Soft Comput (2009) 13:497–509
DOI 10.1007/s00500-008-0364-2

FOCUS

Processing time estimations by variable structure TSK rules learned through genetic programming

Manuel Mucientes · Juan C. Vidal · Alberto Bugarín · Manuel Lama

Published online: 2 September 2008

Abstract Accuracy in processing time estimation of different manufacturing operations is fundamental to get 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.

Keywords Genetic programming · Context-free grammar · TSK fuzzy rules · Production planning · Processing time estimation · Manufacturing industry

M. Mucientes (✉) · J. C. Vidal · A. Bugarín · M. Lama
Department of Electronics and Computer Science,
University of Santiago de Compostela,
15782 Santiago de Compostela, Spain
e-mail: manuel.mucientes@usc.es

1 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 (Herrmann and Chincholkar 2001). An accurate estimation of the manufacturing time of the product will allow competitive prices, higher profits, and increment the client portfolio too. Nevertheless, task scope is even broader. On the one hand, it fixes future material and storage capacity requirements. On the other hand, it constitutes the basis for any 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, like 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.

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 must take into account two premises: high accuracy, but providing the expert with valuable and easy to extract information. Takagi–Sugeno–Kang (TSK) fuzzy rules (Takagi and Sugeno 1985; Sugeno and Kang 1988) seem to fit perfectly for the requirements of processing times estimation in this case. In a TSK rule, the output is obtained as a polynomial of the input variables. So, if the input variables verify the antecedent part of the rule, the product belongs to that class with a certain degree, and the processing time can be estimated with the polynomial in the consequent of the rule.

Learning of fuzzy knowledge bases through the use of evolutionary algorithms has shown to be a powerful technique (Cordón et al. 2001). Evolutionary algorithms have several advantages over other learning methods in this field: first, the rules of the knowledge base can be represented in many different ways, due to the flexibility in the representation of the solutions. On the other hand, another important advantage is that we can manage the tradeoff between simplicity to extract information and accuracy of the learned rules through the use of different algorithms.

Several authors have learned TSK knowledge bases with evolutionary algorithms: in (Cordón and Herrera 1999), authors present a two-stage evolutionary process for designing TSK fuzzy rule-based systems from examples. They combine a generation stage based on a $(\mu; \lambda)$-evolution strategy, and a refinement stage, in which both the antecedent and consequent parts of the fuzzy rules in the previous knowledge base are adapted by a hybrid evolutionary process. In (Hoffmann and Nelles 2001), a genetic programming approach is used for structure identification of local lineal neuro-fuzzy models through TSK rules. The objective is to identify an optimal partition of the input space into gaussian, axis-orthogonal fuzzy sets. In (Papadakis and Theocharis 2006) a genetic-based algorithm for generating simple and well-defined TSK models is described. The model building process comprises three stages: structure learning, rule base simplification, and fine-tuning. Another approach is proposed in (Yen et al. 1998), with a learning algorithm that integrates global and local learning. The algorithm uses the idea of local weighted regression and local approximation in nonparametric statistics. The method is capable of adjusting its parameters based on the users preference, generating models with good tradeoff in terms of global fitting and local interpretation. Finally, in (Lin and Xu 2006) TSK-type fuzzy controllers are learned with a hybrid approach that consists of a self-clustering algorithm and a dynamic-form symbiotic evolution. First, the internal structure is identified, and then a sequential-search-based dynamic evolution method is applied.

Our approach for processing time estimation is based on the use of TSK rules. However, TSK rule consequents are usually first-order polynomials, and we need more flexibility in the structure of the consequent of the rules, as the terms of the polynomials can be sums of variables or products of variables to keep the knowledge structure demanded by the expert. Accordingly, we have to define a TSK rule model able to estimate processing times of a machine.

The design of a TSK rule of variable structure is a hard task, as we have to define the labels of the antecedent part, the coefficients of the polynomial, but also the functions of the input variables for each term of the polynomial. To learn such a knowledge base, the algorithm must have the ability to represent rules with different structures, and this facility is provided by genetic programming. Genetic programming is an evolutionary algorithm that represents each chromosome of the population as a tree of variable length.

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. 2 introduces the problem of processing time estimation in the wood furniture industry, Sect. 3 presents the TSK rule model with variable structure in the consequent, while Sect. 4 describes the evolutionary approach used for learning: a context-free grammar together with a genetic programming algorithm. Section 5 shows the results of the experiments with different examples sets, and compares the results with other methodologies. Finally, Sect. 6 points out the conclusions.

2 Processing times in the furniture industry

Throughput time is the interval that elapses when the manufacturing system performs all of the operations

necessary to complete a work order. Its accurate estimation is hard 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 throughput time 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. Much effort is therefore spent to reduce throughput time by improving manufacturing planning, control systems, and developing more sophisticated scheduling procedures (Herrmann and Chincholkar 2001; Kusiak and He 1999). The objective is to define work plans that minimize the resources queuing and maximize resources capacity, constrained to the material availability and product requirements. In this context, the feasibility, time, and cost of most promising plans is analyzed (Gupta and Nau 1995; Minis et al. 1999).

