3rd International Workshop on Genetic and Evolving Fuzzy Systems
Witten-Bommerholz, Germany, March 2008

Processing times estimation in a manufacturing industry through genetic programming

Manuel Mucientes, Juan C. Vidal, Alberto Bugarín and Manuel Lama

Abstract— **Accuracy in processing time estimation of manufacturing operations is fundamental to achieve more competitive prices and higher profits in an industry. The manufacturing times of a machine depend on several input variables and, for each class or type of product, a regression function for that machine can be defined. Time estimations are used for implementing production plans. These plans are usually supervised and modified by an expert, so information about the dependencies of processing time with the input variables is also very important. Taking into account both premises (accuracy and simplicity in information extraction), a model based on TSK (Takagi–Sugeno–Kang) fuzzy rules has been used. TSK rules fulfill both requisites: the system has a high accuracy, and the knowledge structure makes explicit the dependencies between time estimations and the input variables. We propose a TSK fuzzy rule model in which the rules have a variable structure in the consequent, as the regression functions can be completely distinct for different machines or, even, for different classes of inputs to the same machine. The methodology to learn the TSK knowledge base is based on genetic programming together with a context-free grammar to restrict the valid structures of the regression functions. The system has been tested with real data coming from five different machines of a wood furniture industry.**

I. Introduction

The estimation of the processing times of each one of the operations involved in the manufacturing of a product is an important task in any manufacturing industry [7]. An accurate estimation of the manufacturing time of the product allows competitive prices, higher profits, and also increase the client portfolio. Nevertheless, the scope of this task is even broader. On one hand, it constraints future material and storage capacity requirements. On the other hand, it constitutes the basis for the production scheduling process and thus will influence the manufacturing or production plan, and the logistic requirements to fulfill client orders in time.

The processing time of a machine can depend on several input variables, such as the dimensions of the product, the material, the speed of the machine for that kind of product, etc. Thus, a machine can have several regression functions (one for each type of product) for the estimation of the processing times. Therefore, a good estimation of the manufacturing time of a product requires not only an accurate regression function, but also the selection of the appropriate function among all the regression functions of the machine.

This work is been carried out in the framework of a R+D contract with Martínez Otero Contract, S.L., supported by the Dirección Xeral de I+D of the Xunta de Galicia through grant PGIDIT04DPI096E.

Manuel Mucientes, Juan C. Vidal, Alberto Bugarín and Manuel Lama are with the Department of Electronics and Computer Science, University of Santiago de Compostela, E-15782 Spain (e-mail: manuel@dec.usc.es).

The estimated processing times are generally used for implementing production plans: schedule a finite set of client orders in the manufacturing workload, given a finite set of resources that can perform the different manufacturing operations of a production plant. Production plans are implemented or modified with the supervision of an expert. Thus, it is fundamental that the expert can easily extract information about how the different processing times have been estimated. The simplicity to extract information from a regression function depends on the way the knowledge contained in that function is represented. In our case, the expert demands regression functions in which it is easy to evaluate the weight or contribution of each variable to the processing time. Moreover, the expert finds also very useful to discover relations among the input variables that affect time estimations. Also, it must be taken into account that the expert structures his knowledge for time estimations with polynomials of the input variables.

In summary, our proposal accomplishes two premises: high accuracy, but providing the expert with valuable and easy to extract information. Takagi–Sugeno–Kang (TSK) fuzzy rules [12], [11] seem to fit well for the requirements of processing times estimation in this case. Flexibility in the structure of the consequents of the TSK rules is fundamental for processing time estimations, but some restrictions in the structures have to be imposed, as not all of them are valid. A compact representation of the valid structures of the consequents can be defined through a context-free grammar. Thus our proposal is a methodology for learning TSK rules of variable structure in its consequent using genetic programming together with a context-free grammar to define the valid structures. The obtained knowledge base is a set of regression functions for processing time estimations in the wood furniture industry.

