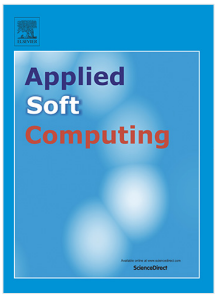
Journal Pre-proof



Application of a genetic algorithm based model selection algorithm for identification of carbide-based hot metal desulfurization

Tero Vuolio, Ville-Valtteri Visuri, Aki Sorsa, Seppo Ollila,
Timo Fabritius

PII: S1568-4946(20)30270-2
DOI: https://doi.org/10.1016/j.asoc.2020.106330
Reference: ASOC 106330

To appear in: *Applied Soft Computing Journal*

Received date : 22 October 2019
Revised date : 28 February 2020
Accepted date : 14 April 2020

Please cite this article as: T. Vuolio, V.-V. Visuri, A. Sorsa et al., Application of a genetic algorithm based model selection algorithm for identification of carbide-based hot metal desulfurization, *Applied Soft Computing Journal* (2020), doi: https://doi.org/10.1016/j.asoc.2020.106330.

Application of a genetic algorithm based model selection algorithm for identification of carbide-based hot metal desulfurization

Tero Vuolio[1)], Ville-Valtteri Visuri[1)], Aki Sorsa[2)], Seppo Ollila[3)] and Timo Fabritius[1)]*

1) Process Metallurgy Research Unit, University of Oulu, P.O. Box 4300, FI–90014, University of Oulu, Finland.

2) Control Engineering, University of Oulu, P.O. Box 4300, FI–90014, University of Oulu, Finland.

3) SSAB Europe Oy, Rautaruukintie 155, P.O. Box 93, FI–92101, Raahe, Finland.

*) Corresponding author. E-mail: tero.vuolio@oulu.fi

Keywords: hot metal desulfurization, mathematical modeling, calcium carbide, variable selection, neural networks, genetic algorithm.

ABSTRACT

Sulfur is considered as one of the main impurities in hot metal. Hot metal desulfurization is often carried out with pneumatic injection of a fine-grade desulfurization reagent using a submerged lance. The aim of this study was to develop a data-driven model for the process. The model selection algorithm carries out a simultaneous variable selection and neural network topology optimization with a combination of binary and integer coded Genetic Algorithm. The objective function applied in the search is repeated Leave-Multiple-Out cross-validation. The model considered is a multi-layer feedforward neural network. In the variable and topology selection phase, the computational load is reduced by making use of Extreme Learning Machine (ELM) architecture. The final model is trained using the Bayesian regularization. The results show that a well-generalizing data-driven model with good prediction performance can be repeatedly selected based on noisy industrial data with the help of a Genetic Algorithm, provided that the model is validated comprehensively with internal and external data sets.

1. Introduction

In steel manufacturing, sulfur is considered as one of the main impurities dissolved in hot metal. The removal of sulfur from the hot metal is important due to the fact that existence of it detoriates the weldability and decreases the corrosion resistance of the final steel product. Hot metal desulfurization is a process, in which the sulfur is removed from the metal phase and transferred to the top slag. Hot metal desulfurization is often carried out using injection of a fine-grade desulfurization reagent with an immersed lance. The carrier-gas used in the injection is typically nitrogen or argon. There are numerous alternatives for reagents, which can be used for hot metal desulfurization, of which this work is concerned with calcium carbide. The main desulfurization reaction with calcium carbide can be given as:[1]

$$< CaC_2 > + [S] \rightarrow (CaS) + 2[C], \qquad [1]$$

where < > denotes solid species, [ ] denotes species dissolved in the metal phase and ( ) denotes species dissolved in the slag phase. During the injection, a complex chemical system called the slag phase is formed on top of the metal. The overall rate of hot metal desulfurization with powder injection is considered to be a summation of the following reaction mechanisms: [1-2]

i. transitory contact reaction that occurs between the injected particles and the hot metal,
ii. permanent contact reaction, which is the reaction between the top slag and the hot metal,
iii. the reaction between the particles trapped inside the bubbles formed in the injection and hot metal.

The contribution of each of the aforementioned reaction mechanisms is difficult to formulate based on the physico-chemical properties of the system only, and major simplifications are still required to reduce the development effort and computational load. Furthermore, several operating factors induce process variance that is not captured by purely mechanistic models, thus reducing their accuracy. As pointed out in the authors' earlier review, the existing mechanistic models for hot metal desulfurization still need at least the following clarifications to be applicable for process control purposes: [3]

- fraction of non-contacted particles,
- residence time of individual particles and the contact time available for mass transfer,

- heterogeneity of metal and slag phases in terms of composition and temperature,
- effect of gas-forming additives on the effective contact area, and
- adhesive behavior of the reagent.

In light of the reasoning above, the data-driven approaches could provide a feasible alternative to the more traditional modeling approaches. As was suggested in our previous studies [4-5], a parametrized 1st order reaction model is applicable in the prediction of kinetics in the case of lime-based desulfurization, if the process operates far from thermodynamic equilibrium [S] > 0.008 wt-%. This attribute results in the rate of desulfurization being controlled by reaction kinetics and not by the thermodynamic driving force, which is why a common simplification is to neglect the thermodynamic driving force. However, if the process operates at low sulfur contents, the role of the thermodynamic driving force increases, which makes the given assumption ill-suited. As the exact determination of the $[S]_{eq}$ is more or less trivial with a thermodynamic approach, the calculation of the rate constant can be considered more or less suggestive, which usually leads to only moderate accuracy. [4] In light of these, a neural network based data-driven modeling approach could provide a feasible alternative for modeling the process.

The earlier data-driven approaches to modeling of carbide-based hot metal desulfurization have showed improved accuracy compared to deterministic approaches. The model proposed by Vinoo *et al.* [6] made use of manual data classification based on the input temperature and variable selection to predict the consumption of the carbide-based reagent with a multivariable linear regression model. However, as the hot metal desulfurization is a non-linear process, a non-linear modeling scheme should be considered. This can be deduced from the relatively large error of prediction for the external data set of 15 points (MAE = 0.0015 wt-%). [6] In the studies of Deo *et al.* [7] and Datta *et al.* [8] prediction accuracies of MAE = 0.0027-0.016 wt-% and R^2 = 0.16-0.60 were obtained with different neural network architectures by making use of a training set including 51 [8] and 146 training data-points [7]. The proposed input variables were the initial content of sulfur in the hot metal ($[S]_0$), the initial mass of the hot metal (m_{Fe}), the flowrate of the carbide ($\dot{m}_r$) and the treatment time (t). Despite the complex model structure, the training and validation errors are relatively large, which suggests that the proposed network architectures (20-20-1) and prediction accuracies are not likely to be entirely consistent. This type of a behavior in neural network training can be usually associated with noisy and sparse data or a non-reliable training algorithm. The study of Rastogi *et al.* [9] made use of Genetic Algorithms for parameter identification to evaluate the significance of different reaction mechanisms based on a parameterized reaction model. In their

study, the genetic algorithm provided a realistic set of parameters to study the contribution of the suggested reaction mechanisms. [9] All in all, the previous authors create a baseline for prediction of the behavior of such a complex process based on solely on non-linear and noisy industrial data.

