Evaluation of Novel Soft Computing Methods for the Prediction of the Dental Milling Time-Error Parameter

Pavel Krömer[1,2], Tomáš Novosád[1], Václav Snášel[1,2], Vicente Vera[4], Beatriz Hernando[4], Laura García-Hernández[7], Héctor Quintián[3], Emilio Corchado[2,3], Raquel Redondo[5], Javier Sedano[6], and Alvaro E. García[4]

[1] Dept. of Computer Science, VŠB-Technical University of Ostrava, Czech Republic
[2] IT4Innovations, Ostrava, Czech Republic
{pavel.kromer,tomas.novosad,vaclav.snasel}@vsb.cz
[3] Departamento de Informática y Automática, Universidad de Salamanca, Spain
escorchado@usal.es
[4] Facultad de Odontología, UCM, Madrid, Spain
{vicentevera,aegarcia}@odon.ucm.es
[5] Department of Civil Engineering, University of Burgos, Burgos, Spain
rredondo@ubu.es
[6] Dept. of AI & Applied Electronics,
Castilla y León Technological Institute, Burgos, Spain
javier.sedano@itcl.es
[7] Area of Project Engineering, University of Cordoba, Spain
ir1gahel@uco.es

Abstract. This multidisciplinary study presents the application of two well known soft computing methods – flexible neural trees, and evolutionary fuzzy rules – for the prediction of the error parameter between real dental milling time and forecast given by the dental milling machine. In this study a real data set obtained by a dynamic machining center with five axes simultaneously is analyzed to empirically test the novel system in order to optimize the time error.

Keywords: soft computing, dental milling, prediction, evolutionary algorithms, flexible neural trees, fuzzy rules, industrial applications.

1 Introduction

Accurate scheduling and planning becomes increasingly important part of modern industrial processes. To optimize the manufacturing of products and schedule the utilization of devices, the product manufacturing time has to be known in advance. However, the predictions given by traditional methods and tools are often less accurate. Precise prediction of product manufacturing time is important for industrial production planning in order to meet, industrial, technological, and economical objectives [2,16]. One of the main goals of a production process is to deliver products on time and utilize the resources at maximum during

V. Snasel et al. (Eds.): SOCO Models in Industrial & Environmental Appl., AISC 188, pp. 163–172.
springerlink.com

production cycles. The production time estimate provided either by production models (i.e. by auxiliary software) or human experts are often less accurate than desirable [2]. Soft computing techniques can be used for flexible and detailed modelling of production processes [5]. The area of soft computing represents a set of various technologies involving non-linear dynamics, computational intelligence, ideas drawn from physics, physicology and several other computational frameworks. It investigates, simulates and analyzes very complex issues and phenomena in order to solve real-world problems: such as the failures detection in dental milling process, which requires a multidisciplinary approach [13].

In this study, a real data set obtained by a dynamic machining center with five axes simultaneously is analyzed by means of two soft computing techniques to empirically test the system in order to optimize the time error. The rest of this paper is organized as follows. Section 2 and section 3 present the background on the methods used to predict dental time-error. Section 4 introduces the experimental application and in section 5 conclusions are drawn .

2 Flexible Neural Tree

Flexible neural tree (FNT) [3] is a hierarchical neural network, which is automatically created in order to solve given problem. Its structure is usually determined using some adaptive mechanism and it is intended to adapt to the problem and data under investigation [11,10,4]. Due to this property of the FNTs, it is not necessary to setup some generic static network structure not related to the problem domain beforehand.

A general and enhanced FNT model can be used for problem solving. Based on the predefined instruction/operator sets, a FNT model can be created and evolved. In this approach, over-layer connections, different activation functions for different nodes and input variables selection are allowed. The hierarchical structure could be evolved by using genetic programming. The fine tuning of the parameters encoded in the structure could be accomplished by using parameter optimization algorithms. The FNT evolution used in this study combines both approaches. Starting with random structures and corresponding parameters, it first tries to improve the structure and then as soon as an improved structure is found, it fine tunes its parameters. It then goes back to improving the structure again and, provided it finds a better structure, it again fine tunes the rules'

