Evolutionary computation method for modeling of material properties

Leo GUSEL
University of Maribor, Faculty of Mechanical Engineering
Slovenia

1. Introduction

In order to reach high quality of the metal forming process and full functionality of the product, the properties of the material which the future product will be made of, have to be determined as precisely as possible. One of very important properties of material is impact toughness, which is defined as the ability of a material to resist fracture under the effect of shock loading. It is defined by the energy required to break a piece of metal of standard shape and with a cross-sectional area of 1 cm^2 (Lange, 2001). A test called the Charpy test is used for evaluation of impact toughness (CVN) of a variety of mass produced materials and is of great value for the selection of materials and for quality control (Tanguy et. al., 2005).
To get precise data and to reduce the cost of the experiment, several modeling methods predicting the values of the dependent output variables (i.e. system behavior) have been used so far (Mondal & Maiti, 2002, Mohanty et. al., 2003, Özel & Karpat, 2005). Traditional methods often employed to solve complex real problems tend to inhibit elaborate explorations of the search space. They can be expensive and often results in sub-optimal solutions. In most traditional modeling methods, such as multiple regression analysis, a prediction model is determined in advance. Merely a set of coefficients has to be found by the deterministic procedure. Because of the pre-specified shape and size of the model, the latter is often not capable of capturing the complex relation among influencing parameters and the model would not be precise enough for industrial applicability.

Evolutionary computation (*EC*) is generating considerable interest for solving real engineering problems. They are proving robust in delivering global optimal solutions and helping to resolve limitations encountered in traditional methods. *EC* harnesses the power of natural selection to turn computers into optimization tools. It is very applicable to different problems in manufacturing industry (Brezocnik et. al., 2005, Dimitriu et. al., 2009, Odugava et. al., 2005, Pierreval et. al., 2003, Sette et. al., 2001). One of the core methods of evolutionary computing is *genetic programming* (*GP*) method, which use the genetic algorithm paradigm to derive computer expressions to solve a given problem. The aim is often to build models capable of predicting output values from input values. It is very important that the influence of the independent input variables on the dependent output variables and, consequently, on the product quality can be examined in the early stage of process planning. Genetic programming method is most often used for complex system modeling (Chang et. al., 2005, Fakhrazd et. al. 2009). But with proper selection of several genetic parameters, genetic programming can also be effectively used for modeling of relatively simple system, such as system described in our paper, since it has a great advantage of being more accurate than conventional methods (e.g. regression analysis).

In addition, when the complexity of the environment is increasing (e.g., more system parameters, more experiments), the adaptation of genetic programming run on such new environment is relative easily. In genetic programming, prevention against quick fall in local optimum (i.e. genetic model, which does not provide a suitable system solution) can be assured by different measures:

- With sufficient number of input variables (i.e. terminal genes *T*), influencing the output variable of the genetic programming process.
- With proper selection of function genes *F* for adequate description of the relation between system variables.
- With adequate mutation probability (assures new genetic material in the population).

In the chapter, an approach completely different from the conventional methods for determination of accurate models for the change of material properties, is presented. This approach is genetic programming method which is one of evolutionary computation methods and is based on imitation of natural evolution of living organisms. Genetic programming is a domain-independent method that genetically breeds a population of computer programs to solve a problem. Specifically, genetic programming iteratively transforms a population of computer programs into a new generation of programs by applying analogs of naturally occurring genetic operations.

The main characteristic of genetic programming is its non-deterministic way of computing. No assumptions about the form and size of expressions were made in advance, but they were left to the self organization and intelligence of evolutionary process. Genetic programming method can automatically create, in a single run, a general (parameterized) solution to a problem in the form of a graphical structure whose nodes or edges represent components and where the parameter values of the components are specified by mathematical expressions containing free variables. That is, genetic programming can automatically create a general solution to a problem in the form of a parameterized topology.

In this chapter an example of genetic programming modeling of impact toughness of formed material is described. First, copper alloy rods were cold drawn under different conditions and then impact toughness of cold drawn specimens was determined by Charpy tests. The values of independent variables (effective strain, coefficient of friction) influence the value of the dependent variable, impact toughness. On the basis of training data, different prediction models for impact toughness were developed by genetic programming. The study showed that a genetic programming approach is suitable for system modeling.

During our research, several different models for impact toughness satisfying the criterion of the success were discovered. The obtained models differ in size, shape, complexity and precision of the solution. Only the best models, gained by genetic programming are presented in the paper. Accuracy of the best models was proved with the testing data set. The comparison between deviation of genetic model results and regression model results concerning the experimental results is presented in the chapter. Because the proposed genetic programming method is general, it can be successfully used for modeling of different materials properties and phenomena where experimental data on the process are available.

The book chapter is organized as follows. A description of genetic programming method is given in Section 2. A description of experimental work and experimental results is given in Section 3. Section 4 shows the fitness measure, genetic operations and values of genetic parameters used for genetic programming method. Results, discussion and comparison between the best genetically developed model results, regression model results and experimental results are given in Section 5. Finally, some concluding remarks are given in Section 6.

2. Methods used

One of the central challenges of computer science is to get a computer to do what needs to be done, without telling it how to do it. Genetic programming addresses this challenge by providing a method for automatically creating a working computer program from a high-level problem statement of the problem. Genetic programming achieves this goal of automatic programming by genetically breeding a population of computer programs using the principles of Darwinian natural selection and biologically inspired operations (Koza et. al., 1999). The operations include reproduction, crossover (sexual recombination), mutation, and architecture-altering operations patterned after gene duplication and gene deletion in nature.
Genetic programming starts from a high-level statement of the requirements of a problem and attempts to produce a computer program that solves the problem. The human user communicates the high-level statement of the problem to the genetic programming system by performing certain well-defined preparatory steps (Koza et. al., 2003). The five major preparatory steps for the basic version of genetic programming require the specification of:

1. the set of terminals (e.g., the independent variables of the problem, zero-argument functions, and random constants) for each branch of the to-be-evolved program,
2. the set of primitive functions for each branch of the to-be-evolved program,
3. the fitness measure (for explicitly or implicitly measuring the fitness of individuals in the population),
4. certain parameters for controlling the run, and
5. the termination criterion and method for designating the result of the run.

