UNIVERSITY OF CAMBRIDGE

Genetic Algorithm for Optimization of Mechanical Properties

Feb. - June 2003

A. Delorme
Department of Materials Science and Metallurgy, University of Cambridge
Pembroke Street, Cambridge CB2 3QZ, U.K.

Abstract

The neural network is a powerful method modelling which can be used to establish physical link between the microstructure and complex mechanical properties depending on a large number of parameters. Because of the large multidimensional space of solutions, if we are interested in finding the best set of parameters for a target value, an efficient searching tool is needed. The genetic algorithm is an optimization technique which simulates biological evolution: an initial set of possible solutions is improved through an iterative process of selection, recombination and mutation. In this study, we first explain in details what are neural networks and genetic algorithms. In a second part, the parameters of genetic algorithms are optimised for our problem. Then, the genetic algorithm is tested on the yield strength of austenitic stainless steels.

Keywords: Genetic Algorithm, Neural Network, Yield Strength, Austenitic Stainless Steel

Contents

Acknowledgements

I would first like to thank my supervisor Prof. H.K.D.H Bhadeshia to allow me to spend four months in his laboratory for this project and for his wise advices during this period.

I am also very grateful to Dr. T. Sourmail for his help, his guidance and his friendship.

Finally, thanks to all the members of this multi-origins laboratory, for their kindness and friendship.

Introduction

The relationship between the microstructure and the mechanical properties is well established qualitatively at a microscopic level. However, establishing a physically based link between a real microstructure and most complex properties such as toughness or creep resistance remains a challenge. To add to the difficulty, prediction of the microstructure itself as a function of composition and treatment is, in most cases, only semi-quantitative.

A more promising approach consist in incorporating some of this knowledge in powerful empirical modelling methods. Such an approach has been applied with great success to a variety of systems and properties through the neural network modelling: toughness of weld metals [1], creep rupture strength of ferritic steels [2] or yield strength and ultimate tensile strength of FeCrNi alloys [3]. One of the great strength of neural network modelling, among others which will be discussed later, is its ability to incorporate the influence of a large number of parameters, both quantitative and qualitative.

However, this strength is at the same time source of difficulties: if one is interested in optimisation, whereby a target value is given and the best set of input parameters sought, one is confronted with the traditional problem of sampling in multidimensional spaces: to visit all the combinations corresponding to a variation in, say, 10 steps in each of 30 variables, 10^{30} computations have to be performed. Among all solutions to the sampling problem, the most easily implemented, and the most easily interpreted, is likely to be genetic algorithm.

In this project, we are concerned with the possibility to use genetic algorithm to the existing empirical models cited earlier in optimisation processes. In a first part, the neural network method itself is presented. Follows a general introduction to genetic algorithms and the details of the implementation in the present study. Finally, results for yield strength of austenitic stainless steels are presented.

1 Neural network combined with a genetic algorithm

1.1 Neural networks

A neural network is a regression analysis technique in which a non-linear function is fitted to experimental data. The database contains several input parameters, like element concentrations, treatment temperature ... and the corresponding output, which can be the yield strength, the ultimate tensile strength ... Each input x_j is multiplied by a weight $w_j^{(1)}$. The sum of all these products with a constant $\theta^{(1)}$ forms the argument of a hyperbolic tangent as shown in equation 1. Each output h_i is itself multiplied by a weight $w_i^{(2)}$. The sum of these hyperbolic tangent with a second constant $\theta^{(2)}$ therefore gives the output y as a non-linear function of x_i as shown in equation 2.

$$h_i = \tanh(\sum_j w_{ij}^{(1)} x_j + \theta_i^{(1)}) \tag{1}$$

$$y = \sum_i w_i^{(2)} h_i + \theta^{(2)} \tag{2}$$

In this way, non-linear, and therefore complex, models are created. An example of a simple two hidden-unit neural network is given figure 1. The inputs and the output are connected through hidden units. These relates to the number of hyperbolic tangents used and hence determine the neural network complexity.

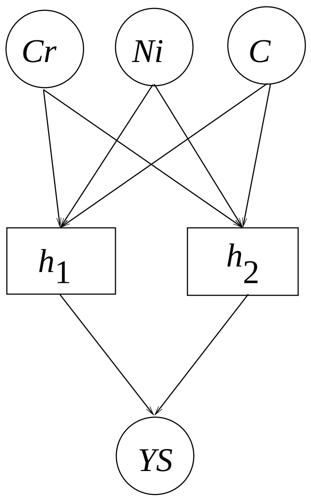

Figure 1: A simple two-hidden units neural network

1.2 Error estimation

The neural network takes into account two errors. Firstly, the overall test error, E_D is calculated by comparing the predicted values y_j with those measured t_j as shown in equation 3.

$$E_D \propto \sum_j (t_j - y_j)^2 \tag{3}$$

Secondly, the neural network evaluates the uncertainty in the fitting parameters [4]. In regions of the input space where data are sparse or noisy, many functions can be fitted without compromising the fit in regions where data are plentiful. $\sigma_y^{(j)}$ is the error related to this uncertainty which is evaluated taking into account the uneven distribution of input data. The error bars are larger where the data are sparse or noisy. The log predictive error (LPE)(eq.4) allows to evaluate models considering those two errors calculations.The LPE penalises less wild predictions that are accompanied by appropriately large error bars.

$$LPE = \sum_j \left[\frac{1}{2} \frac{(t^{(j)} - y^{(j)})^2}{\sigma_y^{(j)2}} + log(\sqrt{2\pi}\sigma_y^{(j)}) \right] \tag{4}$$

1.3 Overfitting

The difficulty with the use of powerful and flexible regression is the possibility of overfitting. To counteract this, the dataset is divided into two sets, a training dataset and a testing dataset. The first set is used to train the network, and the training error decreases as the model complexity increases. An overcomplex model can be defined, which fits all the data in the training dataset but does not take into account of the noise (fig. 2). So, the test dataset is then used and the test error increases as the model complexity increases, e.g when overfitting occurs.

