Genetic programming for the prediction of tensile strength of cast iron

J. Duda*, A. Stawowy
AGH University of Science and Technology, Faculty of Management,
Gramatyka 10, 30-067 Krakow, Poland
*Corresponding author. E-mail address: jduda@zarz.agh.edu.pl

Received 21.06.2011; accepted in revised form 27.07.2011

Abstract

In this paper we propose genetic programming (GP) to predict tensile strength of ductile cast iron. The chemical composition and pouring temperature were used as explanatory input variables (parameters), while tensile strength as dependent output variable (response). On the basis of real data set collected in a one of the Polish foundries, two different models for output variable were developed by genetic programming. Statistical analysis of obtained results and two test cases were employed to compare the accuracy of the GP model with the neural network (NN) model and a linear multiple regression model. The comparison demonstrated that the GP outperforms regression techniques, while it is generally worse than NN. Nevertheless GP can be a powerful tool for predicting the mechanical properties of cast iron as it provides a mathematical model, which can be further analyzed.

Keywords: Application of information technology to the foundry industry, Metal parameters prediction, Genetic programming

1. Introduction

The mechanical properties of ductile iron are very important element of finished good quality. The purpose of our research was to create a prediction model of mechanical behavior of ductile iron based on chemical composition and pouring temperature. In this work we propose a genetic programming (GP) approach to predict tensile strength of ductile cast iron. GP is evolutionary computation method which imitates biological evolution of living organisms. Since proposed by Koza [1], GP has attracted notice in terms of its ability to model non-linear relationships.

A great part of the researches proposes the multiple regression (MR) method or artificial neural networks (ANN) to predict mechanical properties of cast iron [2, 3, 4, 5]. Because of the fixed size and shape of the MR model the latter is often not capable enough to capture complex relation between influencing parameters. On the other hand, ANN has been argued primarily as a “black box” approach due to the massive number of nodes and connections within its structure, giving in result completely unaccountable model for the user.

In the field of GP application to prediction of materials’ properties, the literature is not extensive. We could found only a few papers dealing with GP approach to model concrete and steel properties [6, 7, 8, 9, 10] that will be discussed in the following section.

2. Genetic programming approach

Genetic programming (GP) is a general name for the class of evolutionary methods, which can solve problems without requiring the user to know or specify the form or structure of the solution in advance [7]. In this sense they are similar to artificial neural networks, however, thanks to the representation of the solutions used in genetic programming they can be applied to much wider range of problems, including linear, tree-based or graph-based problems. Moreover, contrary to ANN they give a
human readable model, which can be further interpreted or modified in order to better describe particular phenomena.

2.1. GP principles

The problem in genetic programming can be seen as any computer program, usually the set of terminal and non-terminal symbols (e.g. mathematical operators and functions) as long as the fitness function for this program (i.e. how good is the program comparing to another one) can be evaluated in any time. The main algorithm of genetic programming is outlined in Fig. 2.

![Diagram of the main algorithm of genetic programming](image)

Fig. 1. Main algorithm of the genetic programming

The initial population consists of solutions (called in evolutionary methods individuals) which are random combination of functional components and terminals from the set of symbols allowed. Next generation is created by genetic recombination of individuals in a current population. The two basic genetic operators in evolutionary computation are crossover and mutation of genes in individuals. Fig. 2 illustrates two parent computer programs and two offspring created as the result of genes exchange. Subtrees in offspring are swapped below randomly selected crossover point.

![Diagram of crossover operator](image)

Fig. 2. Crossover operator creating two

Some genes in offspring can be altered with a given probability by the means of two kinds of mutation operators. Point mutation simply changes a randomly chosen function or terminal symbol into another from the set of possible symbols.

![Diagram of subtree mutation operator](image)

Fig. 3. Subtree mutation operator

In the subtree mutation a subtree below mutation point is changed into another randomly generated subtree what has been shown in Fig. 3.

After the process of genetic recombination new individuals are evaluated according to the fitness (quality) function. Evolution is continued until satisfactory value of the fitness function is found or a given number of generations is reached.