Figure 1 shows the five major preparatory steps for genetic programming method. The preparatory steps (shown at the top of the figure) are the human-supplied input to the genetic programming system. The computer program is the output of the genetic programming system.

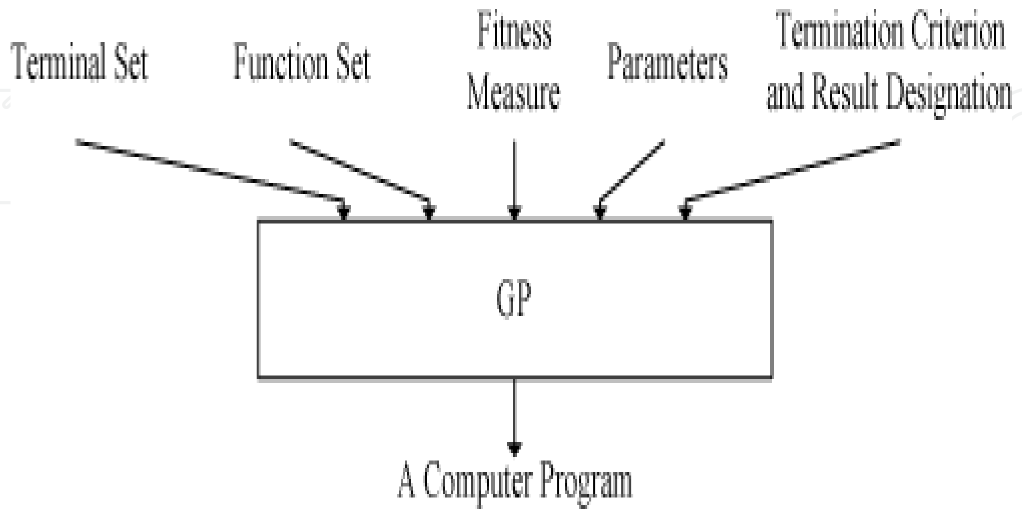

Fig. 1. Major preparatory steps for genetic programming (GP) method

The first two preparatory steps specify the ingredients that are available to create the computer programs. A run of genetic programming is a competitive search among a diverse population of programs composed of the available functions and terminals.

Function set and terminal set:

The identification of the function set and terminal set for a particular problem is usually a straightforward process. For some problems, the function set may consist of merely the arithmetic functions of addition, subtraction, multiplication, and division as well as a conditional branching operator. The terminal set may consist of the program's external inputs (independent variables) and numerical constants. This function set and terminal set is useful for a wide variety of problems. For many other problems, the ingredients include specialized functions and terminals (Koza, 1994).

Fitness Measure:

The third preparatory step concerns the fitness measure for the problem. The fitness measure specifies what needs to be done. The fitness measure is the primary mechanism for communicating the high-level statement of the problem's requirements to the genetic programming system. It is, for many practical problems, multi-objective in the sense that it combines two or more different elements. The different elements of the fitness measure are often in competition with one another to some degree. The first two preparatory steps define the search space whereas the fitness measure implicitly specifies the search's desired goal.

Control Parameters:

The fourth and fifth preparatory steps are administrative. The fourth preparatory step entails specifying the control parameters for the run. The most important control parameter is the population size. In practice, the user may choose a population size that will produce a reasonably large number of generations in the amount of computer time we are willing to devote to a problem (as opposed to, say, analytically choosing the population size by somehow analyzing a problem's fitness landscape). Other control parameters include the

probabilities of performing the genetic operations, the maximum size for programs, and other details of the run.
Termination:
The fifth preparatory step consists of specifying the termination criterion and the method of designating the result of the run. The termination criterion may include a maximum number of generations to be run as well as a problem - specific success predicate (Koza, 1992). In practice, one may manually monitor and manually terminate the run when the values of fitness for numerous successive best-of-generation individuals appear to have reached a plateau. The single best-so-far individual is then harvested and designated as the result of the run. After the human user has performed the preparatory steps for a problem, the run of genetic programming can be launched. Once the run is launched, a series of well-defined, problem-independent executional steps (that is, the flowchart of genetic programming) is executed.
Genetic programming starts with an initial population of computer programs composed of functions and terminals appropriate to the problem. The individual programs in the initial population are typically generated by recursively generating a rooted point-labeled program tree composed of random choices of the primitive functions and terminals (provided by the human user as part of the first and second preparatory steps of a run of genetic programming). The initial individuals are usually generated subject to a pre-established maximum size (specified by the user as a minor parameter as part of the fourth preparatory step). In general, the programs in the population are of different size (number of functions and terminals) and of different shape.
Each individual program in the population is executed. Then, each individual program in the population is either measured or compared in terms of how well it performs the task at hand (using the fitness measure provided in the third preparatory step). For many problems, this measurement yields a single explicit numerical value, called *fitness*. The fitness of a program may be measured in many different ways, including, for example, in terms of the amount of error between its output and the desired output, the amount of time required to bring a system to a desired target state, etc.. The execution of the program sometimes returns one or more explicit values. Alternatively, the execution of a program may consist only of side effects on the state of a world (e.g., a robot's actions).
The creation of the initial random population is, in effect, a blind random search of the search space of the problem. It provides a baseline for judging future search efforts. Typically, the individual programs in generation 0 all have exceedingly poor fitness. Nonetheless, some individuals in the population are usually more fit than others. The differences in fitness are then exploited by genetic programming. Genetic programming applies Darwinian selection and the genetic operations to create a new population of offspring programs from the current population.
The genetic operations include crossover, mutation, reproduction, and the architecture-altering operations. These genetic operations are applied to individual(s) that are probabilistically selected from the population based on fitness. In this probabilistic selection process, better individuals are favored over inferior individuals. However, the best individual in the population is not necessarily selected and the worst individual in the population is not necessarily passed over (Koza et. al., 1999).