The paper is structured as follows: Sect. II introduces the problem of processing time estimation in the wood furniture industry, Sect. III presents the TSK rule model with variable structure in the consequent, while Sect. IV describes the evolutionary approach used for learning: a context-free grammar together with a genetic programming algorithm. Sect. V shows the results of the experiments with different examples sets, and compares the results with other methodologies. Finally, Sect. VI points out the conclusions.

II. Processing times in the furniture industry

Throughput time (TT) is defined in industrial production systems as the interval that elapses when the manufacturing system performs all of the operations necessary to complete

978-1-4244-1613-4/08/$25.00 ©2008IEEE

a work order. An accurate estimation of TT is hard to obtain in industries where custom-designed products are dominant, such as the furniture industry. The lack of previous manufacturing experiences and the complexity in the production makes time estimations difficult.

The reduction of TT has many benefits: lower inventory, reduced costs, improved product quality (problems in the process can be found more quickly), faster response to customer orders, and increased flexibility. The estimation of the processing time is one of the most important tasks in the product design life cycle.

This work is framed in a R+D project developed for a a wood furniture industry in which the experts make processing time estimations based on a polynomial approach. The values of polynomial variables are taken from the technical designs. In this paper, we will focus on the primary and secondary processing operations such as: sawing, drilling, sanding or laminating. For these operations, the input variables will be the dimensions of that pieces. This selection of the input variables is due to the way the manufacturing is carried out in the company. For example, the same sawing speed is always used for cutting solid wood independently of its moisture. In the next section, the model that has been followed for processing times estimation in this wood furniture industry is detailed. The model is based on a TSK fuzzy rule base in which the consequents follow a variable structure provided by an expert.

III. TSK RULE MODEL FOR PROCESSING TIMES ESTIMATION

The approach presented in this paper for processing times estimation looks for a high accuracy regression model, but maintaining the knowledge structure provided by an expert. The objective of the last premise is to make easy for the expert the extraction of information from the regression functions. Processing time estimations provide the expert with information to implement or modify a production plan. Thus, the simplicity in information extraction of the regression functions associated to a machine is as important as accuracy in time estimations. Of course, the type of information that an expert demands depends on the characteristics of the industry the system is being developed for. Also, sometimes, different experts ask for different types of information. Thus, the representation of the regression functions must fulfill the requirements of the expert from the point of view of information extraction. The expert demands the following characteristics for the representation of the functions:

- The weight or contribution of each input variable to the processing time estimation must be explicit.
- A regression function could depend on a new generated variable. This variable represents relations among input variables. For example, the product of the three dimensions of a piece of furniture generates the variable *volume*.

Also, in this particular case, the expert structures his knowledge for processing time estimations with polynomials of the input variables like:

$$Processing\ time = 100 \cdot length + 200 \cdot volume \tag{1}$$

With this type of knowledge the expert can estimate the times, but also extract that the processing time for a machine with such a regression function is sensible to all the dimensions of the piece of furniture. Nevertheless, if the processing time depends on the surface instead of the volume, the expert could modify a production plan taking into account that the thickness of the piece does not affect the time for that machine. For example, the thickest pieces could be assigned to the last machine, and the thinner ones to the other machine. So, from the expert's point of view, it is very important to maintain this knowledge representation to preserve simplicity for information extraction from the regression functions.

Following these prerequisites, processing times of a machine can be described as polynomials of several input variables. These variables can be combined in many different ways:

$$\sum_i \alpha_i \cdot \prod_{j=1}^{na} x_j^{\delta_{i,j}} \tag{2}$$

where α_i are the coefficients, x_j, $j = 1, \ldots, na$, are the input variables, and $\delta_{i,j}$ is an indicator variable defined by:

$$\delta_{i,j} = \begin{cases} 1 & \textit{if } x_j \in \textit{i-th term of the polynomial} \\ 0 & \textit{otherwise} \end{cases} \tag{3}$$

Moreover, a number of different polynomials, each one of them representing the estimation of the processing time of a class of the input variables, are needed to model a given machine. For example, in an specific machine, processing times of pieces with a thickness over a threshold are estimated with a polynomial, and under that threshold with another polynomial. Therefore, the learning process has to obtain a regression function for each class of the input variables to a machine.