2.2. GP models

Genetic programming proposed by Koza [1] operated on representation of computer programs in the form of trees, particularly so called syntax (or grammar) trees. Such syntax tree comprises the sequence of functional components in such sense that tree root represents computer program output, nodes represent functions and their arguments are placed in subsequent nodes which are also functions or leaves which represents terminal symbols e.g. name of variables or constants. Such solution can be easily translated to LISP expression as proposed by Koza.

Fig. 4 illustrates a simple computer program, which in LISP language is denoted as (+ (* x x) (/ y 2)), but can be easily transformed to more straightforward mathematical notation as f(x,y) = x*x + y/2.

![Diagram of a computer program as a syntax tree](image)

Fig. 4. A computer program represented as the syntax tree

The sets of terminal symbols and functions determine how computer program can look like. In the variant of GP called gene expression programming method (GEP) proposed by Ferreira [12], which was used to generate models in this paper, a decision maker may set additional parameters like acceptability of constants, type and range of constants, number of components (subprograms) and a mathematical operator connecting them. Moreover, the preference of the particular functions can be also set.

2.3. GP for materials parameters modelling

A it was mentioned earlier, only few papers dealing with the application of genetic programming for modelling physical properties of metals, alloys and other materials have been published in last decade.

A series of papers was provided by Brezocnik and Kovacic. In their first paper [10] they propose to apply GP to model a forming efficiency of steel cylinders that depended on friction coefficient, workpiece geometry and geometric strain. The used a huge population of 500 individuals and counted up to 600 generations in four experiments with different sets of terminal symbols. They showed that the best model for describing forming efficiency was the one with exp (constant to the power of x) function, but they concluded that the computational time was very high and should be shortened.

In their next two papers they modeled roughness of aluminum cube as a function of cutting parameters such as spindle speed, feed
rate, depth of cut, and vibrations [12] and the adequacy of steel for the use as tool steel [7]. In both cases the best model was generated with the set of standard mathematical operators \( +, - , \times, / \) and floating-point constants. The average deviation of the data set and training sets from the observed values for the roughness surface model was about 7.5%, while the most complex model for the second problem had the reliability, measured as a percentage of proper answers (adequate or inadequate for tools) of 97.26%.

Another paper by Kovacic et al. [6] describes genetic programming applied for prediction of bending capability for rolled titanium zinc metal sheet. The problem is similar to the one presented in this paper as the depended value is estimated on the basis of chemical characteristic of the material (percentage of Cu, Ti, and Fe). They concluded that GP could be a powerful tool for predicting mechanical parameters of steel materials.

Puncrek but et al. [10] dealt also with the prediction on the basis of chemical characteristic of lead-free solder alloys, but this time to determine their melting point and solidification range. A weighted percentage of Sn, Ag, Cu, Bi, and In were used as independent variables and root mean square error (RMSE) was calculated as a fitness function. The set of allowed functions consisted of basic operators, logarithm function, power function and absolute value function \( +, -, \times, \log, \text{pow}, \text{abs} \). The results showed, that RMSE for GP was lower than the results achieved by the phase diagrams (CALPHAD) approach.

Although in the most recent paper Tsai and Lin[8] used GP to determine the parameters of concrete, instead of steel or alloy, they work is worth to be mentioned as they compared the GEP variant of genetic programming, which was also used in this paper with another method called weighted genetic programming (WPG). In some specific cases WPG provided a little bit better model than GEP in the sense of RMSE value, however, the difference was not very significant. They showed that both methods could be successfully applied to the prediction of concrete strength, but both achieved rather poor results in prediction of concrete slump.

3. Numerical experiments

We compared genetic programming approach and models obtained in our previous work [4] in the development of a prediction model using empirical data. Data for 34 melts were collected in a one of Polish foundry. The error back-propagation algorithm was used to train the multilayer feed-forward network, while Excel add-in Essential Regression was used to create multiple regression models.