The Figure 2 is a flowchart showing the executional steps of a run of genetic programming (Koza, 1994). The flowchart shows the genetic operations of crossover, reproduction, and mutation as well as the architecture-altering operations. This flowchart shows a two-offspring version of the crossover operation.

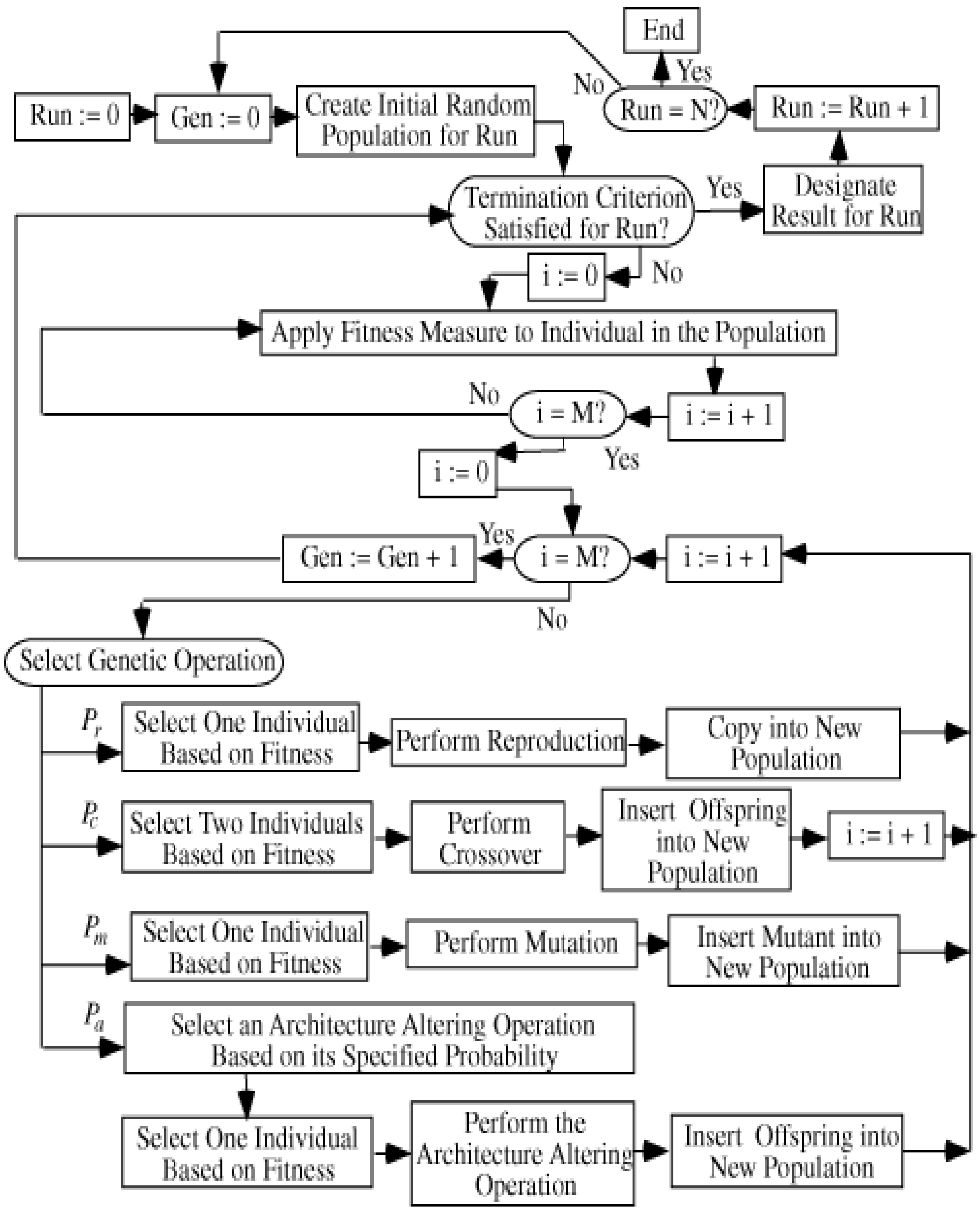
End
Yes
No
Run := 0
Gen := 0
Create Initial Random Population for Run
Run = N?
Run := Run + 1
Termination Criterion Satisfied for Run?
Yes
Designate Result for Run
i := 0
No
Apply Fitness Measure to Individual in the Population
No
i = M?
i := i + 1
Yes
i := 0
Yes
Gen := Gen + 1
i = M?
i := i + 1
No
Select Genetic Operation
P_r
Select One Individual Based on Fitness
Perform Reproduction
Copy into New Population
P_c
Select Two Individuals Based on Fitness
Perform Crossover
Insert Offspring into New Population
i := i + 1
P_m
Select One Individual Based on Fitness
Perform Mutation
Insert Mutant into New Population
P_a
Select an Architecture Altering Operation Based on its Specified Probability
Select One Individual Based on Fitness
Perform the Architecture Altering Operation
Insert Offspring into New Population

Fig. 2. Flowchart of genetic programming method

In the crossover operation (Fig. 3), two parental programs are probabilistically selected from the population based on fitness. The two parents participating in crossover are usually of different sizes and shapes. A crossover point is randomly chosen in the first parent and a crossover point is randomly chosen in the second parent. Then the subtree rooted at the crossover point of the first, or receiving, parent is deleted and replaced by the subtree from the second, or contributing, parent. Crossover is the predominant operation in genetic programming (and genetic algorithm) work and is performed with a high probability.

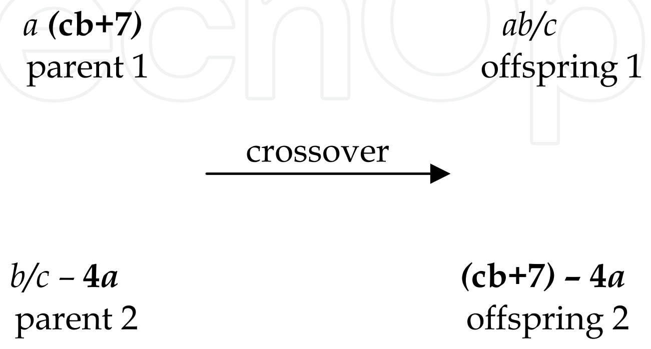

Fig. 3. Crossover operation in genetic programming.