Methodologies like design for manufacturing and assembly (DFMA) are used to reduce the cost and improve the quality of products (Boothroyd 1991; Boothroyd et al. 1994), since the product design has a huge impact on the throughput time in custom-designed products. Note that the throughput time has many components, including move, queue, setup, and processing times (Allahverdi et al. 2008; Shabtay and Steiner 2007; Cheng et al. 2004). In this work we will focus on the estimation of the processing times, which is a critical piece in each manufacturing step.

The estimation of the processing time is one of the most important tasks in the product design life cycle. For example, time estimations are taken into account for redesigning the product if the predicted time is longer than expected. DFMA approaches include many models and techniques for estimating the processing times of a manufacturing step based on the product design. For a detailed design, highly detailed process planning, manufacturing process simulation, or time estimation models can be employed (Minis et al. 1999; Boothroyd 1991; Boothroyd et al. 1994). For a conceptual design, however, less detailed models must depend on a more limited set of critical design information (Boothroyd 1991; Boothroyd et al. 1994). Both approaches are applied in the furniture industry since the definition of a detailed design is cost and time expensive.

The process structure is depicted in Fig. 1: the material, manufacture and assembly requirements and costs are inferred from the technical or conceptual design of the furniture. Length, width, thickness and other inputs that define a workpiece are some of the variables that influence the processing time of an operation. These variables extracted from the product design can also be combined to get new variables such as the surface or the volume. For

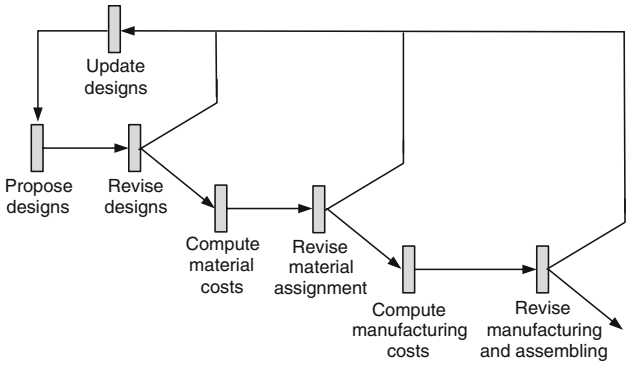

Fig. 1 Diagram of the product design task

example, the processing time to varnish a piece of furniture depends on its surface. If the varnishing process is by hand then the manipulation time is also influenced by the volume of the piece.

As an example, we will describe the way in which the processing time is calculated for a table with the conceptual design shown in Fig. 2. Time estimation is computed in the *compute manufacturing costs* task depicted in Fig. 1. This task, as also happens with the product design, is based on the route selected to manufacture the product and on the order of the operations that this route defines. Although the way in which we select the manufacturing route is out of the scope of this paper, we have to mention that this selection is based on a set of rules based on woodworking knowledge. These rules take into account the constructive decisions, joints used to assemble the furniture, and finishing or quality standards. Once the operations and their ordering are defined, the *compute manufacturing costs* task estimates the times of each of the manufacturing steps. For the table example, it must compute the time to:

- Cut the workpieces from large wood boards.
- Plan and calibrate their surface and thickness.
- Cut each corner at a 45-degree angle.
- Cut a groove in each end of the pieces to be mitered.
- Finishing.

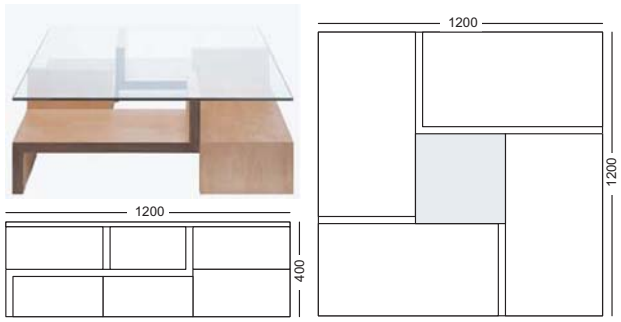

Fig. 2 Conceptual design from which furniture parts and manufacturing steps can be extracted

– Hold the spline in the mitered joint with top-grade adhesive.
– Assemble and pack the furniture.

The processing time of each step is influenced by the operation and, also, by the resource that will perform the operation. For example, the wood pieces of the table example must be calibrated to get the required thickness and precision. Figure 3 depicts the calibrating and sanding machine that will perform the operation. Calibrating machinery specifies the feed speed, the abrasive belt speed, the abrasive belt size, the working thickness, and the maximum and minimum workpiece dimensions. Nevertheless, only the feed speed has influence on the processing time since the other parameters only constraint which resource can perform the operation. In perfect conditions, processing time could be computed by multiplying the speed by the length of the pieces that will be processed. However, the manipulation of the pieces reduce the speed ratio. In our example, although machine feed speed is 5 m/min, time measurements showed that the real speed was significantly lower.