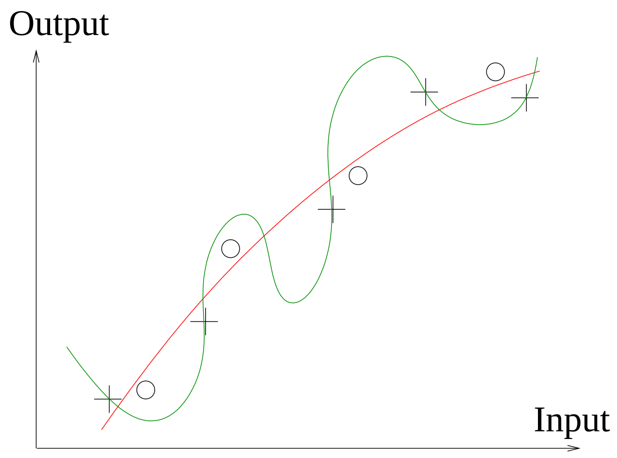

Figure 2: A overcomplex model (green curve) has overfitted the training set (+), resulting on a larger error on the testing data (o) than for a model which does not fit the noise (red curve)

1.4 Committee Model

Several models of varying complexity are produced and an error is associated with each one. It is shown that a committee of models used together could reduce the overall test error and allow more reliable predictions than the best single model with the lowest error [5].

1.5 Genetic algorithm

The genetic algorithm is a model of machine learning based on the mechanism of natural selection and natural genetics [6], [7]. This is done by a random creation of a population of individuals, represented by chromosomes. This individuals are evaluated and undergo a process of evolution which start with a natural selection, inspired by Darwin's theory of evolution: the best individuals of a population are selected. Then a biological process occurs: some recombinations, as crossover and mutation, are made in order to create a new generation of individuals with the hope that this new one is better. The genetic algorithm is stopped when a target value is reached.
Genetic algorithms are viewed as optimization tools and allow to solve problems for which an extremum solution is searched.

1.5.1 Process

We are looking for an input set (x_1,x_2,...,x_j) which will give a desired output y. The genetic algorithms are based on biological theory and so, we can assimilate this set as a chromosome. Each input x_i is the equivalent of a gene. In a first time, a population is randomly generated. This population contains n chromosomes (x_1,x_2,...,x_j). These inputs are entered in a model, created by a neural network, and we obtain the output y and the associated error for each chromosome. The chromosomes are then ranked according to their fitness. The best chromosomes are selected and subjected to operations, as well in the biological system, as crossover or mutation. In this way, we obtain the second generation. This new sets of chromosomes are then still evaluated and undergo selection, crossover and mutation mechanisms, as previously. We obtain our third generation and this process takes place for many generations until the fitness value is reached (fig. 3).

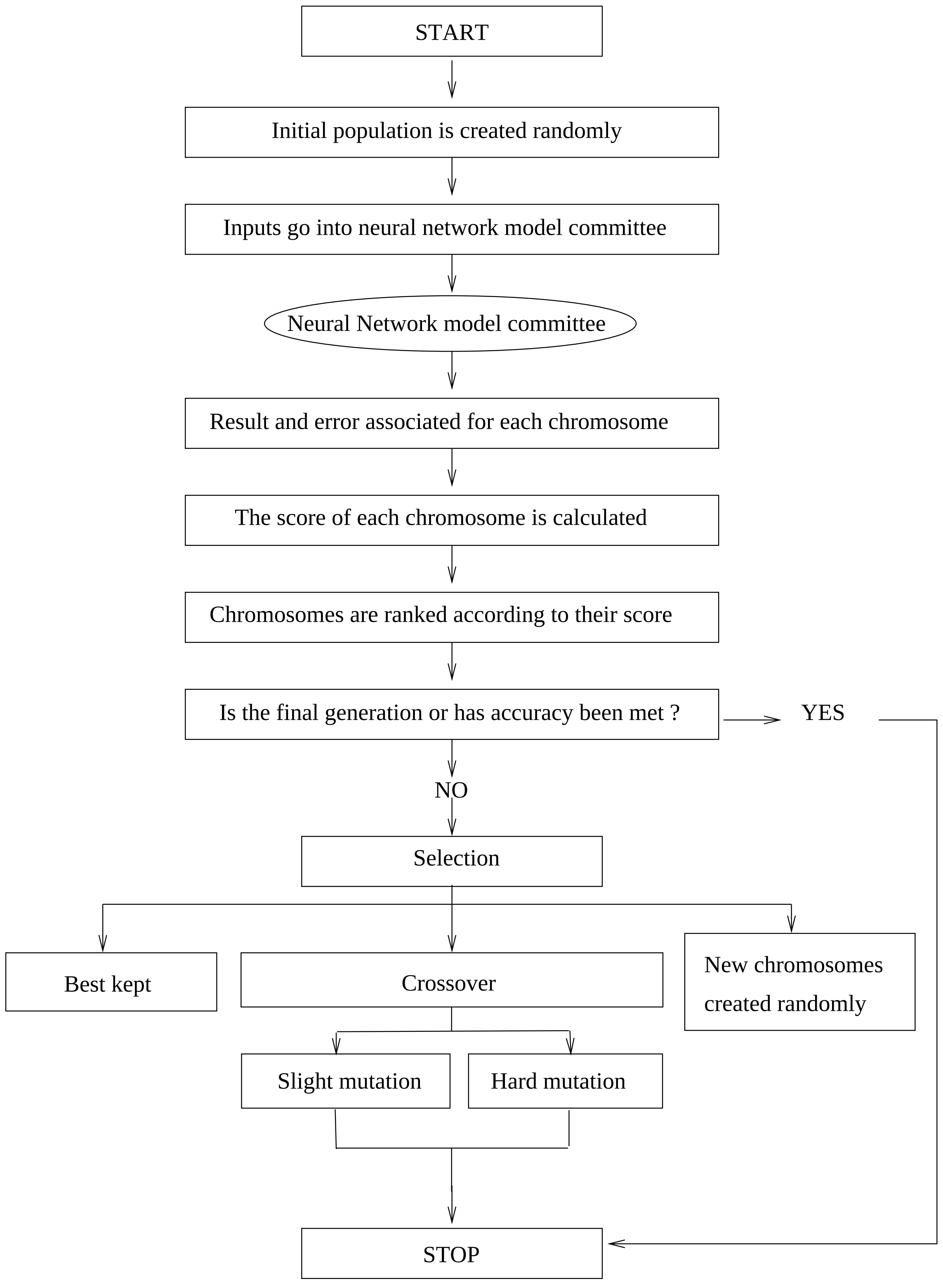
START
Initial population is created randomly
Inputs go into neural network model committee
Neural Network model committee
Result and error associated for each chromosome
The score of each chromosome is calculated
Chromosomes are ranked according to their score
Is the final generation or has accuracy been met ?
YES
NO
Selection
Best kept
Crossover
New chromosomes created randomly
Slight mutation
Hard mutation
STOP