Although there are some studies in the literature in which a multilayer neural network has been used to predict the state of metallurgical processes [10-17], there are no remarks in the literature of automated model selection based approach in modeling of hot metal desulfurization. In the study of Petterson *et al.* [10], a feed-forward neural network was used to predict the blast furnace burden distribution. The authors performed a simultaneous training and topology optimization of the network for a pre-selected variable subset by making use of a genetic algorithm. [10] Saxén and Petterson[11] studied the problem of input variable selection for neural network models to predict dynamic silicon content in the blast furnace process. They showed that an intuitive pruning algorithm can be successfully applied for silicon content prediction based on sparse and noisy blast furnace data. The importance of a variable candidate was associated with the importance of its connective weight with respect to prediction error. [11] Later, Saxén and Pettersson [13] coupled their model selection algorithm with statistical reasoning, which was based on the Akaike Information Criterion (AIC). [13] In the studies of Pettersson et al. [12] and Mahanta and Chakraborti [17], an evolutionary neural network combined with a bi-objective genetic algorithm was applied for selection of pareto-optimal model structure for comprehensive modeling of the blast furnace. [12, 17]

In the study of Wang *et al.* [14], the variable selection was carried out manually by using a multiple linear regression as a model basis. The resulting network was then trained using a Genetic Algorithm and a moderate prediction accuracy was achieved. [14] In the study by Wang *et al.* [15], the Random Forest (RF) algorithm was used for prediction of the silicon content in a blast furnace. In their study, the measure of variable importance was related to classification accuracy of the model candidate when applied to an external data set. The algorithm selected the relevant input variables amongst 28 candidate variables. [15] Wang *et al.* [16] presented a variable selection algorithm for a Support Vector Machine (SVM) model designed for the end-point prediction of the BOF blow. The importance of a predictor variable candidate was measured with mutual information and selected if a threshold value of the quantity was exceeded. [16]

The primary objective in a data-driven prediction model identification is to find a model that describes the changes in the selected output variable with sufficient accuracy and has a short computational time, which makes such models well-suited for real-life applications. Consequently,

the aim of this study was to develop an algorithm that performs the model identification steps for a given set of data in a sophisticated manner by making use of an automatic variable selection algorithm coupled with a multi-layer feedforward neural network. To illustrate the capability of the approach, the algorithm was applied to an industrial data set containing a high number of candidate prediction variables, and tested for predicting the sulfur content at the end of the hot metal desulfurization treatment. The effect of the hyperparameters of the algorithm on the computational results is evaluated by making use of repetitive testing and statistical reasoning. The relevance of the selected variables and their contribution to the hot metal desulfurization is also discussed.

2. An algorithm for a neural network model identification

Data-driven model identification can be roughly divided into the following steps: 1) data gathering, 2) variable/feature extraction, 3) variable/feature selection, 4) model parameter identification and 5) model validation. In the selection of a neural network model, step 4 is often called the training of the network. As the term feature is often used in the context of signal or image processing, the columns of the data matrix X are here referred to as variables. For this aim, a Genetic Algorithm prediction model identification algorithm is presented. The flowchart of the algorithm is presented in **Figure 1**. In the algorithm, the variable selection and network topology selection are carried out simultaneously by making use of a binary and integer-coded Genetic Algorithm. The Genetic Algorithm applies Leave-Multiple-Out (LMO) cross-validation as the fitness function. In LMO cross-validation, the data is repeatedly split into training and internal validation sets for N times, and the suitability of the variable set and network topology are assessed based on the average of the repeated cross-validation error. After the variable and topology selection, the model is trained and regularized, after which the final modeling error is assessed based on an external validation data set. Prior to variable selection, the training data was standardized such that $\mu_X = 0$ and $\sigma_X = 1$. It should be noted that the standardization was carried out after the extraction of the external validation set, and the standardization of the external set was carried out by making use of the mean and standard deviation calculated based on the training data only.

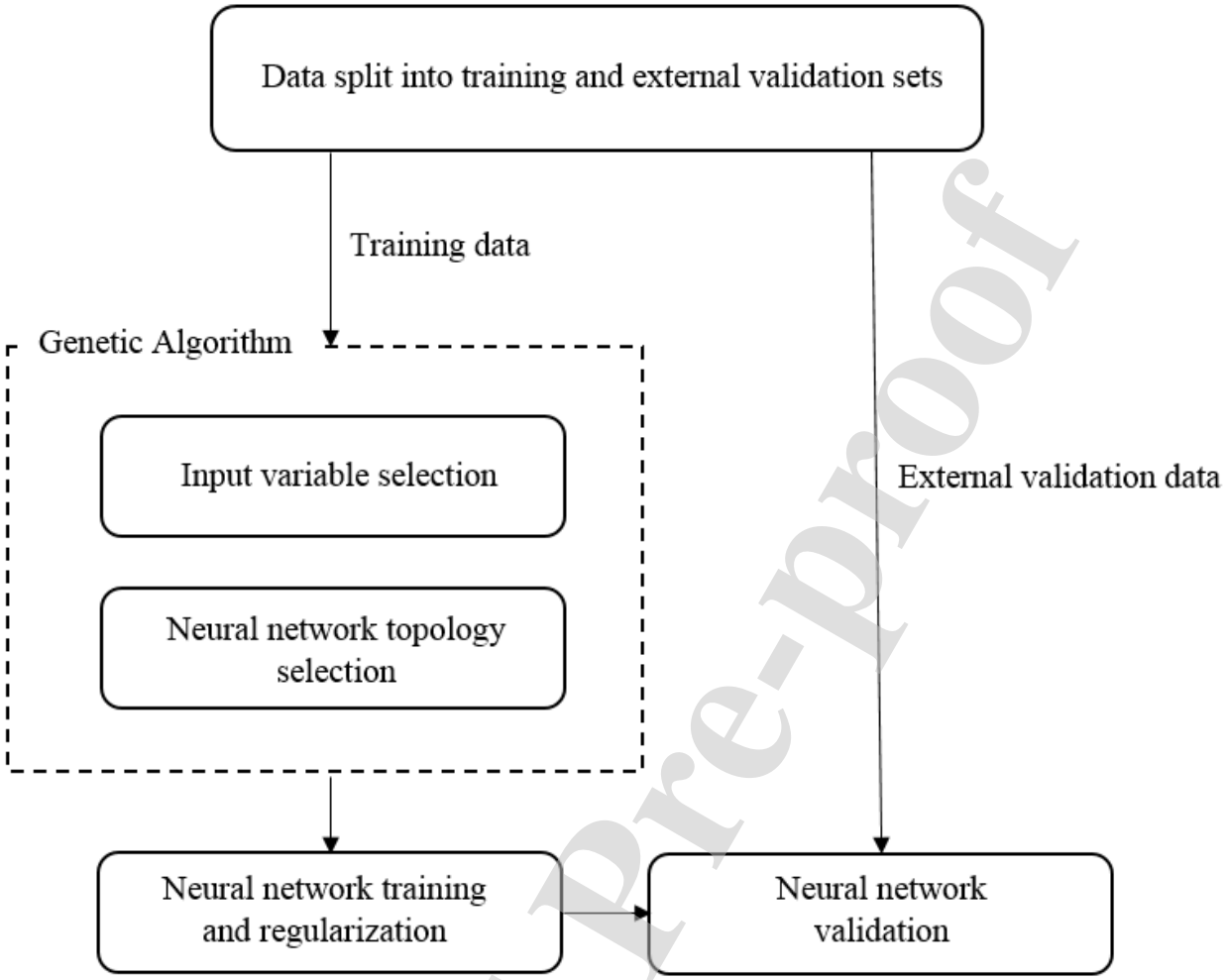


Figure 1. Flowchart of the model identification algorithm.