We are currently working with 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. Basically, they use pieces dimensions and material features. However, this approach has several drawbacks: first, it is difficult to identify which variables may influence the processing time when a machine is performing an operation. Secondly, the weight of these variables is obtained from the experience and from manual timekeeping and is therefore uncertain (Cao et al. 1999). Finally, the previously mentioned characteristics of the pieces of furniture, such as their dimensions and variety of materials, makes difficult the definition of a unique function for the range of possible values. In fact, it is usual to identify different classes of pieces based on the values that the variables can take. For each class, a different polynomial may be defined.

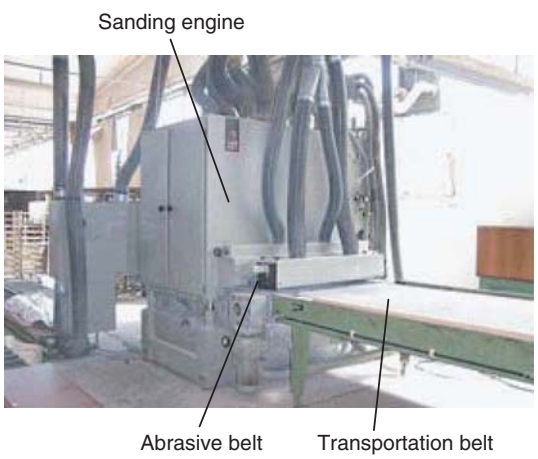

Fig. 3 Calibrating and sanding machine

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.

3 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:

$$\text{Processing time} = 100 \cdot \text{length} + 200 \cdot \text{volume} \quad (1)$$

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:

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

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

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

Moreover, for a given machine, there can coexist different polynomials, each one representing the estimation of the processing time of a class of the input variables. For example, in an specific machine, processing times of pieces with a thickness over a threshold could be estimated with a polynomial, and under that threshold with another polynomial. Thus, 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 (Takagi and Sugeno 1985; Sugeno and Kang 1988). 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:

$$R_k : \text{IF } X_1 \text{ is } A_{l_1^k}^1 \text{ and } \ldots \text{ and } X_{\text{na}} \text{ is } A_{l_{\text{na}}^k}^{\text{na}} \quad \text{THEN } Y \text{ is } \sum_i \alpha_i \cdot f_i(x_1, \ldots, x_{\text{na}}) \tag{4}$$

where X_j is a linguistic variable, $A_{l_j^k}^j$ is a linguistic label of that variable, $l_j^k = 1,...,nl_j$, Y is the output variable, and f_i $(x_1,...,x_{\text{na}})$ are functions of the inputs variables (x_j).

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

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

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 very 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 that has been used to learn TSK rules following the model of Eq. 4 for processing time estimation in the wood furniture industry.

4 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 (Koza 1992). 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.

According to (Cordón et al. 2001), evolutionary learning of knowledge bases has different approaches to represent the solution to the problem: Pittsburgh, Michigan, iterative rule learning (IRL), and cooperative-competitive approaches. In the Pittsburgh approach (Carse et al. 1996), each chromosome codifies a complete knowledge base. The length of the chromosomes can be variable to represent rule bases with a variable number of rules. This methodology has a high computational cost, as in each iteration many knowledge bases have to be evaluated. It would be more adequate for our purpose to codify a single rule in each individual of the population. The other three approaches follow this codification.

In the Michigan approach (Kovacs 2004) rules evolve along time due to their interaction with the environment using the evolutionary algorithm and reinforcement learning. The distribution of the payoff is usually complex. On the other hand, in the IRL approach (Cordón and Herrera 2001), there is not payoff distribution because a single rule is learned by the evolutionary algorithm and not the whole rule base. After each sequence of iterations, the best rule is selected and added to the final rule base. The selected rule must be penalized in order to induce niche formation in the search space.

Finally, in the cooperative-competitive approach (Giornada and Neri 1995) rules evolve together (cooperative, which is not the case of the IRL approach) but competing among them to obtain the higher fitness. In this approach it is fundamental to include a mechanism to maintain the diversity of the population (niche induction). The mechanism must warrant that there is competition among individuals of the same niche, but also has to avoid the deletion of those weak individuals that occupy a niche not covered by other individuals of the population. The proposed evolutionary algorithm is based on the cooperative-competitive approach.