Summarizing, given an input, first the system has to select a regression function for the machine, and then estimate the processing time. The regression function must follow the structure of Eq. 2, as one of the premises of the approach is to maintain the simplicity for information extraction. The model that best suites for processing time estimations under these conditions is a TSK fuzzy rule-based system [12], [11]. In a TSK rule, the output is obtained as a function of the input variables. This is exactly what is necessary to estimate processing times, maintaining the structure of the knowledge provided by the expert. The proposed rule model is:

$$\begin{aligned} R_k: \ & \text{IF } X_1 \textit{ is } A^1_{l^k_1} \text{ and } \ldots \text{ and } X_{na} \textit{ is } A^{na}_{l^k_{na}} \\ & \text{THEN } Y \textit{ is } \sum_i \alpha_i \cdot f_i(x_1, \ldots, x_{na}) \end{aligned} \tag{4}$$

where X_j is a linguistic variable, $A^j_{l^k_j}$ is a linguistic label of that variable, $l^k_j = 1, \ldots, nl_j$, Y is the output variable, and $f_i(x_1, \ldots, x_{na})$ are functions of the inputs variables (x_j).

These functions, f_i, can be defined in two different ways:

$$f_i(x_1,\ldots,x_{na}) = \begin{cases} \sum_{j=1}^{na} x_j \cdot \delta_{i,j} \\ \prod_{j=1}^{na} x_j^{\delta_{i,j}} \end{cases} \quad (5)$$

allowing all the possible combinations that are needed for processing time estimations.

The generation of a knowledge base with TSK rules of this type (Eq. 4) requires the definition of several linguistic labels, coefficients (α_i), and also functions (f_i) with different structures. The use of expert knowledge can help to reduce the search space, for example limiting the valid structures for the functions. Anyway, the generation of a knowledge base of this kind by an expert is really very complex and, accordingly, rules should be obtained with a learning algorithm. Evolutionary algorithms have flexibility in the representation of the solutions, and also we can manage the tradeoff between accuracy and simplicity for information extraction. In the next section, we describe in detail the evolutionary algorithm we have used to learn TSK rules following the model of Eq. 4 for processing time estimation in the wood furniture industry.

IV. Evolutionary algorithm

The structure of the consequent of the rules for processing time estimation can be very different from one rule to another. Also, this structure has restrictions: for example, a function (f_i) cannot be a combination of sums and products of variables (only sums or products). For these reasons, the most appropriate evolutionary algorithm is genetic programming [8]. In genetic programming, an individual is a tree of variable length. Each individual in the population can have a different structure, and the introduction of restrictions in that structure of the chromosome can be solved using, for example, a context-free grammar.

The proposed algorithm is based on the cooperative-competitive approach [6]: rules evolve together (cooperative) but competing among them to obtain the higher fitness. We have chosen token competition [9] as the mechanism for maintaining diversity of the population.

The learning process is based on a set of training examples. These examples are represented by tuples: (length, width, thickness, processing time), where the first three variables are the dimensions of a piece of furniture, and the output variable is the processing time for that piece in the machine. In token competition, each example of the training set has a token. All the individuals that cover an example have to compete to seize its token, but only one of them (the stronger one) can get it. During the evolutionary process, the individual with the highest strength in the niche will exploit it, trying to obtain the maximum number of tokens of the niche to increase its strength (and its fitness). On the other hand, the weaker individuals reduce their strength as they can not compete with the best individual in the niche.