2.1. Input variable selection with Genetic Algorithm

It is known that the performance of the prediction models can be drastically improved by applying the right subset of variables. The variable selection methods can be divided into model-based and model-free methods. When considering prediction model identification, the variable selection phase is crucial and computationally the most expensive step, especially when the number of candidate variables is high. Furthermore, if no expert knowledge is available, manual variable selection techniques become too time-consuming and often lead to suboptimal variable subsets. A general problem in simple search methods, such as in forward selection or backward elimination is that the methods tend to converge to a local optimum. In the context of variable selection this means for example a model, which includes all the relevant variables, but also a set of redundant or irrelevant variables. [18] The incorporation of irrelevant variables in neural network models has been reported to cause instability, low interpretability and convergence problems during training, resulting in poor model prediction performance. [19]

Genetic Algorithms are a family of metaheuristic optimization methods, which apply the principles of natural selection for exploration of the search space. In Genetic Algorithms, the search domain is presented in the form of a chromosome population, in which each chromosome represents an answer to the problem being studied. [20] The coding of the chromosomes can be based on binary, real or integer numbers. The ability of Genetic Algorithms to solve variable or feature selection problems [5, 21-22], as well as to perform neural network topology optimization [10, 23] and simultaneously both of the aforementioned problems [17] has been proven in various studies. In the studies of Vuolio et al. [5] and Sorsa et al.[21], Genetic Algorithms were applied to Multiple Linear Regression (MLR) model identification, whereas the study of Barros and Rutledge [22] studied the selection of principal components for principal component regression. In the study of Barros and Rutledge [22], the results acquired with the Genetic Algorithm were found to be similar to the results of exhaustive search. However, execution time of the Genetic Algorithm was significantly shorter. [22] In the context of neural networks, the selection of the right prediction variables with a Genetic Algorithm has proven efficient for example in improving the specificity of an Extreme Learning Machine based classifier in the study of Chyzhyk et al.. [24] In the study, the authors applied a Genetic Algorithm and LOO-cross-validation for feature selection to improve the classification results of the ELM model for classification of Alzheimer's disease CAD based on MRI data [24]. Based on the general properties of the algorithm and on the evidence given in the selected studies,

the success of the algorithm in such combinatory optimization tasks can be associated with binary encoding of the solution space and with the capability of exploring a large search space in a computationally efficient manner.

The steps of the binary Genetic Algorithm can be divided roughly into 1) initialization, 2) evaluation, 3) selection, and 4) recombination. In a variable selection problem, each gene in the chromosome corresponds to a candidate variable. The value of the chromosome determines whether the variable is selected in the model; 1 being the variable is selected. The initialization step of a binary-coded chromosome population, i.e. the initialization of the model candidates, is carried out by tossing a biased or a non-biased coin [20].

The evaluation of an individual in the population is carried out by making use of an appropriate fitness function, which in the case of minimization problems is formulated as an inverse of the objective function. Regardless of the variable selection algorithm applied, the objective function is the one that leads the search algorithm in the right direction. [25] A widely discussed fact is that the data-driven models whose identification is based on the training set only are optimistically biased, and thus need a comprehensive validation step in order to generalize well and to be sufficiently stable. [26] The usability of cross-validation as a selection strategy for a neural network model over other methods, such as information criteria and hypothesis testing, is apparent as it demands no statistical assumptions [27]. One of the cross-validation techniques is to use a repeated Leave Multiple Out (LMO) cross-validation in the model selection, in which the data set is repeatedly split into training and validation sets, and the average of *N* repetitions is taken as the validation error. As was illustrated by Baumann [25, 28], the LMO-cross-validation outperforms Leave-One-Out (LOO) cross-validation as the model selection strategy.[25,28] In such a case, the objective function that is based on minimizing the repeated LMO-cross-validation error for *N* split repetitions, can be given as:

$$\mathrm{MSSE}_{\mathrm{CV}} = \min \frac{1}{N} \sum_{l=1}^{N} \sum_{j=1}^{k} (y_{j,CV_l} - f(X))^2, \qquad [2]$$

where MSSE stands for mean of the sum of squared errors for *N* repetitions, *N* is the number of repetitions in the inner loop of the variable selection algorithm, *k* is the number of data points in the internal validation set, *y* is the dependent variable, CV is the cross-validation data-set and *f*(*X*) is a model candidate. Repeated cross-validation serves to eliminate the dependency of the modelling

error from the partitioning data. [28] In the present authors' previous work [24] a Genetic Algorithm that makes use of a repeated cross-validation as the objective function was found to provide sufficiently accurate and robust modeling results based on small and noisy industrial data sets. The performed number of in the internal validation loop was 4*N*. [5] Similar findings were provided for MLR models by Sorsa and Leiviskä [29] and for PLS models by Kepplinger et al. [30]. However, the reliability of results is often a compromise between computational resources. Even though the validation step is carried out several times for each of the model candidates, cross-validation in the variable selection phase only provides the internal validity of the model. For this reason, the model needs to be validated with a completely independent data set referred here to as an external validation data set.

The selection of an individual for the recombination stage is carried out to favor the selection of the fittest individuals [20], in other words the ones with the lowest $MSSE_{CV}$. Thus the population is expected to evolve towards better solutions. In this study, the selection is carried out with a roulette-wheel selection and the number of selected individuals is $n_{pop}/2$. In the recombination stage, the new generation is produced from the selected individuals. The usual recombination operators are crossover and mutation. The formulation of the recombination operators depends on the coding of the chromosome. In the case of binary-coded chromosomes, the operations are carried out by modifying the bit strings. A standard technique is to carry out the crossover by swapping some parts of the selected parent chromosomes with each other. The rate of crossover is regulated with a crossover probability [20]. which is set constant in this work. Usually binary coded mutation is carried out in such a way that each of the bits has a certain probability of inverting from 0 to 1 or vice versa. The probability of the inversion is controlled with mutation probability. This solution can sometimes lead to a non-efficient mutation operator especially with low mutation probabilities. To exclude the local optima, the algorithm follows a deterministic mutation schedule. A deterministic mutation schedule has been observed to be able to efficiently generate new information to the existing population [31].

2.2. Selection of neural network topology with Genetic Algorithm

Usually, the variable or feature selection for a neural network is carried out with a constant number of hidden neurons. Example of this kind of approach can be found in [32]. In this study the chromosomes in the variable and the network topology selection are treated as a hybrid of binary and integer-coded chromosomes. In the population, each chromosome consists of a binary string

representing the variables and an integer value representing the number of hidden neurons. The reasoning behind integer-based coding is that if the full topology of the network were expressed as a binary coded string, the number of bits in a single chromosome would be excessively large, as the number of weight coefficients in the network is $n_{\text{bits}} = 2jk + 2k + 1$, where j is the number of input variable candidates and k is the maximum number of hidden neurons. To exemplify, a network of 23 input variables and 30 hidden neurons corresponds to 721 bits in an individual. A description of a binary-coded topology optimization can be found for example in [23]. Also, as the length of the chromosome determines the maximum number of hidden neurons, the complexity of the model is constrained by it. Alongside this, the large number of bits would assumedly make the binary crossover operators non-efficient, which would make the algorithm computationally very expensive, especially when keeping in mind that the variable selection is carried out simultaneously. On the other hand, the binary-coding could be based on the number of neurons, which still demands relatively large computational resources for large topologies, as the recombination operators demand a for-loop to carry out the bit modifications. In the light of the reasoning above, binary-coded topology optimization is suitable in a situation where the network topology is optimized to a pre-selected set of input variables. In view of these considerations, the computational load of network architecture selection can be drastically reduced by applying integer-based coding to the topology.

In the case of a fully connected single layer network architecture, the topology can be expressed as a single integer value that corresponds to number of hidden neurons. A certain downside of this approach is that the model could contain irrelevant weight connections in the selection phase. The treatment of this matter is discussed in more detail in **Sections 2.3. and 2.4.**. The recombination of the network architectures is based on real-coded crossover and mutation operators. For crossover, the arithmetic crossover was used, in which the offspring of two parents is given by:

$$\gamma_{\text{OC}}^{(1)} = \left[r_1\gamma_{\text{P}}^{(1)} + (1 - r_1)\gamma_{\text{P}}^{(2)}\right], \qquad [3]$$

$$\gamma_{\text{OC}}^{(2)} = \left[r_1\gamma_{\text{P}}^{(2)} + (1 - r_1)\gamma_{\text{P}}^{(1)}\right],$$

where γ is the network topology based chromosome, OC stands for the offspring, P stands for a parent, r_1 is a uniformly distributed random number $r_1 = \{0,1\}$ and the brackets denote the nearest

integer rounding function. The mutation operator for the real-coded part of the chromosome was chosen as a Mäkinen-Periaux-Toivanen-mutation [33]. In their mutation operator, the mutated offspring is given by: [33]

$$\gamma_{\mathrm{OC,M}}^{(1)} = [(1 - t_m)l + t_m u], \qquad [4]$$

where t_m is the decision variable for the mutation, l is the minimum value of the number of hidden neurons in the network and u is the maximum number of hidden neurons in the network. The decision variable t_m is defined as: [33]

$$t_m = \begin{cases} t - t\left(\dfrac{t - r_2}{t}\right)^p & , \quad \text{if } t < \beta \\ t & , \quad \text{if } t = \beta \\ t + (1 - t)\left(\dfrac{r_2 - t}{1 - t}\right)^p & , \quad \text{if } t > \beta \end{cases} \qquad [5]$$

where r_2 is a uniformly distributed random number $r_2 = \{0,1\}$, t is a decision threshold value that is given for an offspring i by $t = (\gamma_{\mathrm{OC}}^{(i)} - l)/(u - l)$ and p is the mutation exponent that defines the distribution of mutation. The crossover and mutation probabilities are equal for both the integer and binary-coded parts of the chromosomes. The encoding of the chromosomes as well as an example of the recombination operators are presented in **Figure 2**. The recombination that is presented in the figure yields two offspring individuals with 14 and 11 hidden neurons and 2 and 4 input variables, respectively. After the mutation, an individual with a network with 14 neurons and 5 input variables is generated. The number of candidate input variables in this example is 7, which corresponds to the length of the binary-coded chromosome.