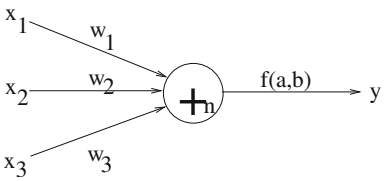


Fig. 1. A flexible neuron operator (instructor)

parameters. This loop continues until a satisfactory solution is found or a time limit is reached. A tree-structural based encoding method with specific instruction set is selected for representing a FNT model in this research. The reason for choosing the representation is that the tree can be created and evolved using the existing or modified tree-structure-based approaches. The function set F and terminal instruction set T that can be used to build a FNT model can be described as follows:

$$S = F \cup T = \{+_2, +_3, \ldots, +_N\} \cup \{x_1, x_2, \ldots, x_n\} \tag{1}$$

where $+_i$ $(i = 2, 3, \ldots, N)$ denote non-leaf nodes' instructions and taking i arguments. Input variables $x_1, x_2, \ldots, x_n$ are leaf nodes' instructions and taking no argument each. The output of a non-leaf node is calculated as a flexible neuron model. From this point of view, the instruction $+_i$ is also called a flexible neuron operator (instructor) with i inputs. A schematic view of the flexible neuron instructor is shown in fig. 1. In the *creation process* of neural tree, if a non-terminal instruction, i.e., $+_i$ is selected, i real values are randomly generated and used for representing the connection strength between the node $+_i$ and its children. In addition, two adjustable parameters a_i and b_i are randomly created as flexible activation function parameters. Activation function can vary according to given task. In this work we use following classical Gaussian activation function:

$$f(a_i, b_i, x) = e^{-(\frac{x-a_i}{b_i})^2} \tag{2}$$

The output of a flexible neuron $+_n$ can be calculated as follows. The total excitation of the $+_n$ is

$$net_n = \sum_{j=1}^{n} w_j \times x_j \tag{3}$$

where x_j $(j = 1, 2, \ldots, n)$ are the inputs to node $+_n$. The output of the node $+_n$ is then calculated by

$$out_n = f(a_n, b_n, net_n) = e^{-(\frac{net_n - a_n}{b_n})^2} \tag{4}$$

A typical evolved flexible neural tree model is shown in fig. 2. The overall output of a flexible neural tree can be computed from left to right by depth-first method, recursively.

The fitness function maps the FNT to a scalar, real-valued fitness values that reflect the FNT's performances on a given task. Firstly the fitness functions should be seen as error measures, i.e. mean square error (MSE) or root mean square error (RMSE). A secondary non-user-defined objective for which algorithm always optimizes FNTs is FNT size as measured by number of nodes. Among FNTs with equal fitness smaller ones are always preferred. MSE and RMSE are given by:

$$MSE(i) = \frac{1}{P} \sum_{j=1}^{P} (y_1^j - y_2^j)^2, \qquad RMSE(i) = \sqrt{MSE(i)} \tag{5}$$

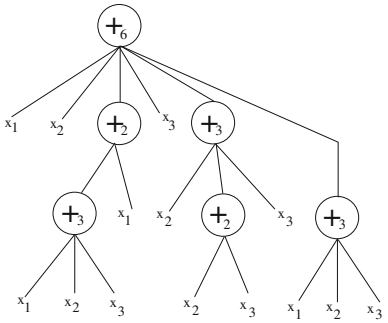


Fig. 2. A typical representation of neural tree with function instruction set $F = \{+_2, +_3, +_4, +_5, +_6\}$, and terminal instruction set $T = \{x_1, x_2, x_3\}$

where P is the total number of samples, y_1^j and y_2^j are the actual time-series and the FNT model output of j-th sample. $MSE(i)$ and $RMSE(i)$ denotes the fitness value of i-th individual.

Finding an optimal or near-optimal flexible neural tree can be accomplished by various evolutionary and bio-inspired algorithms [11,10,4]. The general learning procedure for constructing the FNT model can be described in high level as follows [3]:

1. Set the initial values of parameters used in the GA algorithms. Set the elitist program as NULL and its fitness value as a biggest positive real number of the computer at hand. Create a random initial population (flexible neural trees and their corresponding parameters)
2. Structure optimization by genetic algorithm, in which the fitness function is calculated by MSE or RMSE
3. If a better structure is found and no better structure is found for certain number of generations, then go to step (4), otherwise go to step (2)
4. Parameter optimization by genetic algorithms. In this stage, the tree structure or architecture of flexible neural tree model is fixed, and it is the best tree taken from the sorted population of trees. All of the parameters used in the best tree formulated a parameter vector to be optimized by local search
5. If the maximum number of local search is reached, or no better parameter vector is found for a significantly long time then go to step (6); otherwise go to step (4);
6. If satisfactory solution is found, then the algorithm is stopped; otherwise go to step (2).

