Decision Making in the Medical Domain: Comparing the Effectiveness of GP-Generated Fuzzy Intelligent Structures.

Athanasios Tsakonas, and Georgios Dounias
University of the Aegean, Dept. of Business Administration,
8 Michalon Str., 82100 Chios, Greece,
Phone: +30-271-35165, Fax: +30-271-93464
e-mail: tsakonas@stt.aegean.gr, g.dounias@aegean.gr

ABSTRACT: In this work, we examine the effectiveness of two intelligent models in medical domains. Namely, we apply grammar-guided genetic programming to produce fuzzy intelligent structures, such as fuzzy rule-based systems and fuzzy Petri nets, in medical data mining tasks. First, we use two context-free grammars to describe fuzzy rule-based systems and fuzzy Petri nets with genetic programming. Then, we apply cellular encoding in order to express the fuzzy Petri nets with arbitrary size and topology. The models are examined thoroughly in four real-world medical data sets. Results are presented in detail and the competitive advantages and drawbacks of the selected methodologies are discussed, in respect to the nature of each application domain. Conclusions are drawn on the effectiveness and efficiency of the presented approach.

KEYWORDS: hybrid computational intelligence, medical diagnosis, aphasia, Aachen Aphasia Test, genetic programming, heuristic classification, evolutionary computation, intelligent systems.

INTRODUCTION

Genetic Programming (GP) is a search methodology belonging to the family of evolutionary computation (EC). These algorithms nowadays have been applied in a wide range of real-world problems. Among successful EC implementations, GP retains a significant position due to its valuable characteristics, such as the flexible variable-length solution representation and the absence of population convergence tendency. Genetic programming in its canonical form enables the automatic generation of mathematical expressions or programs. Grammar-guided genetic programming (G^3P) for knowledge discovery is an extension to the original GP concept and it makes possible the efficient automatic discovery of empirical laws. It relates to the Machine Discovery framework, originally described by Langley [17], which incorporated inductive heuristics and suffered from limitations regarding ill-conditioned data and large search spaces [27]. Genetic programming however can avoid these problems due to its stochastic nature.

In the present paper, grammar-guided genetic programming is applied in medical diagnosis. Various computational intelligent approaches have been applied to medical problems in the past, including artificial neural networks (NN) and fuzzy systems [18] and evolutionary algorithms [23]. The problems addressed in this work have been used extensively as benchmarking data in the machine learning society. This data consists of four diagnosis problems from the Proben1 collection [25] of real-world data sets. We applied two methods. The first approach fuzzy rule-based classifiers. The second approach creates fuzzy Petri-nets (FPN). For the latter methodology, we describe the cellular encoding paradigm in the GP grammar, in order to allow arbitrary network sizes and topologies. The models we selected to test have different properties for the discovery of empirical laws, a fact that makes useful the comparison of their effectiveness in dissimilar problems. Their difference is noteworthy in terms of their structure (competitive bases, networks). Specifically, the fuzzy model generates competitive fuzzy rule-based classifiers. On the other hand, the fuzzy Petri-nets methodology creates cooperative fuzzy rule networks.

The objective of this work is first to demonstrate the applicability of the G^3P in a range of different medical problems. Second, to compare the effectiveness of the approaches and try to point out advantages and drawbacks of each approach in respect to the characteristics of each application domain. Additionally, the selected problems make possible for the reader to measure their relative success by further comparing the results to those found in literature [4, 25].

The paper is organized as follows. In section II we introduce the theoretical background of GP. We also present the G3P principles, context-free grammars and current research in the field of cellular encoding. In section III we give a short description of the data used in our experiments. Section IV presents the experimental setup of our models. Our results regarding effectiveness and training time are presented in section V. The last section of this paper concludes with comparison of results and with a short discussion on future perspectives regarding this work.

BACKGROUND

As part of the Evolutionary Computation algorithms, genetic programming has been widely applied in a series of real-world problems. In general, evolutionary models were inspired by the Darwinian theory of evolution. According to the most common implementations, a population of candidate solutions is maintained, and after a generation is accomplished, the population is expected to be better fitted for a given problem. Three genetic operators are mostly used in these algorithms:
reproduction: copies an individual without affecting it
recombination (crossover): exchanges genetic material between two individuals
mutation: exchanges a part of a randomly selected genetic material

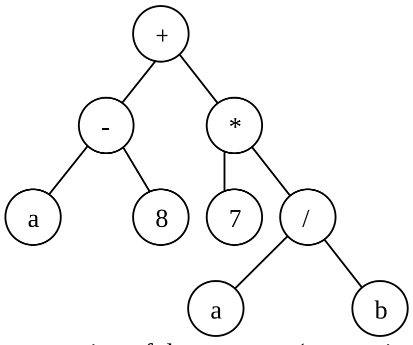

Figure 1. Tree representation of the program (expression): (a-8)+7*(a/b).

An evolutionary algorithm is summarized in the following steps:
1. Initialize a population of individuals at random.
2. Evaluate randomly an individual and compare its fitness to other (this fitness determines how closely is an individual to the desired goal).
3. Modify an individual with a relatively high fitness using a genetic operator.
4. Repeat steps 2-3 until a termination criterion is met.
Usual termination criterions appear to be the accomplishment of a number of generations, the achievement of a desired classification error, etc. Genetic programming uses tree-like individuals that can represent mathematical expressions, making valuable the application of GP in symbolic regression problems. Such a GP individual is shown in Figure 1.