Figure 3: The process of the genetic algorithm

1.5.2 Operators

Fitness function The evaluation of a chromosome has to take into account the result and the error of the committee of models given by the neural network. Each model i created by the neural network gives a result $y^{(i)}$ and the associated error $\sigma_y^{(i)}$. The average prediction of a committee of L models is:

$$p = \frac{1}{L} \sum_i y^{(i)} \tag{5}$$

The standard deviation error (σ) of p is as follows:

$$\sigma^2 = \frac{1}{L} \sum_i \sigma_y^{(i)^2} + \frac{1}{L} \sum_i (t - p)^2 \tag{6}$$

where t is the desired output. The score of a chromosome could be σ. However, in order to have a better score for the better chromosome, we invert the error and the fitness f is defined as follows:

$$f = \frac{1}{\sigma} \tag{7}$$

Selection After being ranked according to their fitness, the chromosomes undergo a process of selection. This mechanism allows the allocation of a greater survival to better individuals: this is the survival-of-the-fittest mechanism we impose on our solution. There is several ways to select the chromosomes which will be recombined. In our genetic algorithm, the roulette wheel selection will be used.

The principle of this method is that the better the chromosomes are, the more chances they have of being selected. Considering n chromosomes, each of them previously evaluated with a fitness value f_i, we calculate the sum S of all the scores, as follows:

$$S = \sum_{i=0}^{n} f_i \tag{8}$$

The probability P_i that a chromosome i will be selected is given simply by the chromosome's fitness value divided by the sum S, as follows:

$$P_i = \frac{f_i}{S} \tag{9}$$

Thus, the best chromosomes, e.g. with better fitness values, will be selected more frequently. A simple example with 5 chromosomes illustrates the roulette wheel selection principle in table 1, where the fitness value f_i and the corresponding probability P_i of being selected are calculated.

Crossover The crossover is a process of taking genes from two parents, mixing them and producing an offspring. The simplest way is to choose randomly a crossover point. The child is produced by copying the first segment from the parent 1 and after the crossover point, the genes of the second parent are copied.

However, there are many others and complex ways to do crossover. The best is to have a lot of crossover points so as to have better chance to take the best from the both parents.

In our genetic algorithm, we decide to use the uniform-crossover, which use $(n-1)$ crossover points for a chromosome containing n genes. For that, a mask is created. Each bit of the mask is a random number between 1 or 2 and thus, determine from which parent each gene will be copied. (figure 4).

Chromosome i	Fitness value f_i	Probability P_i (%)
Chromosome 1	12.3	52.1
Chromosome 2	6.4	27.1
Chromosome 3	3.2	13.5
Chromosome 4	1.5	6.4
Chromosome 5	0.2	0.9
Sum S	**23.6**	**100**

Table 1: The fitness value f_i of each chromosome i and the corresponding probability P_i of selection

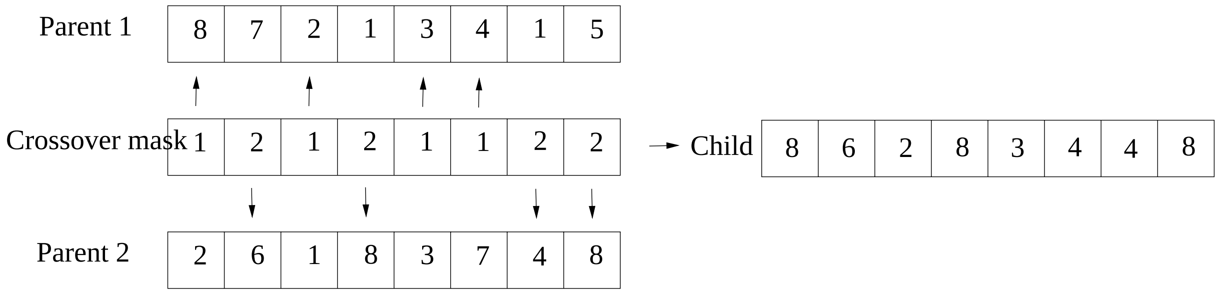

Figure 4: Principle of the Uniform-crossover

Mutation The mutation occurs after the crossover operation. It creates variants of few offsprings previously recombined and so introduce new genetic material. The probability of mutation must be low to stay in the neighbourhood of the current solution. Otherwise, the GA will not perform any better than a random search. In our study, we choose to mutate one offspring, selected randomly, in adding Δx_i to one of its genes. The mutation is slight, between 0 and 0.2% so as to stay close to the current solution.