We have chosen token competition (Wong et al. 2000; Leung et al. 1992) as the mechanism for maintaining the diversity. According to (Berlanga et al. 2005), this mechanism is adequate for genetic programming, as in this kind of evolutionary algorithms the structure of the individuals can be completely different and, thus, the evaluation of the similarities is hard. Mechanisms to maintain diversity, like crowding or fitness sharing, have to estimate the similarities between pairs of individuals, but not token competition.

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 will reduce their strength as they can not compete with the best individual in the niche.

4.1 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. 4. 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

- $V = \{$ rule, antecedent, ant_{length}, ant_{width}, $\text{ant}_{thickness}$, consequent, expression, sumExp, multExp, cvar_{length}, cvar_{width}, $\text{cvar}_{thickness}$ $\}$
- $\Sigma = \{ \alpha, A_1^{length}, A_2^{length}, A_1^{width}, A_2^{width}, A_1^{thickness}, A_2^{thickness}, x_{length}, x_{width}, x_{thickness}, (,), +, \cdot, \lambda \}$
- $S =$ 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. 4 Context-free grammar

$$\begin{aligned}&\text{IF} \quad X_{length} \text{ is } A_1^{length} \text{ and}\\ &\quad X_{width} \text{ is } A_2^{width} \text{ and } X_{thickness} \text{ is } A_1^{thickness}\\ &\text{THEN time is } 120 \cdot (x_{length} \cdot x_{width}) +\\ &\quad 240 \cdot (x_{length} + x_{width} + x_{thickness})\end{aligned}$$

Fig. 5 A typical rule for processing time estimation

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 $\texttt{cvar}_j$.

Figure 5 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.

4.2 Genetic programming algorithm

The genetic programming algorithm for processing time estimation is described in Fig. 6. First, the population has to be initialized.

4.2.1 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 leafs 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. 6 Evolutionary algorithm

Figure 7 shows a typical chromosome corresponding to the rule shown in Fig. 5. Terminal symbols (leafs of the tree) are represented by circles, and variables are shown as flatted circles. Starting at node consequent, there is a unique rule to apply. This rule generates five children nodes, and three of them are variables (expression). In each one of them, four rules can be selected. Randomly, rule 2 is applied to the first variable, rule 1 to the second, and rule 4 to the third. The process is repeated again, until all the leaves of the tree are terminal symbols. Terminal symbol α can take different values in each of the nodes it appears. Initially, each value of α is picked out randomly. After the generation of this preliminary population, called *examples population*, individuals must be evaluated.

4.2.2 Individual evaluation

For each individual and each example in the training set, the following steps are repeated:

1. Obtain the degree of fulfillment of the antecedent part of the rule (individual) for this example.
2. If the example is covered by the rule:
   (a) Parse the string of the consequent part of the rule and obtain the processing time estimation, $pt_{\text{est}}^{c,e}$, where c is the individual and e the example.
   (b) Calculate the error in the time estimation as:
   $$\text{error}_{c,e} = (\text{pt}_{\text{est}}^{c,e} - \text{pt}^{e})^2 \tag{6}$$
   where pt^e is the processing time for example e.

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

$$\text{fitness}_{\text{raw}}^{c} = \frac{ne_c}{\sum_{e=1}^{ne_c} \text{error}_{c,e}} \tag{7}$$

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. The algorithm to decide which individual seizes each example is the following:

- For each example e, choose the individual with lower $\text{error}_{c,e}$
- If there are several individuals with equal $\text{error}_{c,e}$
  - Choose the individual with higher $\text{fitness}_{\text{raw}}^{c}$