GRAMMAR-GUIDED GENETIC PROGRAMMING

The prime advantage of genetic programming over genetic algorithms, is the ability to construct functional trees of variable length. This property enables the search for very complex solutions that are usually in the form of a mathematical formula - an approach that is commonly known as symbolic regression. Later paradigms extended this concept to calculate any boolean or programming expression. Thus, complex intelligent structures, such as fuzzy rule-based systems or decision trees have already been used as the desirable intention in genetic programming approaches [1, 15, 31, 32, 33]. The main qualification of this solving procedure is that the feature selection, and the system configuration, derive in the searching process and do not require any human involvement. Moreover, genetic programming, by inheriting the genetic algorithms' stochastic search properties, does not use local search -rather uses the hyperplane search-, and so avoids driving the solution to any local minimum. The potential gain of an automated feature selection and system configuration is obvious; no prior knowledge is required and, furthermore, not any human expertise is needed to construct an intelligent system. Nevertheless, the task of implementing complex intelligent structures into genetic programming functional sets in not rather straightforward. The function set that composes an intelligent system retains a specific hierarchy that must be traced in the GP tree permissible structures. This writing offers two advantages. First, the search process avoids candidate solutions that are meaningless or, at least, obscure. Second, the search space is reduced significantly among only valid solutions. Thus, a genotype - a point in the search space- corresponds always to a phenotype - a point in the solution space. This approach -known as legal searchspace handling method [36]- is applied in this work using context-free grammars. As we will discuss in the next paragraph, the implementation of constraints using a grammar can be the most natural way to express a family of allowable architectures. While each intelligent system -such as a fuzzy system- has a functional equivalent -by means of being composed by smaller, elementary functions-, what defines and distinguishes this system is its grammar.

CONTEXT-FREE GRAMMARS

Although powerful in its definition, the genetic programming procedure may be proved greedy in computational and time resources. Therefore, when the syntax form of the desired solution is already known, it is useful to restrain the genetic programming from searching solutions with different syntax forms [10, 20]. The most advantageous method to implement such restrictions among other approaches [21], is to apply syntax constraints to genetic programming trees, usually with the help of a context-free grammar declared in the Backus-Naur-Form (BNF) [9, 14, 22, 29]. The BNF-grammar consists of terminal nodes and non-terminal nodes and is represented by the set {N,T,P,S} where N is the set of non-terminals, T is the set of terminals, P is the set of production rules and S is a member of N corresponding to the starting symbol. The use of the terms terminal and non-terminal in a BNF-grammar, does not correspond to what Koza defines as terminal and function. Rather, a function -a non-terminal node in terms of the GP tree architecture- is expressed as terminal in a BNF grammar. To avoid confusion, the use of the terms GPFunction and GPTerminal -instead of the ambiguous terms function and terminal- has been proposed [34] and is adapted throughout this paper. The construction of the production rules can be the most critical point in the creation of a BNF grammar, since these production rules express the permissible structures of an individual. An example grammar expressing a class of individuals, which can produce the program in Figure 1, is composed by the following sets:
N = {EXPR, OP}
T = {-,*,/,a,b,7,8}
S = <EXPR>

Then, P is expressed as shown in Table I.

Table I. Grammar used for a simple example tree

Symbol	Rule
<EXPR>	::= <EXPR> <OP> <EXPR> | <VAR> | <NUMBER>
<OP>	::=- | * | /
<VAR>	::= a | b
<NUMBER>	::= 7 | 8

CELLULAR ENCODING

Although mapping decision trees or fuzzy rule-based systems to specific grammars can be relatively easy to implement, the execution of massively parallel processing intelligent systems -such as the neural networks- is not forthright. In order to explore variable sized solutions, usually a kind of *indirect encoding* is applied. The most common one is the *cellular encoding* [8], in which a genotype can be realized as a descriptive phenotype for the desired solution. More specifically, within such a function set, there are elementary functions that modify the system architecture together with functions that calculate tuning variables. Current implementations include encoding for feedforward and Kohonen neural networks [8,12] and fuzzy Petri-nets [35]. In his original work, Gruau also used a context-free grammar - a BNF grammar- to encode indirectly the neural networks. On the other hand, in [35] a logic grammar - a context-sensitive one- is adapted to encode fuzzy Petri-nets. In our work, we show that as long as the depth-first execution of the program nodes of a GP tree is ensured -which is the default-, a context-free grammar such as a BNF grammar is adequate for expressing neural networks. Gruau's original work has been facing some skepticism [11] on the ability to express arbitrarily connected networks. Later developments [10] seem to offer less restrictive grammar, though the *cut* function[1] still maintained bounded effect. In our approach, we inherit present grammar advances proposed in [35] in his logic grammar for fuzzy Petri-nets and we suggest a BNF grammar for neural networks that is more descriptive than previous works.

THE DATA SETS

[1] an analytical explanation of the *cut* function is given in *Section IV*.

