Available online at www.sciencedirect.com
SCIENCE DIRECT®
Engineering Applications of Artificial Intelligence 17 (2004) 199–204
Engineering Applications of ARTIFICIAL INTELLIGENCE
www.elsevier.com/locate/engappai

Comparing learning classifier systems and Genetic Programming: a case study

S. Sette[a,*], B. Wyns[b], L. Boullart[b]

[a] ICT, Hogeschool West-Vlaanderen Sint-Martens-Latemlaan 2A, Kortrijk B-8500, Belgium
[b] Department of Electrical Energy, Systems and Automation, Ghent University Technologiepark 913, Zwijnaarde B-9052, Belgium

Abstract

Genetic Algorithms has given rise to two new fields of research where (global) optimisation is of crucial importance: '*genetic based machine learning*' (GBML) and '*genetic programming*' (GP). An advanced implementation of GBML (Fuzzy Efficiency based Classifier System, FECS, developed by the authors) and GP (as defined by Koza) are both applied to the case study 'fibre-to-yarn production process'. Results for both systems are presented and compared. Finally, the GP generated equations are transformed into rule-sets similar to those obtained from FECS.

Keywords: Genetic algorithms; Genetic programming; Textile production process; Learning classifier systems; Rule-based machine learning

1. Introduction

Genetic Algorithms(GAs) are generally used as an optimisation technique to search the global optimum of a function. However, this is not the only possible use for GAs. Other fields of applications where robustness and global optimisation are needed could also benefit greatly from the use of GAs. The two most important domains using GAs as the underlying methodology are *Genetic Based Machine Learning* (GBML) (Goldberg, 1989) and *Genetic Programming* (GP) (Koza, 1992).

GBML is a GA driven implementation of Rule Based Machine Learning (RBML). The goal in RBML is to generate a set of rules using an automated learning algorithm. This set of rules should allow the machine to perform optimally in its environment. This paper considers an advanced GBML algorithm, called Fuzzy Efficiency based Classifier System (FECS) as has been introduced by Boullart and Sette (1998), Sette (1998) and by Sette and Boullart (2000). GP is basically a GA applied to a population of Computer Programs (CP). While a GA operates on (coded) strings of numbers a GP operates on computer programs. The purpose of this paper is to compare GP (as described by Koza) with the aforementioned FECS. To this end a real-life industrial production problem is introduced.

One of the important production processes in the textile industry is the spinning process. Starting with cotton fibres, yarns are (usually) created on a rotor or ring-spinning machine. The spinnability of a fibre and (if spinnable) the resulting yarn strength is dependent on the fibre quality and on the machine settings of the spinning machine (BRITE, 1990–1993). FECS and GP will be applied to generate, respectively, classifiers (rule-sets)/mathematical equations for predicting spinnability and yarn strength departing from a certain product quality and machine settings.

FECS generates a rule set of 123 classifiers with a predictive accuracy of 94% for spinnability and a rule set of 119 classifiers with a predictive accuracy of 91% for yarn strength. Moreover, each classifier has a real life meaning (IF … THEN …) and includes a parameter (membership degree) which gives an important indication about the applicability of the corresponding rule. GP generates mathematical equations (1 for spinnability and 1 for yarn tenacity) allowing a correct overall prediction in all cases of at least 90%. Moreover, the resulting equations have a limited complexity (eliminating several input parameters) and give the user an insight into the importance of the database parameters.

Not only are similar accuracies obtained using GP and FECS, but a more detailed analysis shows

*Corresponding author. Tel.: +32-56/241299; fax: +32-56/241292.
E-mail address: stefan.sette@howest.be (S. Sette).

0952-1976/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.engappai.2004.02.006

also that the GP selected parameters correspond to the FECS evaluation of the most important input parameters.