In the mutation operation, a single parental program is probabilistically selected from the population based on fitness. A mutation point is randomly chosen, the subtree rooted at that point is deleted, and a new subtree is grown there using the same random growth process that was used to generate the initial population. Mutation operation is typically performed sparingly with a low probability during each generation of the run.
The reproduction operation copies a single individual, probabilistically selected based on fitness, into the next generation of the population.
After the genetic operations are performed on the current population, the population of offspring (i.e., the new generation) replaces the current population. This iterative process of measuring fitness and performing the genetic operations is repeated over many generations. The main generational loop of a run of genetic programming consists of the fitness evaluation, Darwinian selection, and the genetic operations. Each individual program in the population is evaluated to determine how fit it is at solving the problem at hand. Programs are then probabilistically selected from the population based on their fitness to participate in the various genetic operations, with reselection allowed. While a more fit program has a better chance of being selected, even individuals known to be unfit are allocated some trials in a mathematically principled way (Koza et. al., 2003). That is, genetic programming is not a purely greedy hill-climbing algorithm.
After a certain number of generations the computer programs are usually much better adapted to the environment. The meaning of the environment depends on the problem dealt with. The evolution is terminated when a termination criterion is fulfilled. This can be a prescribed number of generations or sufficient quality of the solution. Since evolution is a non-deterministic process, it does not end with a successful solution in each run (i.e., civilization). In order to obtain a successful solution, the problem must be processed in several independent runs. The number of runs required for the satisfactory solution depends on the difficulty of the problem.

LISP language is especially well suited for *GP* because there is no syntactic distinction between programs and data (Mitchell, 1997). However, any other programming language that can manipulate computer programs as data and that can then compile, link and execute the new programs can be successfully used for *GP* method.
Regression analysis is deterministic method of modeling, which allows with small number of experimental points the greatest information about influence to mathematical model of the process (Barnes, 1994). First, it is necessary to determine one dependent variable and one or more independent variables. The independent variables {x_1, x_2, x_3, ...} are under our control in the sense that we can set the values of x_1, x_2, x_3,…to whatever we wish (within the physical limitations imposed by the process under study). The dependent variable, Y, is a variable that attains its realization, y, in response to the selected setting of {x_1, x_2, x_3, .. }. For this reason, Y is also called the response variable (Montgomery et. al., 2001).

3. Experimental work

The aim of the experimental work was to determine the influence of the effective strain ε_e and coefficient of friction μ in cold drawing to the change of impact toughness (CVN) of cold drawn copper alloy CuCrZr. This is a copper–chrome–zirconium alloy with 0.71% Cr, 0.05% Zr and Cu as a base. It has high electrical and thermal conductivity and excellent mechanical and physical properties also at elevated temperatures.
Copper alloy rods were deformed by cold drawing under different conditions. The drawing speed was 20 m/min and the angle of drawing die was δ = 28°. Copper alloy rods were drawn from initial diameter D= 20 mm to six different diameters (i.e. six different effective strains). Three different lubricants with different coefficients of friction (μ=0.07, μ=0.11 and μ=0.16) were used for the drawing process. The coefficients of friction were determined by ring tests (Lange, 2001). In order to evaluate the impact toughness, standard Charpy V-notch specimens (with a central 45° V-notch of 2mm depth and a 0.25 mm notch root radius) were prepared from the location in the middle of drawn rods. In this way we got 18 different experimental specimens.
Then, the impact toughness (CVN) of all specimens was determined by the standard Charpy test (SIST EN 10 045-1, V- notch) at the 20°C temperature. Three Charpy tests for each specimen were carried out to provide reliable results. The results (average values) for impact toughness are presented in Table 1**.** Experimental data serve as an environment which, during simulated evolution, the models for the impact toughness have to be adapted to.
The number of experimental data is sufficient for our problem, because the relation between two independent variables and impact toughness is relatively simple and the difference between the highest (165 J) and the lowest (143.3 J) value of impact toughness (output variable) is very small. Such problems can be solved also with smaller genetic environment.

Nr.	Effective strain ε_e	Coef. of friction μ	Impact toughness $CVN\,[\,J\,]$
initial spec.	/	/	169.3
1	0.10	0.07	165
2	0.21	0.07	161
3	0.32	0.07	159
4	0.44	0.07	155
5	0.57	0.07	151
6	0.71	0.07	148.3
7	0.10	0.11	163
8	0.32	0.11	155.3
9	0.71	0.11	145
10	0.10	0.16	160
11	0.44	0.16	152
12	0.71	0.16	143.3
13	0.21	0.11	159
14	0.44	0.11	153.3
15	0.57	0.11	149.3
16	0.21	0.16	158.3
17	0.32	0.16	154
18	0.57	0.16	147

Table 1. Experimental results

4. Modeling of impact toughness by genetic programming

In the *GP* method the initial random population *P(t)* consists of randomly generated organisms which are in fact mathematical models. The variable *t* represents the generation time. Each organism in the initial population consists of the available function genes *F* and terminal genes *T*. Terminal genes are in fact independent variables: strain and coefficient of friction. The random floating-point numbers from the range [-10, 10] are added to the set of terminals to increase genetic diversity of the organisms. Function genes *F* are basic arithmetical functions, exponential function and cosine function.