The models are tested in three different setups of four data sets from the medical domain. These data sets have been taken unmodified by a collection of real-world benchmark problems, the Proben1 [25] that has been established for neural networks. The original data are derived by the UCI Machine Learning Repository [2]. In the Proben1 data set, modifications were applied for processing with NN and better comparability results. Specifically, first the values of every data attribute were standardized in the range [-1,1]. Then, nominal values were substituted using binary encoding, incrementing this way the number of inputs or outputs (see Table II). Missing values were substituted by standard -zero-values. Three different sequences of the examples were created. The outputs were represented using 1-of-n encoding where n outputs were used to express n classes. This encoding facilitated the winner-takes-all approach that is also followed in our experiments. Table II shows the problem complexity of these data sets. The first problem to be addressed is the Wisconsin Breast Cancer data. The goal is to diagnose between benign and malignant breast tumor. The second problem is the Pima Indians Diabetes data, where we diagnose between positive or negative diabetes. The problem of the Heart disease follows, where the goal is to diagnose if the diameter of a heart vessel is reduced by more than 50% or not. The last problem is to diagnose between thyroid hyperfunction, hypofunction or normal function.

Table II. Problem Description

Problem	Attributes	Inputs		Classes	Records used
		conti-nuous	discre-te		
cancer	9	9	0	2	696
diabetes	8	8	0	2	764
heart	13	6	29	2	916
thyroid	21	6	15	3	7196

DESIGN AND IMPLEMENTATION

Each data set was separated into a training set, a validation set and a test set. The training set consists of 50% of the data and the rest 50% is divided equally between the validation set and the test set. The separation of the examples into training, validation and test sets was performed in a loop manner. Specifically, the first two examples were assigned to the training set, then the next to the validation set and the fourth to the test set. This process was repeated until all the examples were assigned a set.

During the training phase, the validation set is typically used to avoid overfitting. A solution that has better classification score in the training set, is adapted as new best solution if and only if the sum of classification scores of both training and validation sets is the same or better than the best solution's respective score. We performed 20 runs for each data set. In all experiments, we used the same GP parameters.

Table III. GP parameters for G^3P

Parameter	Value
Population:	2,000 individuals
GP implementation:	Steady-state G^3P
Selection:	Tournament with elitist strategy
Tournament size:	6
Crossover Rate:	0.35
Overall Mutation Rate:	0.65
Node Mutation Rate:	0.4
Shrink Mutation Rate:	0.6
Killing Anti-Tournament size:	2
Maximum allowed individualsize:	650 nodes
Maximum number of generations:	100

It is accepted that the G3P procedure may suffer size problems during initialization [27]. Although the fine-tuning of our algorithm was not the main concern of this paper, we investigated various initialization approaches. Without claiming optimality, the GP parameters are presented in Table III. This setup, together with function selection probability optimization, offered for the presented grammars stable and effective runs throughout experiments. As it can be observed, this setup denotes our preference for significantly high mutation rates, especially shrink mutation [30] that slows down the code bloat caused by crossover operations. The optimization of function selection probabilities is consisted of giving more selection probability to GPTerminals rather than GPFunctions. Although the initialization of the population is random, using this probability bias the algorithm is 'forced' to generate individuals of acceptable size.

This optimization was decided after experimentation for each of the four implementations, since it was not possible to obtain a general principle regarding the most proper probability values.

G^3P FOR FUZZY RULE-BASED SYSTEMS

A fuzzy if-then rule [13], can be in the form:

$$if\ x\ is\ A\ then\ y\ is\ B\ with\ C,\ \ C \in [0,1]$$

where the *"x is A"* is the antecedent (or premise) set, *"y is B"* is the consequent (or conclusion) set, and *C* is the *certainty factor*. In fuzzy reasoning, the traditional two-valued logic, the *modus ponens,* is used in a generalized form. Namely, a fact may be more or less true, based on the truth of another fact. A fuzzy set is defined as:

$$A = \{(x, \mu_A(x)) \mid x \in X\}$$

where, the $\mu_A(x)$ is a *membership function* for the fuzzy set. Fuzzy sets are seen an extension to the classic sets that have a crisp boundary, where the transition for a value from belonging to a set and not belonging to the set is gradual and characterized be the membership function.

Table IV. Production Rules for Fuzzy Rule-Based Systems

Symbol	Rule
<TREE>	::=<RL> |<RULE>
<RL>	::=RL <TREE> <TREE>
<RULE>	::=RULE <COND> <CLASS>
<COND>	::=<IF> | <AND>
<IF>	::=IF <INP> <FS>
<AND>	::=AND <COND> <COND>
<CLASS>	::=THEN <OUT> <CLASS_VALUE>
<FS>	::=S | M | L
<INP>	::=X1 | X2 | X3 | X4
<CLASS_VALUE>	::=CLASS1 | CLASS2 | CLASS3 …
<OUT>	::=Y

The membership functions are described as a mathematical formula. The *X* is called the *universe of discourse*, and it may be comprised by discrete or continuous values. When the universe of discourse X is a continuous space, several fuzzy sets are used, most times covering the X uniformly. These fuzzy sets often are given linguistic terms such as "Small" or "Medium", thus they are called *linguistic variables*. These linguistic variables are used in fuzzy rules, which are interpreted as fuzzy relations using *fuzzy reasoning*.