A. Description of the context-free grammar

As has been pointed out, in genetic programming each individual is a tree of variable size. Thus, the structure of the individuals can be very different among them. In order to generate valid individuals of the population, and to produce right structures for the individuals after crossover and mutation, some restrictions have to be applied. With a context-free grammar all the valid structures of a tree (chromosome) in the population can be defined in a compact form. A context-free grammar is a quadruple (V,Σ,P,S), where V is a finite set of variables, Σ is a finite set of terminal symbols, P is a finite set of rules or productions, and S is an element of V called the start variable.

The grammar that defines the structure of the learned rules for processing time estimations, has been designed using expert knowledge. The information provided by the expert is:

- Input and output variables.
- Number of linguistic labels of the input variables.
- Valid structures for the consequent of the rules.

The grammar is described in Fig. 1. The first item enumerates the variables, then the terminal symbols, in third place the start variable is defined, and finally the rules for each variable are enumerated. When a variable has more than one rule, rules are separated by symbol |. Variable *rule* is the start variable of the grammar and generates two new nodes in the tree: *antecedent* and *consequent*. *antecedent* codifies the antecedent part of the rule with three propositions (ant_j), one for each of the input variables: length, width and thickness. Each proposition is defined by a linguistic label, $A_{l_j}^j$, or by symbol λ, which represents that no linguistic label is selected. The linguistic labels of each variable of the antecedent part have been obtained with a uniform partition of the universe of discourse of each input variable.

Variable *consequent* represents the consequent part of the rule as the sum of three (the number of input variables) mathematical expressions (*expression*). Each variable *expression* represents one of the elements of the summatory of the consequent part of a rule in Eq. 4: terminal symbol α codifies each of the α_i, and f_i (Eq. 5) is represented by variables *sumExp* and *multExp*, by symbol α (the mathematical expression is only a coefficient), or by λ (this function is not taken into account). Finally, a mathematical expression can contain each of the input variables (x_j), or skip some of them with symbol λ ($\delta_{i,j} = 0$, Eqs. 3, 5). This is represented with the rules of $cvar_j$.

Fig. 2 shows an example of the rules learned with the proposed evolutionary algorithm and the described grammar. The rule has three propositions in the antecedent part (all the linguistic variables are used to classify the input), and two terms in the polynomial of the consequent: one considers the surface of the piece, and the other one adds all the dimensions of the piece.

B. Genetic programming algorithm

The genetic programming algorithm for processing time estimation is described in Fig. 3. First, the population has to

- V = { rule, antecedent, ant_{length}, ant_{width}, $\text{ant}_{thickness}$, consequent, expression, sumExp, multExp, cvar_{length}, cvar_{width}, $\text{cvar}_{thickness}$ }
- Σ = { α, A_1^{length}, A_2^{length}, A_1^{width}, A_2^{width}, $A_1^{thickness}$, $A_2^{thickness}$, x_{length}, x_{width}, $x_{thickness}$, (,), +, ·, λ }
- S = rule
- Productions:
  - rule $\longrightarrow$ antecedent consequent
  - antecedent $\longrightarrow$ ant_{length} ant_{width} $\text{ant}_{thickness}$
  - $\text{ant}_{length} \longrightarrow A_1^{length} \mid A_2^{length} \mid \lambda$
  - $\text{ant}_{width} \longrightarrow A_1^{width} \mid A_2^{width} \mid \lambda$
  - $\text{ant}_{thickness} \longrightarrow A_1^{thickness} \mid A_2^{thickness} \mid \lambda$
  - consequent $\longrightarrow$ expression + expression + expression
  - expression $\longrightarrow \alpha \cdot (\text{ sumExp }) \mid \alpha \cdot (\text{ multExp }) \mid \alpha \mid \lambda$
  - sumExp $\longrightarrow$ cvar_{length} + cvar_{width} + $\text{cvar}_{thickness}$
  - multExp $\longrightarrow$ $\text{cvar}_{length} \cdot \text{cvar}_{width} \cdot \text{cvar}_{thickness}$
  - $\text{cvar}_{length} \longrightarrow x_{length} \mid \lambda$
  - $\text{cvar}_{width} \longrightarrow x_{width} \mid \lambda$
  - $\text{cvar}_{thickness} \longrightarrow x_{thickness} \mid \lambda$