2. Fuzzy efficiency based classifier system

The three basic components to construct a LCS (Goldberg, 1989) are:

- *A rule and message system*, which allows the machine to interact with its environment (accepting information from the environment and generating actions towards the environment).
- *A reward mechanism* to evaluate the rule set. This mechanism allows separating successful rules from unsuccessful or meaningless rules.
- *A GA* is used to generate new (more optimal) rules for the rule set.

A schematical overview of a LCS is given in Fig. 1.

The goal of this part was to generate rules between [quality/machine settings] and [spinnability/yarn strength]. However, the original LCS algorithm (by Goldberg) is limited to discrete values and shows also some basic deficiencies (Sette et al., 1998; Sette, 1998).

Fuzzy Efficiency Classifier System introduced a number of additional features to correct the aforementioned problems:

- introduction of additional performance (called efficiency) parameters for the classifiers and for the global system;
- making the reward dependant on the global performance of the system;
- elitist reproduction based on two parameters (strength and performance);
- introduction of a so-called 'guided mutation operator', generating likely better mutations based on performance characteristics;
- introducing fuzzy sets to overcome the limitation of parameters to be discrete.

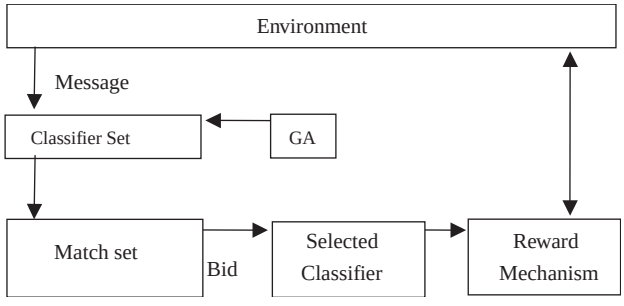

Fig. 1. Basic classifier system.

The *efficiency* of a classifier (or more general, the total classifier system) can be described using the following parameters:

- $a(t)$ the *accuracy* or number of accumulated successful rewards of the classifier or classifier system
- $g(t)$ the *generalism* or total number of accumulated selections (successful or not) of the classifier or classifier system

Effficiency $E(t)$ is then defined as

$$E(t) = \frac{a(t)}{g(g)}$$

with $0 \leqslant E(t) \leqslant 1.0$.

$E(t)$ is the maximum (1.0) indicating that all selections were successful and minimum (0.0) indicating that no successful selections were made up to time t.

The following specific efficiency parameters can be defined:

- *Global efficiency* $E_g(t)$: the efficiency of the whole classifier system at time t is calculated as the ratio of all the successful rewards to all presented environment messages. It describes the efficiency of the whole classifier system at time t. A 1.0 efficiency would indicate a classifier system which responds to all environment messages with 100% successful response.
- *Virtual efficiency* $E_v(i,t)$: all classifiers i within the match set M are considered (virtually) selected and for each of them it is determined if they would be eligible for reward, resulting in an virtual efficiency at time t. The resulting ratio determines $E_v(i,t)$. They are not necessarily selected within the BBA for effective reward.
- *Real efficiency* $E_r(x,t)$: only the classifiers $x \in M$ selected by the BBA (for possible reward) are considered when calculating real efficiency at time t.

An ECS cycle can basically be described as follows:

The environment generates a message, which is presented to the classifier system. From the classifier set S the matching classifiers are selected and presented in the matching set M. During this phase, the value for $E_v(i,t)$ is upgraded for all matching classifiers i.

From the match set M the highest strength classifier x is selected and checked if it is eligible for reward. At the same time $E_r(x,t)$ (for the selected classifier x) and $E_g(t)$ (for the whole classifier system) are upgraded. The strength reward of the classifier x (if successful) is calculated based upon $E_g(t)$. After a fixed number of such cycles has occurred the GA will introduce a number of new classifiers within the classifier set based upon the classifier strength, $E_r(x,t)$, $E_v(i,t)$ and $E_g(t)$.