Table V.Functions for the simulation of a Mamdani-model classifier

Function	Pseudo-code
RL (arg1,arg2)	If absolute(arg1)>absolute(arg2) then return arg1; else return arg2
RULE(arg1,arg2)	Return arg1*arg2
IF(arg1,arg2)	Fuzzify (arg1), based on the (arg2) value, return weight
AND(arg1,arg2)	Return minimum(arg1,arg2)
THEN(arg1,arg2)	If arg1=arg2 then return 1; else return -1
L, M, L, etc.	Return a constant value (e.g -1 for L, 0 for M etc.).
CLASS1, CLASS2, etc.	Return a constant value (e.g 1 for CLASS1, 2 for CLASS2, etc.).
X1,X2, etc.	System inputs (assuming a numerical value)
Y	System output (assuming a numerical value)

Fuzzy reasoning contains inference rules that derive conclusions from a set of fuzzy rules and input data. In the Mamdani classifier model using the max-min composition, several steps are followed to perform fuzzy reasoning. Firstly, we compare the input data with the antecedent sets of the fuzzy rules and we get the degrees of compatibility (called *weights*) with respect to these antecedent sets. Then, we combine these degrees using fuzzy AND or OR to obtain a *firing strength*, which shows the degree that a rule is satisfied. The firing strength corresponds to the certainty factor presented above. The max-min criterion, when only AND operators are used, will assign as firing strength the smaller of the antecedent degrees of compatibility. Finally, we obtain the overall output between the consequent sets of the rules. When the max-min composition is used, the rule with the larger firing strength will be the system's output. The definition of the grammar we used is shown in *Table IV* [1].

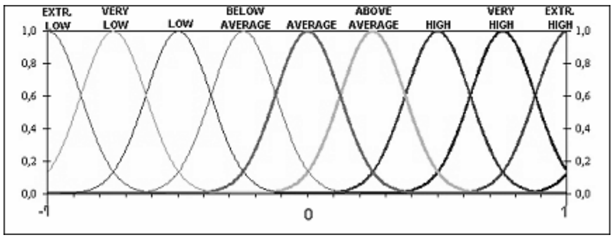

Figure 2. Membership functions of the fuzzy rule-based system.

This grammar describes a fuzzy system model with four inputs and one output. The GPFunctions used to describe the fuzzy mechanism, correspond to the words with bold in *Table IV*. We suggested the working shown in *Table V*, in order to simulate a Mamdani classifier.

The fuzzification is applied in *IF* nodes. The implementation uses Gaussian membership functions, and for a given Gaussian range *a* (standard for the *IF* nodes) , a center *c=arg2* and a value *x=arg1,* the function output will be the following:

$$n = e^{-\frac{1}{2}\left(\frac{x-c}{a}\right)^2}$$

In order to offer more degrees of freedom, we selected to use nine (9) membership functions, which are presented *in Figure 2*. The *THEN* node returns 1 if for the examining example the output (*arg1*), belongs to the class described by *arg2* and -1 otherwise. The reason to use this mechanism, together with the RL working, is to be able to know (when the tree evaluation is complete) whether the rule that fired was true or false. If the fired rule describes a false consequent set, the program value will be negative. While an individual represents a complete rule base, when examining an example during the training phase, this procedure will produce either positive or negative values indicating correct or wrong classification..

G^{3}P FOR FUZZY PETRI-NETS

A fuzzy Petri net [19] can be seen as a network that is constructed by input places, transitions and output places. The topology of a simple fuzzy Petri net is depicted in *Figure 3*. The framework of a fuzzy Petri net is finely correlated with the classification process of any pattern recognition task [24]. The input places are associated with the values of the features. These values are processed by the transitions of the network. The levels of firing of the network, depend on the parameters that are associated with each transition.

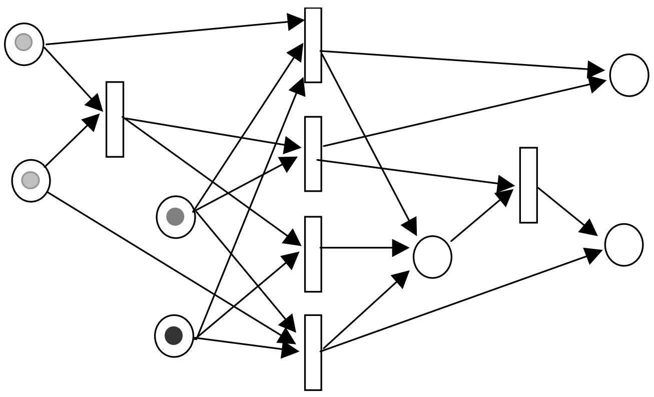
Figure 3. A fuzzy Petri net.

Consequently, an output place corresponds to a class. As with the case of NN in the previous paragraph, in order to handle a number of classes larger than two, this system adopts the *winner-takes-all* approach. Formally, a fuzzy Petri net is described by the following 8-tuple :

$FPN = (\boldsymbol{P}, \boldsymbol{T}, \boldsymbol{D}, I, O, cf, a, \beta)$

where $\boldsymbol{P} = \{P_1, P_2, ..., P_n\}$ is a finite set of places,

$\boldsymbol{T} = \{T_1, T_2, ..., T_n\}$ is a finite set of transitions,

$\boldsymbol{D} = \{D_1, D_2, ..., D_n\}$ is a finite set of propositions,

$\boldsymbol{P} \cap \boldsymbol{T} \cap \boldsymbol{D} = \varnothing$,

$I : \boldsymbol{T} \to \boldsymbol{P}^{\infty}$ is the input function, a mapping from transitions to bags of places,

$O : \boldsymbol{T} \to \boldsymbol{P}^{\infty}$ is the output function, a mapping from transitions to bags of places,

$cf : \boldsymbol{T} \to [0,1]$ is an association function, a mapping from transitions to real values in the range [0,1],

$a : \boldsymbol{P} \to [0,1]$ is an association function, a mapping from places to real values in the range [0,1],

$\beta : \boldsymbol{P} \to \boldsymbol{D}$ is an association function, a mapping from places to propositions.