Population size The population size represent the number of chromosomes in a population. Of course, the more chromosomes there are, the more chances exist to have good solutions. However, the time for finding a solution has to be considered and if there are too many chromosomes, the genetic algorithm slows down. Sizes 20-30 are reported as best. Moreover, it has been shown that the best population size depends on size of chromosomes [8]. For these reasons, we choose a population size of 20 in the following study.

2 Optimisation of genetic algorithm parameters

The performance of the genetic algorithm depends on several parameters and today, there are no general studies describing the most appropriate set for any particular problem. Indeed, it is difficult to estimate the parameter settings because of the complexity of the research. Studies give general recommendations and the best way to improve a specific genetic algorithm is to experiment several values for each parameter. As described in the first part of this report, the population size is fixed at 20 chromosomes; one gene is mutated with a maximum mutation rate of 0.2%. In this second part, the number of populations, the number of generations and the crossover rate are optimized, resulting in a faster convergence of the genetic algorithm.

2.1 Number of generations

The number of generations determines how many times each population undergoes the genetic iteration. A larger number enhances the score, with a result close to the target and a minimised error. However, beyond a given point, the score saturates or changes little.

Five simulations are considered, running respectively for 3, 30, 300, 3000 and 5000 generations. The score of the best population for each simulation is shown figure 5. It is clear that the number of generations is helpful in reaching a better score. Yet, the curve saturates after a large number of generations,as illustrated in figure 6. This figure represents the result and the associated error with which the score is calculated (eq 6, 7). The target value is indicated by a straight dashed line. The result after 3000 generations is good enough and adding 2000 further generations doesn't lead to a substantial change in accuracy. Bearing this in mind, the following study uses 3000 generations.

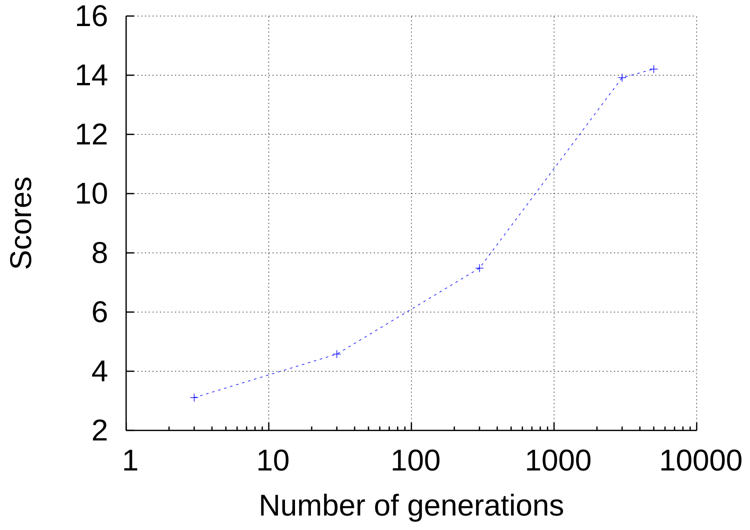

Figure 5: The best scores obtained after 3, 30, 300, 3000 and 5000 generations

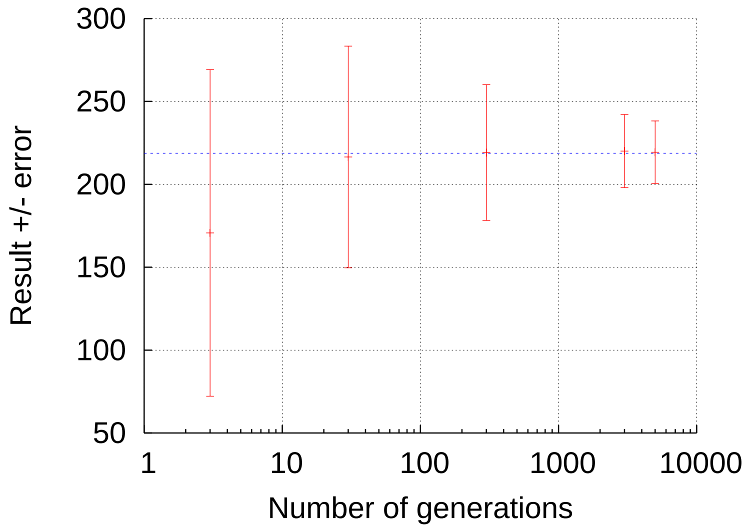

Figure 6: The result and the error corresponding to the scores shown figure 5 after 3, 30, 300, 3000 and 5000 generations

2.2 Number of populations

The genetic algorithm could work with only one population. However, using several populations each subjected to the genetic algorithm can improve the efficiency. If one population has poor individuals, the solution can be found by another one as they typically explore different searching-space. The populations are allowed to mix after several generations, e.g. the crossover will occur between one chromosome of a given population and one chromosome of another population. From this perspective, a greater number of populations gives a greater chance of finding good solutions. However, it is also necessary to think in terms of the computational speed.

The influence of the number of populations has been studied by using the algorithm on a number of populations varying from 1 to 10. When n populations are used in a genetic algorithm, the result is provided by the best element of all populations. Each program is itself run 4 times. The average of the resulting 4 best scores is shown in figure 7. The extrema of the error bars are the upper and the lower limit of the best results.

In a general way, the scores enhance with the number of populations. The best score is obtained with the 8 populations simulation. The same result have been found in a precedent study [3]. The 10 populations simulation gives good results and the third best average is achieved by the 3 populations program. In terms of speed, the time to complete a generation increases with the number of populations. From this perspective, it would be more fruitful to run a program with 3 populations, as long as the number of generations is increased. Therefore, in the present work, the genetic algorithm involves 3 populations, allowing mixing after 200 generations.