4.1 Fitness measure, genetic operations and evolutionary parameters

The absolute deviation *R* (*i*, *t*) of individual model *i* (organism) in generation time *t* for the *GP* approach, was introduced as the standard raw fitness measure (Koza, 1992):

$$R(i,t) = \sum_{j=1}^{n} \left| E(j) - P(i,j) \right|, \tag{1}$$

Where *E*(*j*) is the experimental value for measurement *j*, *P*(*i*, *j*) is the predicted value returned by the individual model *i* for measurement *j*, and *n* is the maximum number of measurements. The aim of the optimisation task is to find such models that equation (1) would give as low absolute deviation as possible.
However, because it is not necessary that the smallest values of the above equation also means the smallest percentage deviation of this model, the average absolute percentage deviation of all measurements for individual model *i* was defined as (Koza, 1992):

$$\Delta(i) = \frac{R(i,t)}{|E(j)|\, n} \cdot 100\% \tag{2}$$

Equation (2) was not used as the fitness measure for evaluation of population, but only to find the best organism in population after completing the run.
In the *GP* method, reproduction, crossover, and mutation operations were used for altering the population *P(t)*. Evaluation and altering of the population *P(t)* were repeated until termination condition has been fulfilled. In this paper, the termination condition was the prescribed maximum number of generation to be run.
The evolutionary processes were controlled by following evolutionary parameters: population size 1000, maximum number of generations to be run 30 (50), probability of reproduction 0.1, probability of crossover 0.7, probability of mutation 0.2, maximum depth for initial random organisms 6, maximum depth of mutation fragment 6, and maximum permissible depth of organisms after crossover 14. Higher mutation probability as usual was used for increasing the population variety, because mutation operation generates entire new sub-trees, which cause a larger population variety. Large population variety is one of the most important conditions for efficient *GP* modeling. The generative method for the initial random population was ramped half-and-half (Koza et. al., 1999). The method of selection for reproduction, crossover and mutation was tournament selection with a group size of 6.

4.2 Execution of the evolutionary process

Modeling of the impact toughness was carried out by the special *GP* system (computer program), which comprises 49 program units (modules), and was programmed in our laboratory. Four personal computers were used for processing. The *GP* system ensures repeated development of the individual civilization (if necessary). This is very useful when it is necessary to repeat evolution of the civilization with a greater number of generations or when evolution is interrupted for any reason.
Each individual *GP* run started with the training phase by the training data set shown in *Table 1* (Nr.1 to Nr.12). The testing data set (*Table 1*: Nr.13 to Nr. 18) was not included within the training range. Each run lasted up to the generation 30 when it was temporarily interrupted. If the average percentage deviation Δ(i) of at least one prediction model (organism) in the population was smaller than 5%, the evolution of the population continued up to generation 50, otherwise it was terminated. After each training phase, the

accuracy of the prediction of the best models was tested with the testing data set. For modeling of impact toughness by *GP* method 400 independent runs were executed. The *GP* models in our research were originally developed as prefix LISP expression. In LISP the program code as well as the data structure are in the form of symbolic expressions (s-expressions), which are constructed in prefix notation. A special program interface then converted them into an infix notation.

5. Results and discussion

5.1 GP models

All successful *GP* solutions in our research, which have fulfilled the fitness criterion, can be classified into two characteristic groups:

(1) Solutions in which the evolutionary process gradually eliminates the independent variable μ out of the developing models. The final solution does not contain the variable μ. Such models are in the rule very simple but not as precise as more complex models which contain variable μ. The best GP model, which does not contains the variable μ, was generated with the set of function genes F = {+, -, *, /} and is presented as:

$$162.139 - 19.05\varepsilon_e \tag{3}$$

It has the average percentage deviation of the training data Δ (i) = 1.73%, and that of the testing data Δ(i) = 1.10 %. Percentage deviation is in fact the percentage error between a single experimental value and the value predicted by the genetic model.

(2) Solutions in which the evolutionary process gradually locates the independent variable μ in the organisms (models). Such models are very precise but also very complex. The best *GP* model was obtained in generation 49 and is presented as LISP expression:

```
(+ (+ (+ (+ (% (% (% (− ε 6.11649) (− −8.68974 ε)) (− 8.42826 ε)) (− (* −1.4476 ε) (* −2.2
8621 ε))) 9.86097) (% (* (− ε 6.11649) (− ε (+ μ −0.143767))) (* (+ (% (% (− ε 6.11649) (−
−8.68974 ε)) μ) (+ −8.16447 μ)) (% (+ −2.55866 ε) μ)))) (% (* (+ −7.05545 μ) (− ε (+ μ −
0.143767))) (* (+ (− −8.68974 ε) (+ −8.16447 μ)) (% (+ −2.55866 ε) ε)))) (* (+ (* (+ (+ −2.
55866 ε) (+ μ ε)) (% (* (+ −2.55866 (* (% μ −6.42028) (+ μ ε))) (− (% μ −1.4476) (+ μ −0.
143767))) (* (+ (% (− ε ε) μ) (+ −8.16447 μ)) (% (% ε −1.4476) μ)))) (* (% (− ε 6.11649) (
− −8.68974 ε)) (* (− 3.54124 μ) (+ μ 8.23564)))) (− (− (− ε (+ −7.05545 μ)) (+ (% μ −1.784
02) (− ε ε))) (+ (− (* μ ε) (% (− ε 6.11649) (− −8.68974 ε))) (* (% μ −8.68974) (+ μ ε))))))
```

After simplification, the above expression is equivalent to:

$$
\begin{aligned}
&9.86+\frac{1.192\varepsilon_e-7.293}{\varepsilon_e\left(\varepsilon_e^2+0.261\varepsilon_e-73.23\right)}+\frac{\left(0.143\varepsilon_e+\varepsilon_e^2-\varepsilon_e\mu\right)(\mu-7.055)}{(\varepsilon_e-2.558)(\mu-\varepsilon_e-16.854)}+\\
&\frac{(\varepsilon_e-6.116)\left(0.143\mu+\varepsilon_e\mu-\mu^2\right)}{(\varepsilon_e-2.558)\left(\mu-8.164-{(\varepsilon_e-6.116)}\big/{(8.68\varepsilon_e+\varepsilon_e\mu)}+\mu\right)}+\\
&\left(7.055+\varepsilon_e-\frac{\varepsilon_e-6.116}{\varepsilon_e+8.689}-0.439\mu-\varepsilon_e\mu+0.11\mu^2+0.11\varepsilon_e\mu\right)\cdot\\
&\left(-\frac{(\varepsilon_e-6.116)(3.54-\mu)(8.23+\mu)}{8.689+\varepsilon_e}-\frac{1}{\varepsilon_e\mu-8.16\varepsilon_e}\left(0.208\mu-2.44\mu^2\right)(\mu+2\varepsilon_e-2.558)(-2.558-0.155\varepsilon_e\mu)\right)
\end{aligned}
\tag{4}
$$

The model (4) was generated with the set of function genes F = {+, -, *, /} and has the average percentage deviation of the training data Δ (i) = 0.22%, and that of the testing data Δ(i) = 0.20%.