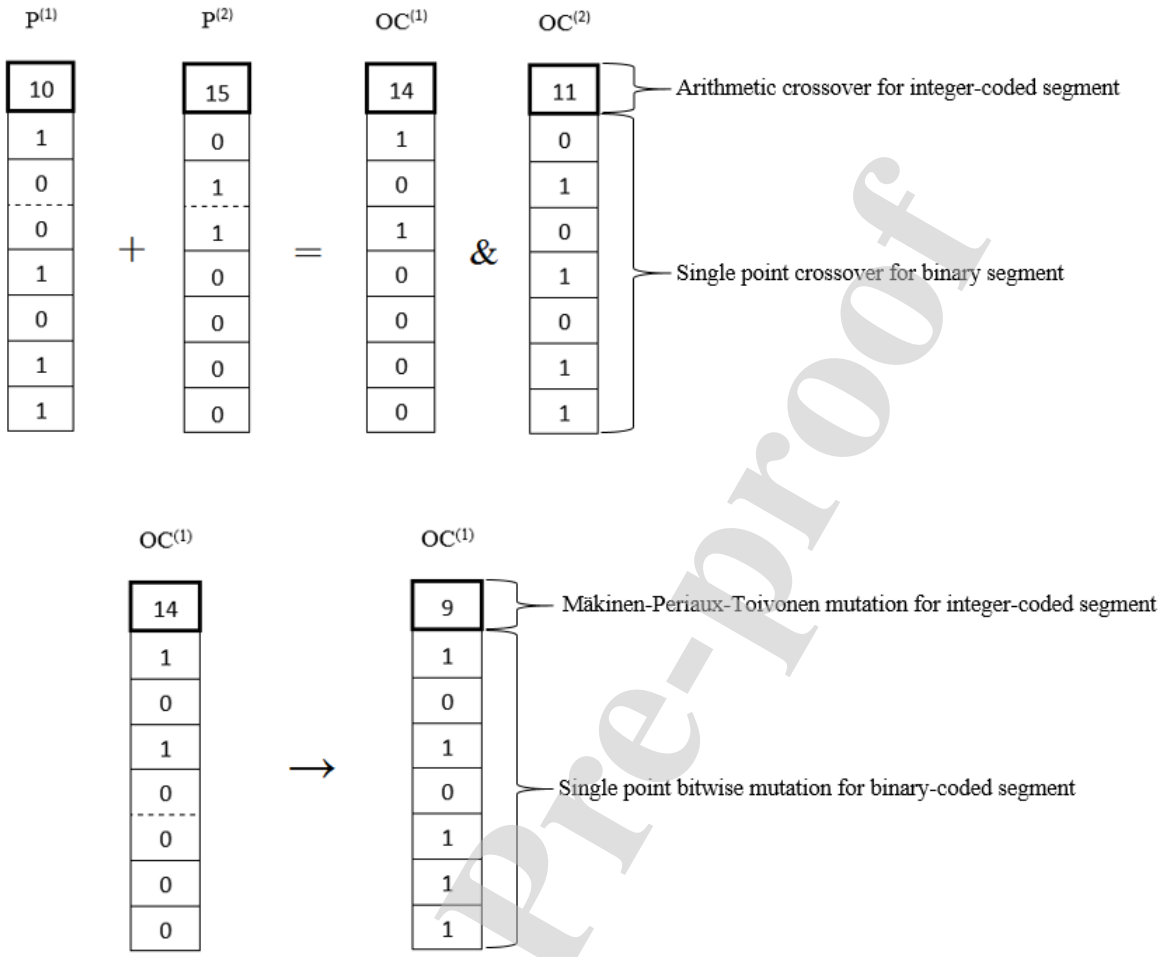


Figure 2. Encoding of the chromosomes and the recombination operators. The crossover and mutation cut off points are randomly selected and illustrated as dashed lines. The crossover and mutation parameters that produce the presented individuals are: $\alpha = 0.2$, $t = 0.4483$, $\beta = 0.2607$, $p = 1$, $u = 30$, $l = 1$.

Neural networks with parsimonious architectures and non-sparse weight coefficients have been found to generalize better for sparse and noisy data than large networks. [34] This can be associated to overfitting of the network. The solutions for the aforementioned problems are presented in **Section 2.4**. Another problem in respect of the generalizability of a network is the problem of multicollinearity among the input variables. As demonstrated by Qin [35], the existence of collinear input variables increases the number of possible weight sets that minimize the objective function. Alongside the fact that this property complicates the training procedure that is based on a random initial guess, it may decrease the prediction performance of the network to an external data set, especially in the case of noisy data. [35] Thus, to avoid the multicollinearity problem in the selected model, the objective function for the variable selection is combined with a penalty term:

$$fitness = \frac{1}{\min \frac{1}{N}\sum_{l=1}^{N}\sum_{j=1}^{n}(y_{j,cV_l} - f(X))^2} - \lambda \mathrm{VIF}_{\max}, \quad [6]$$

where VIF_{max} is the largest diagonal element of the inverse correlation matrix. The constraining parameter λ is chosen such that the maximum penalty is a pre-defined percentage of the maximum fitness of an individual. In this study, 90% was applied. For clarification, the pseudocode for evaluating the population fitness by making use of repeated LMO cross-validation is presented in **Table 1**.

353 **Table 1.** Evaluation of the fitness of the population by making use of repeated cross-validation.

Population fitness evaluation
For $i = 1$ **to** $i = n_{pop}$
1) Select an individual i – An ELM network based model candidate with j variables and k hidden neurons.
For $l = 1$ **to** $l = N$
2) Divide the training data set into a training subset and internal validation data set.
3) Initialize the hidden layer weights randomly from a uniform distribution $W^{(1)} = \{-1,1\}$.
4) Train the selected ELM network with the training set.
5) Calculate the network output and the sum of squared error for the internal validation data set.
6) Return to 2.
End
7) Calculate the arithmetic mean of the internal validation error over N repetitions
8) Evaluate the fitness of the individual i with the fitness function
End

