressed by the symbol "+" at the header of the third column, encoding one of the three elementary actions that may operate on the rules defining the adaptive device. Another elementary action is the "-", which removes rules. This is also applied in the specification of the adaptive version of the *Mutation operator*, in section 4.3. The *AdaptCrossover* operator creates a new rule for which all intervals with index less than *p* or greater or equal than *q*, will inherit from rule R_i, and the intervals with index greater or equal than *p* and less than *q*, will inherit from the rule R_j.

		+
	AdaptCrossover	
	I	
	J	
	P	
	Q	
$x_1 \geq$		$A_{i1} + (A_{j1} - A_{i1})(U_1(p) - U_1(q))$
$x_1 \leq$		$B_{i1} + (B_{j1} - B_{i1})(U_1(p) - U_1(q))$

$x_2 \geq$		$A_{i2} + (A_{j2} - A_{i2})(U_2(p) - U_2(q))$
$x_2 \leq$		$B_{i2} + (B_{j2} - B_{i2})(U_2(p) - U_2(q))$
...		...
$x_k \geq$		$A_{ik} + (A_{jk} - A_{ik})(U_k(p) - U_k(q))$
$x_k \leq$		$B_{ik} + (B_{jk} - B_{ik})(U_k(p) - U_k(q))$

Table 4 Adaptive Decision Table for *AdaptCrossover*

The rules can be resumed as:

$$x_h \geq \begin{cases} A_{ih} & ; \text{if } h < p; A_{jh} \\ ; \text{if } p \leq h < q; A_{ih} & ; \text{if } h \geq q \end{cases}$$

and

$$x_h \leq \begin{cases} B_{ih} & ; \text{if } h < p; B_{jh} \\ ; \text{if } p \leq h < q; B_{ih} & ; \text{if } h \geq q \end{cases}$$

for all $1 \leq h \leq k$.

4.3 Adaptive Implementation of Mutation Operator

The *Mutation genetic operator* takes one individual and modifies its genetic information, at the genetic level, producing a mutation of the original individual; in the case of the rules of the GARP algorithm, this operation alters the limits of the intervals defined within the rules.*Table* **5** specifies the adaptive function *AdaptMutation* as a decision table [2]. The cell in the first row of the first column is used to identify the function's name only, leaving unused any other cells in that row; in the second column, the rows 2-5 specifies the parameters of *AdaptMutation*; parameter *j* is an integer, representing the index associated to the rule; parameters *k* and *g* are pseudo random integer numbers; *k* specifies a gene to be changed; *g* specifies whether the *k*-th gene is to be changed or not; parameters *a* and *b* are pseudo random integer numbers, representing new limits for the interval chosen to be changed.

The remaining rows specifies which conditions the rule must follow, as described below:

$$x_h \geq \begin{cases} A_{jh} & ; \text{if } r < g; a \\ ; \text{if } r = g; A_{jh} & ; \text{if } r > g \end{cases}$$

and

$$x_j \leq \begin{cases} B_{jh} & ; \text{if } r < g; b \\ ; \text{if } r = g; B_{jh} & ; \text{if } r > g \end{cases}$$

for all $1 \leq h \leq k$.

		-	+
	AdaptMutation		
	j		
	r		
	g		
	a		
	b		
$x_1 \geq$		A_{j1}	$A_{j1}U_r(g) - (a - A_{j1})U_r(g + 1)$
$x_1 \leq$		B_{j1}	$B_{j1}U_r(g) - (b - B_{j1})U_r(g + 1)$
$x_2 \geq$		A_{j2}	$A_{j2}U_r(g) - (a - A_{j2})U_r(g + 1)$
$x_2 \leq$		B_{j2}	$B_{j2}U_r(g) - (b - B_{j2})U_r(g + 1)$
...			
$x_k \geq$		A_{jk}	$A_{jk}U_r(g) - (a - A_{jk})U_r(g + 1)$
$x_k \leq$		B_{jk}	$B_{jk}U_r(g) - (b - B_{jk})U_r(g + 1)$

Table 5 Adaptive Decision Table for *AdaptMutation*

4.4 Putting all together: AdaptGARP

The *Table 6* shows the general setup to encode the GARP genetic algorithm in a corresponding adaptive decision table.