Figure 4 shows the percentage deviation curve Δ between the best model of individual generation and experimental results when the set of function genes F = {+, -, *, /} was used. It is obviously that in early generations the best models are not as precise as the models generated in late generations. The relatively slow improvement of the best models in late generations is due to unification trends of the population leading to shortage of new genetic ideas.

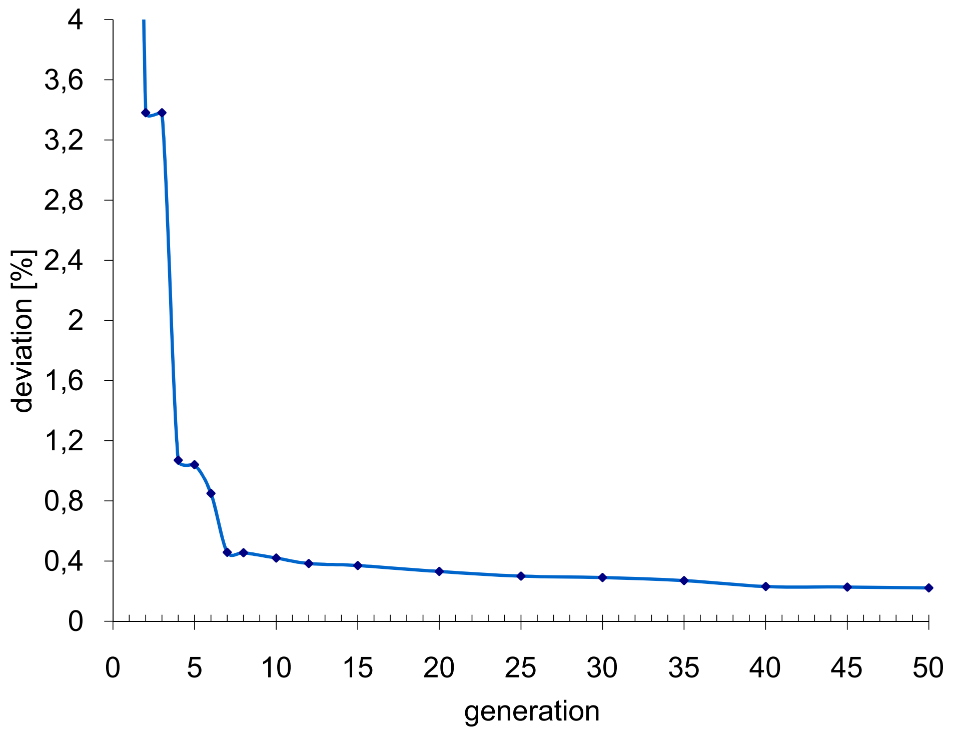

Fig. 4. Percentage deviation curve between the best model of each generation and experimental results (*F*=+, -,*, /).

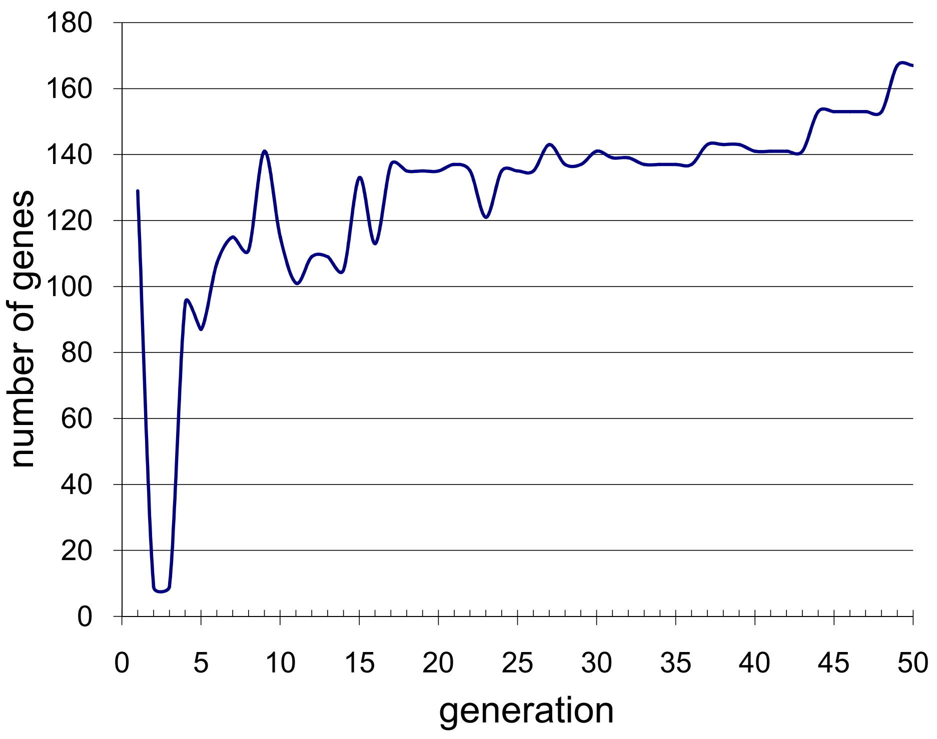

Fig. 5. Curve of all genes of the best *GP* model in each generation (*F*=+, -,*, /).

