Hot–Strength of Ferritic Creep–Resistant Steels Comparison of Neural Network and Genetic Programming

R. C.Dimitriu[†], H. K. D. H. Bhadeshia[†], C. Fillon[‡] and C. Poloni[‡]

[†]Materials Science and Metallurgy, University of Cambridge
Cambridge CB2 3QZ, U.K., www.msm.cam.ac.uk/phase–trans

[‡]Electrical Engineering and Computer Science, University of Trieste
Via Valerio, 10, 34127 Trieste, Italy

Materials and Manufacturing Processes 24 (2009) 10–15.

Abstract

We present here an analysis of a complex set of data on the strength of steels as a function of chemical composition, heat treatment and test temperature. The steels represent a special class designed to resist deformation at elevated temperatures (500–600°C) over time periods in excess of 30 years, whilst serving in hostile environments. The aim was to compare two methods, a neural network based on a Bayesian formulation, and genetic programming in which the data are formulated in an evolutionary procedure. It is found that in the present context, the neural network is able more readily to capture greater complexity in the data whereas a genetic program seems to require greater intervention to achieve an accurate representation.

1 Introduction

We have shown in recent work that interesting fundamental phenomena can be revealed by looking for patterns in large and multivariate datasets [1]. In some cases robust models which capture the physical principles underlying the behaviour of the data can lead to new developments, such as the δ–TRIP steel which is a major departure from conventional thinking in the context of alloys which rely on transformation plasticity [2]. It is true therefore, to assert that methods such as neural networks, genetic programming and optimisation techniques have made a mark on the development of materials.

In our continuing research on steels for the energy production industries [3], we recently analysed how the strength of creep–resistant ferritic steels depends on the test–temperature and a variety of other parameters such as the chemical composition of the steel and the heat treatment prior to

testing [1]. Such steels form the back–bone of steam turbines used in electricity generation and have to resist creep–deformation for many decades at temperatures up to 600°C.

The most interesting outcome was that the rate at which the strength (σ_Y) decreased with temperature (T) became much more pronounced beyond $T_C \simeq 800\,\mathrm{K}$, Fig. 1. Furthermore, the slope $\partial\sigma_Y/\partial T$ for $T > T_C$ was found to be similar for hot–strength and for creep rupture data, as illustrated in Fig. 1. This behaviour is believed to be associated with the onset of significant atomic mobility, such that dislocation climb becomes much more feasible at temperatures beyond T_C. From a technological point of view, the similarity in $\partial\sigma_Y/\partial T$ for hot–strength and creep–rupture provides a ready method for estimating expensive creep data using ordinary tensile tests [1].

Another useful outcome is that $|\partial\sigma_Y/\partial T|$ in the high temperature regime correlates directly with the tensile strength measured at ambient temperature, Fig. 2. An alloy which is strong at ambient temperature loses more of its strength as the temperature is raised above T_C. Fig. 2 could in principle be used to determine $|\partial\sigma_Y/\partial T|$ for an arbitrary steel, and hence to work out the temperature–dependent strength simply by measuring the ambient temperature strength:

$$\sigma_Y\{T\} = \sigma_Y\{T_A\} + \frac{\partial\sigma_Y}{\partial T} \times (T - T_A) \tag{1}$$

where T_A is the ambient temperature.

Note that for $T < T_C$, $|\partial\sigma_Y/\partial T|$ seems independent of alloy (Fig. 2) and corresponds to the temperature sensitivity of the solid–solution strength and that of pure iron [1].