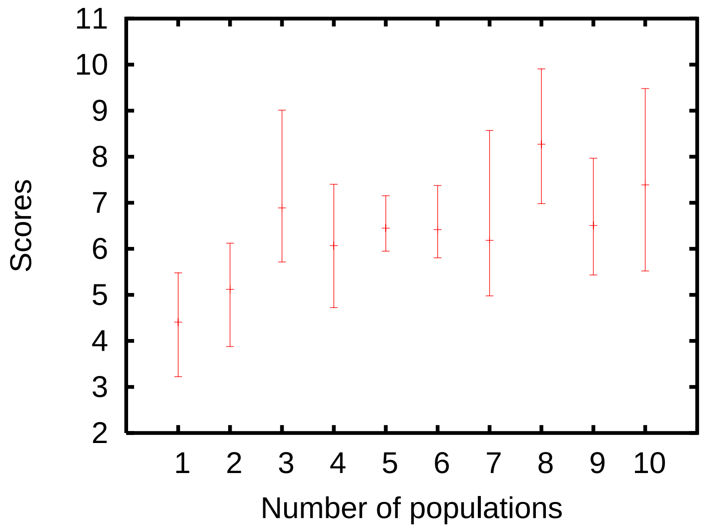

Figure 7: The scores of the best population for simulation programs involving from 1 to 10 populations

2.3 Crossover rate

As explained in the first part, the method chosen is uniform crossover of which the rate determines how many chromosomes will be recombined through a generation. A rate of 100% means that all the chromosomes will be recombined to form the new population. In this case, it is likely that the best chromosomes from the old population will be lost. To avoid this, it is necessary to use an elitism method, which keeps unchanged the best chromosome(s) between generations. Alternatively, if the rate is too low, there is little change between generations, preventing improvement. Moreover, new genetic material has to be introduced in a population at each generation, to avoid staying in a local minimum. Typically, the crossover rate is high, about 80-95% [8].

In our study, 7 simulations were conducted with crossover rates of 10, 25, 50, 60, 70, 80 and 90%. The remaining parameter settings were identical; notably 3 populations evolved during each simulation. As before, we consider the population among the 3 which has the best score. These programs were run 4 times each and therefore, the average of the best score was made. The results are shown in figure 8.

In general, better scores were obtained at the crossover rate at 90%, which was selected for subsequent calculations. In the present study, this corresponds to 18 chromosomes combining with each other among 20. The best chromosome is copied without change to the new population. The chromosome with the worst score is killed and a new one is generated randomly, introduced to the new population and subjected to the genetic algorithm alongside the others.

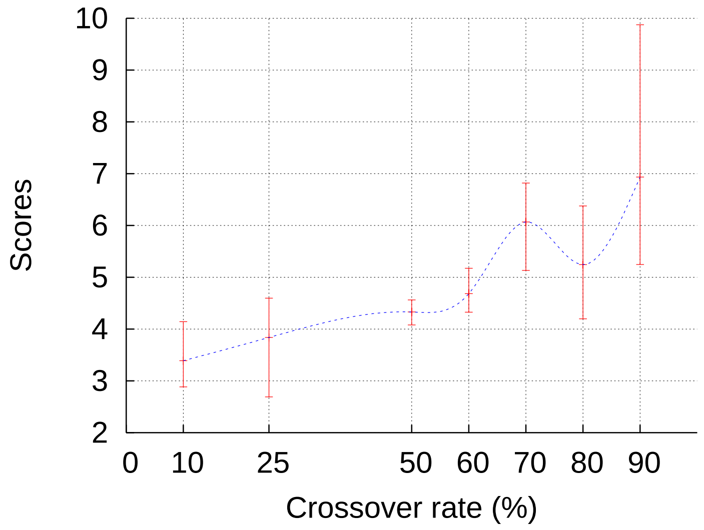

Figure 8: The scores of the best population in function of the crossover rate

3 Results

According to the first and second part of this report, a genetic algorithm has been developed in language C considering the following parameters:

Number of populations = 3

Number of generations = 3000

Population size = 20 chromosomes

When a new generation is created, the following steps are followed: after ranking the 20 chromosomes according to their scores, the first chromosome is copied without change. The chromosomes 2 to 19 are recombined with each others. One gene of one of these chromosomes is mutated between $\pm 0.2\%$. The chromosome 20, with the worst score, is killed and a new random chromosome is generated and incorporated in the new population.

This program can calculate the best set $(x_1,x_2,...,x_j)$ of input parameters for a desired output y, which is in this study, the yield strength of austenitic stainless steels, for which a neural network model was developed in a previous study [3]. In this part, a brief introduction to austenitic stainless steels and the yield strength is given. Then, the neural network developed in [3] is explained in details. At last, the results of our study are presented.

3.1 Modelling the yield strength of austenitic stainless steels

3.1.1 Austenitic stainless steels

Austenitic stainless steels contain between 18-30 wt% Cr, 8-20 wt% Ni and 0.03-0.1 wt% C. The high level of chromium contributes to the excellent corrosion resistance: beyond a concentration of 10.5 wt%, a thin, protective, chromium oxide layer is formed, which passivates the steel. However, chromium is a ferrite stabiliser. To balance this, nickel is added, to ensure a fully austenitic structure after a quenching to room temperature. However nickel is an expensive alloying element and can lead to allergies[9]. So attempts have been made to reduce its concentration using other austenite stabilisers such as manganese or nitrogen.

A slow cooling or a reheating between 550 and 800°C can allow chromium carbides formation, which nucleates at the grain boundaries. It can lead in these regions to a depletion of chromium and thus, favour intergranular corrosion. One way of reducing or eliminating the formation of $Cr_{23}C_6$ is to introduce titanium and niobium with which carbon forms much more stable carbides than $Cr_{23}C_6$. In this way, titanium and niobium will combine first with the carbon, preventing the $Cr_{23}C_6$ nucleation.

Many other alloying elements can be added to enhance mechanical properties.