354
355

356 **2.3. Extreme learning machine (ELM)**

357

358 Even though the encoding and decoding of the variable selection phase is intended to reduce the
359 computational load of the algorithm, it is well-known that the training of a neural network is
360 computationally very intensive, because the number of parameters to be optimized is often very
361 large even for relatively small networks. As the cross-validation needs to be repeated several times,
362 the number of neural network training iterations would be $N{\cdot}iter_{max}$ for a single inner loop of the
363 variable selection phase. To tackle this, it is assumed the input output relations with a given variable
364 set and network architecture can be approximated with the Extreme Learning Machine (ELM)
365 architecture proposed in. [36] If a linear output layer is considered, the connective weights between
366 the hidden and the output layer can be obtained with: [36]

$$w^{(k)} = (Z^T Z)^{-1} Z^T y, \qquad [7]$$

where $w^{(k)}$is the weight matrix that connects the output layer and the previous hidden layer, Z is a non-linear transformation of the input data matrix obtained from the hidden layer and y is the desired output vector in the training set. As the applied activation function is the hyperbolic tangent, each column in Z is given as:

$$z_j = \frac{e^{\bar{x}_j} - e^{-\bar{x}_j}}{e^{\bar{x}_j} + e^{-\bar{x}_j}}, \quad [8]$$

$$\bar{x}_j = \sum_{i=0}^{k} w_{i,j}\, x_i + b_j, \quad [9]$$

where z_j is the output of a hidden neuron j, $\bar{x}$ is the weighted input for a hidden neuron j, x_i is an input variable, $w_{i,j}$ is a weight coefficient between input variable i and hidden neuron j and b_j is a bias-term of a hidden neuron j. In the study of Huang et al.[36], it was proven that the ELM is a significantly faster and reasonably accurate training algorithm compared to traditional backpropagation methods, to produce complex input-output mappings by randomly initializing the hidden layer connective weights. However, as the hidden layer weights remain untrained, the usual case is that a large number of hidden neurons is needed. For this reason, the final training of the selected network is carried out by making use of Bayesian regularization. [34] The original authors found that the cross-validated error of the ELM is not sensitive to the number of hidden neurons, and thus, the emphasis in the variable and network architecture selection phase is to find an architecture that is complex enough to produce an estimate of an output vector based on the given set of inputs. [36]

2.4. Training and regularization of the network

The training of the final network is carried out by applying the SSE as the objective function. For the training, a Levenberg-Marquardt (LM) algorithm coupled with Bayesian regularization is applied. The coupling of LM with Bayesian regularization was originally introduced by Foresee and Hagan [34]. The iterative step in the LM-algorithm is given as: [37]

$$\Delta w = \left(J^{\mathrm{T}} J + \lambda I\right) J^{\mathrm{T}} e, \quad [10]$$

where J is the Jacobian matrix, I is the identity matrix, λ is the damping parameter and e is the prediction residual. The damping parameter evolves within iterations, as presented by Hagan and Menhaj [37]. The Jacobian matrix J consists of the first-order partial derivatives of the prediction residual with respect to network weights and biases. The prediction residual is given by:

$$e = y - \hat{y}, \tag{11}$$

where $\hat{y}$ is the network output. The partial derivatives can be calculated with the backpropagation algorithm by applying a chain-rule of derivation. [34] The derivatives of the prediction residuals in respect to hidden layer weights and biases are given by:

$$\frac{\mathrm{d}e}{\mathrm{d}w_{i,j}^{(1)}} = -w_j^{(2)}\left(1 - {z_j}^2\right)x_{i,j}, \tag{12}$$

$$\frac{\mathrm{d}e}{\mathrm{d}b_j^{(1)}} = -w_j^{(2)}\left(1 - {z_j}^2\right). \tag{13}$$

The derivatives of the prediction residuals in respect to the output layer weights and bias are given by:

$$\frac{\mathrm{d}e}{\mathrm{d}w_j^{(2)}} = -z_j, \tag{14}$$

$$\frac{\mathrm{d}e}{\mathrm{d}b_1^{(2)}} = 1. \tag{15}$$

A typical problem in training the neural network is overfitting, which means that the network does not generalize well to an independent data set. The usual methods of improving generalizability are pruning, regularization and early stopping, of which this study considers Bayesian regularization. Overfitting has been associated with overly large networks and with large variances and magnitudes of the network weight coefficients. [34] The Bayesian regularization algorithm attempts to minimize simultaneously the sum of squared error as well as the variance of the network parameters, which often leads to more stable predictions. For this aim, the objective function to be minimized can be written as: [34]

$$F(W) = \beta E_{\mathrm{D}} + \alpha E_{\mathrm{W}}, \quad [16]$$

where E_{D} is the sum of squared errors, E_W is the sum of squared weights, whereas α and β represent the regularization parameters that steer the optimization towards either minimizing the network error (α<<β) or minimizing the sum of the squared weight coefficients (α>>β), of which the latter often yields smoother predictions. The algorithm to minimize the objective function proceeds as follows: [34]

1. Initialize the objective function parameters such that $\alpha = 0$ and $\beta = 1$.
2. Compute one step of minimizing *F*(W).
3. Compute the effective number of parameters by $\gamma = n - 2\alpha \mathrm{tr}(\mathrm{H}^{-1})$. The Hessian matrix can be approximated with $\mathrm{H} \approx 2\beta \mathrm{J}^{\mathrm{T}}\mathrm{J} + 2\alpha \mathrm{I}$.
4. Compute the optimal estimates for the parameters of the objective function with $\alpha = \frac{\gamma}{2E_{\mathrm{W}}(W)}$ and $\beta = \frac{n-\gamma}{2E_{\mathrm{D}}(W)}$.

In the above formulation *n* is the number of parameters in the network and γ measures the number of effective parameters in reducing the error value of the objective function. It should be noted that as the Jacobian matrix is readily available in the LM algorithm, so the implementation of the Bayesian regularization is straightforward. For the interpretation of γ, Foresee and Hagan [34] suggested that if γ is sufficiently larger than the total number of parameters in the network, the network is large enough to map the input-output relations. [34] Consequently this means that if γ, computed for a trained network, is significantly smaller than the number of parameters in the fully connected network based on the ELM architecture, the topology that is selected in the first phase of the algorithm can be considered to be large enough to describe the changes in the sulfur content. This matter is discussed in more detail in **Section 4**.

3. Experimental data and model performance evaluation

The experimental data was gathered from the secondary hot metal desulfurization process at SSAB Europe Oy in Raahe, Finland, and consists of 551 treatments. The considered data matrix consists of 23 candidate variables, which makes the number of model candidates very large (2^n-1). As the number of data splits is high in the internal validation loop, an exhaustive search is practically ruled out as the variable selection strategy. The input candidate variable data consists of hot metal analyzes at the beginning of the treatment, activities of the species in hot metal, temperatures before the treatment as well as the control variables such as the reagent flowrate. The dependent variable, which is the sulfur content, is measured at the end of each of the treatments. In the plant practice, the analysis of the hot metal samples is carried out with the C-S-combustion method and X-Ray Fluorescence (XRF). The evaluation of the model performance was based on standard figures of merit which were the coefficient of determination (R^2) and the mean absolute error (MAE).

4. Results and discussion

The variable selection algorithm was employed for different combinations of computational parameters in order to evaluate their significance. It was also noticed that the selection algorithm was not sensitive to crossover probability. This property was associated with a relatively high and deterministically evolving mutation probability during the first 40 iterations, during which the mutation probability decreased from $p_M = 0.40$ to $p_M = 0.1$. The maximum number of iterations was not pre-defined as the homogeneity of the population was chosen as the convergence criterion. However, it was observed that the algorithm is most sensitive to the number of individuals and the number of data splits in the internal validation loop, which is why the importance of these parameters is systematically evaluated with computer model experiments.

4.1. Model identification and modeling accuracy

An exemplified convergence of the best individual during the iterations is presented in **Figure 3**. It is seen that in this case the algorithm converges to a stationary state after approximately 70 iterations. It can be observed from the figure that the high rate of mutation at the beginning of the search modifies the best individual in the population in an efficient manner, which can be deduced from the oscillation of the objective function value.