Table VI. FPN Place-manipulating functions

	Name	Description	Number of arguments
Input place	sp1	Sequential division	3
	pp1	Parallel division	2
	in	Initialize the value	2
Intermediate place	sp2	Sequential division	3
	pp2	Parallel division	3
	stop	Terminate the modification	0

The incorporation of fuzzy Petri nets into G^3P was originally presented by [35]. That implementation made possible the description of fuzzy Petri nets of arbitrary size and topology. Unlike in [35], where a context-sensitive grammar is used to guide the genetic process, we present a context-free version of that grammar. The context-free equivalent, which applied in this work, satisfies the same descriptive rules. According to the methodology described in [35], manipulating functions are used to insert additional places and transitions. Since there are two types of modifiable places, the input and intermediate ones, we classify the place-manipulating functions into two types. These functions are presented in *Table VI*. Moreover, we have manipulating functions that are used on transitions. These transition-manipulating functions are shown in *Table VII*.

Table VII. FPN Transition-manipulating functions

Name	Description	Number of arguments
st	Sequential division	3
pt	Parallel division	2
cut	Remove one of the incoming edges	2
setcf	Set the certainty factor	1

Two variables were used to assist the simulation of the *breadth-first* execution, as with the model of the previous paragraph: a parameter array *Q* and a parameter value *V*. The working of these functions can be summarized as follows:

The *sp1* function takes three arguments. The first and the third argument can be *pp1* or *sp1* (or *in*) functions. The second argument is a transition manipulating function such as *pt*, *st* or *cut*. It calls sequentially the three arguments. Its application to the developing fuzzy Petri net is to add sequentially a new input place next to the place that is applied.

The *pp1* function has two arguments. They can be *pp1*or *sp1* (or *in*) functions. It feeds the arguments with copies of the array *Q*. It then saves the concatenation of them to array *Q*. It does not affect explicitly the variable *V*. Its modification to the developing fuzzy Petri net is to create an input place in parallel to the place that is applied.

The *in* function has one argument. This argument is one of the FPN inputs. It fuzzifies the input and initializes the array *Q* and the variable *V* to this value. We used five (5) Gaussian membership functions, which are shown in *Figure 4*.

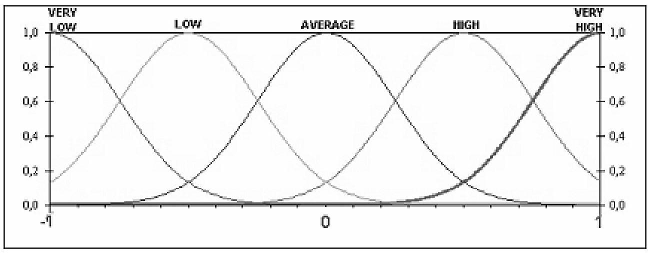

Figure 4. Membership functions of the fuzzy Petri net.

The *sp2* function has three arguments. The first and the third argument can be *pp2* or *sp2* (or *stop*) functions. The second argument is a transition manipulating function (*pt*, *st*, etc.). It calls sequentially the three arguments. Its application to the developing FPN is to add sequentially a place next to the place that is applied (in a similar fashion of the *sp1*).

The *pp2* function takes three arguments. The first and the third argument can be *pp2* or *sp2* (or *stop*) functions. The second argument is a transition manipulating function. Initially, it feeds the three arguments with copies of the array *Q* . Then, it saves the concatenation of the first and the third argument to the array *Q* . Its modification to the developing FPN is to create a place in parallel to the place that is applied, similar to *pp1*. It does not affect explicitly the variable *V*.

The *stop* function has no arguments. It signals the end of further modification. The array *Q* is initialized with the value *V*.

The *st* function has three arguments. The first and the third argument are transition manipulating functions. The second argument is a place manipulating function. It calls sequentially the three arguments. Its application to the developing neural network is to add sequentially a transition next to the transition that is applied.

The *pt* function has two arguments. They can be transition manipulating functions. It feeds the arguments with copies of the array *Q*. Its modification to the developing fuzzy Petri net is to create an transition in parallel to the place that is applied.

The *cut* function has two arguments. The first argument is an integer. The second one is a transition manipulating function. It cuts the connection that corresponds to the (*I mod N)* value -where *I* is the number of inputs to this place and *N* is the integer-, *iff* the number of inputs to this place is greater than 1. It passes down to the second argument the parameters *Q* and *V*.

The *setcf* function takes one argument. This argument is a *certainty factor* value. It performs the transition of the input value. The result is saved to variable *V*.