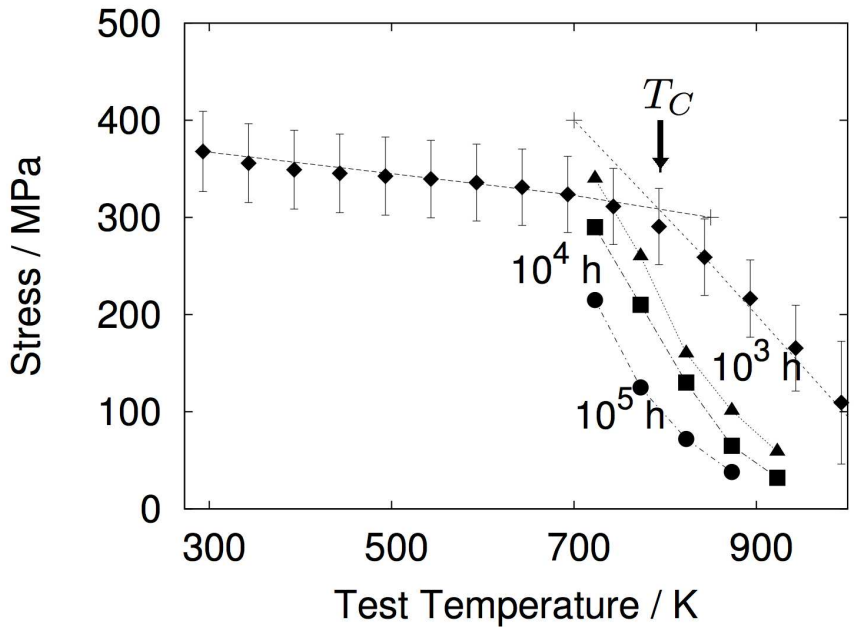

Figure 1: Comparison of temperature–sensitivity of creep–rupture and proof strength for $2\frac{1}{4}$Cr1Mo. The times associated with creep–rupture are also indicated [1]. The detailed parameters for the steel are 0.15C-0.18Si-0.63Mn-0.024Ni-2.23Cr 0.97Mo-0.02Cu-0.01Al-0.0083N wt%, 1193 K for 480 min, air cooled, tempered at 993 K for 360 min.

It is important to emphasise that the use of the neural network in identifying the regime where diffusion controls deformation is unique in that both plasticity and temperature are accounted for in the context of practical steels which typically contain precipitates, boundaries, myriad of solutes and a variety of defects. It is encouraging that it has been possible to reach clear conclusions in spite of the complexity of each of the alloy systems studied [1]. The work also lays open the possibility of

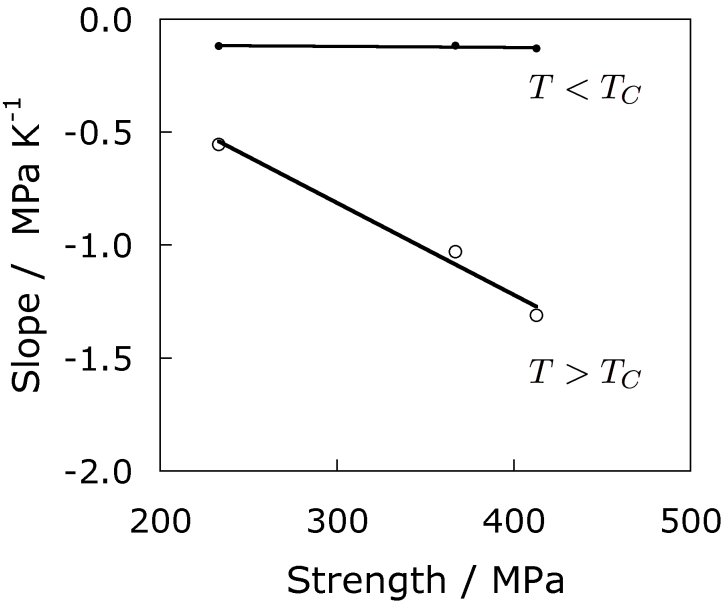

Figure 2: Plot of $\partial\sigma_Y/\partial T$ as a function of σ_Y measured at ambient temperature. Two cases are illustrated, for $T > T_C$ and vice versa. The alloys considered are the classical $2\frac{1}{4}$Cr1Mo, 5Cr and 9Cr1Mo. Data from [1].

defining a fundamental temperature limit to the applicability of ferritic steels in power plant. This would require work to relate the depression of the creep–rupture curve relative to the hot–strength curve on diagrams such as Fig. 1.

Given the utility of the phenomena described above, our aim in this paper was to see whether the (empirical) neural network method used in the analysis [1] leads to results which can be replicated by an alternative empirical method. We have therefore used the same data for the genetic programming. The neural method has been described elsewhere so it is only introduced here to the extent that it makes the paper readable; the main focus is on the other technique on the metallurgical significance of the results.

2 The Methods Used

2.1 Bayesian Neural Networks

A general method for treating complex data is the neural network in a Bayesian framework. This has been documented thoroughly [5–8] and applied extensively in the study and design of steels [9–17]. For this reason only specific points of relevance are introduced here.