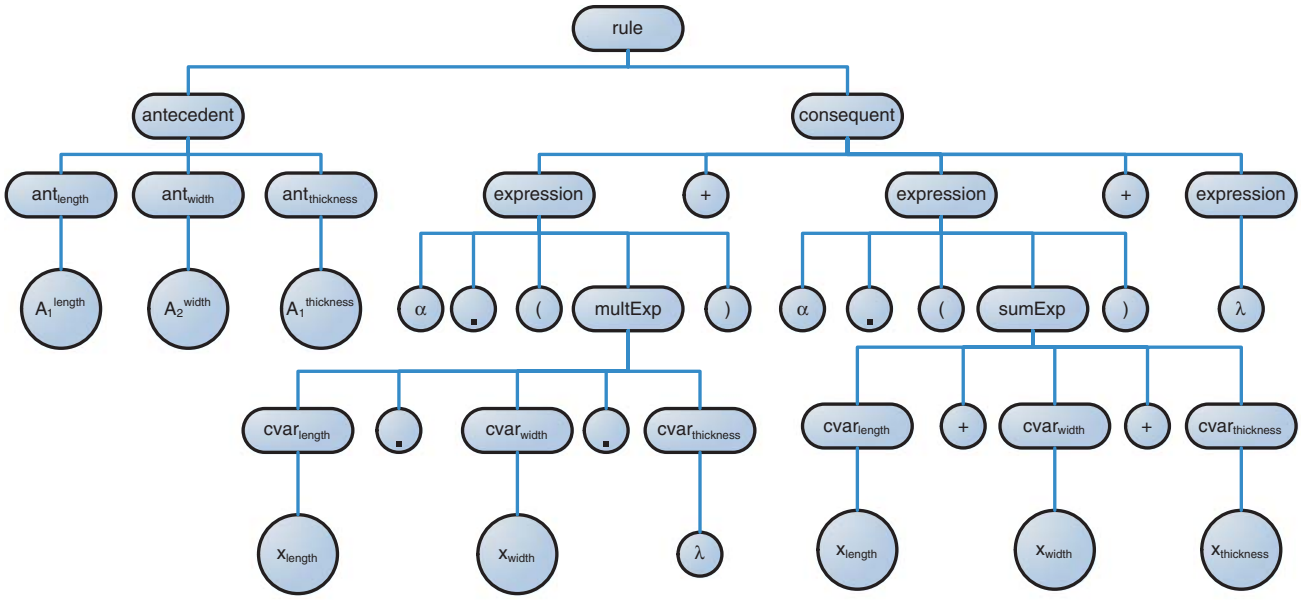

Fig. 7 A typical chromosome

– If there are several individuals with equal $\text{fitness}^c_{\text{raw}}$
  - Choose the individual with higher ne_c

Thus, the individual that generates a lower error for the example will seize it. If there are several individuals with the same error, then the stronger individual of the niche will seize the example. And, finally, if there are also several individuals in this situation, the one that covers more examples will be selected. Fitness is defined as:

$$\text{fitness}^c = \text{fitness}^c_{\text{raw}} \times \frac{\text{seized}_c}{\text{covered}_c} \tag{8}$$

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

4.2.3 Population resize (i)

The last step in the initialization consists in eliminating those individuals of 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 will double the size of the previous population, as it will contain the original individuals plus their offspring (due to crossover and mutation).

4.2.4 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. Figure 8 shows an example of crossover between two individuals. First, cp_1 is selected at node `sumExp`. As this node does not exist in the second of the individuals, the new cp_1 is the parent node of the previous cp_1: node `expression`. There are three possible cp_2, as there are three nodes of the same type as cp_1 in the second of the individuals. Randomly, the node with white background is selected, and the subtrees that have as root nodes cp_1 and cp_2 are interchanged.