ECS (and other LCS configurations) are suitable for the generation and evaluation of rules that are composed from discrete parameters. But the great majority of the 'real' environments and processes has continuous parameters where the conversion to a functional classifier format is less evident. To overcome this deficit a solution is suggested in which the continuous parameters are divided in 'fuzzy sets' where the parameter is split in a number of fuzzy classes and the ECS algorithm is extended for the processing of classifiers on the basis of their membership degree (the so-called Fuzzy Efficiency Classifier System). Each (continuous) message of the environment that is presented to the ECS can be compared to all (discrete) classifiers (belonging to the classifier list) where a corresponding membership function is calculated. The membership function is a measure for the applicability of the classifiers on the presented example, introducing fuzzy classes to allow handling of continuous values.

A more in-depth discussion of FECS is outside the scope of this paper but a detailed analysis, including a discussion of fuzzy membership degrees and defuzzification of the results, can be found by Sette and Boullart (2000), Sette (1998) and by Boullart and Sette (1998).

3. Genetic programming

3.1. Introduction

A schematical overview of the GP algorithm is given in Fig. 2. The following steps can be distinguished:

1. Generation of a random population of CP.
2. Evaluation of the fitness of all CPs in the population. Moreover, if a certain criterion is reached (for example a certain fitness threshold), the algorithm is terminated and the CP with the highest fitness is selected as the final result.
3. Replacing the current population by a new population by means of applying genetic operators (reproduction, crossover and mutation) probabilistically.
4. Return to step 2.

It is clear that this procedure is nearly identical to the one followed in a 'classic' GA. The major difference will be the representation and the corresponding fitness evaluation of the CP and this will be discussed in the next paragraph.

3.2. Representation of the computer program (cp)

In a GA a population member is an (often binary) coded representation of a number. However, a population member in GP is a hierarchically structured (computer) program consisting of functions and terminals. The functions and terminals are selected from a set of functions and a set of terminals. For example, a function set F could contain the basic arithmetic operations: $F = \{+, -, *, /\}$. However, the function set may also include other mathematical functions, Boolean operators, conditional operators or any user defined operators. For each of the operators the number of arguments ('arity') has to be defined. For F the corresponding number of operators is given by {2, 2, 2, 2}. The terminal set T contains the arguments for the functions. For example $T = \{x, y\}$ with x and y two independent variables.

A CP can now be depicted as a rooted, point-labelled tree with ordered branches, using operations (internal points of the tree) from the function set and arguments (leaves of the tree) from the terminal set.