The network is a non–linear regression method which, because of its flexibility, is able to capture enormous complexity in the data, whilst at the same time avoiding overfitting. There are a number of interesting outputs other than the coefficients which help recognise the significance of each input. First, there is the *noise* in the output, associated with the fact the input set is unlikely to be comprehensive – i.e., a different result is obtained from identical experiments. Secondly, there is the *uncertainty of modelling* because many mathematical functions may be able to adequately represent

known data but which behave differently when extrapolated. A knowledge of this uncertainty helps make the method less dangerous in extrapolation.

The mathematical structure of the network used is easily explained; in combination with the weights (w, θ) fully describes the relationships between the inputs x_I and the output y:

$$y = \sum_i w_{ij}^{(2)} h_i + \theta^{(2)} \qquad \text{with} \qquad h_i = \tanh\left\{ \sum_j w_{ij}^{(1)} x_i + \theta_i^{(1)} \right\} \tag{2}$$

As emphasised earlier, the details are described elsewhere but one major outcome of the method is that the Bayesian framework indicates a *modelling uncertainty* in addition to the perceived level of *noise* in the output. Noise refers to the case where a different outcome is obtained when an experiment is repeated, because some variable which influences the output has not been controlled. The modelling uncertainty on the other hand describes the range of predictions possible when using a variety of models, each of which is able to reasonably represent the same data. Another way of describing the modelling uncertainty is to say that instead of using a best–fit set of weights to derive equation 2, we associate a distribution of weights. The width of such a distribution in a localised region of the input space is then related to the modelling uncertainty when making a prediction in that region. Unlike the noise, the modelling uncertainty is not therefore a constant parameter and varies according to the values of the inputs used in calculating the output. The uncertainty is large when the data on which the models are based are absent, sparse or noisy – it therefore gives a firm indication of potential problems associated with the extrapolation of non–linear methods.

The Bayesian approach offers an indicator of the network-perceived significance of each input. The measure provided is a function of the values of the regularisation constants for the weights associated with an input σ_w. This measure is similar to a partial correlation coefficient in that it represents the amount of variation in the output that can be attributed by any particular input.

2.2 Genetic Programming

In genetic programming, relationships are reached through an evolutionary search process [18, 19], with a variety of possibilities for the actual algorithm involved, for example, an expression, a formula, a plan, a control strategy, a decision tree or a learning model depending on the problem [20]. The learning problem of interest here is the mapping of inputs to outputs given a dataset. Genetic programming in that context begins by creating a population of algorithms at random, followed by an assessment of the fitness of the program with respect to its ability to reach a solution. Programs which are particularly fit are selected for recombination to produce a new population by using genetic operators(selection, crossover and mutation) [21]. The process is repeated until an acceptable solution is reached, or for a predetermined number of runs. The evolutionary cycle is illustrated in Figure 3.

The programs can be viewed as a syntax tree, where a branch node is an element from a function set, which in turn may contain arithmetic functions, trigonometry and logic operators with at least one argument [18]. Some nodes which are at the ends of branches represent elements from a which contains independent variables and constants fixed according to the prior knowledge of the

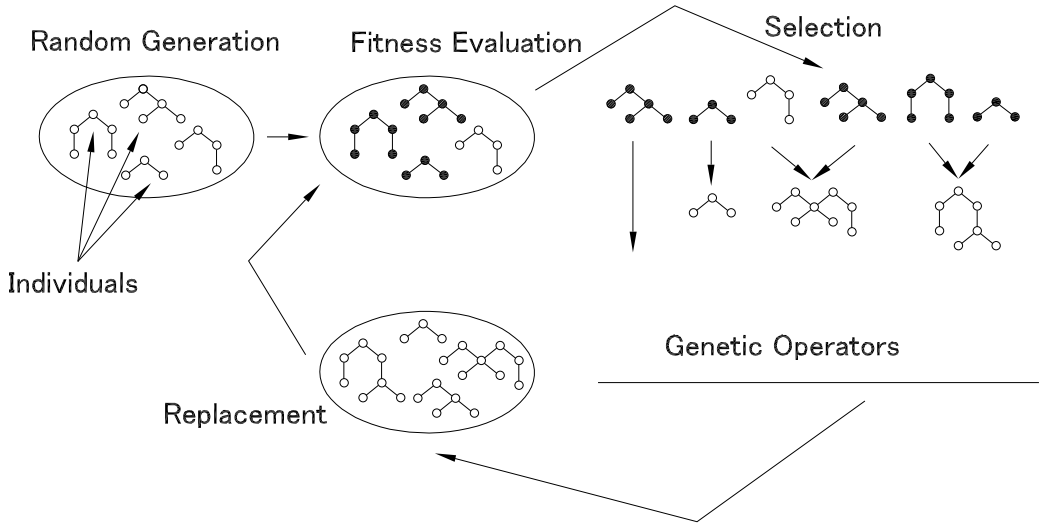

Figure 3: The artificial evolution cycle.