The second column of the first five rows labeled as "Condition rows", defines the conditions controlling the operation of the adaptive decision table. To the column marked as *0* corresponds the first three rows marked as "Actions", and both are used to set up initial values to the adaptive decision table operation. The columns marked as *1* and *2* controls the operation of the adaptive decision table, corresponding to the four Action rows. The columns marked as *3F* and *4F* indicates halt conditions for the operation of the adaptive decision table. Also, last seven condition rows encode, in the columns beyond column 5, the sample data available (columns marked as D_1, D_2, D_3, … , D_h) and reserve room for the rules the Adaptive Decision will produce (columns marked as R_1, R_2, R_3, … , R_h).

As each rule must specify one interval for each of the environmental variables, there are *2k* rows for each rule to be produced; this way, the pair $A_{ij,}$ B_{ij} represents the extreme values of the interval for the *i*-th environmental variable of the rule *Rj*, where $h+4 \leq i \leq h+4+n$ and $1 \leq j \leq k$.

As the available data sample has only punctual data, we required that $A_{ij,} = B_{ij}$ when $5 \leq i \leq h+4$ and $1 \leq j \leq k$.

5 Experiments

Fourteen experiments were performed, evolving two species and data sets, for the both algorithms, GARP and *AdaptGARP*.

The data applied for the experiments were the test set of the openModeller system referent to the species *Furcata boliviana* and another data set of a species under study by the biological researchers of the project, named as *Species B*, were applied.

- *Furcata boliviana*: data set containing 66 presence points with coordinates of longitude, latitude and altitude.
- *Ouratea spectabilis*: data set contains 34 presence points with coordinates of longitude and latitude.

In order to define performance metrics, the growth of the number of iterations as a function of the required precision of the fitness function, were applied. In fact, the fitness function is based on a-priory probability calculated over the presence points – usually, half the sample – used to generate the model compared with the posteriori probability calculated, taking into account all the available data in the sample.

		AdaptCrossover specification			*AdaptMutation* specification									D_1	D_2	D_3	...	Dh					
								0	1	2	3F	4F		5	6	7	...	h+4	R_1	R_2	R_3	..	R_n
Condition rows	**iterations == 0**	**H**	**+**		**H**	**-**	**+**	-	**T**	-	-	-											
	fitness < 75%							-	-	**T**	-	-											
	iterations < IterMax							-	-	**T**	-	-											
	fitness ≥ 75							-	-	-	**T**	-											
	iterations ≥ IterMax							-	-	-	-	**T**											
	x1≥		$A_{i1} + (A_{j1} - A_{i1})(U_1(p) - U_1(q))$			A_{j1}	$A_{j1}U_r(g) - (a - A_{j1})U_r(g + 1)$							A_{11}	A_{21}	A_{31}	...	A_{h1}					
	x1≤		$B_{i1} + (B_{j1} - B_{i1})(U_1(p) - U_1(q))$			B_{j1}	$B_{j1}U_r(g) - (b - B_{j1})U_r(g + 1)$							B_{11}	B_{21}	B_{31}	...	B_{h1}					
	x2≥		$A_{i2} + (A_{j2} - A_{i2})(U_2(p) - U_2(q))$			A_{j2}	$A_{j2}U_r(g) - (a - A_{j2})U_r(g + 1)$							A_{12}	A_{22}	A_{32}	...	A_{h2}					
	x2≤		$B_{i2} + (B_{j2} - B_{i2})(U_2(p) - U_2(q))$			B_{j2}	$B_{j2}U_r(g) - (b - B_{j2})U_r(g + 1)$							B_{12}	B_{22}	B_{32}	...	B_{h2}					
	...		**...**											...	...	...	...	...					
	xk≥		$A_{ik} + (A_{jk} - A_{ik})(U_k(p) - U_k(q))$			A_{jk}	$A_{jk}U_r(g) - (a - A_{jk})U_r(g + 1)$							A_{1k}	A_{2k}	A_{3k}	...	A_{hk}					
	xk≤		$B_{ik} + (B_{jk} - B_{ik})(U_k(p) - U_k(q))$			B_{jk}	$B_{jk}U_r(g) - (b - B_{jk})U_r(g + 1)$							B_{1k}	B_{2k}	B_{3k}	...	B_{hk}					
Actions	**IterMax = 400**							**T**															
	iterations = 0							**T**															
	n = PopSize							**T**															
	Colonize								**T**	**T**													
	iterations++								**T**	**T**													
	Selection								**T**	**T**													
	Evaluate								**T**	**T**													
Adaptive Functions	***AdaptCrossover***	**A**								**T**													
	AdaptMutation				**A**					**T**													
Parameters	**i**	**P**								i_0													
	j	**P**			**P**					i_0													
	p	**P**								p_0													
	q	**P**								r_0													
	r				**P**					r_0													
	g				**P**					g_0													
	a				**P**					a_0													
	b				**P**					b_0													