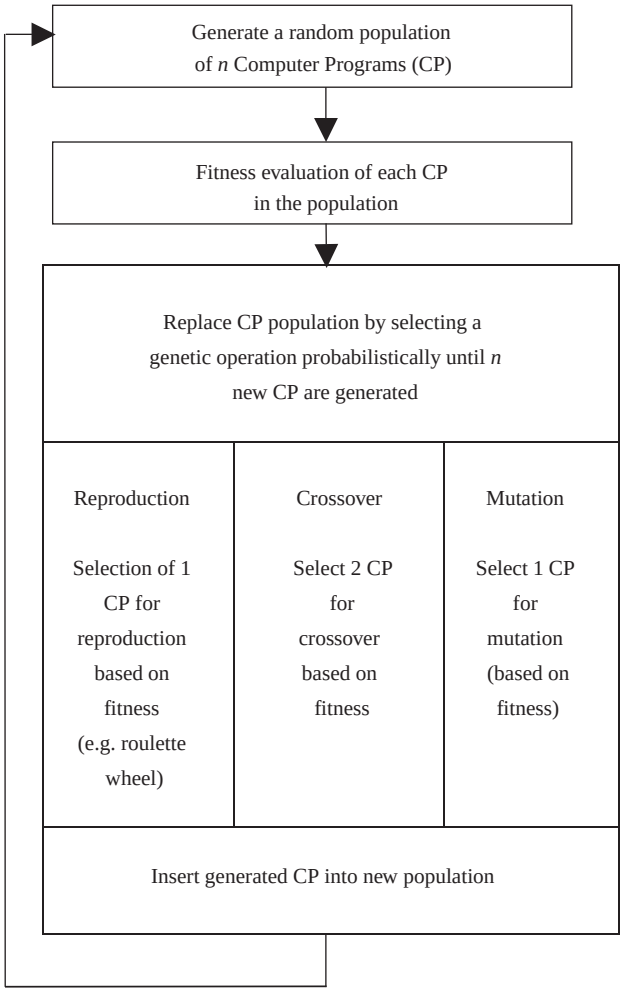

Fig. 2. Schematical overview of GP.

4. Experimental setup

The dataset for the fibre-to-yarn process was selected from the database described by Sette et al. (1996, 1998).

This database was originally built within the framework of a BRITE/EURAM project (BRITE, 1990–1993) and consisted of twenty different cotton types for which five different spinning machine settings were systematically changed. For all these settings it was experimentally determined whether the fibre was spinnable or not, resulting in a dataset of 1260 spinnable and 900 unspinnable experiments. The whole fibre-to-yarn database (2160 data samples) was used with regard to the prediction of spinnability, while the 1260 spinnable experiments were used in the prediction of yarn strength.

The dataset used for modelling is a subset of the aforementioned database and was constructed as follows:

- 5 machine parameters m_i (m_1 = yarn count, m_2 = twist, m_3 = navel, m_4 = breaker and m_5 = rotor) with several possible *discrete* settings were used to set up the machine;
- 5 different *continuous* fibre characteristics f_i were selected (f_1 = length, f_2 = uniformity, f_3 = strength, f_4 = elongation and f_5 = micronaire).

For FECS, the rules were all coded using a ternary alphabet {0, 1, #} with the '#' symbol representing either a 0 or 1.

For the GP implementation, the LISP code is a modified version of the code presented by Koza (1991). It now supports multiple dimensional input parameters, reading learning data from an ASCII file and more functions in the GP function set. The function set consisted of:

- the basic functions '+, −, ∗, ∕'
- a maximum/minimum function 'max, min' selecting maximum/minimum of two parameters
- a step function S which rounds a parameter to 1.0 if the parameter is larger then 0.5 or 0.0 otherwise.
- a (protected: only positive values) square root function
- cos, sin functions

5. First results

Using *FECS*, 123 rules (also called classifiers) have been generated, resulting in a total accuracy of 94% for spinnability. For yarn strength a rule set of 119 classifiers was generated with a predictive accuracy of 91%. Tables 1 and 2 give the 10 most important rules generated by FECS for predicting spinnability and yarn strength (ordered according to importance: rule 1 being the least important and rule 10 the most important).

As demonstrated in the legend of Table 1, each classifier has *a* (*real life*) *meaning*. An indication of the importance of each input parameter is given in Table 3, by summing up its meaningful selections (at least one 0 or 1) in the 10 most important rules.

Using *GP* for spinnability Sp, the following GP generated equation showed an accuracy of 90.1% towards the whole database:

$$\mathrm{Sp} = S(\mathrm{Max}(\mathrm{Max}(\mathrm{Max}(m_1, m_3 + m_2), S(m_1)) \times (-4.6/m_5)), m_3 - m_5). \quad (1)$$

As a consequence, following the GP, spinnability is only dependent on the machine parameters m_1, m_2, m_3, m_5: i.e. 90% of the database is correctly described without the use of any fibre parameter f_i.

Using *GP* for tenacity Str, the following GP generated equation showed an accuracy of 90% towards the whole database:

$$\mathrm{Str} = \sin(f_3\sqrt{|\sin(\cos(f_3 - 6.0)) - \sin(m_1)|}). \quad (2)$$

This equation shows that a reasonable good accuracy for predicting the tenacity can be reached using only *two* parameters: fibre strength f_3 and yarn count m_1.