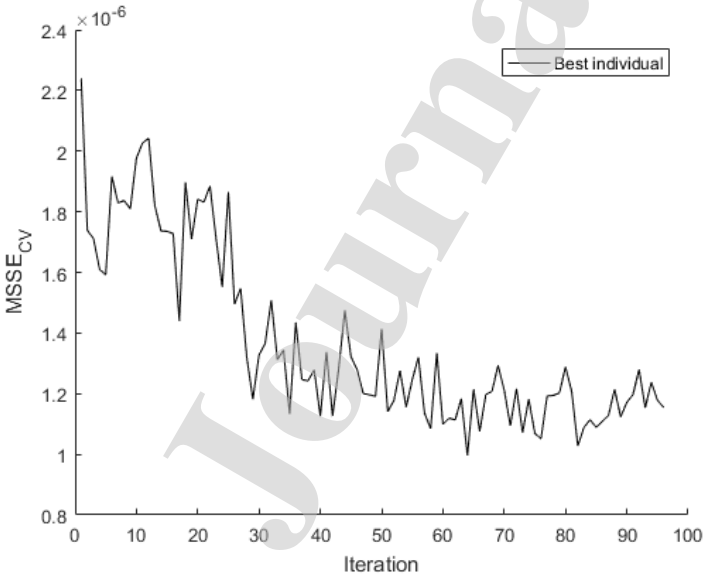


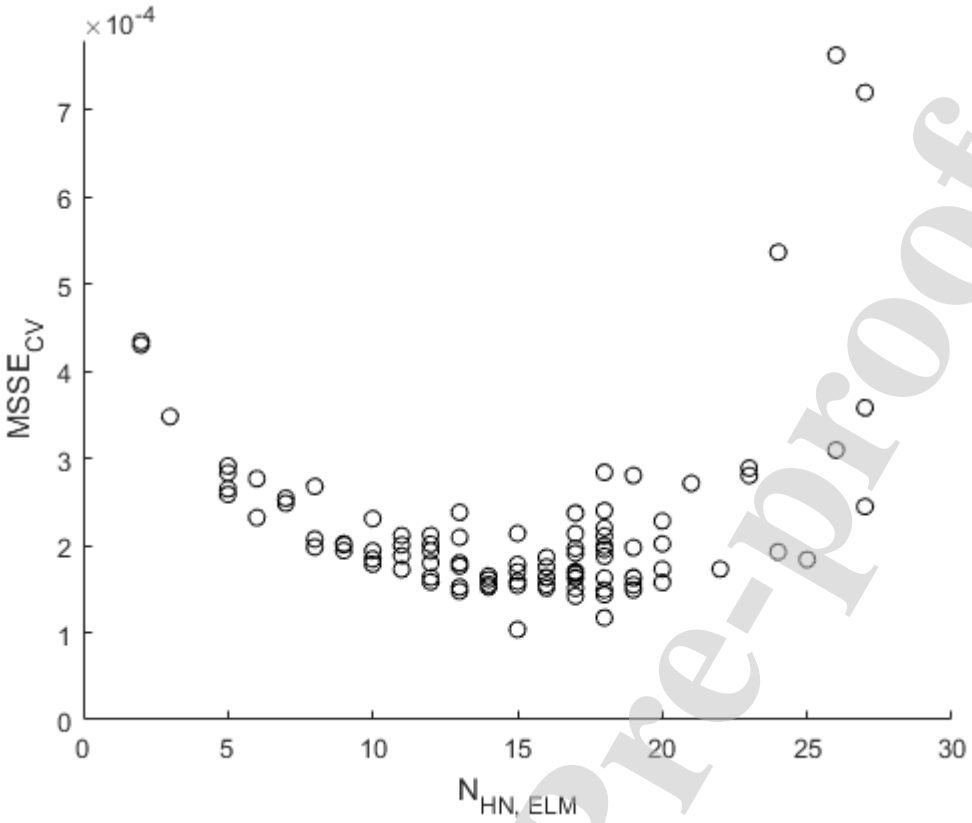
Figure 3. The objective function value as a function of iterations during the variable selection phase.

The fitness values of the end population as a function of the number of hidden neurons in the ELM network is presented in **Figure 4**. It should be noted that the end population is homogeneous with respect to variables, which means that every single chromosome presents the same variable subset. Thus, the mean squared error for cross-validation ($MSSE_{CV}$) depends only on the representability of the internal validation set and the network architecture.

It is apparent that if the cross-validation is carried out for a sufficient number of randomly selected validation subsets with resampling, the variance in the $MSSE_{CV}$ obtained with different network architectures becomes small, and consequently, overfitting and chance correlation can be excluded. It should be noted that each of the models are evaluated for $N \cdot k$ different data splits, where k is the number of similar individuals in the population, which consequently results that the fittest individuals are evaluated with $N \cdot k \cdot i$ splits, where i is the number of iterations in which the model appears in the population. To exemplify, a model that survives the last 20 iterations goes through 1000k cross-validation rounds. For this reason, the selected final model can be considered to be validated exhaustively. It can also be observed that the repeated cross-validation, complemented with the penalty factor, favors the selection of more parsimonious models with respect to network architecture, but still complex enough when it comes to capturing the non-linear interactions in the process.

A notable thing is that the ELM model is not very sensitive to the number of hidden neurons, but at least 12 neurons is needed to avoid underfitting. The stability of the ELM model is in agreement with remarks of Huang et al. [36]. However, as the number of neurons increases, the probability of non-stable predictions could increase, as the probability of occurrence of a non-stable weight coefficient increases. This emphasizes the need of finding the trade-off between the model complexity and the modeling error. [17] This could be the reasoning behind the exceptionally large error values for two of the worst individuals with large number of hidden neurons, regardless of the fact that a high number of cross-validation repetitions is performed. This property, on the other hand, supports the need for regularization in the final model training, but does not significantly affect the selection result as the fittest individual in the end population is selected as the candidate for final training and the probability of chance correlation is low due to the high number of data split repetitions in the internal validation.

503



504

505

506 **Figure 4.** Mean sum of squared error for cross-validation ($MSSE_{CV}$) set as an average of 50 repetitions as a function of the number of hidden neurons in the Extreme Learning Machine (ELM) architecture.

507

508

509 The training of the selected network was carried out as described in **Section 2.4**. The hypothesis
510 testing for the normality of the distribution reveals that a random sample drawn from the training
511 residual is from a normal distribution with a risk-level of 5%. The expected value of the residual is
512 near zero and the standard deviation is 6.9 ppm. Consequently, the outcome for the final network
513 can be given as:

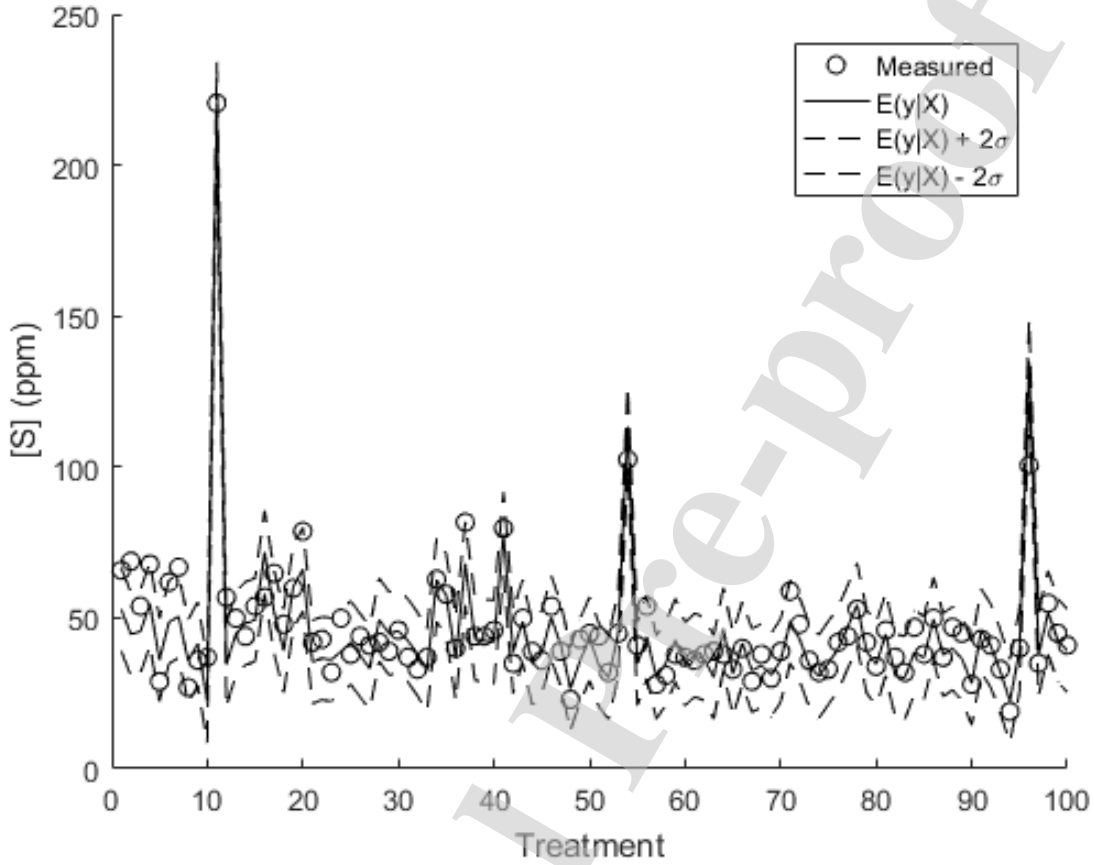
514

$$E(\hat{y}|X) = f(W,X) + \varepsilon = f(W,X) + N(0, 0.00069)$$

515

516 The prediction results with corresponding confidence limits for each of the treatments for the
517 external validation data set are presented in **Figure 5**. It is seen from the figure that the model can
518 predict the sulfur content at the end of the desulfurization treatment with fair accuracy (R^2 = 0.91;
519 MAE = 5.46 ppm). Nearly all predictions for the external validation set are within $\pm 2\sigma$, which
520 corresponds to a 95% confidence interval (CI). The figures of merit for the model as well as for the

521 other models presented in literature are presented in **Table 2.** It can be seen that the present model
522 outperforms the ones proposed in earlier studies. The results of this study can thus be considered
523 encouraging, as the identification is based on noisy real-life data.



524

525 **Figure 5.** The measured and predicted sulfur contents for the external validation set with
526 corresponding confidence limits (CI = 95.4%).

527 **Table 2.** Comparison of existing data-driven models for carbide-based hot metal desulfurization
528 with corresponding figures of merit.

Authors	Model Type	Variable selection	R^2	MAE (ppm)	n
Datta *et al*.	ANN (BP)	Manual	0.60	27.0	11
Rastogi *et al*.	PMR (GA)	Manual	0.69	19.0	12
Deo *et al*.	ANN (BP + GSS + LT)	Manual	0.39	27.8	45
Vinoo *et al*.	MLR	Manual	0.48	15.3	15
This work	ELM – LM + BR	GA	0.91	5.46	100

529 Notes: (ANN = Artificial Neural Network; MLR = Multiple Linear Regression; GAS = Genetic
530 Algorithm; LT = Logistic Transformation of Input Data; PMR = Parameterized Mechanistic
531 Reaction Model; LM = Levenberg-Marquardt; BR = Bayesian Regularization; BP =
532 Backpropagation; GSS = Golden Section Search; *n* = number of points in the internal or external
533 validation data set).

To analyze the repeatability of the model selection, the algorithm was run through 30 times with constant parameters. The parameters chosen for the repetitions were N = 50 *and* n_{pop} = 100. The reasoning behind the aforementioned hyperparameter values is analyzed in more detail in **Section 4.2**. The result of these runs are presented in **Table 3.** It can be seen from the table that under the conditions of this study, the algorithm is able to repeatedly identify a prediction model that is able to capture the variance in the sulfur content, if the hyperparameters (the population size and number of split repetitions) are selected correctly.

Table 3. Results of repetitive testing of the model selection algorithm. The values have been calculated based on 30 repetitions. The parameters chosen were N = 50 and n_{pop} = 100.

	Training		External validation	
	R^2	MAE (ppm)	R^2	MAE (ppm)
Mean	0.84	5.02	0.84	6.39
Median	0.86	5.44	0.86	6.16
Best	0.92	5.33	0.91	5.46
Worst	0.85	6.61	0.82	6.85

4.2. Repeatability and robustness of the algorithm

To evaluate the importance of the variable candidates and the robustness of the algorithm, a repeated test was carried out. The repeated test was carried out based on a full factorial experimental design matrix. The design vectors for the number of individuals and the number of split repetitions were n_{pop} = [20 40 60 80 100]^T and N = [0 10 30 50] ^{T}. The variable selection algorithm was repeated 100 times for each of the design levels, which consequently yields 2000 runs for the algorithm. **Figure 6** presents the average of the selected variables with corresponding standard deviations as a function of population size. However, an adequate number of individuals is assumed to be dependent on the number of input candidate variables, which could be confirmed in further studies with multiple different data-sets.

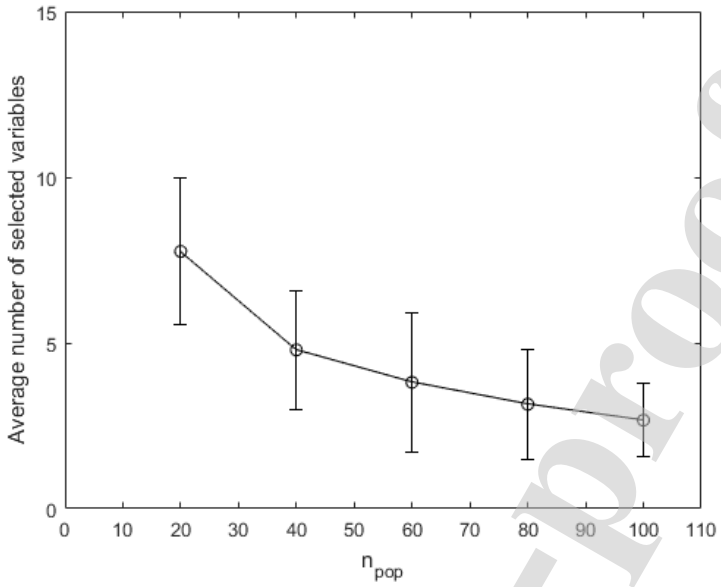


Figure 6. Average number of selected variables for 100 repetitions as a function of population size and for $N = 50$.

The results of the repetitive testing were analyzed with MLR based metamodeling. For modeling, the following hypotheses were postulated:

- $H_{0,1}$ – The size of the population has no effect on the search result,
- $H_{\alpha,1}$ – The increase in the size of the population decreases the expected value for the number of selected variables,
- $H_{0,2}$ – The number of data splits does not improve the repeatability of the algorithm,
- $H_{\alpha,2}$ – The number of data splits reduces the standard deviation between the repetitions.

The modeling results for both the mean and standard deviations of the number of selected variables are given in **Table 4**. From the table it can be seen that increasing the population size has a negative logistic effect on the expected value of the number of selected variables. It is seen from the table that the estimate of the regression coefficient related to the number of individuals is not saturated, as the standard error of the estimate is only 5.7%, which results int the *p*-value of the estimate being significantly smaller than the selected risk level ($\ll 0.05$). Thereby, the analysis also confirms the information provided in **Figure 6**. This result indicates that the suitable number of individuals is

dependent on the number of candidate variables, i.e. the number of possible model candidates, and consequently fulfills the postulated hypothesis $H_{\alpha,1}$.

The analysis of the standard deviations of the selected variables was carried out similarly to the analysis of the expected value. As seen in **Table 4**, the increase in the population size does not necessarily improve the robustness, as for a small number of data split repetitions, a relatively good model with a large number of input candidates can be selected by chance. This can be associated both to relatively stable behavior of the ELM model with respect to redundant variables and the number of neurons. However, as stated by Baumann [25,28], the increased number of repetitions decreases the probability of selecting a good model by chance. [25,28] This is clearly seen in the results; the increase in the number of splits reduces the standard deviation of the search (*p*-value < 0.05).