The BNF grammar is shown in *Table VIII*. The starting symbol is the <CLAUSE> symbol. This grammar corresponds to a binary decision fuzzy Petri net.

Table VIII.Production Rules for Fuzzy Petri Nets

Symbol	Rule
<CLAUSE>	::=<FPN>
<FPN>	::=<PROG> <PROG>
<PROG>	::=<PLACE1><TRAN>
<PLACE1>	::= SP1 <PLACE1> <TRAN> <PLACE2>
	| PP1 <PLACE1> <PLACE1>
	| <INIT>
<INIT>	::=<ATTR>
<PLACE2>	::=SP2 <PLACE2> <TRAN> <PLACE2>
	| PP2 <PLACE2> <TRAN> <PLACE2>
<TRAN>	::=PT <TRAN> <TRAN>
	| ST <TRAN> <PLACE2> <TRAN>
	| CUT <NUMBER> <TRAN>
	| SETCF <CF>
<NUMBER>	::=integer in [1,256]
<CF>	::=real in [0,1]
<ATTR>	::=data attribute (system input)

RESULTS AND DISCUSSION

In *Table IX* we present the training classification error (CE) for each model and data set. The presentation of these results is useful since, as stated previously, during the training phase we the best solution is the one that has the lower sum of the classification errors in both training and validation set. The validation and test CEs are shown in *Table X*. This setup seems to be restrictive for the G^{3}P for FPN model. More specifically, as it was expected, a high *correlation* between the size of the produced FPN and the training classification result was noticed. Hence, we consider that a larger GP model should be appropriate for the G^{3}P for FPN methodology in order to achieve competitive results. As it can be seen from *Table XV*, the average best solution size for fuzzy Petri nets, in our configuration is less than those of the FRBSs. The G^{3}P for FRBS is highly competitive. Its success comes in the `diabetes` and the `cancer` data.

Table IX. Classification Errors in Medical Data Training Sets.

Problem	G^{3}P for Fuzzy Rule-Based Systems			G^{3}P for Fuzzy Petri Nets		
	best	avg.	stddev	best	avg.	stddev
cancer1	1.14	2.08	0.55	3.16	3.44	0.25
cancer2	0.57	1.59	0.48	1.43	3.16	1.54
cancer3	0.28	1.01	0.46	1.14	2.61	1.62
diabetes1	15.70	17.25	1.11	20.41	23.62	1.96
diabetes2	17.01	19.89	2.08	21.46	23.82	1.84
diabetes3	16.49	19.80	2.23	21.72	23.76	2.03
heart1	14.62	16.24	1.52	19.43	20.18	0.99
heart2	15.28	17.14	2.04	20.74	20.89	0.13
heart3	15.06	15.65	0.77	21.17	21.25	0.16
thyroid1	5.33	6.66	1.09	6.19	7.19	0.86
thyroid2	5.58	6.37	1.03	5.78	6.71	0.91
thyroid3	5.80	6.75	1.33	5.78	6.41	0.74

In general, classification competence of fuzzy systems over crisp systems is expected in data sets having continuous features, a principle that is verified in our experiments. In *Table XI*, we show the average effective generations for the models and the data sets. These results show that in more difficult problems -such as ill-conditioned or large sized (e.g. thyroid data) problems- the algorithm needed slightly more training time to find the best solution. This is the same result found in [4] concerning the effective time of *linearGP*.

Table X. Validation and Test Classification Error Rates of G^{3}P for Fuzzy Rule-Based Systems and Fuzzy Petri Nets in Medical Data Sets.

Problem	G^{3}P for Fuzzy Rule-Based Systems						G^{3}P for Fuzzy Petri Nets					
	Validation CE (%)			Test CE (%)			Validation CE (%)			Test CE (%)		
	best	avg.	stddev	best	avg.	stddev	best	avg.	stddev	best	avg.	stddev
cancer1	1.72	2.87	0.83	2.29	4.39	1.42	2.29	3.06	0.69	2.87	4.31	0.94
cancer2	2.87	4.48	0.84	1.72	4.45	1.23	2.29	5.17	2.20	2.29	4.83	1.19
cancer3	4.02	5.08	0.61	3.44	4.90	0.83	4.02	6.26	1.49	2.29	4.42	1.43
diabetes1	26.17	28.75	1.44	21.98	26.47	3.40	24.08	27.53	2.26	23.03	26.82	2.56
diabetes2	24.08	25.47	0.97	23.56	24.78	1.22	24.60	27.78	2.22	23.03	27.08	2.65
diabetes3	23.56	24.95	1.02	21.99	24.25	1.64	26.17	28.63	2.19	24.08	28.21	2.16
heart1	23.14	25.29	1.25	18.34	22.12	1.93	24.89	25.76	0.79	23.14	25.63	1.44
heart2	16.59	18.55	1.44	20.96	23.68	1.89	21.83	22.12	0.25	21.83	25.32	3.73
heart3	17.90	25.76	3.08	25.32	25.76	0.61	21.39	22.85	2.15	24.89	25.90	1.76
thyroid1	5.55	6.77	0.90	5.28	6.08	0.57	6.28	7.17	0.77	5.55	6.26	0.69
thyroid2	5.94	6.54	0.82	5.44	5.95	0.52	6.03	6.75	0.79	5.44	5.95	0.60
thyroid3	6.00	6.80	1.13	5.44	6.03	0.82	6.17	6.51	0.61	4.22	5.49	0.84