Table 1
Ten Most important rules for predicting spinnability

Input										Output	Rule nr	Reward
m_1	M_2	m_3	m_4	m_5	f_1	f_2	f_3	f_4	f_5			
#0	01	0	##	1	0	0	#	#	0	1	1	44
##	0#	#	10	0	0	#	1	#	#	1	2	45
0#	#0	1	##	1	#	#	#	0	#	0	3	46
00	01	1	##	1	#	#	#	#	#	0	4	51
0#	10	1	0#	0	#	#	#	#	#	1	5	80
0#	00	1	##	1	#	#	#	#	#	0	6	86
##	01	1	0#	0	#	#	#	#	#	1	7	115
01	01	0	##	#	#	#	#	#	#	1	8	117
#1	10	0	##	#	#	#	#	#	#	1	9	118
##	##	1	##	1	#	0	0	#	#	0	10	214

E.g. rule number 10 {## ## 1 ## 1 # 0 0 # # : 0} is interpreted as follows: if navel is high ($m_3 = 1$), rotor speed is high ($m_5 = 1$) and fibre uniformity and strength are low ($f_2 = 0$ and $f_3 = 0$) then all other input parameters do not matter (##, #) the fibre is not spinnable (output = 0).

Table 2
Ten most important rules for predicting the yarn strength

Input										Output	Rule nr	Reward
m_1	M_2	m_3	m_4	m_5	f_1	f_2	f_3	f_4	f_5			
1#	#0	#	00	0	#	1	1	#	#	2	1	14.8
01	01	0	0#	#	#	1	1	#	#	2	2	15.4
01	00	#	00	0	#	#	1	#	#	2	3	16.8
10	##	#	0#	1	#	#	#	0	#	1	4	16.9
#1	#0	1	0#	0	0	#	0	#	#	0	5	17.9
1#	00	0	##	#	#	#	1	#	#	2	6	20.5
0#	00	0	0#	#	0	#	0	0	#	0	7	21.9
0#	00	0	01	1	0	#	0	#	#	0	8	23.8
10	#1	#	##	#	#	#	1	#	#	2	9	48.2
00	0#	1	0#	0	0	#	0	#	#	0	10	117.2

Table 3
Meaningful selections (FECS) for each input parameter

	m_1	m_2	m_3	m_4	m_5	f_1	f_2	f_3	f_4	f_5
Spinnability	7	9	9	3	8	2	2	2	1	1
Strength	10	9	6	8	6	4	2	9	2	0

Comparing the results from Table 3 with Eq. (1) it is clear that the top scoring parameters for spinnability (m_1, m_2, m_3 and m_5) are effectively the only ones used in the GP generated Eq. (1) for spinnability. The same can be said for the prediction of yarn strength (Eq. (2)), although somewhat less pronounced (some other parameters not included in the GP equation also have considerable values).

When comparing the results from both algorithms it becomes clear that:

- GP is slightly less accurate then FECS (somewhere between 1% and 4%).
- Although supplied with 10 input parameters, GP selected only a few parameters (4 for spinnability and 2 for yarn strength) to reach good accuracy, while FECS used all the 10 available input parameters. This basically means that 90% of the database can be accurately modelled using only 2–4 parameters. An increase of only 3% accuracy requires at least an additional 6 parameters. (Using a neural network and 17 input parameters, accuracy can still be improved by another 3%.) In this case, accuracy seems to be highly dependable on the number of parameters, which are taken into consideration, as could be expected.
- The results of GP and FECS showed several similarities. The GP selected parameters correspond to the FECS evaluation of the most important input parameters.

6. Comparing the rule sets from genetic-based machine learning and genetic programming

In order to have a better insight in the comparison of the GP and FECS results, the GP equations can be converted to a rule-alike form. Eq. (2) for yarn strength can easily be converted (with some loss of accuracy) to the ruleset given in Table 4 by implementing, respectively, 2 and 3 (fuzzy) classes for f_3 and m_1.