4.2.5 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

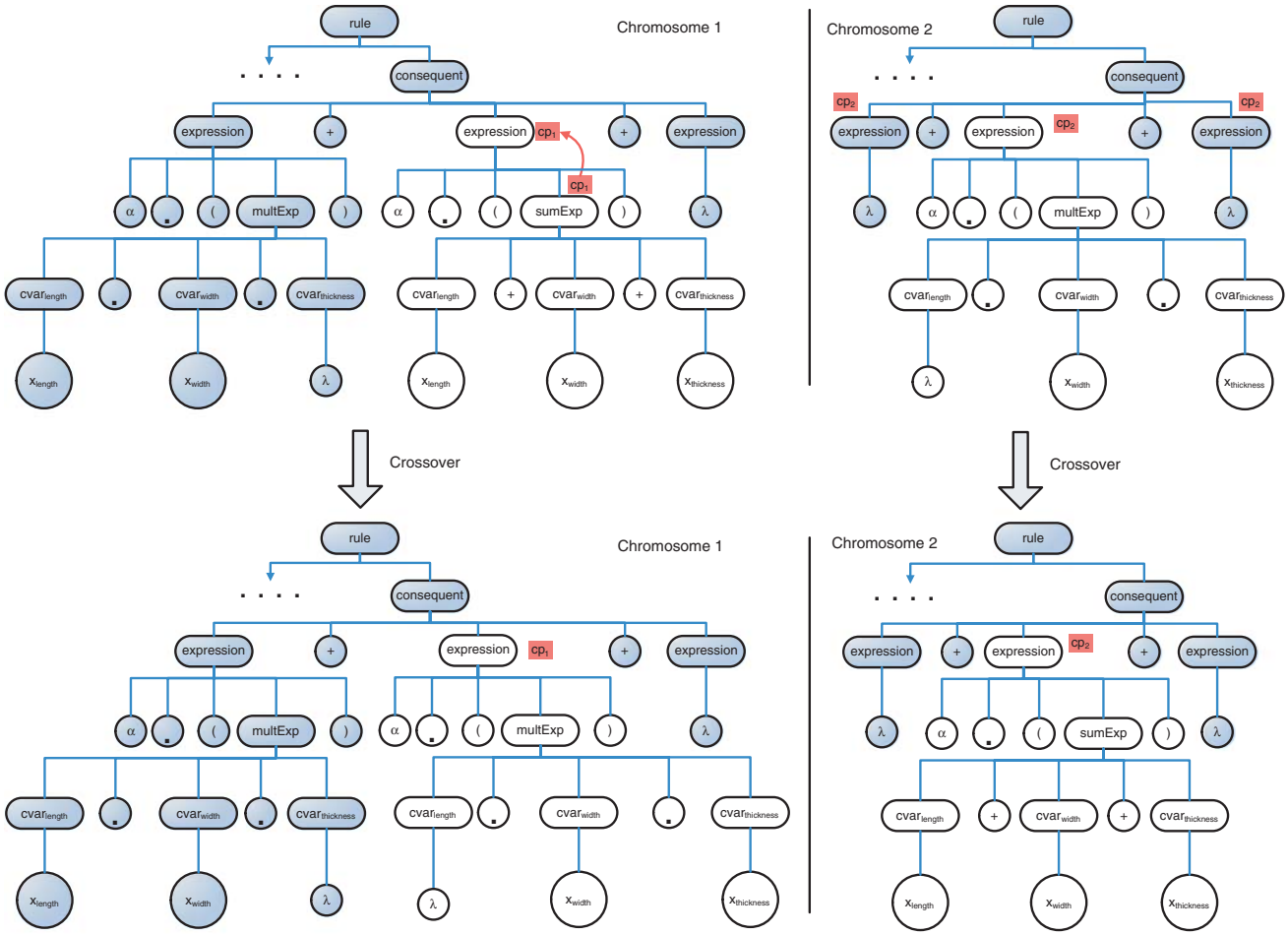

Fig. 8 Crossover of two chromosomes

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 equal 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.

4.2.6 *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 $\mathrm{pop}_{\mathrm{size}}$: first, individuals with null fitness are eliminated. Whenever the size of the population is under $\mathrm{pop}_{\mathrm{size}}$, 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. If the population is still under $\mathrm{pop}_{\mathrm{size}}$ (this may occur in the last iterations of the algorithm), then copies and mutated copies of the best individuals are inserted.

4.2.7 *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.

5 Results

The proposed algorithm has been 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

Table 1 Characteristics of the data sets

Machine	Length (m)		Width (m)		Thickness (m)		Time (s)	
	$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ
RS-I	1.971	1.132	1.239	0.737	0.251	0.144	1,125	580
ACM	1.971	1.132	1.239	0.737	0.251	0.144	385	265
VS	1.971	1.132	1.239	0.737	0.251	0.144	579	230
RS-II	1.971	1.132	1.239	0.737	0.251	0.144	526	272
CSLB	1.971	1.132	1.239	0.737	0.251	0.144	736	658

factory along several years, have been used to generate the examples sets. The dimensions of each of the pieces were obtained and then, for each of the machines, the processing time was measured. These times are very noisy because many of the operations require some kind of manipulation by an operator and, also, because this operator has to measure the times manually.

Each example has values of length, width and thickness of the piece of furniture, as well as the measured processing time for that piece in the machine. Table 1 shows the mean ($\bar{x}$) and standard deviation (σ) for each variable in each of the data sets. 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 equal 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 2–6 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. Learns TSK rules with global semantics using genetic programming with a context-free grammar. Token competition is used to maintain diversity in the population. 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 (Wang and Mendel 1992): the well-known Wang and Mendel ad hoc data driven method that generates Mamdani type fuzzy rules with global semantics. We have used five labels to partition each linguistic variable.
- COR (Casillas et al. 2002, 2005b): an ad hoc data driven method that learns Mamdani type fuzzy rules with

[1] Results of methods WM, MOGUL-TSK, and NN-MPCG have been obtained using the software KEEL (Alcalá-Fdez et al. 2008).

Table 2 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 3 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 4 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 5 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

Table 6 Results of the five-fold cross-validation for machine CSLB

Method	#R		MSE_{tru}		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

global semantics. COR consists of two stages: search space construction, and selection of the most cooperative fuzzy rule set. The combinatorial search for the selection stage was implemented with the best-worst ant system (BWAS), an ant colony optimization algorithm. The algorithm was run with the standard parameter values and five labels for each linguistic variable.

- COR+TUN (Casillas et al. 2005a): consists in the previously described COR algorithm plus a final tuning stage. This final stage is a constrained tuning of the membership function parameters using a genetic algorithm. It is performed by using variation intervals to preserve meaningful fuzzy sets.
- MOGUL-TSK (Alcalá et al. 2007): a two-stage evolutionary algorithm based on MOGUL (a methodology to obtain Genetic fuzzy rule-based systems under the IRL approach). The first stage performs a local identification of prototypes to obtain a set of initial local semantics-based TSK rules, following the IRL approach and based on an evolutionary generation process within MOGUL. Then, a postprocessing stage is applied. It consists in a genetic niching-based selection process to remove redundant rules and a genetic tuning process to refine the fuzzy model parameters. The method was run with the standard parameter values, and the initial partition of each variable was of three labels.
- NN-MPCG (Moller 1990): a multilayer Perceptron network that uses the conjugate gradient, a non-linear optimization method, to adjust weights. In our experiments, the network has one hidden layer with ten neurons.