3.1.2 Yield strength

To characterise the mechanical properties of a material, the strength, ductility and toughness are usually considered at first. In this study, we are interested in the yield strength which can easily be assessed by tensile tests. A uniform uniaxial load is applied on a standard sample until it fractures. The engineering curve relates this test as a strain-stress graph, fig. 9. At first, the sample undergoes an elastical linear deformation. If the load is released, the sample returns to its initial state. Beyond the elastic limit, the material becomes plastic and is subject to work hardening. The tensile curve reaches a maximum engineering stress beyond which the sample develops a neck. This is soon followed by fracture.

The yield strength is sometimes difficult to define so the proof stress, corresponding to 0.2% plastic strain, is often substituted.

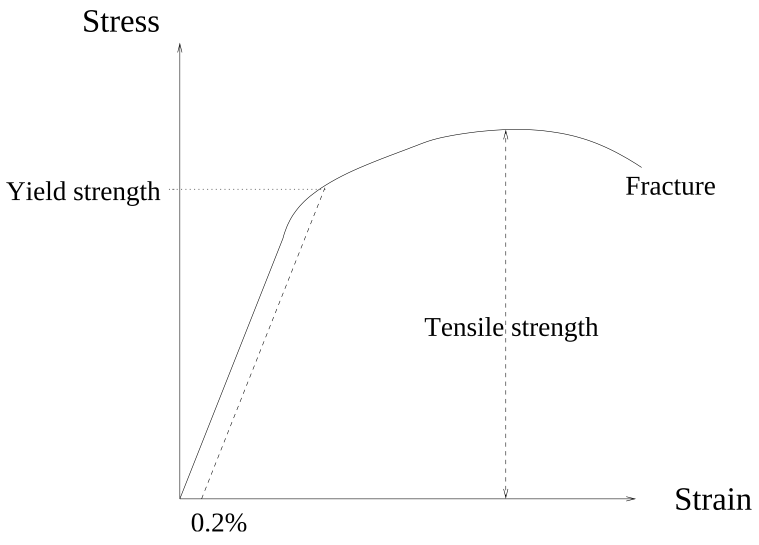

Figure 9: Engineering stress-strain curve

3.1.3 Models

The neural network developed by [3] predict the yield strength of austenitic stainless steels as a function of several variables, as chemical composition, stoichiometric stabilisation ratio of Ti and Ni additions, treatment temperature and test temperature. A large database has been used to develop models. The range of each input and output from the database can be found in table 2.

To evaluate the carbides formation, the stoichiometric stabilisation ratio for titanium and niobium additions is defined as follows:

$$\text{Ratio} = \frac{(C_{\text{Ti}}/4) + (C_{\text{Nb}}/8)}{C_{\text{C}} + C_{\text{N}}} \tag{10}$$

All the variables used within the neural network model were normalised between -0.5 and $+0.5$ in order to be able to compare each others easier. In the genetic algorithm, the same agreement is made. The normalisation of a value x is done through the following equation:

$$x_n = \left(\frac{x - x_{min}}{x_{max} - x_{min}} \right) - 0.5 \tag{11}$$

where x_n is the normalised value between -0.5 and $+0.5$, x_{min} and x_{max} are respectively the minimum and the maximum value of the dataset. Nevertheless, during the genetic algorithm process, each value can vary between -1 and $+1$, while the corresponding unnormalised value remains positive. In this way, the genetic algorithm can explore data outside the dataset and find new compositions.

Variable	Minimum	Maximum
Chromium (wt%)	15.90	21.06
Nickel (wt%)	8.40	34.45
Molybdenum (wt%)	0.00	2.91
Manganese (wt%)	0.61	1.82
Silicon (wt%)	0.00	1.15
Niobium (wt%)	0.00	0.95
Titanium (wt%)	0.00	0.56
Vanadium (wt%)	0.00	0.06
Copper (wt%)	0.00	0.35
Nitrogen (wt%)	0.00	0.08
Carbon (wt%)	0.01	0.12
Boron (wt%)	0.00	0.02
Phosphorus (wt%)	0.00	0.04
Sulphur (wt%)	0.00	0.05
Cobalt (wt%)	0.00	0.54
Aluminium (wt%)	0.00	0.52
Ratio	0.00	3.0625
Heat Treatment Temperature (K)	1279	1473
Test Temperature (K)	293	1273
Yield Strength (MPa)	35	341.27

Table 2: Analysis of the dataset

As well, the desired value of yield strength is normalised following the equation 11. Thus, a normalised value of -0.5 correspond to the minimum value of yield strength saved in the database, e.g. 35 MPa. Likewise, a normalised value of +0.5 correspond to the maximum value in the database, which is 341 MPa.

3.2 Results

3.2.1 Target yield strength of 218 MPa

The first simulation is made to check the behaviour of the genetic algorithm. The target value of yield strength is set to 0.1, which correspond to an unnormalised value of 218 MPa. The dataset provides such values of yield strength and the aim of this simulation is to check the results of the genetic algorithm. The only parameter fixed is the test temperature, which is chosen as 300 K. The 18 others parameters are allowed to vary, between -1 and $+1$ during the genetic algorithm process. After 3000 generations, the best results obtained is shown table 3.

According to table 3, the genetic algorithm has managed to reach the target after 3000 generations. Moreover, the associated error obtained is very reasonable.