problem or may be randomly generated [22]. Figure 4 illustrates a syntax tree for the mathematical expression $x \times ytimes1.3 - x$.

The genetic operators alter the structure of the trees and modify the content of the nodes or terminations with the help of the operators in an effort to discover individuals with improved fitness. The simplest operator is duplication since it replicates an individual without modification. Crossover is an operation for the probabilistic exchange of genetic information between two randomly picked parents, facilitating a global search for the optimum in the process. Mutation in an operator inducing a small probabilistic change in the genetic makeup, resulting in a local search. It begins by selecting a point at random within the tree, and then the mutation operation replaces the function or the terminal set with the same type of element.

The fitness provides a measure of how an individual matches to a particular environment, a criterion to select individuals that participate in generating a new population [23]. Examples of fitness functions include the mean square and root mean–square errors [24, 25]. Selection procedures may take, for example, the following forms:

- roulette-wheel, where the probability that an individual is selected depends on its normalised fitness value [26];
- ranking, which is based on the ranking of the fitness values of individuals [27];
- tournament, where n (≥ 2) individuals are sampled from the population and the one with the greatest fitness is selected [28];

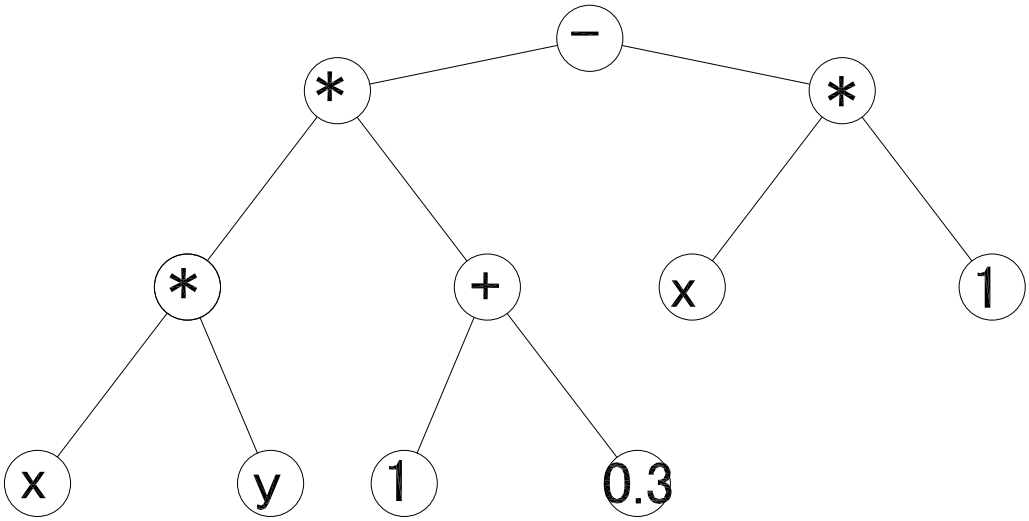

Figure 4: Tree representation of $x \times ytimes1.3 - x$.

- elitism involves the copying of the best few individuals in the next generation. It can increase performance by avoiding the loss of fit individuals [29].

3 The Database

The database used has its origins in published data [30] on the hot-strength (0.2% proof strength) of ferritic creep–resistant steels including the classical $2\frac{1}{4}$Cr1Mo, 5Cr, 9Cr1Mo and 12Cr1Mo type steels. The hot–strength is a function of the microstructure and solid solution strengthening, both of which depend on chemical composition and heat treatment; the relevant variables are listed in Table 3.

4 Comparison of Methods

In making these comparisons the calculations are based on data used for creating the models. This is because the neural network methods has already been shown to faithfully reproduce those data [1] – the genetic programming should therefore at least replicate those outcomes or reveal new structure within the generally verified trends.