Evolutionary methods [1] are in this study used for FNT structure optimization as well as for activation function parameters and tree nodes weights optimization. The selection, crossover and mutation operators used are the same as those of standard genetic programming [1]. A genetic algorithm starts with selection of two parents from current population. The product of crossover operator can be one or more offspring - two in this study. The mutation of offspring is performed

at the last step of genetic algorithm. After these three steps we have new offspring which is placed into a newly created population. The process is repeated until desired new population is built. As soon as the new population is built, the new population is evaluated and sorted according to the fitness function.

Selection is in the FNT evolution implemented using the weighted roulette wheel algorithm and the tree structure crossover is implemented as an exchange of randomly selected subtrees of parent chromosomes. The crossover of node weights and activation function parameters is done in a similar way as in previous studies applying genetic algorithms to neural network training [6]. A variety of FNT mutation types were used:

1. Changing one terminal node: randomly select one terminal node in the neural tree and replace it with another terminal node.
2. Changing one function node: randomly select one function node and replace it with a newly generated subtree.
3. Growing: select a random function node in hidden layer of the neural tree and add newly generated subtree as a new child.
4. Pruning: randomly select a node in the neural tree and delete it in the case the parent node has more than two child nodes.

The mutation of tree weights and activation function parameters is the same as in the genetic algorithms for artificial neural networks [6].

3 Fuzzy Rules Evolved by Genetic Programming

Fuzzy rules (FR) [7,8,15] inspired by the area of fuzzy information retrieval (IR) [9] and evolved by genetic programming have been shown to achieve interesting results in the area of data mining and pattern analysis.

The fuzzy rules use similar data structures, basic concepts, and operations as the fuzzy information retrieval but they can be used for the analysis (i.e. classification, prediction) of general data. A fuzzy rule has the form of a weighted symbolic expression roughly corresponding to an extended Boolean query in the fuzzy IR analogy. The rule consists of weighted feature (attribute) names and weighted aggregation operators. The evaluation of such an expression assigns a real value from the range $[0, 1]$ to each data record. Such a valuation can be interpreted as an ordering or a fuzzy set over the data records. The fuzzy rule is a symbolic expression that can be parsed into a tree structure. The tree structure consists of nodes and leaves (i.e. terminal nodes). An example of fuzzy rule is give below:

feature1:0.5 and:0.4 (feature2[1]:0.3 or:0.1 ([1]:0.1 and:0.2 [2]:0.3))

In the fuzzy rule syntax can be seen three types of nodes: the feature node is defined by feature name and its weight (*feature1:0.5*) and represents a requirement on current value of a feature, past feature node is defined by feature name, index of previous record, and weight (*feature2[1]:0.3*) and it is requirement on previous value of a feature. Finally, the past output node is defined by the index

of previous output and weight (*[1]:0.5*) and represents a requirement on previous value of the predicted output variable. Clearly, such a fuzzy rule can be used for the analysis of both, data sets consisting of independent records and time series.

The fuzzy rules are evaluated using the formulas and equations from the area of fuzzy IR and fuzzy sets (see e.g. [7,8,15]). The terminal node weights are interpreted as threshold for data feature values and operator nodes are mapped to fuzzy set operators. The fuzzy rule predicting certain value for a given data set is found using standard genetic programming that evolves a population of tree representations of the rules in a supervised manner. The whole procedure is very similar to the evolution of the FNT structure described in section 2 but it differs in the choice of the fitness function which is taken from the area of fuzzy IR. The correctness of search results in IR can be evaluated using the measures precision P and recall R. Precision corresponds to the probability of retrieved document to be relevant and recall can be seen as the probability of retrieving a relevant document. Precision and recall in the extended Boolean IR model can be defined using the $\Sigma-$count $\|A\|$ [19]:

$$\rho(X|Y) = \begin{cases} \frac{\|X \cap Y\|}{\|Y\|} & \|Y\| \neq 0 \\ 1 & \|Y\| = 0 \end{cases}, \; P = \rho(REL|RET), \; R = \rho(RET|REL) \quad (6)$$

where REL stands for the fuzzy set of all relevant documents, RET for the fuzzy set of all retrieved documents, and $\|A\|$ is the $\Sigma-$count, i.e. the sum of the values of characteristic function μ_A for all members of the fuzzy set $\|A\| = \sum_{x \in A} \mu_A(x)$ [19]. The F-score F is among the most used scalar combinations of P and R:

$$F = \frac{(1 + \beta^2)PR}{\beta^2 P + R} \quad (7)$$

For the evolution of fuzzy rules [7,8,15] we map the prediction given for training data set by the fuzzy rule to RET and the desired values to REL. F corresponds to the similarity of two fuzzy sets and a fuzzy rule with high F provides good approximation of the output value.

4 Dental Miling Time-Error Prediction in Industry

FNTs and FRs were used for the estimation of the time-error parameter in a real dental milling process. The data was gathered by means of a Machining Milling Center of HERMLE type-C 20 U (iTNC 530), with swivelling rotary (280 mm), with a control system using high precision drills and bits.

The models were trained using an initial data set of 98 samples obtained by the dental scanner in the manufacturing of dental pieces with different tool types (plane, toric, spherical and drill). The data set contained records consisting of 8 input variables (Tool, Radius, Revolutions, Feed rate X, Y and Z, Thickness, Initial Temperature) and 1 output variable (Time Error for manufacturing) as shown in table 1. Time error for manufacturing is the difference between the

Table 1. Description of variables in the data set

Variable (Units)	Range of values
Type of tool	Plane, toric, spherical and drill
Radius (mm.)	0.25 to 1.5
Revolutions per minute (RPM)	7,500 to 38,000
Feed rate X (mm. by minute)	0 to 3,000
Feed rate Y (mm. by minute)	0 to 3,000
Feed rate Z (mm. by minute)	50 to 2,000
Thickness (mm.)	10 to 18
Temperature (°C)	24.1 to 31
Real time of work (s)	6 to 1,794
Time errors for manufacturing (s)	-28 to -255

time estimated by the machine itself and real production time. Negative values indicate that real time exceeds estimated time. The goal of this study was to evaluate the ability of evolutionary evolved FNTs and FRs to predict the dental milling time-error from the data. The parameters used for the evolution of the FNT and FR are shown in table 2. They were selected on the basis of initial experiments and past experience with the methods.

Because the number of records in the data set was small, a 10-fold cross-validation schema was selected. The final model is obtained using the full data set. Next, several different indexes were used to validate the models [18,17] such as the percentage representation of the estimated model, the loss (error) function ($\mathcal{V}$) and the generalization error value.

The percentage representation of the estimated model was calculated as the normalised mean error for the prediction (FIT1, FIT) using the validation data set and full data set respectively. The loss function $\mathcal{V}$ is the numeric value of the MSE that was computed using the training data set, the generalisation error value is the numeric value of the normalised sum of square errors (NSSE) that was computed using the test data set [12,14].

The results of both methods are shown in table 3. The presented values are averages after 10 independent runs for each of the 10 folds. Clearly, the FNT method was significantly better than FRs which in turn delivered results similar to those by the previously used soft computing methods [16]. Visual illustration of the time-error prediction by FNT and FR for first fold is shown in fig. 3 and fig. 4 respectively. Note that both methods are stochastic and the results may vary for independent runs.

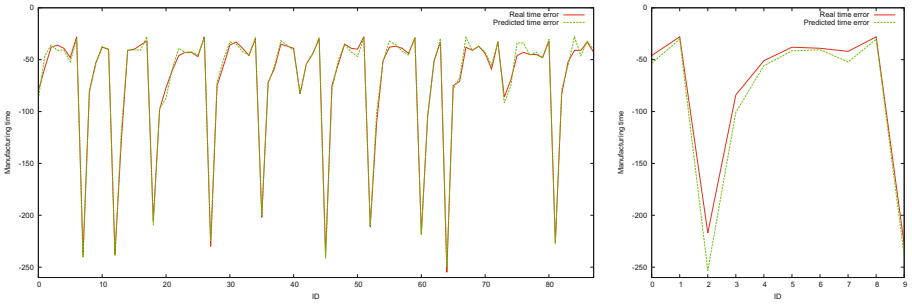
FNT and FR have shown a good ability to learn the relations hidden in the data as shown in fig. 3a and fig. 4a and indicated by high FIT and low $\mathcal{V}$ in table 3 [16]. The good generalization ability of the methods is illustrated in fig. 3b and fig. 4b and supported by low NSSE and high FIT1 in fig. 3b. The results obtained by the FNT model are best-so-far for the dental milling time-error parameter prediction.