To check if the given compositions correspond to an austenitic steel, we calculate the Cr equivalent and the Ni equivalent, following the equations 12 and 13. This allows to inspect the influence of alloying elements, considering if they are gamma or alpha stabilisers. With these values of Cr equivalent and Ni

Variable	Result	Input of the database
Cr (wt%)	18.88	17.85
Ni (wt%)	11.31	12.00
Mo (wt%)	0.00	0.04
Mn (wt%)	1.97	1.71
Si (wt%)	1.24	0.60
Nb (wt%)	0.72	0.74
Ti (wt%)	0.03	0.02
V (wt%)	0.01	0.03
Cu (wt%)	0.13	0.05
N (wt%)	0.03	0.03
C (wt%)	0.12	0.07
B (wt%)	0.002	0.000
P (wt%)	0.02	0.02
S (wt%)	0.04	0.01
Co (wt%)	0.27	0.29
Al (wt%)	0.06	0.02
ratio	0.65	0.975
HTT (K)	1436	-
Result (MPa)	**219**	**219**
Error (MPa)	**21**	-

Table 3: The first column gives the compositions of the best result for a desired yield strength of 218 MPa at 300 K. The second column gives the values of a steel coming from the database

equivalent, the structure of the steel can be found, thanks to the Shaeffler-Schneider diagram, figure 10.

$$\mathrm{Cr}_{eq} = x_{\mathrm{Cr}} + 2x_{\mathrm{Si}} + 1.5x_{\mathrm{Mo}} + 5x_{\mathrm{V}} + 5.5x_{\mathrm{Al}} + 1.75x_{\mathrm{Nb}} + 1.5x_{\mathrm{Ti}} + 0.75x_{\mathrm{W}} \tag{12}$$

$$\mathrm{Ni}_{eq} = x_{\mathrm{Ni}} + x_{\mathrm{Co}} + 0.5x_{\mathrm{Mn}} + 0.3x_{\mathrm{Cu}} + 25x_{\mathrm{N}} + 30x_{\mathrm{C}} \tag{13}$$

where x_i is the weight percentage of element i present. For the composition defined in table 3, we obtain the values of Cr_{eq} and Ni_{eq} shown table 4.

Cr_{eq}	23.05
Ni_{eq}	16.95
Result (MPa)	**219**
Error (MPa)	**21**

Table 4: Values of Cr_{eq} and Ni_{eq} for the best set of parameters

According to the Schaeffler-Schneider diagram, the steel belongs to the austenitic domain.

The chemical compositions are as well characteristic of an austenitic stainless steel. A input of the database is compared to our result in table 3. The yield strength obtained for this steel is 219 MPa at

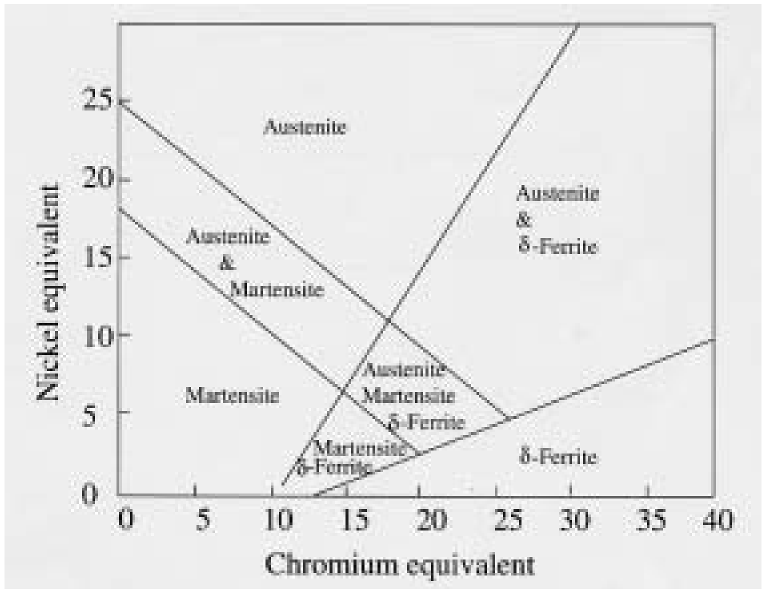

Figure 10: Schaeffler-Schneider diagram giving the basic effects of alloying elements on the structure of Cr-Ni steels [10]

373 K. The contents are characteristic of a 347 type austenitic stainless steel and similar to our result except for the silicon content which is twice higher.

Although this first simulation gives very good results, it would be interesting to try to reach a value of yield strength which is not in the dataset.

3.2.2 High values of yield strength

Target value of yield strength of 494 MPa

For a Fe-0.01Nb-0.04Ti-0.17Cu-0.025P-0.013S steel at 298 K A previous study [3] has develop a genetic algorithm for the yield strength of austenitic stainless steel and the program has been tested, among other values, for a normalised target value of yield strength of 1, which correspond to 494 MPa. The program considers a Fe-0.01Nb-0.04Ti-0.17Cu-0.025P-0.013S steel at 298 K and tries to optimise the compositions in Cr, Ni, Mo, Mn, Si, N, C, B and the heat treatment temperature.

Our genetic algorithm has been develop from this previous study. Large modifications of the program have been made in order to improve it. Comparing our results to these from [3] would be a good mean to check improvements. For that, we decide to run the genetic algorithm with the same fixed parameters, e.g. the target value of yield strength is 494 MPa, the test temperature is 298 K and the compositions fixed are Fe-0.01Nb-0.04Ti-0.17Cu-0.025P-0.013S. The others compositions elements (Cr, Ni, Mo, Mn, Si, N, C, B) are allowed to vary, as well as the heat treatment temperature.

The results obtained in this study and these obtained in [3] are compared in table 5.

Our yield strength and the associated error are better than those resulting from the genetic algorithm developed by [3], which obtain quite high error. Indeed, the program undergoes large modifications

Input	Our study	Previous study [3]
Cr (wt%)	19.00	18.69
Ni (wt%)	19.69	9.80
Mo (wt%)	3.28	1.74
Mn (wt%)	0.44	2.42
Si (wt%)	0.74	0.58
N (wt%)	0.02	0.04
C (wt%)	0.08	0.10
B (wt%)	0.007	0.005
ratio	0.01	0.24
Result (MPa)	**484**	**396**
Error (MPa)	**50**	**171**

Table 5: Best results after 2000 generations for Fe-0.01Nb-0.04Ti-0.17Cu-0.025P-0.013S steel at 298 K

as notably the addition of a selection process and the change of the crossover rate, which have been successful.

The chromium equivalent and the nickel equivalent have both been calculated. The values are shown in table 6.