CONCLUSIONS AND FURTHER RESEARCH

This work presented a comparison of two grammar-guided genetic programming approaches in medical diagnosis. Namely, fuzzy rule-based systems and fuzzy Petri nets were implemented using context-free grammars. Cellular encoding was adopted in order to describe arbitrary network topology for the fuzzy Petri nets. Four data sets were used from the medical domain. We performed twenty runs in three variations of these data for each of the approaches. The results allow driving conclusions on the effectiveness of this methodology. Specifically it was shown that fuzzy Petri nets approach should be used with a larger than the examined model, in terms of genetic population and training time. We suggest that a large GP model could make this *cellular encoding* model (FPN) more antagonistic in terms of classification accuracy. For this work, we encountered data sets each with two or three classes. We observed that the FRBS approach performed less better in data sets that had discrete features and it was more effective in data sets consisted of only continuous attributes. Furthermore, we consider that experiments with data sets having larger number of classes (such as the `mushroom` data from the *UCI Machine Learning repository*) would offer interesting observations on the relative performance of the models. In this work, the large number of the experiments led us to adopt a relatively small maximum number of generations. We suggest that a larger number (e.g. 200-300 generations) would offer fairly lower classification error of the best solution.

Table XI Average Best Solution Size of G^3P in Medical Data.

Problem	G^3P for Fuzzy Rule-Based Systems		G^3P for Fuzzy Petri Nets	
	avg.	stddev	avg.	stddev
cancer1	214	116	83	80
cancer2	193	135	92	76
cancer3	160	118	92	75
diabetes1	274	125	45	26
diabetes2	172	150	44	25
diabetes3	207	208	51	17
heart1	378	139	45	11
heart2	372	168	89	53
heart3	315	119	64	31
thyroid1	165	212	39	20
thyroid2	188	208	32	17
thyroid3	193	178	34	22

Research will follow this work regarding larger models that would enable to both methodologies to perform competitively. In the absence of a standard GP benchmarking data set, other real-world domains will be used in order to obtain transparent results on the classification success of this approach. More intelligent models -such as Kohonen networks- can be implemented in context-free grammars to offer more generalized knowledge discovery attempt. Finally, tuning properly the initialization procedure of a grammar-guided genetic programming will enable better exploration of the search space for a given size of the GP model.

Table XII. Effective training time of G^3P in Medical Data (in generations).

Problem	G^3P for Fuzzy Rule-Based Systems		G^3P for Fuzzy Petri Nets	
	avg.	stddev	avg.	stddev
cancer1	48	32	47	15
cancer2	43	29	54	30
cancer3	48	27	50	37
diabetes1	71	25	45	27
diabetes2	42	12	46	29
diabetes3	40	31	52	30
heart1	77	24	30	26
heart2	56	40	24	7
heart3	40	23	30	11
thyroid1	84	30	52	46
thyroid2	75	37	41	31
thyroid3	85	21	34	43

REFERENCES

[1] Alba E., Cotta C., Troya J.M., "Evolutionary Design of Fuzzy Logic Controllers Using Strongly-Typed GP", In *Proc. 1996 IEEE* Int'l Symposium on Intelligent Control., pp. 127-132. New York, NY., 1996

[2] Blake C., Keogh E., Merz C.J., UCI Repository of machine learning databases. Dept. Inform. Comp. Sci. Univ. California, Irvine, CA. [http://www.ics.uci.edu/~mlearn/ML-Repository.html]

[3] Bojarczuk C.C., Lopes H. S., Freitas A. A., "Genetic Programming for Knowledge Discovery in Chest-Pain Diagnosis", in IEEE Engineering in Medicine and Biology, pp. 38-44, July 2000

[4] Brameier M., Banzhaf W., "A comparison of Linear Genetic Programming and Neural Networks in Medical Data Mining", in IEEE Trans. on Evol.Comp., vol.5,no.1, Feb 2001,pp 17-26

[5] Dounias G., Tsakonas A., Jantzen J., Axer H. , Bjerregaard B. ,v.Keyserlingk D. G. , "Genetic Programming for the Generation of Crisp and Fuzzy Rule Bases in Classification and Diagnosis of Medical Data", in Proc. of First Int. NAISO Congress on Neuro Fuzzy Technologies, NF-2002, Habana, Cuba, Jan. 16, 2002

[6] Elstein A. S., Shulman Lee S., Sprafta S. A.,“Medical problem solving: an analysis of clinical reasoning“, Cambridge, Harvard University Press 1978

[7] Freitas A.A., "Genetic Programming Framework for Two Data Mining Tasks: Classification and Generalized Rule Induction", in Genetic Programming 1997: Proc. 2nd Annual Conf. (Stanford University, July 1997), pp 96-101, Morgan Kaufmann, 1997

[8] Gruau F., "Neural Network Synthesis using Cellular Encoding and the Genetic Algorithm", Ph.D. Thesis, Ecole Normale Superieure de Lyon, anonymous ftp:lip.ens-lyon.fr (140.77.1.11) pub/Rapports/PhD PhD94-01-E.ps.Z

[9] Gruau F., "On Using Syntactic Constraints with Genetic Programming", in P.J.Angeline, K.E.Jinnear,Jr., "Advances in Genetic Programming", MIT,1996