It can be seen that all FECS rules, except rule 4, have an immediate equivalent in the above GP derived rules. For example: the most important FECS rule {00 0# 1 0# 0 0 ♯ 0 # # : 0} (number 10 from Table 1, the positions in bold correspond with f_3 , m_1 and strength) is clearly a more specific version of the first rule {0♯ 0 : 0} in Table 4.

A similar procedure can be applied to the GP Equation (1) for spinnability, resulting in the ruleset given in Table 5.

Again all FECS rules have a corresponding GP derived rule. It can be seen however that the GP derived rules are more general and therefore convert to simpler 'rules of thumb'. For example:

The first rule for yarn strength {0# 0 : 0} reads 'If the fibre strength is low and the yarn count is low or medium then the resulting yarn strength will be low'

The first rule for spinnability {## ## 1 1 : 0} reads 'If the navel and rotor speed are high, yarn count and twist do not matter, the yarn is not spinnable'.

Finally, an overview of some of the advantages and disadvantages of GP/FECS is depicted in Table 6.

Table 4
Rule reduction of strength equation

M_1	f_3	Strength	FECS rules nr
0#	0	0	5, 7, 8 and 10
10	0	1	(4)
##	1	2	1, 2, 3, 6 and 9

Table 5
Rule reduction of spinnability (Spin) equation

m_1	m_2	m_3	m_5	Spin	FECS rules
##	##	1	1	0	3,4,6,10
##	##	1	0	1	5,7,(2)
#1 or 1#	##	0	#	1	1,8,9,(2)

Table 6
Overview of advantages/disadvantages of GP/FECS

	FECS	GP
Learning time	High	Very high
Executing speed	Fast	Very fast
Data compression	High (a few dozen rules, 10 parameters)	Very high (1 equation, 2 to 4 parameters)
Accuracy	High	Medium
Physical information	More detailed	More general

7. Conclusions

FECS generates a rule sets with a predictive accuracy of 94% for spinnability and 91% for yarn strength. The GP generated mathematical equations allowed a correct overall prediction in all cases (prediction of spinnability and yarn strength) of at least 90%. Also, the resulting equations have a limited complexity (eliminating several input parameters) and give the user an insight into the importance of the database parameters. This corresponded closely with the results obtained using genetic based machine learning. Moreover, the GP equations could be converted into similar (but more general) rules as those generated by FECS. GP gives a more elementary model using (very) few parameters, but otherwise with similar results to FECS.

References

Boullart, L., Sette, S., 1998. High performant learning classifier systems. Proceedings EIS (Engineering of Intelligent Systems). Fuzzy Logic/Genetic Algorithms, Vol. 1, pp. 249–256.

BRITE/EURAM project BREU 00052-TT (1990–1993). Research for a mathematical and rule based system which allows to optimise a cotton mixture, based on the interdependence of significant fibre properties, process parameters, yarn properties and spinning machinery performances.

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimisation, and Machine Learning. Addison-Wesley, Reading, MA.

Koza, J.R., 1992. Genetic Programming, On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge.

Sette, S., 1998. Lerende systemen door middel van evolutionaire algoritmen (Learning systems by means of evolutionary algorithms). Ph.D. Thesis, University Ghent, Faculty of Applied Sciences, Department of Textiles (in Dutch).

Sette, S., Boullart, L., 2000. An implementation of Genetic algorithms for Rule Based Machine Learning. EAAI (Engineering Applications of Artificial Intelligence) 13 (4), 381–390.

Sette, S., Boullart, L., Van Langenhove, L., 1996. Optimising a production process by a neural network/genetic algorithm approach. EAAI (Engineering Applications of Artificial Intelligence) 9 (6), 681–689.

Sette, S., Boullart, L., Van Langenhove, L., 1998. Using genetic algorithms to design a control strategy of an industrial process. Control Engineering Practice 6, 523–527.