Table 6 AdaptGARP

The fourteen experiments with each data sample showed no relevant changes in the performance. For the *Furcata boliviana* only, the results obtained were in a different value of iterations for a required precision of 0.007%, as showed in *Table 7*.

Fitness precision %	GARP	AdaptGARP
0.050	22	22
0.040	27	27
0.030	35	35
0.020	52	52
0.010	102	102
0.009	113	113
0.008	127	127
0.007	144	145
0.006	168	168
0.005	200	200
0.004	252	252
0.003	335	335
0.002	502	502
0.001	1002	1002

Table 7 Comparison of results for *Furcata boliviana*

6 Conclusions

This paper described how to implement genetic algorithms using adaptive decision tables and, as a proof of concept, the *AdaptGARP* was designed, as an alternative to the GARP genetic algorithm. In order to compare the performance of the proposed implementation, several experiments were executed, comparing both algorithms performances, using the same two data samples and parameters, and only a minimal lost of performance were observed, leading to the conclusion that the method is practicable. Future works includes made this implementation available for the biological researchers, in order to have its relevance evaluated by the community.

This work is result of the collaboration of the Adaptive Technologies Laboratory with the openModeller project. When this collaboration begins our aims was assure that Adaptive Devices can be used to obtain results compatibles on performance with those obtained using genetic algorithms. In this work we prove this claim and gain knowledge about the internals details of the GARP algorithm implementation, in particular, and about the openModeller system in general. Now we are ready to explore the design of new algorithms to contribute to the openModeller project.

Acknowledgements

Our thanks to FAPESP funding under the process 2004/11012-0. The first author wants to thanks to FAPESP funding under the process 2006/05797-0.

References

[1] Myers, H. J., “Compiling optimized code from decision tables”, *IBM Journal of Research and Development*, 16, 5 (Sep. 1972), 489-503.

[2] Neto, J. J., “Adaptive Rule-Driven Devices - General Formulation and Case Study”, *Lecture Notes in Computer Science*, Springer, 2494 (July 2001), 234-250.

[3] Pollack, S.L., Hicks, H.T., and Harrison, W.J., *Decision Tables: Theory and Practice*, Wiley, New York, 1971.

[4] Reinwald, L. T. and Soland, R. M, “Conversion of Limited-Entry Decision Tables to Optimal Computer Programs I: Minimum Average Processing Time”, *Journal of the ACM (JACM)*, v.13 n.3, p.339-358, July 1966.

[5] Reinwald, L. T. and Soland, R. M., “Conversion of Limited-Entry Decision Tables to Optimal Computer Programs II: minimum storage requirement”, *Journal of the ACM (JACM)*, v.13 n.3, p.339-358, July 1966.

[6] Stockwell, D., Peters, D, “The GARP modelling system: problems and solutions to automated spatial prediction”, *International Journal of Geographical Information Science*, 13, 25 (1999), 143-158.

[7] Stockwell, D. R. B., Beach, J. H., Stewart, A., Voronstsov, G., Vieglais, D. and Pereira, R. S., 2006. The use of the GARP genetic algorithm and internet grid in the Lifemapper world atlas os species biodiversity. *Ecological Modelling*. **Volume 195, Issues 1-2, 15 May 2006, Pages 139-145.**