[10] Gruau F., Whitley D., Pyeatt L., "A Comparison between Cellular Encoding and Direct Encoding for Genetic Neural Networks", in Koza J.R., Goldberg D.E., Fogel D.B., Riolo R.L. (eds.), Genetic Programming 1996: Proceedings of the First Annual Conf., pp 81-89, Cambridge, MA, 1996, MIT Press

[11] Hussain T., "Cellular Encoding: Review and Critique", Technical Report, Queen's University, 1997, http://www.qucis.queensu. ca/home/hussain/web/1997_cellular_encoding_review.ps.gz

[12] Hussain T., Browse R., “Attribute Grammars for Genetic Representations of Neural Networks and Syntactic Constraints of Genetic Programming”, in AIVIGI’98:, Workshop on Evol.Comp., Vancouver BC, 1998

[13] Jang J.-S.R., Sun C.-T., Mizutani E., “Neuro-Fuzzy and Soft Computing”, Matlab Curicculum Series, 1997

[14] Janikow C.Z., "A Methodology for Processing Problem Constraints in Genetic Programming", in Computers Math.Applic. Vol.32:8,pp 97-113, 1996

[15] Koza J.R., "Genetic Programming: On the Programming of Computers by Means of Natural Selection", Cambridge, MA, MIT Press., 1992

[16] Koza J.R., Bennett III F.H., Andre D., Keane M.A., "Genetic Programming III", Morgan Kaufmann Publ., Inc.,1999

[17] Langley P., Simon H.A., Bradshaw G.L., "Rediscovering chemistry with the Bacon system", in Machine Learning: an artificial intelligence approach, Vol 1, Morgan Kaufmann, 1983

[18] Abbod M.F., von Keyserlingk D.G., Linkens D.A., Mahfouf M., "Survey of Utilization of Fuzzy Technology in Medicine and Health Care", in Fuzzy Sets and Systems, Vol 120, Issue 2:1, June 2001, pp. 331-349

[19] Looney, C.G., 1988 Fuzzy Petri nets for rule-based decision-making. IEEE Trans. Systems,Man and Cybernet. 18, 178-183

[20] Montana D.J., "Strongly Typed Genetic Programming", in Evolutionary Computation, vol. .3, no. 2, 1995

[21] N.Paterson, M.Livesey,"Evolving Caching Algorithms in C by GP", in Genetic Programming 1997, pp 262-267, MIT Press,1997

[22] Naur P., "Revised report on the algorithmic language ALGOL 60", Commun. ACM, Vol 6, No 1, pp 1-17, Jan 1963

[23] Ngan P.S., Wong M.L., Lam W., Leung K.S., Cheng J.C.Y.,"Medical data mining with evolutionary computation", in Artif.Intel.in Medicine, 16 (1999), 73-96

[24] Pedrycz W., "Generalized fuzzy Petri nets as pattern classifiers", in Pattern Recognition Letters 20 (1999), 1489-1498

[25] Prechelt L., "Proben1 - A set of neural network benchmark problems and benchmarking rules", Tech.Rep. 21/94, Univ. Karlsruhe, Karlsruhe, Germany, 1994

[26] Quinlan, J.R., “Induction of Decision Trees”, Machine Learning, Vol. 1, pp. 81-106, 1986.

[27] Ratle A., Sebag M., "Genetic Programming and Domain Knowledge: Beyond the Limitations of Grammar-Guided Machine Discovery"

[28] Riedmiller M., Braun H., "A direct adaptive method for faster backpropagation learning: The RPROP algorithm", in Proc. IEEE Intl. Conf. Neural Networks, San Francisco, CA,1993,pp 586-591

[29] Ryan C., Collins J.J., O'Neil M. , "Grammatical Evolution: Evolving Programs for an Arbitrary Language", in W.Banzhaf, R.Poli, M.Schoenauer, T.C.Fogarty (Eds.), "Genetic Programming", Lecture Notes in Computer Science, Springer, 1998

[30] Singleton A., "Genetic Programming with C++", BYTE Magazine, Feb 1994

[31] Tsakonas A., Dounias G., "Hierarchical Classification Trees Using Type-Constrained Genetic Programming", in Proc. of 1st Intl. IEEE Symposium in Intelligent Systems, Varna, 2002

[32] Tsakonas A., Dounias G., Axer H., von Keyserlingk D.G., "Data Classification using Fuzzy Rule-Based Systems represented as Genetic Programming Type-Constrained Trees", in Proc. of the UKCI-01, Edinbourgh, 2001, pp 162-168

[33] Tsakonas A. Dounias G., "A Scheme for the Evolution of Feedforward Neural Networks using BNF-Grammar Driven Genetic Programming", in Proc. of Eunite-02, Algarve, 2002

[34] Whigham P., "Search Bias, Language Bias and Genetic Programming", in Genetic Programming 1996, pp 230-237, MIT Press, 1996

[35] Wong M.L., "A flexible knowledge discovery system using genetic programming and logic grammars", Decision Support Systems, 31, 2001, pp 405-428

[36] Yu T., Bentley P., "Methods to Evolve Legal Phenotypes". In Lecture Notes in Comp.Science 1498, Proc. of. Parallel Problem Solving from Nature V, 1998, pp 280-291