Fig. 1. Context-free grammar

IF X_{length} *is* A_1^{length} and X_{width} *is* A_2^{width} and $X_{thickness}$ *is* $A_1^{thickness}$
THEN time is $120 \cdot (x_{length} \cdot x_{width}) + 240 \cdot (x_{length} + x_{width} + x_{thickness})$

Fig. 2. A typical rule for processing time estimation. In this case rule means that processing time depends on the area (with weight 120) and the sum of the dimensions (with weight 240) of the piece

be initialized.

1) Initial rule generation: For each example in the training set, an individual (rule) is generated as follows: the antecedent part of the rule is created selecting the labels that best cover the input values of the example. On the other hand, the consequent part is obtained starting with variable *consequent* and applying randomly selected productions (rules of the context-free grammar) recursively until all the leaves of the tree are terminal symbols.

1) Initialize population
   a) Generate rules
   b) Evaluate population
   c) Resize population (i)
2) for iteration = 1 to *maxIterations*
   a) Crossover and mutation
   b) Evaluate population
   c) Resize population (ii)
3) Select rules for the final knowledge base

Fig. 3. Evolutionary algorithm

2) Individual evaluation: For each individual and each example in the training set, the degree of fulfillment of the antecedent part of the rule is obtained. If the example is covered by the rule, the error in time estimation is calculated as:

$$error_{c,\,e} = (pt_{est}^{c,\,e} - pt^{e})^2 \tag{6}$$

where pt^e is the processing time for example e.

Finally, it is necessary to calculate the raw fitness of each individual as:

$$fitness_{raw}^{c} = \frac{\sum_{e=1}^{ne_c} error_{c,e}}{ne_c} \tag{7}$$

where ne_c is the number of examples covered by individual c. Raw fitness measures the strength of the individual for the examples it covers. To calculate the fitness of each individual, first it is necessary to implement the token competition: for each example e, the individual with lower $error_{c,\,e}$ will seize the example. Fitness is defined as:

$$fitness^{c} = fitness_{raw}^{c} \cdot \frac{seized_c}{covered_c} \tag{8}$$

where $seized_c$ is the number of examples seized by individual c and $covered_c$ the number of examples covered.

3) Population resize (i): The last step in the initialization consists of eliminating the individuals in the *examples population* that do not cover any example. Finally, pop_{size} individuals are picked out from the *examples population* to build the initial population, and the iterative part of the algorithm starts.

This iterative part is repeated *maxIterations* times, and starts with the crossover and mutation of the individuals of the population. There is no selection. A couple of individuals is randomly picked up (all the individuals of the previous population have to be chosen once), individuals are crossed with probability p_c, then mutated with probability p_m, and finally added to the population. At the end of the process the population doubles the size of the previous population, as it contains the original individuals plus their offspring (due to crossover and mutation).

4) Crossover: Crossover of two individuals is implemented with a one-point crossover operator. The cross point of the first individual (cp_1) is selected randomly among all the genes of the chromosome (i.e., nodes of the tree) that are variables of the context-free grammar. Then, the algorithm looks for a node of the same variable in the second of the individuals. If there is not such a node, then the parent node of cp_1 is chosen as cp_1. This process is repeated until there is at least one node in the second individual equal to node cp_1. If there are several candidates to select cp_2 (cross-point of the second individual), one of them will be randomly selected. Once cp_1 and cp_2 have been determined, the subtree with root node cp_1 is grafted at node cp_2 of the second chromosome and vice versa.