Data set was divided into training data (28 items), validation data (4), and testing data (2). The chemical composition and pouring temperature were used as independent input variables (parameters), while tensile strength as dependent output variable. The values of variables during this investigation ranged from 406.2-714.9 MPa tensile strength (TS), 1483-1511 °C temperature (T), 92-93% iron (Fe), 3.56-3.94% carbon (C), 2.25-3.2% silicon (Si), 0.244-0.562% manganese (Mn), 0.028-0.05% phosphorus (P), 0.0-0.14% sulphur (S), 0.064-0.214% chromium (Cr), 0.003-0.019% molybdemenum (Mo), 0.029-0.064% nickel (Ni), 0.012-0.024% aluminium (Al), 0.034-0.093% copper (Cu), 0.0-0.11% tin (Sn), 0.006-0.029% titanium (Ti), 0.034-0.054% magnesium (Mg), and 0.0-0.12% zirconium (Zr).

3.1. GP models for tensile strength

Two different models for tensile strength were developed genetically on the basis of training data set. First time only four basic mathematical operators where used \( +, -, \times, / \) and model (1) was generated by genetic programming algorithm.

\[
TS = 545.95 \times (C \times Ti^2 - Ti + Si) - 535.23 \times \frac{Cr}{Sn + S} + \\
+ 6959.68 \times Mn \times Ni \times C \times Cr + \\
+ \frac{Al \times (Mn + Mo + Cr - Si)}{Mg \times Mn \times P}
\]

(1)

The value of R-squared coefficient for this model was \( R^2=0.74 \) and 12 explanatory variables were used.

As this result was not satisfactory we decided to introduce also some functions to the set of allowed non-terminal symbols for genetic programming. Following functions were added: logarithm, natural logarithm, exp function and finally sine and cosine trigonometric functions. After calculating 10,000 generations model (2) was obtained.

\[
TS = \frac{Mn \times (Cr + Mo + S)}{S \times \sin(Temp)} + \log_{10} \frac{Ni}{(Cr - P) \times (Mo + Mn)} + \\
\ln P \times \frac{\cos(Cr - Mn)}{\sin(Mn)} + 599
\]

(2)

This time R-squared coefficient was \( R^2=0.90 \) and only 8 explanatory variables were used in the model.

3.2. Comparison with regression and ANN

The models obtained by GP were compared with two regression models developed in work [4] – simple linear regression model (3):

\[
TS = 23344.68 - 540.6 C + 3290.43 Ni + \\
- 321.16 Si - 177.82 Fe - 2.38 Temp
\]

(3)

and multiplicative regression model (4):

\[
TS = 5115.53 + 533622.98P \times Cr \times Ni - 16.92C^2 \times Si + \\
- 0.000316 Temp \times Fe^2 + 1680550.10Al \times Cu \times Mg + \\
- 37639.46Cr^2 \times Cu
\]

(4)

Eventually all models were compared with the MLP artificial neural network described in paper [4] with two hidden layers consisted of 10 and 1 neuron respectively.

The results gathered in Table 1 presents RMSE, R-squared (not calculated for ANN) and the estimation for two test cases. Well trained artificial neural network gave the best results in terms of RMSE and the deviation from the observed values in two test cases.
However, it is known that ANN usually very well describes the model for the values that were used for training and validation, while for out-of-sample values it is not so good.

All in all genetic programming can successfully compete to the regression analysis methods providing better results both in terms of R-squared coefficient and the estimation of our test cases.

4. Conclusions

The presented results have shown that it is possible to quite successfully apply genetic programming model in estimating tensile strength of cast iron, based on chemical composition and pouring temperature. Correlation in the testing data set, which shows how well the GP model is able to estimate tensile strength, was high.

In further research we plan to investigate the influence of chemical composition and microstructure on mechanical properties of ductile cast iron. Although the interpretation of a genetic programming model is less intuitive then regression one, genetic programming seems to be a promising approach to develop prediction model for diagnostic and prognostic purposes.

References