5.1 Analysis of accuracy

The methods can be ranked in increasing order of accuracy: WM, COR, COR+TUN, MOGUL-TSK[2], GP-TSK, and NN-MPCG. If we compare the average values of MSE_{tst} of our method (GP-TSK) with the other ones, GP-TSK is the best method in two of the machines and the second best method in the other three. Going into the details, the differences in MSE_{tst} between GP-TSK and WM range from more than two times higher (machine VS, Table 4) to more than seven times higher (machine ACM, Table 3). For COR, MSE_{tst} is more than two times higher in machine VS (Table 3) and more than five times higher for machine ACM (Table 4). In the same way, COR+TUN test error ranges from a 33% higher in machine VS (Table 3) to a 153% higher than GP-TSK for machine ACM (Table 4).

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 4), while for machine VS (Table 3) 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 2), with an reduction of 23% under GP-TSK MSE_{tst}, VS (Table 4) with a 21% improvement, and CSLB (Table 6) with a 38% lower error. On the other hand, in machine ACM (Table 3), the neural network error is a 39% higher than GP-TSK, and in machine RS-II (Table 5) the improvement of GP-TSK is around 4%.

5.2 Discussion about knowledge structure

As has already been mentioned, in this system it is of high importance to keep the knowledge structure provided by the expert. The reason is that an expert must be able to analyze the regression functions that generated a processing time estimation. With that information, the expert could, for example, modify a production plan, discard a proposed production plan, etc. 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

[2] Really, COR+TUN outperforms the MSE_{tst} values of MOGUL-TSK in three of the data sets, but with a much higher number of rules. With a similar number of rules, MOGUL-TSK has better accuracy than COR+TUN.

1. If length is high and width is low and thickness is high then time is 120 · length + 60 · width + 180 · (length · width · thickness)
2. If length is low then time is 120 · length + 240 · (length · width · thickness)
3. If width is low and thickness is low then time is 120 · length + 60 · thickness + 180 · (length · width · thickness)
4. If width is low then time is 120 · length + 240 · (length · width · thickness)

Fig. 9 A typical rule base for machine ACM

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.

Finally, GP-TSK obtains always a very low number of rules (between 3 and 6), while MOGUL-TSK gets between 20 and 24 rules in the different machines. A knowledge base with a reduced number of rules is more interpretable. Also, a low number of rules helps the expert to extract information from the regression functions as, for a piece of furniture, a lower number of rules will generally be fired. For all these reasons, in the context-free grammar (Fig. 4) that picks up the expert's knowledge, only two linguistic labels per input variable have been defined. Of course, with a higher number of rules, accuracy could be augmented. However, in this case the expert has preferred to loose accuracy in exchange for an improvement in interpretability.

Figure 9 shows a typical rule base learned with GP-TSK. This rule base fulfills the two premises we were looking for:

- Simplicity for information extraction: the structure of the rules follow the requirements of the expert. This is not fulfilled by any of the other tested methods (only MOGUL-TSK fulfills it partially). The characteristics that make easy to extract information from the knowledge bases are:
  - The contributions of each of the input variables are explicit.
  - Rules represent associations of variables (like the volume) that are meaningful for processing time estimations, while avoiding the generation of new variables with no meaning for the expert.
  - The number of rules is very reduced.
- Accuracy of the processing time estimators obtained by GP-TSK has proved to be very high, and has only been outperformed by a neural network approach in three of the five data sets. Nevertheless, from the expert's point of view, a neural network approach is not acceptable, due to the impossibility to extract valuable information.

6 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 has been 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.

Acknowledgments Authors wish to acknowledge Xunta de Galicia and Martínez Otero Contract, S.A. for their financial support under grants PGIDIT06SIN20601PR and PGIDIT04DPI096E.

References

Alcalá R, Alcalá-Fdez J, Casillas J, Cordón O, Herrera F (2007) Local identification of prototypes for genetic learning of accurate tsk fuzzy rule-based systems. Int J Intell Syst 22:909–941

Alcalá-Fdez J, Sánchez L, García S, del Jesus MJ, Ventura S, Garrell JM, Otero J, Romero C, Bacardit J, Rivas VM, Fernández JC,

Herrera F (2008) Keel: a software tool to assess evolutionary algorithms to data mining problems. Soft Comput. doi:10.1007/s00500-008-0323-y

Allahverdi A, Ng CT, Cheng TCE, Kovalyov MY (2008) A survey of scheduling problems with setup times or costs. Eur J Oper Res 187(3):985–1032

Berlanga FJ, del Jesus MJ, Herrera F (2005) Learning compact fuzzy rule-based classification systems with genetic programming. In: Proceedings of the 4th conference of the European society for fuzzy logic and technology (EUSFLAT), Barcelona (Spain), pp 1027–1032