Figure 5 shows the complexity curve of the best models (generated with function genes F = {+, -, *, /}) in each generation. In generation 0 created randomly, the best models consists of 129 genes (i.e. complexity is 129). Then, at generations 2 and 3 complexity decreases and reaches the value of only 8. In the next generations complexity increases significantly and the best model (4), obtained in generation 49 contains 167 genes. Higher number of genes means higher complexity of the genetic model.

5.2. Regression model

A mathematical model for regression method was chosen according to (Montgomery et. al., 2001):

$$y(x) = b_0 + \sum_{i=1}^{n} b_i x_i + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} b_{ij} x_{ij} + \sum_{i=1}^{n} \sum_{j=i}^{n} b_{ij} x_{ij}^2 \tag{5}$$

Where y is dependent variable, x_i, x_{ij} are independent variables, while b_0, b_i, b_{ij} are coefficients to be determined by using regression analysis. In our case dependent variable was impact toughness (CVN), while effective strain ε_e and coefficient of friction μ were independent variables. Coefficients b_0, b_i and b_{ij} were determined by using regression analysis program SPSS. By inserting the computed values of coefficients into the equation (5), the regression model for impact toughness can be presented as:

$$169.589 - 25.412\, \varepsilon_e - 38.771\, \mu - 1.531\, \varepsilon_e^2 - 6.825\, \mu^2 - 9.758\, \varepsilon_e \mu \tag{6}$$

Equation (6) represents a mathematical model of the influence of effective strain and coefficient of friction on impact toughness for chosen material in experimental area. It has the average percentage deviation of the training data set Δ(i) = 0.44 % and that of the testing data set Δ(i) = 0.38 %.

5.3. Comparison between *GP* and regression models

The main characteristics of the best *GP* and regression models are collected in *Table 2*. For both the training and testing data sets the descriptions of columns are as follows: the first column indicates the average percentage deviation of all sample data $\Delta(i)$, the second column indicates the greatest deviation of single sample data $\Delta(i,j)_{max}$. The valuation of the model complexity, based on the number of numerical constants and variables of the individual model, is indicated in the last column.

The *GP* model (4) has a very complex form and also the smallest average percentage deviation of all sample data and the smallest maximum deviation $\Delta(i,j)_{max}$ of single sample data. For the training data set the prediction precision of the model *GP* (4) is 1.0 times greater and for the testing data set 1.9 times greater then the prediction precision of regression model. On the other hand, the structure form of regression model (6) and especially that of the *GP* model (3) is very simple. These models are very adequate for practical use in the case that there are no extremely high demands about accuracy.

	Training data set		Testing data set		
Model	$\Delta(i)$ [%]	$\Delta(i,j)_{max}$ [%]	$\Delta(i)$ [%]	$\Delta(i,j)_{max}$ [%]	Complexity
GP model (4)	*0.22*	*0.38*	*0.20*	*0.33*	*complex*
GP model (3)	*1.73*	*2.80*	*1.10*	*0.33*	*very simple*
REG model (6)	*0.44*	*1.12*	*0.38*	*0.71*	*simple*

Table 2. Comparison between best *GP* and regression models: $\Delta(i)$ - the average percentage deviation of all sample data, $\Delta(i,j)_{max}$ - the greatest deviation of single sample data.

Figure 6 represents the percentage error between a single experimental value and the value predicted by the two *GP* models and regression model for the training and testing data set. The x-axis represents numbers of measurements (see *Table 1*) and the y-axis the relevant percentage deviations of single sample data Δ (i) produced by each model. The *GP* model (4) has very small deviation amplitude. All deviations of predicted impact toughness are smaller than 0.5%. The greatest deviation value is 0.38% in measurement point 2. There are also no significant differences in deviation amplitude between training and testing data sets. Obtained GP model is suitable for accurate prediction of the impact toughness of copper alloy not only for data, which were included in the testing, but for every value of the effective strain in the range from $\varepsilon_e = 0.1$ to $\varepsilon_e = 0.71$ and for every value of the coefficient of friction in the range from $\mu = 0.07$ to $\mu = 0.16$. For regression model (6) and especially for simple *GP* model (3) deviations are much higher. Most deviations for regression model are in the range from 0.3% to 1% and for the simple *GP* model (3) in the range from 1% to 2.5%.

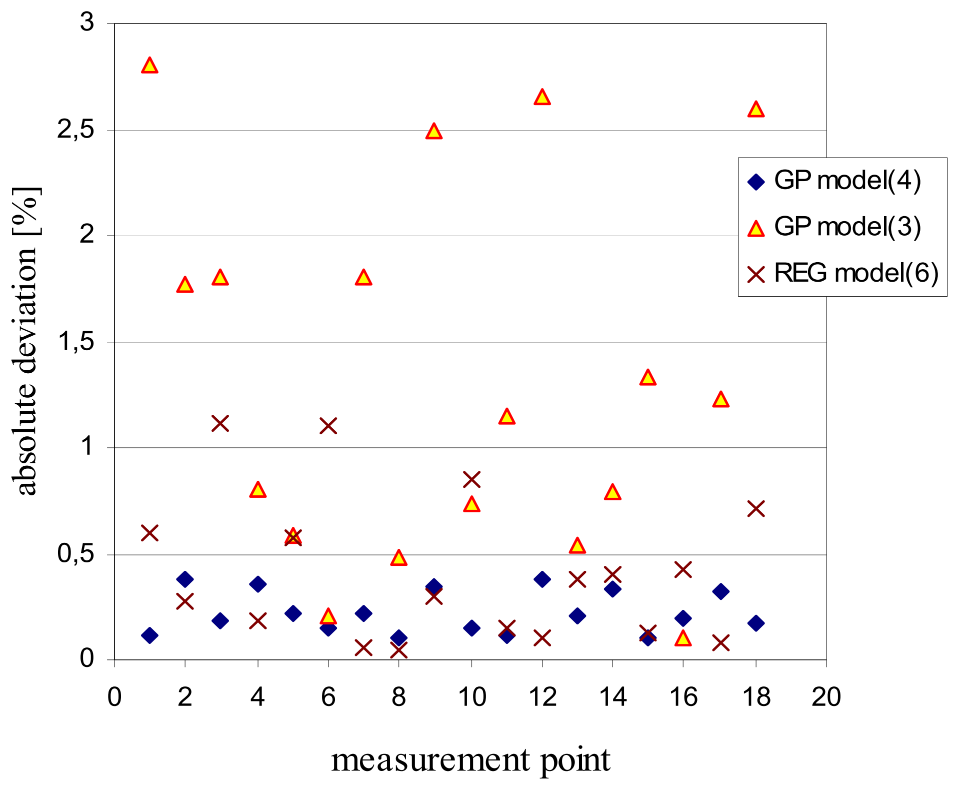

Fig. 6. Absolute deviation of model results from the experimental results in a singular measurement point.