5) Mutation: Mutation operation starts selecting randomly a gene. If the gene is a variable, then a rule for this variable is randomly applied and the resulting subtree replaces the original one. On the other hand, if the gene is a terminal symbol that can take different values (α in our grammar), its value can be mutated in two different ways: random mutation and step mutation. Random mutation is selected with probability p_{rm}, and chooses a new value randomly. On the other hand, step mutation increases or decreases (with the same probability) the value of the gene in a quantity, called $prec_g$ (where g is the gene), that represents a meaningful change in the gene.

6) Population resize (ii): After crossover and mutation, individuals are evaluated in the same way as for the evaluation stage in the initialization step. Finally, population must be resized to pop_{size}: first, individuals with null fitness are eliminated. Whenever the size of the population is under pop_{size}, new individuals are added. These individuals are chosen from the *examples population*, selecting those rules that cover examples that have not been seized yet by the individuals of the population.

7) Final knowledge base construction: Once the algorithm has finished, the knowledge base has to be defined selecting some of the rules of the final population. First, the individuals are sorted in a decreasing fitness order. Starting with the best individual, an individual is added to the knowledge base if it covers at least one example not covered yet by the rules of the knowledge base.

V. Results

Our algorithm was validated with a subset of the machines that are currently being used in the production plans of a wood furniture industry. These machines are: rip saw for edging and ripping I (RS-I), abrasive calibrating machine (ACM), veneer slicers (VS), rip saw for edging and ripping II (RS-II), and commissioning system for large boards (CSLB). Data of 1,500 different custom pieces of furniture, that have been build in the factory along several years, have been used to generate the examples sets. The dimensions of each of the pieces was obtained and then, for each of the machines, the processing time was measured.

The algorithm has to learn the knowledge base (rule model of Eq. 4) that minimizes the error of processing time estimation for the machine, but keeping the rule structure provided by the expert and summarized in the context-free grammar. Experiments have been performed with a five-fold cross-validation for each of the examples sets. Each set was divided in five subsets of the same size, and the learning process was run five times, using as training set four of the subsets, and as test set the remaining one. The test set was different in each of the runs.

We have compared our approach with other regression techniques proposed by other authors. Tables I-V show, for each of the machines (or data sets), the mean and standard deviation of the number of rules (#R), the mean square error of training (MSE_{tra}), and the mean square error of test (MSE_{tst}) of the five-fold cross-validation experiments for each technique[1]. In each table, the lower average values for #R, MSE_{tra}, and MSE_{tst} are marked in boldface. The methodologies compared in the tables are:

- GP-TSK: the approach described in this paper. The algorithm has the following parameters: *maxIterations* = 100, $pop_{size} = 500$, $p_c = 0.8$, $p_m = 0.5$ (per chromosome), $p_{rm} = 0.25$.
- WM [13]: the well-known Wang and Mendel ad hoc data driven method.
- COR [5], [4]: an ad hoc data driven method that learns Mamdani type fuzzy rules with global semantics.
- COR+TUN [3]: consists in the previously described COR algorithm plus a final tuning stage.
- MOGUL-TSK [1]: a two-stage evolutionary algorithm based on MOGUL (a methodology to obtain Genetic Fuzzy Rule-Based Systems under the Iterative Rule Learning approach).
- NN-MPCG [10]: a multilayer Perceptron network.

[1]Results of methods WM, MOGUL-TSK, and NN-MPCG have been obtained using the software KEEL [2].