Boothroyd G (1991) Assembly automation and product design. Marcel Dekker, August

Boothroyd G, Dewhurst P, Knight W (1994) Product design for manufacture and assembly. Marcel Dekker, February

Cao Q, Patterson JW, Bai X (1999) Reexamination of processing time uncertainty. Eur J Oper Res 164(1):185–194

Carse B, Fogarty TC, Munro A (1996) Genetic algorithms and soft computing. Studies in fuzziness and soft computing. In: Evolving temporal fuzzy rule-bases for distributed routing control in telecommunication networks, vol 8. Physica-Verlag, Heidelberg, pp 467–488

Casillas J, Cordón O, Herrera F (2002) Cor: A methodology to improve ad hoc data-driven linguistic rule learning methods by inducing cooperation among rules. IEEE Trans Syst Man Cybern B Cybern 32(4):526–537

Casillas J, Cordón O, del Jesus MJ, Herrera F (2005a) Genetic tuning of fuzzy rule deep structures preserving interpretability and its interaction with fuzzy rule set reduction. IEEE Trans Fuzzy Syst 13(1):13–29

Casillas J, Cordón O, Fernández de Viana I, Herrera F (2005b) Learning cooperative linguistic fuzzy rules using the best-worst ant system algorithm. Int J Intell Syst 20:433–452

Cheng TCE, Ding Q, Lin BMT (2004) A concise survey of scheduling with time-dependent processing times. Eur J Oper Res 152:1–13

Cordón O, Herrera F (1999) A two-stage evolutionary process for designing TSK fuzzy rule-based systems. IEEE Trans Syst Man Cybern B 29(6):703–715

Cordón O, Herrera F (2001) Hybridizing genetic algorithms with sharing scheme and evolution strategies for designing approximate fuzzy rule-based systems. Fuzzy Sets Syst 118:235–255

Cordón O, Herrera F, Hoffmann F, Magdalena L (2001) Genetic fuzzy systems: evolutionary tuning and learning of fuzzy knowledge bases. In: Advances in fuzzy systems—applications and theory, vol 19. World Scientific, Singapore

Giornada A, Neri F (1995) Search-intensive concept induction. Evol Comput 3(4):375–416

Gupta SK, Nau DS (1995) A systematic approach for analyzing the manufacturability of machined parts. Comput Aided Des 27(5):323–342

Herrmann JW, Chincholkar MM (2001) Reducing throughput time during product design. J Manuf Syst 20(6):416–428

Hoffmann F, Nelles O (2001) Genetic programming for model selection of TSK-fuzzy systems. Inf Sci 136(1-4):7–28

Kovacs T (2004) Strength or accuracy: credit assigment in learning classifier systems. Springer, Berlin

Koza JR (1992) Genetic programming: on the programming of computers by means of natural selection. MIT Press, Cambridge

Kusiak A, He W (1999) Design of components for schedulability. Eur J Oper Res 164(1):185–194

Leung KS, Leung Y, So L, Yam KF (1992) Rule learning in expert systems using genetic algorithm: 1, concepts. In: Proceedings of the 2nd international conference on fuzzy logic and neural networks, Iizuka (Japan), pp 201–204

Lin CJ, Xu YJ (2006) The design of TSK-type fuzzy controllers using a new hybrid learning approach. Int J Adapt Contr Signal Process 20(1):1

Minis I, Herrmann JW, Lam G, Lin E (1999) A generative approach for concurrent manufacturability evaluation and subcontractor selection. J Manuf Syst 18(6):383–395

Moller F (1990) A scaled conjugate gradient algorithm for fast supervised learning. Neural Netw 6:525–533

Papadakis SE, Theocharis JB (2006) A genetic method for designing TSK models based on objective weighting: application to classification problems. Soft Comput 10(9):805–824

Shabtay D, Steiner G (2007) A survey of scheduling with controllable processing times. Eur J Oper Res 155:1643–1666

Sugeno M, Kang GT (1988) Structure identification of fuzzy model. Fuzzy Sets Syst 28:15–33

Takagi T, Sugeno M (1985) Fuzzy identification of systems and its application to modeling and control. IEEE Trans Syst Man Cybern SMC 15:116–132

Wang L-X, Mendel JM (1992) Generating fuzzy rules by learning from examples. IEEE Trans Syst Man Cybern 22(6):1414–1427

Wong ML, Lam W, Leung KS, Ngan PS, Cheng JCY (2000) Discovering knowledge from medical databases using evolutionary algorithms. IEEE Eng Med Biol Mag 19(4):45–55

Yen J, Wang L, Gillespie CW (1998) Improving the interpretability of TSK fuzzy models by combining global learning and local learning. IEEE Trans Fuzzy Syst 6(4):530–537