6. Conclusion

The presented genetic programming approach for modeling of material properties (in our case impact toughness) strongly differs from the conventional methods since it does not use strict mathematical rules and does not derive equations in a rational human way of thinking. The evolutionary process is non-deterministic and involves asynchronous, uncoordinated, and self-organizing activities that are not centrally controlled. During our research, several different models for impact toughness satisfying the criterion of the success were discovered. The obtained models differ in size, shape, complexity and precision of the solution.

The study showed that a *GP* approach is suitable for system modeling. A comparison was made between *GP* and regression models. In modeling with the regression method, the form of the prediction models was pre-specified. On the contrary, in modeling with the *GP*, the form, size, and complexity of the models were left to simulated evolution. The resulting models were tested by the testing data set and compared with respect to different criteria. It is a disadvantage of the *GP* approach that the modeling lasts longer than in regression approach and that the models are much more complicated.

This is due to the fact that evolution is a stochastic process, and therefore parsimony in the development of the models is rare. But in many metal forming processes it is not the model complexity but the accuracy of prediction that is of vital importance. However, the best model developed by *GP* gives a more accurate prediction than the pre-specified model optimised by regression method, due to the fact that in *GP* a much wider solution space can be analysed since the structure of the models is not prescribed in advance but is left to the evolutionary process.

Accuracy of solutions obtained by *GP* depends on applied evolutionary parameters and also on the number of measurements and the accuracy of measuring. In general, more measurements supply more information to the evolution which improves the structure of models. At the same time, the greater number of measurements causes the time-consuming computer processing and the execution of experiments is very expensive and requires much time. Because of the high precision of the models developed by the *GP* approach, an excessive number of experiments and computations can be avoided, which leads to reduction of the costs of product development. Because the proposed *GP* method is general, it can be successfully used for modeling of different materials properties and phenomena where experimental data on the process are available.

References

Barnes, W.(1994). Statistical Analysis for Engineers and Scientists – a computer based approach, The University of Texas at Austin, McGraw – Hill, New York.

Brezocnik M., Kovacic, M., & Gusel, L. (2005). Comparison between genetic algorithm and genetic programming approach for modeling the stress distribution. Materials and Manufacturing Processes, 20, 497 – 508.

Chang, Y.S., Kwang, S.P., & Kim, B.Y. (2005). Nonlinear model for ECG R-R interval variation using genetic programming approach. Future Generation Computer Systems, 21, 1117 – 1123.

Dimitriu, R. C., Bhadeshia, H. K. D. H., Fillon, C., & Poloni,C. (2009). Strength of Ferritic Steels: Neural Networks and Genetic Programming, Materials and Manufacturing Processes, 24/1, 10 – 15.

Fakhrzad, M.B., & Khademi Zare, H. (2009).Combination of genetic algorithm with Lagrange multipliers for lot-size determination in multi-stage production scheduling problems. Expert Systems with Applications, 36 (6), 10180-10187.

Koza, J.R. (1992). Genetic programming. (1st ed.). The MIT Press, Massachusetts.

Koza, J.R. (1994) Genetic programming II, The MIT Press, Massachusetts.

Koza, J.R., Keane, M.A., Streeter, M. J., Mydlowec, W., Yu, J., & Lanza, G. (2003). Genetic Programming IV: Routine Human - Competitive Machine Intelligence, Kluwer Academic Publishers.

Koza, J.R., Bennett III, F. H., Andre, D., & Keane, M. A. (1999).Genetic Programming III: Darwinian Invention and Problem Solving, Morgan Kaufmann Publishing.

Lange, K. (2001). Handbook of metal forming, McGraw Hill, New York.

Mitchell, T.M. (1997). Machine learning, McGraw-Hill, New York.

Mohanty, S., Mahanty, B., & Mohapatra, P.K.J. (2003). Optimization of Hot Rolled Coil Widths Using a Genetic Algorithm. Materials and Manufacturing Processes, 18(3), 447-462.

Mondal, S., & Maiti, M. (2002). Multi-item fuzzy EOQ models using genetic algorithm, Computers & Industrial Engineering 44, 105–117.

Montgomery, D.C., Runger, G.C., & Hubele, N.F. (2001). Engineering statistics. (2nd ed.). New York. John Wiley & Sons.

Odugava, V., Tiwari, A., & Roy, R. (2005). Evolutionary computing in Manufacturing industry: an overview of recent applications, Applied Soft Computing, 5, 281 – 299.

Özel, T., & Karpat, I. (2005). Predictive modeling of surface roughness and tool wear in hard turning using regression and neural networks. International Journal of Machine Tools and Manufacture, 45 (4-5), 467 – 479.

Pierrevall, H., Caux, C., Paris, J.L., & Viguier, F. (2003). Evolutionary approaches to the design and organization of manufacturing system. Computers & Industrial Engineering, 44, 339 – 364.

Sette S., & Boullart, L. (2001). Genetic programming: principles and applications, Engineering Applications of Artificial Intelligence 14, 727 – 736.

Tanguy, B., Besson, J., Piques, R. & Pineau, A. (2005). Ductile to brittle transition of an A508 steel characterized by Charpy impact test – Part 1: experimental results.Engineering Fracture Mechanics, 72, 49 – 72.

Engineering the Future
edited by Laszlo Dudas
INTECH