TABLE I
Results of the five-fold cross-validation for machine RS-I

Method	#R		MSE_{tra}		MSE_{tst}	
	$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ
GP-TSK	**5**	1	5,313	468	5,592	1,113
WM	125	1	17,827	1,907	19,155	2,360
COR	96	3	12,547	339	14,113	362
COR+TUN	96	3	7,029	889	8,277	1,122
MOGUL-TSK	24	2	5,891	448	6,662	879
NN-MPCG	—	—	**4,266**	219	**4,297**	387

TABLE II
Results of the five-fold cross-validation for machine ACM

Method	#R		MSE_{tra}		MSE_{tst}	
	$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ
GP-TSK	**3**	2	**618**	106	**609**	121
WM	125	1	4,665	564	4,654	521
COR	95	3	2,853	112	3,295	221
COR+TUN	95	3	1,223	88	1,541	297
MOGUL-TSK	21	3	1,445	418	1,691	761
NN-MPCG	—	—	831	122	846	169

TABLE III
Results of the five-fold cross-validation for machine VS

Method	#R		MSE_{tra}		MSE_{tst}	
	$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ
GP-TSK	**6**	1	1,287	191	1,274	168
WM	125	1	2,971	88	3,169	401
COR	102	3	2,172	33	2,479	234
COR+TUN	102	3	1,376	79	1,691	201
MOGUL-TSK	23	2	1,303	153	1,440	135
NN-MPCG	—	—	**990**	35	**1,003**	93

TABLE IV
Results of the five-fold cross-validation for machine RS-II

Method	#R		MSE_{tra}		MSE_{tst}	
	$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ
GP-TSK	**5**	1	**1,088**	113	**1,096**	189
WM	125	1	4,811	398	4,980	718
COR	94	4	3,066	77	3,601	374
COR+TUN	94	5	1,511	97	1,953	264
MOGUL-TSK	20	3	2,363	861	2,729	1,001
NN-MPCG	—	—	1,096	81	1,139	153

The analysis of the average values of MSE_{tst} shows that GP-TSK is the best method in two of the machines and the second best method in the other three. Going into the details, GP-TSK clearly outperforms WM, COR and COR+TUN.

TABLE V
RESULTS OF THE FIVE-FOLD CROSS-VALIDATION FOR MACHINE CSLB

Method	#R		MSE_{tra}		MSE_{tst}	
	$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ
GP-TSK	**3**	1	4,192	363	4,545	1,301
WM	125	1	18,860	2,006	19,731	2,217
COR	105	3	11,755	360	14,852	1,276
COR+TUN	105	3	5,167	761	7,112	980
MOGUL-TSK	21	2	8,407	2,189	9,612	2,034
NN-MPCG	—	—	**2,723**	135	**2,816**	624

The comparison between MOGUL-TSK and GP-TSK is more informative, as both methods learn TSK rules, although with different structures in the consequent. GP-TSK obtains again better values of MSE_{tst} in all the machines. The differences take the lower value, a 13% higher, for machine ACM (table III), while for machine VS (table II) MSE_{tst} is more than two times higher.

Finally, the neural network approach (NN-MPCG) obtains the best MSE_{tst} results in machines RS-I (table I), with an reduction of 23% under GP-TSK MSE_{tst}, VS (table III) with a 21% improvement, and CSLB (table V) with a 38% lower error. On the other hand, in machine ACM (table II), the neural network error is a 39% higher than GP-TSK, and in machine RS-II (table IV) the improvement of GP-TSK is around 4%.

We can distinguish four different levels of similarity between the information that can be extracted from the regression functions of each of the methodologies and the knowledge structure provided by the expert. The lower similarity corresponds to the neural network (NN-MPCG): the expert has no information about how an estimated time was generated. WM, COR, and COR+TUN methods provide the expert with more information: *"if length is low and width is low then time is low"*. With this rule, the expert can extract that variable *thickness* does not influence the time estimations of this machine, and also that pieces with low lengths and widths will generate low processing times. Nevertheless, the expert can not deduce the contribution of each of the variables to time estimations.

This is partially solved with MOGUL-TSK, as the expert knows the contribution of each of the input variables to time estimation: *"if length is low and width is low then time is 50 + 100 · length + 300 · width"*. MOGUL-TSK consequents are first-order polynomials, so it is not possible to extract from rules relations among variables (for example surface, lateral surface or volume). Also, MOGUL-TSK generates knowledge bases with local semantics, while GP-TSK has global semantics. Moreover, GP-TSK rules also provide the expert with information about the relations among variables, as higher order polynomials can be generated by the grammar: *"if length is low and width is low then time is 60 · (length · width)"*.