Table 2. FNT and FR evolution parameters

Method	Parameters
FNT	pop. size 100, crossover probability P_C 0.8, mutation probability P_M 0.2, limiting number of 10 generations, fitness function $RMSE$, Gaussian activation function with a, b, and weights from the range $[0, 1]$
FR	pop. size 100, crossover probability P_C 0.8, mutation probability P_M 0.2, limiting number of 1000 generations, no past feature nodes and no past output nodes allowed, fitness function F-Score with $\beta = 1$

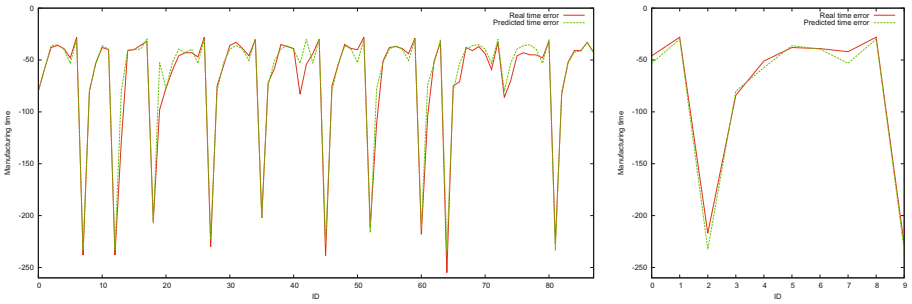
Table 3. Dental milling time-error prediction indexes

Method	FIT1[%]	FIT[%]	$\mathcal{V}$	NSSE
FNT	95.89	92.02	0.0041	0.0150
FR	86.80	86.75	0.0079	0.0888



(a) Training data set (b) Test data set

Fig. 3. Example of visual results of training and prediction by FNT (fold 1)



(a) Training data set (b) Test data set

Fig. 4. Example of visual results of training and prediction by FRs (fold 1)

5 Conclusions

This study presents the comparison of some performance indexes of two well known soft computing methods for the prediction of the dental milling time-error parameter. Both soft computing models were trained on a real-world data set describing the production of a dental milling machine and their ability to adapt to the data was compared. To provide a good analysis of the performance of the methods, a 10-fold cross-validation was performed. The results of the cross-validation showed that the FNT managed to find models with significantly better average accuracy in terms of FIT, FIT1, $\mathcal{V}$, and NSSE. The FNTs will be further studied as predictors of the dental milling time-error and other parameters such as accuracy.

Acknowledgements. This research is partially supported through a projects of the Spanish Ministry of Economy and Competitiveness [ref: TIN2010-21272-C02-01] (funded by the European Regional Development Fund). The authors would also like to thank to ESTUDIO PREVIO and TARAMI (both from Madrid, Spain) for their collaboration in this research. This work was also supported by the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070) and by the Bio-Inspired Methods: research, development and knowledge transfer project, reg. no. CZ.1.07/2.3.00/20.0073 funded by Operational Programme Education for Competitiveness, co-financed by ESF and state budget of the Czech Republic.