The experimental data were in each case split randomly into equal parts to create the training and

Variable	Minimum	Maximum
Aluminium / wt%	0.001	0.04
Carbon / wt%	0.09	0.48
Copper / wt%	0.0001	0.25
Chromium / wt%	0.0001	12.38
Manganese / wt%	0.38	1.44
Molybdenum / wt%	0.01	1.05
Nickel / wt%	0.0001	0.6
Nitrogen / wt%	0.001	0.04
Silicon / wt%	0.18	0.86
Austenitising time / min	10	5400
Tempering time / min	30	660
Austenitising temperature / K	1143.15	1243.15
Tempering temperature / K	898.15	1023.15
Test temperature / K	293.15	973.15
Hot strength / MPa	69	660

Table 1: Variables used to develop the models.

test datasets, the latter being used to assess the ability of the model to predict unseen data; the test error is calculated as follows:

$$E = \sum_j (t_j - y_j)^2 \tag{3}$$

where y_j is a predicted value and t_j the target value; to calculate this error we normalised the output to be in the range ± 0.5. The neural network model used here is in a Bayesian framework and hence gives a modelling uncertainty associated with each prediction [5–8]. This is extremely helpful avoid the usual dangers of extrapolation. For the genetic programming, the modelling uncertainty was improvised by plotting the standard deviation of three models within a committee of models. The three component models were created simply by starting the training process three times; on each occasion a different model is obtained because of the randomness–based components of the genetic programming process.

The training process for the genetic programming begins by applying only the elementary $(+ - * /)$, and if necessary after assessment, more complex models can be created by including other functions such as (sin, cos, tan, tanh, log, exp...). In what follows, *all functions* refers to the case where the elementary and other functions have all been made available to the genetic program, whereas *trigonometric functions* is self–explanatory. The test errors for the different approaches are listed in Table 2; it should be noted that in all cases the committee of models has smaller test errors than any of the individual models. The genetic programs are seen from Table 2 to perform less well when compared with the neural network.

The best way of assessing the performance of non–linear models is to use them to make predictions. Figure 5 shows prediction for a $2\frac{1}{4}$Cr steel, and Figure 6 shows similar data for a prediction for a

Table 2: Test errors for committees of models.

Model	Test error
All functions genetic program, 3–member committee	0.611
Trigonometric functions genetic algorithm, 3–member committee	0.361
Neural network, 13–member committee	0.293

12Cr steel.

The predictions from the genetic program trained with arithmetic, elementary and trigonometric functions, failed to capture the complexity of the data. The different temperature dependency of stress on temperature on either side of T_C is not captured – instead, a rather stiff model is produced because the search for the final function was restricted to relatively simple formulae. With this limitation removed, the genetic program based on just the trigonometric functions is able to capture the stress–temperature behaviour correctly, probably because it includes hyperbolic tangents which are know to be very flexible functions. Indeed, the neural network uses the hyperbolic tangent transfer function as indicated in equation 2.

5 Summary

It may be concluded from this work that in the particular application considered, the two methods are similar in being able to create sufficiently complex non–linear functions to represent real experimental information in creep–rupture resistant steels. In many respects, both methods require trials to find the optimum overall function, but the genetic programming was more computer intensive and did not generalise as well as the highly–flexible neural network method.

From a metallurgical point of view, the clear, method–independent identification of two regimes in the temperature dependence of strength is a useful confirmation which can in principle be exploited in the design of steels for elevated temperature applications.

6 Acknowledgements

We would like to thank the Marie Curie Early Stage Research Training Programme of the European Commission for making this work possible. The work by C. Fillon and C. Poloni was supported by the Marie–Curie RTD network AI4IA, EU contract MEST–CT–2004–514510 (December 14th 2004) and by CAD–CAE laboratory of the Mechanical Department of the University of Trieste.