Both this rule and the rule generated by MOGUL-TSK estimate processing times using variables *length* and *width*. But the GP-TSK rule provides the expert with information about the relation between both variables, generating a new variable very intuitive for the expert: the surface of the piece. Also, the GP-TSK approach avoids the existence of rules considered as difficult to interpret by the expert. For example, a rule with the square of variable *length* in the consequent part could be generated by a MOGUL-TSK approach with higher order polynomials, but not by GP-TSK as the context-free grammar does not generate that rule structure.

VI. CONCLUSIONS

A methodology for the estimation of processing times in the wood furniture industry has been presented. The algorithm is based on a genetic programming approach together with token competition to maintain the diversity of the population. The system is composed of a set of TSK rules with different structures in the consequent. Expert knowledge has been incorporated through the use of a context-free grammar that imposes restrictions to the structure of these rules. The system was tested with real data of five different machines of the production plant. Our approach generates a set of rules with an structure that makes easy for the expert to extract valuable information, while obtaining also a very high accuracy in time estimations. Moreover, a comparison with other methods has been done, showing the best tradeoff between accuracy and simplicity in information extraction.

REFERENCES

[1] R. Alcalá, J. Alcalá-Fdez, J. Casillas, O. Cordón, and F. Herrera. Local identification of prototypes for genetic learning of accurate tsk fuzzy rule-based systems. *International Journal of Intelligent Systems*, 22:909–941, 2007.

[2] J. Alcalá-Fdez, L. Sánchez, S. García, M.J. del Jesus, S. Ventura, J.M. Garrell, J. Otero, C. Romero, J. Bacardit, V.M. Rivas, J.C. Fernández, and F. Herrera. Keel: A software tool to assess evolutionary algorithms to data mining problems. *Soft Computing*, In press.

[3] J. Casillas, O. Cordón, , M.J. del Jesus, and F. Herrera. Genetic tuning of fuzzy rule deep structures preserving interpretability and its interaction with fuzzy rule set reduction. *IEEE Transactions on Fuzzy Systems*, 13(1):13–29, 2005.

[4] J. Casillas, O. Cordón, I. Fernández de Viana, and F. Herrera. Learning cooperative linguistic fuzzy rules using the best-worst ant system algorithm. *International Journal of Intelligent Systems*, 20:433–452, 2005.

[5] J. Casillas, O. Cordón, and F. Herrera. Cor: A methodology to improve ad hoc data-driven linguistic rule learning methods by inducing cooperation among rules. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, 32(4):526–537, 2002.

[6] A. Giornada and F. Neri. Search-intensive concept induction. *Evolutionary Computation*, 3(4):375–416, 1995.

[7] J.W. Herrmann and M.M. Chincholkar. Reducing Throughput Time during Product Design. *Journal of Manufacturing Systems*, 20(6):416–428, 2001.

[8] J.R. Koza. *Genetic programming: on the programming of computers by means of natural selection*. The MIT Press, 1992.

[9] K.S. Leung, Y. Leung, L. So, and K.F. Yam. Rule learning in expert systems using genetic algorithm: 1, concepts. In *Proceedings of the 2nd International Conference on Fuzzy Logic and Neural Networks*, pages 201–204, Iizuka (Japan), 1992.

[10] F. Moller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6:525–533, 1990.

[11] M. Sugeno and G.T. Kang. Structure identification of fuzzy model. *Fuzzy Sets and Systems*, 28:15–33, 1988.

[12] T. Takagi and M. Sugeno. Fuzzy identification of systems and its application to modeling and control. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-15:116–132, 1985.

[13] L.-X. Wang and J.M. Mendel. Generating fuzzy rules by learning from examples. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(6):1414–1427, 1992.