References

1. Affenzeller, M., Winkler, S., Wagner, S., Beham, A.: Genetic Algorithms and Genetic Programming: Modern Concepts and Practical Applications. Chapman & Hall/CRC (2009)
2. Chang, P., Liao, T.: Combining som and fuzzy rule base for flow time prediction in semiconductor manufacturing factory. Applied Soft Computing 6(2), 198–206 (2006)
3. Chen, Y., Abraham, A.: Flexible Neural Tree: Foundations and Applications. In: Chen, Y., Abraham, A. (eds.) Tree-Structure Based Hybrid Computational Intelligence. ISRL, vol. 2, pp. 39–96. Springer, Heidelberg (2010)
4. Chen, Y., Yang, B., Meng, Q.: Small-time scale network traffic prediction based on flexible neural tree. Appl. Soft Comput. 12(1), 274–279 (2012)
5. Custodio, L.M.M., Sentieiro, J.J.S., Bispo, C.F.G.: Production planning and scheduling using a fuzzy decision system. IEEE Transactions on Robotics and Automation 10(2), 160–168 (1994)
6. Ding, S., Li, H., Su, C., Yu, J., Jin, F.: Evolutionary artificial neural networks: a review. Artificial Intelligence Review, 1–10 (2011), doi:10.1007/s10462-011-9270-6
7. Krömer, P., Platoš, J., Snášel, V., Abraham, A.: Fuzzy classification by evolutionary algorithms. In: IEEE International Conference on Systems, Man, and Cybernetics, pp. 313–318. IEEE System, Man, and Cybernetics Society (2011)

8. Krömer, P., Platoš, J., Snášel, V., Abraham, A., Prokop, L., Mišák, S.: Genetically evolved fuzzy predictor for photovoltaic power output estimation. In: 2011 Third International Conference on Intelligent Networking and Collaborative Systems (INCoS), pp. 41–46. IEEE (2011)
9. Pasi, G.: Fuzzy sets in information retrieval: State of the art and research trends. In: Bustince, H., Herrera, F., Montero, J. (eds.) Fuzzy Sets and Their Extensions: Representation, Aggregation and Models. STUDFUZZ, vol. 220, pp. 517–535. Springer, Heidelberg (2008)
10. Peng, L., Yang, B., Zhang, L., Chen, Y.: A parallel evolving algorithm for flexible neural tree. Parallel Computing 37(10-11), 653–666 (2011)
11. Qi, F., Liu, X., Ma, Y.: Synthesis of neural tree models by improved breeder genetic programming. Neural Computing & Applications 21, 515–521 (2012), doi:10.1007/s00521-010-0451-z
12. Sedano, J., Corchado, E., Villar, J., Curiel, L., de la Cal, E.: Detection of heat flux failures in building using a soft computing diagnostic system. Neural Network World 20(7), 883–898 (2010)
13. Sedano, J., Curiel, L., Corchado, E., de la Cal, E., Villar, J.R.: A soft computing method for detecting lifetime building thermal insulation failures. Integr. Comput.-Aided Eng. 17(2), 103–115 (2010)
14. Sedano, J., Curiel, L., Corchado, E., de la Cal, E., Villar, J.R.: A soft computing method for detecting lifetime building thermal insulation failures. Integr. Comput.-Aided Eng. 17(2), 103–115 (2010)
15. Snášel, V., Krömer, P., Platoš, J., Abraham, A.: The Evolution of Fuzzy Classifier for Data Mining with Applications. In: Deb, K., Bhattacharya, A., Chakraborti, N., Chakroborty, P., Das, S., Dutta, J., Gupta, S.K., Jain, A., Aggarwal, V., Branke, J., Louis, S.J., Tan, K.C. (eds.) SEAL 2010. LNCS, vol. 6457, pp. 349–358. Springer, Heidelberg (2010)
16. Vera, V., Corchado, E., Redondo, R., Sedano, J., Garcia, A.: Applying soft computing techniques to optimize a dental milling process. Neurocomputing (submitted)
17. Vera, V., Garcia, A.E., Suarez, M.J., Hernando, B., Corchado, E., Sanchez, M.A., Gil, A.B., Redondo, R., Sedano, J.: A bio-inspired computational high-precision dental milling system. In: NaBIC, pp. 423–429. IEEE (2010)
18. Vera, V., Garcia, A.E., Suarez, M.J., Hernando, B., Redondo, R., Corchado, E., Sanchez, M.A., Gil, A.B., Sedano, J.: Optimizing a dental milling process by means of soft computing techniques. In: ISDA, pp. 1430–1435. IEEE (2010)
19. Zadeh, L.A.: Test-score semantics dor natural languages and meaning representation via Pruf. In: Empirical Semantics. Quantitative Semantics, vol. 1, pp. 281–349. Studienverlag Brockmeyer, Bochum (1981)