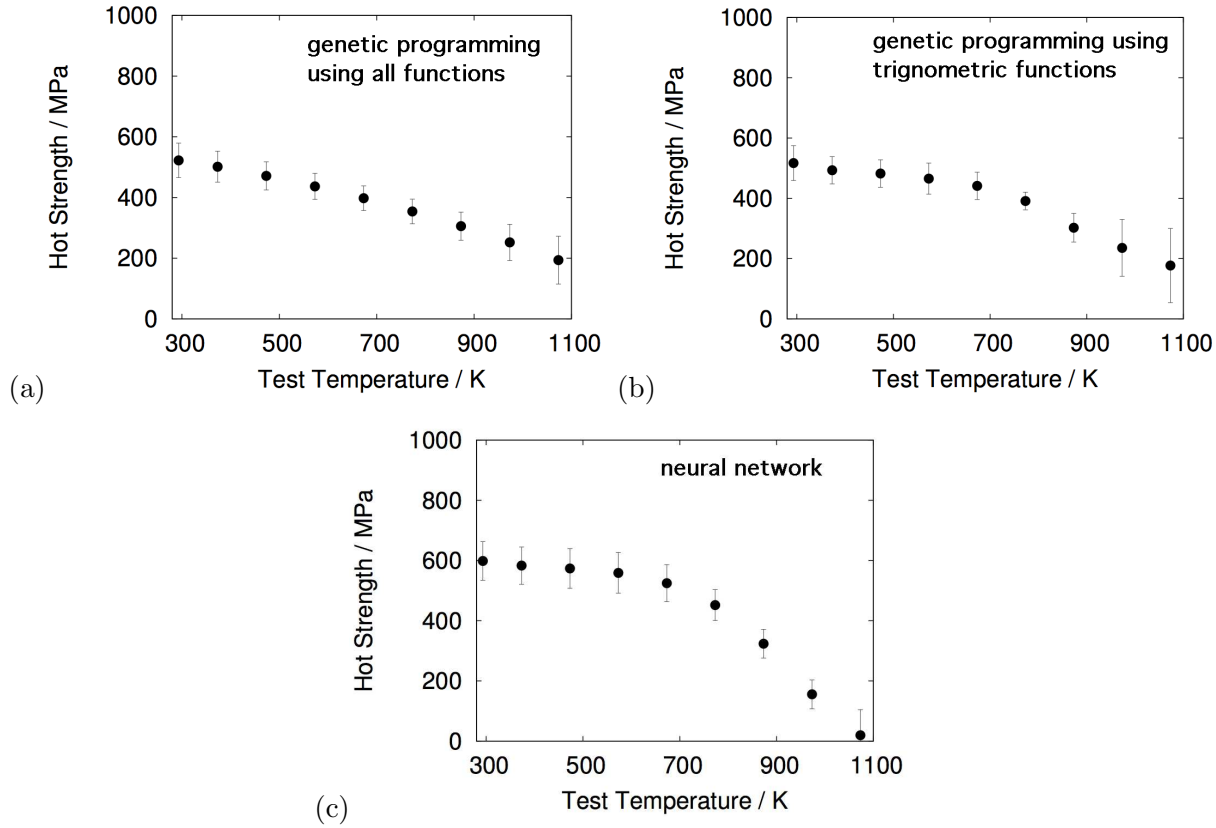

Figure 5: Calculations for steel containing 0.15C 0.25Si 0.61Mn 0.32Ni 2.35Cr 0.96Mo 0.03Cu 0.01Al 0.01N wt%, 1193 K for 210 min, air cooled, tempered at 988 K for 300 min. (a) Genetic programming using arithmetic, elementary and trigonometric functions. (b) Genetic programming using trigonometric functions. (c) Corresponding outcome using the neural network.