	Our study	Previous study [3]
Cr_{eq}	25.48	22.54
Ni_{eq}	22.86	15.06
Result (MPa)	**484**	**396**
Error (MPa)	**50**	**171**

Table 6: Values of Cr_{eq} and Ni_{eq} for the best set of parameters of our study and of [3]

Regarding the Schaeffler diagram figure 10, the steel obtained after the study of [3] does not seem to be fully austenitic but contain austenite and ferrite. Our result is fully austenitic.

In this simulations, several parameters were fixed and even the target yield strength was outside the dataset, the value was close to the reality and likely. We decide to investigate further from the dataset.

No parameters fixed The present calculation is run with the same test temperature fixed at 298 K, for the same number of generations. However, in this case, all the compositions and the heat treatment temperature are allowed to vary. The best set of parameters obtained is shown table 7.

The resulting yield strength corresponds to the target value. Comparison of the results obtained in the both cases shows similar error bars and obtention of the required yield strength. Given that the number of variables, and therefore the searching space, in the second problem is considerably larger than in the first one, we can speculate that convergence was reached long before the 2000 iterations in the first case.

Although we try to reach a yield strength of 484 MPa, this target is reasonable and we decide to investigate further.

Variable	Result
Cr (wt%)	18.93
Ni (wt%)	16.13
Mo (wt%)	2.06
Mn (wt%)	0.96
Si (wt%)	1.17
Nb (wt%)	0.25
Ti (wt%)	0.45
V (wt%)	0.01
Cu (wt%)	0.05
N (wt%)	0.01
C (wt%)	0.19
B (wt%)	0.016
P (wt%)	0.00
S (wt%)	0.00
Co (wt%)	0.04
Al (wt%)	0.20
ratio	0.70
HTT (K)	1419
Result (MPa)	**486**
Error (MPa)	**58**

Table 7: The best score for a desired yield strength of 484 MPa at 298 K

Target value of yield strength of 800 MPa The test temperature is fixed at 298 K. In the database, the highest yield strength in this order of temperature is about 340 MPa. Trying to reach such a value is quite a challenge. Thus, the number of generations is increased until 4000 generations. The best result obtained is described table 8.

The result yield strength is 774 MPa, which is nearby 800 MPa. The error bar remains large because we are too far from the upper limit of the dataset. However, the compositions could correspond to an austenitic stainless steel. An input of the database has been compared with our results table 8. Obviously, the yield strength for the steel coming from the dataset is quite low but the test temperature is 1273 K. Moreover, the chemical compositions are almost similar to our result, except for the molybdenum content which is higher. Increasing molybdenum enhances the resistance to pitting and corrosion by making the passive film at the surface of the steel stronger and contributes to a higher strength.

Variable	Result	Dataset
Cr (wt%)	23.90	20.24
Ni (wt%)	33.01	31.83
Mo (wt%)	3.74	0.09
Mn (wt%)	1.19	1.16
Si (wt%)	0.82	0.50
Nb (wt%)	0.32	0.00
Ti (wt%)	0.62	0.46
V (wt%)	0.03	0.00
Cu (wt%)	0.33	0.00
N (wt%)	0.07	0.00
C (wt%)	0.13	0.10
B (wt%)	0.007	0.000
P (wt%)	0.05	0.02
S (wt%)	0.01	0.01
Co (wt%)	0.18	0.37
Al (wt%)	0.44	0.5
ratio	0.98	1.15
HTT (K)	1503	1413
Result (MPa)	**774**	**63**
Error (MPa)	**99**	-

Table 8: The first column gives the best score for a desired yield strength of 800 MPa at 298 K. The second column gives chemical compositions of a steel at 1213 K coming from the database

Conclusion

In this study, we have developed a genetic algorithm in language C. The program can be obtained from the web: http://www.msm.ac.uk/map/map.html. This genetic algorithm has been tested for the yield strength of austenitic stainless steels but can be easily adapted for any mechanical properties as long as a neural network model exists.

Predictions made can reach a target yield strength set inside or close to the database with reasonable error bars and element compositions close to those of a typical austenitic stainless steel. When the target value is far from the existing values of the dataset, it is more difficult to reach the target without increasing the error bars. However, this large error bars can be due to the neural network model itself and the best way to decrease it would be to develop larger database and build a new neural network.

Finally, the genetic algorithm constitutes a good tool to search in a complex and large search space. An essential step to valid the genetic algorithm would be to produce the steels founded by the program and test mechanical properties. Moreover, the idea would be to allow the search among more parameters as the price of elements or others mechanical values.

References

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

[2] A. G. Sheard H. K. D. H. Bhadeshia D. Cole, C. Martin-Moran and D.J.C. MacKay. Modelling Creep Rupture Strength of Ferritic Steel Welds. *Science and Technology of Welding and Joining*, 5:81–90, 2000.

[3] I. Shah. *Tensile Properties of Austenitic Stainless Steel.* PhD thesis, University of Cambridge, 2002.

[4] D. J. C. MacKay. A Practical Bayesian Framework for Backpropagation Networks. *Neural Computation*, 4:448–472, 1992.

[5] T. Sourmail. *Simultaneous Precipitation Reactions in Creep-Resistant Austenitic Stainless Steels.* PhD thesis, University of Cambridge, 2002.

[6] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, 1989.

[7] D. Whitley. A Genetic Algorithm Tutorial. In *Statistics and Computing*, volume 4, pages 65–85, 1994.

[8] M. Obitko and P. Slavik. Visualization of Genetic Algorithms in a Learning Environment. pages 101–106, Bratislava: Comenius University, 1999. Spring Conference on Computer Graphics, SCCG'99.

[9] R. Magdowski P. J. Uggowitzer and M. O. Speide. Nickel Free High Nitrogen Austenitic Steels. *ISIJ Int.*, 36(7):901–908, 1996.

[10] Schneider and Climax Molybdenum Co. *Foundry Trade J.*, 108:562, 1960.