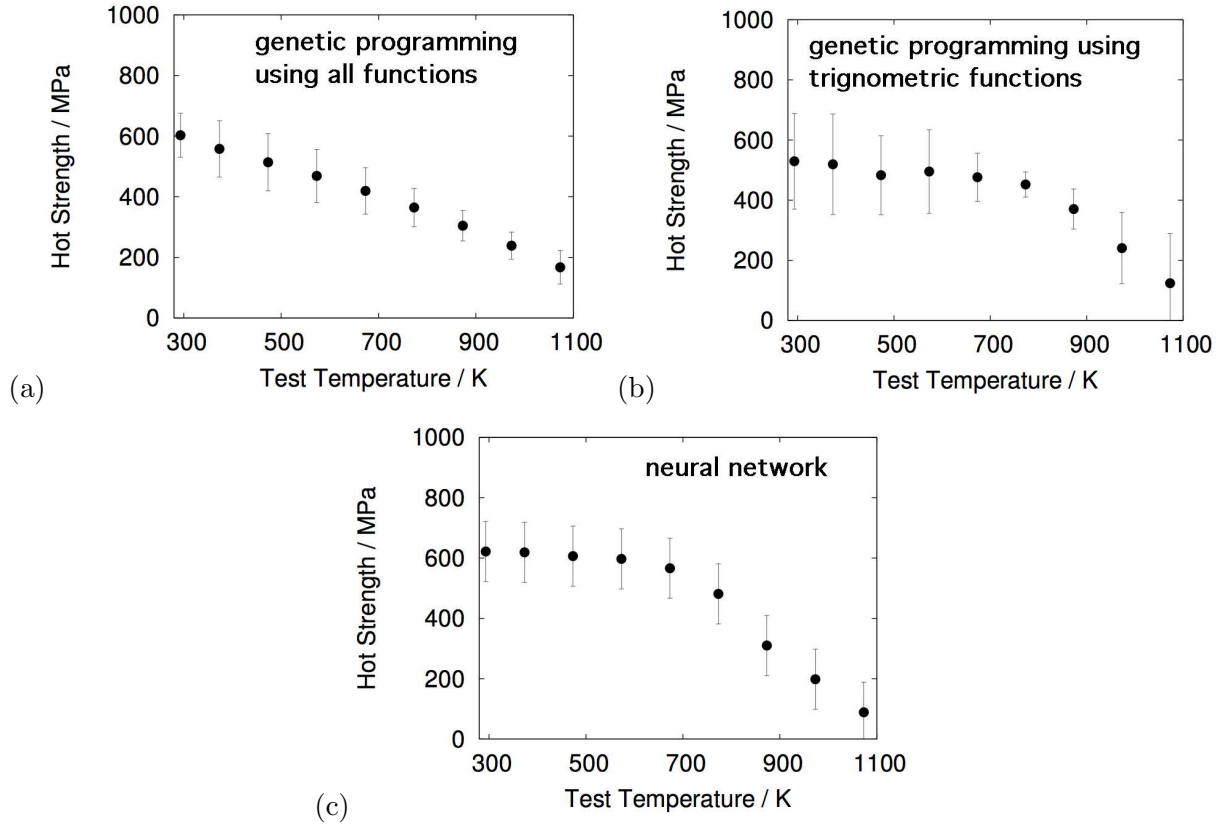

Figure 6: Calculations for steel containing steel containing 0.21C 0.44Si 0.62Mn 0.85Ni 11.64Cr 0.97Mo 0.06Cu 0.03Al 0.02N wt%, 1323 K for 25 min, air cooled, tempered at 913 K for 60 min. (a) Genetic programming using arithmetic, elementary and trigonometric functions. (b) Genetic programming using trigonometric functions. (c) Corresponding outcome using the neural network.

References

[1] R. Dimitriu and H. K. D. H. Bhadeshia. Hot–strength of creep–resistant ferritic steels and relationship to creep–rupture data. *Materials Science and Technology*, 23:1127–1131, 2007.

[2] S. Chatterjee, M. Murugananth, and H. K. D. H. Bhadeshia. δ–TRIP steel. *Materials Science and Technology*, 23:819–827, 2007.

[3] H. K. D. H. Bhadeshia. Design of ferritic creep–resistant steels. *ISIJ International*, 41:621–640, 2001.

[4] J. Fridberg, L.-E. Torndähl, and M. Hillert. Diffusion in iron. *Jernkontorets Annaler*, 153:263–276, 1969.

[5] D. J. C. MacKay. Practical bayesian framework of backpropagation networks. *Neural Computation*, 4:448–472, 1992.

[6] D. J. C. MacKay. Bayesian interpolation. *Neural Computation*, 4:415–447, 1992.

[7] H. K. D. H. Bhadeshia. Neural networks in materials science. *ISIJ International*, 39:966–979, 1999.

[8] D. J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[9] H. K. D. H. Bhadeshia, D. J. C. MacKay, and L.-E. Svensson. The impact toughness of C–Mn steel arc–welds – a Bayesian neural network analysis. *Materials Science and Technology*, 11:1046–1051, 1995.

[10] T. Goswami. Prediction of low cycle fatigue lives of low alloy steels. *ISIJ International*, 36:354–360, 1996.

[11] S. B. Singh and H. K. D. H. Bhadeshia. Estimation of bainite plate-thickness in low-alloy steels. *Materials Science and Engineering A*, A245:72–79, 1998.

[12] J. M. Vitek, Y. S. Iskander, and E. M. Oblow. Improved ferrite number prediction in stainless steel arc welds using artificial neural networks. *Welding Journal, Research Supplement*, 79:33s–50s, 2000.

[13] J. Tenner, D. A. Linken, P. F. Morris, and T. J. Bailey. Prediction of mechanical properties in steel heat treatment process using neural networks. *Ironmaking and Steelmaking*, 28:15–22, 2001.

[14] D. Dunne, H. Tsuei, and Z. Sterjovski. Artificial neural networks for modelling of the impact toughness of steel. *ISIJ International*, 44:1599–1607, 2004.

[15] Z. Guo and W. Sha. Modelling the correlation between processing parameters and properties of maraging steels using artificial neural network. *Computational Materials Science*, 29:12–28, 2004.

[16] M. Mukherjee, S. B. Singh, and O. N. Mohanty. Neural network analysis of strain induced transformation behaviour of retained austenite in TRIP-aided steels. *Materials Science and Engineering A*, 434A:237–245, 2006.

[17] S. Datta and M. K. Banerjee. Optimizing parameters of supervised learning techniques (ANN) for precise mapping of the input–output relationship in TMCP steels. *Scandanavian Journal of Metallurgy*, 33:310–315, 2004.

[18] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[19] J. R. Koza. *Genetic Programming 2: Automatic Discovery of Reusable Programs*. MIT Press, 1994.

[20] I. Kushchu. Genetic programming and evolutionary generalisation. *IEEE Transactions on Evolutionary Computation.*, pages 431–442, 2002.

[21] N. Chakraborti. Genetic algorithms in materials design and processing. *International Materials Reviews*, 49:246–260, 2004.

[22] M. O'Neill and C. Ryan. Gramatical evolution in evolutionary computation. *IEEE Transactions*, pages 349–358, 2001.

[23] S. Luke, G.C. Balan, and L. Panait. Population implosion in genetic programming. *Proceedings of the Genetic and Evolutionary Computation Conference GECCO'03*, pages 1729–1739, 2003.

[24] T. Sawa. Exact bias and mean square error of k-classe estimators. *Econometrica*, pages 138–146, 1970.

[25] C. C. Kilgus and W. C. Gore. Root mean square error in encoded digital telemetry. *IEE Transactions on Comunications*, pages 315–320, 1972.

[26] C. Laquerbe, P. Floquet, S. Domenech, and L. Pibouleau L. Development of separation sequences using a genetic algorithm. *Rairo-Recherche Operationnelle-Operations Research*, pages 375–397, 1997.

[27] B. Naudts and L. Kallel. A comparison of predictive measures of problem difficulty in evolutionary algorithms. *Evolutionary Computation, IEE Transactions*, pages 1–15, 2000.

[28] J. P. Yang and C. K. Soh. Structural optimization by genetic algorithms with tournament selection. *Journal of Computing in Civil Engineering*, pages 897–903, 1997.

[29] S. M. Bhandarkar, Y. Q. Zhang, and W. D. Potter. An edge-detection technique using genetic algorithm-based optimization. *Pattern Recognition*, pages 1159–1180, 1994.

[30] Y. Kojchi, I. Hiroshi, T. Hideo, Y. Masayoshi, K. Osamu, K. Kiyoshi, and K. Kazuhio. NRIM data sheets 1b, 3b, 8b, 11b, 12b, 17b, 18b, 19b, 20b, 21b. Technical report, National Research Institute for Metals, Tokyo, Japan, 1994.

[31] A. A. B. Sugden and H. K. D. H. Bhadeshia. A model for the strength of the as-deposited regions of steel weld metals. *Metallurgical Transactions A*, 19A:1597–1602, 1988.

[32] D. Kalish, S. A. Kulin, and M. Cohen. Bainitic structures and thermomechanical treatments applied to steel. *Journal of Metals*, 17:157–164, 1965.

[33] M. Peet, S. S. Babu, M. K. Miller, and H. K. D. H. Bhadeshia. Three–dimensional atom probe analysis of carbon distribution in low–temperature bainite. *Scripta Materialia*, 50:1277–1281, 2004.

[34] W. C. Leslie. Iron and its dilute substitutional solid solutions. *Metallurgical Transactions A*, 3:5–23, 1972.