Table 4. MLR models for analysis of algorithm performance.

y	Variable	Estimate	Std. Error	t-value	p-value
avg(num_{select})	bias	15.60	0.66	23.74	$1.70 \cdot 10^{-14}$
	ln (n_{pop} + 1)	-2.82	0.16	-17.64	$2.29 \cdot 10^{-12}$
	ln (N + 1)	-0.01	0.05	-0.16	0.87
std(num_{select})	bias	4.39	0.95	4.60	$2.50 \cdot 10^{-4}$
	ln (n_{pop} + 1)	-0.37	0.23	-1.59	0.13
	ln (N + 1)	-0.30	0.09	-3.54	$2.53 \cdot 10^{-3}$

When the reasoning above is complemented with the selection histograms presented in **Figures 7 a), b), c) and d)**, it can be postulated that if the number of individuals is sufficiently high and the uncertainty in the internal validation is low enough, the algorithm is able to find the relevant variables, and also to minimize the number of redundant variables in the model with a very high probability. Based on the repetitive tests, the most significant variables are:

1) Sulfur content in the hot metal before treatment (x_3) – Hit-rate = 100%,
2) The amount of reagent injected (x_{15}) – Hit-rate = 72%,
3) The injection time (x_2) – Hit-rate = 35%.

It should be noted that two of the most important variables are to some extent interrelated, as the injection flowrate (kg/min) determines the injection time. However, their VIF indexes are ~2.5 on average, which indicates a non-significant multicollinearity. Surprisingly, the chemical composition of the metal phase explains hardly any observable variance in the sulfur end-point, but some hot

metal components, such as Si, Mn, V and Ti appear irregularly in the search results. However, the appearance of these variables in the search may be by chance, as the effect of the variables on the modeling results was found to be small. This is supported with the fact that the selection probability of these variables was found to decrease while increasing the population size and number of internal validation repetitions.

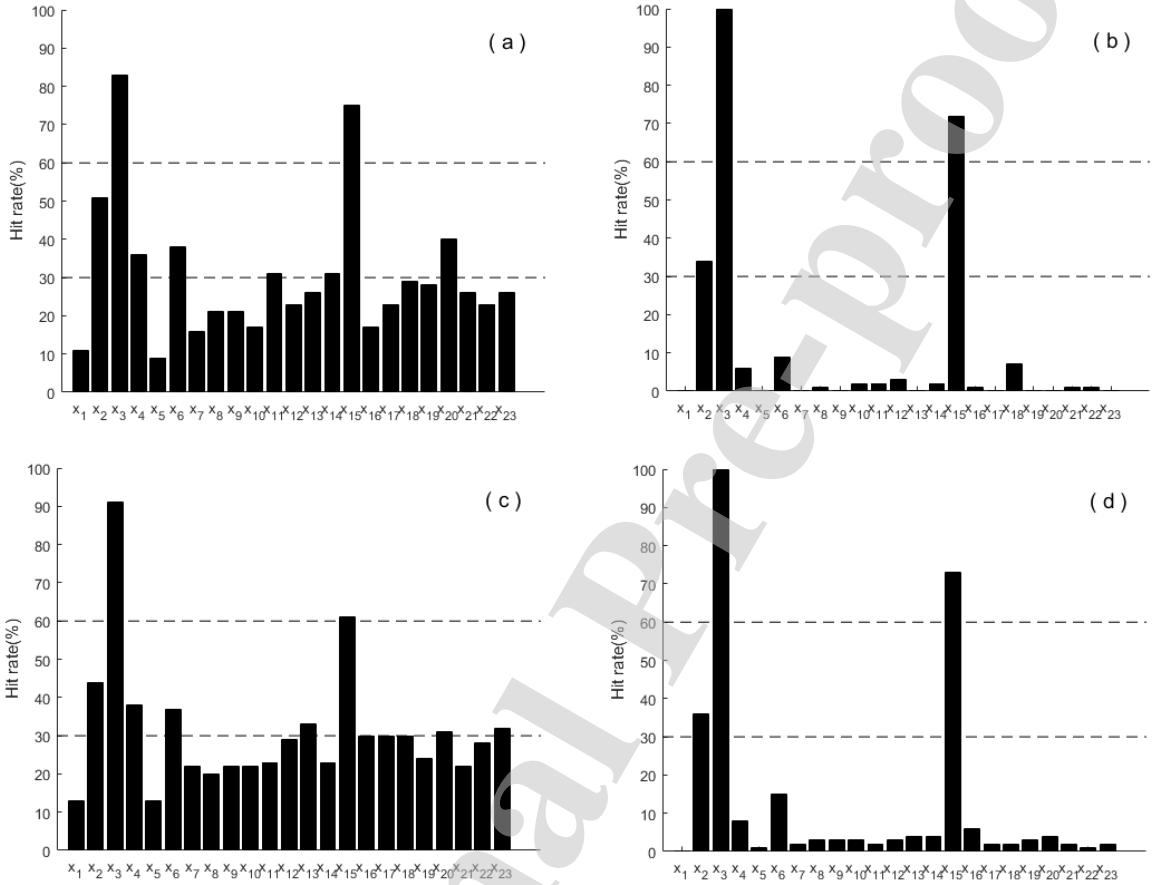


Figure 7. Selected variables for 100 repetitions of the variable selection algorithm. a) N = 50, n_{pop} = 20; b) N = 50, n_{pop} = 100; c) N = 10, n_{pop} = 20; d) N = 10, n_{pop} = 100.

4.2.1. Selection of the number of split repetitions

As the reliability of the selection algorithm depends on the representability of the cross-validation data set, and thus is often a compromise with the computational load, the number of repetitions is an important tunable hyper parameter. It was observed that realistic prediction results were obtained when the data was split so that 57% was used for training, 25% for internal validation and 18% for external validation. It should be noted that as the LMO-cross-validation is considered as a resampling method, the data used for model selection is 82% of all data. To make the selection

result independent of the data split, the cross-validated error needs to be repeatedly evaluated in the variable selection phase. [28]

As the exhaustive testing of all possible split combinations is not computationally feasible, the trade-off between the feasible number of splits and uncertainty of the mean error needs to be computed. In repetitive testing, the uncertainty for a 95.4% confidence limit is given as: [38]

$$u_{\mathrm{CV}} = 2\frac{\sigma_{\mathrm{MSSE_{CV}}}}{\sqrt{N}}. \qquad [17]$$

The relative uncertainty is then defined as the ratio of uncertainty and the average MSSE of the repetitions:

$$\bar{u}_{\mathrm{CV}} = \frac{u_{\mathrm{CV}}}{\mu_{\mathrm{MSSE_{CV}}}} \qquad [18]$$

In **Figure 8,** the relative uncertainty in the MSSE of the repeated cross-validation is presented as a function of cross-validation repetitions in the internal validation loop of the variable selection phase. The error values are computed for the best individual in the final population. The fitted curve reveals that to yield reliable results with an uncertainty of less than 5%, at least 37 repetitions in the internal validation loop must be carried out for this data.

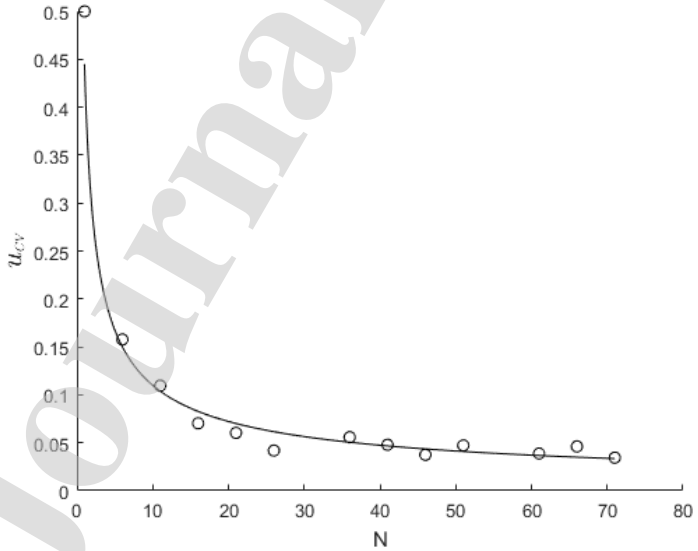


Figure 8. Relative uncertainty of cross-validation error as a function of split repetitions in the internal validation loop.

4.3. Implications on selection results

In the analysis of the selected variables it seems that the flowrate of the injected reagent (kg/min) does not explain the changes in the end content of sulfur. This particular fact is contradictory to our previous studies, but can be attributed to the differences in the reagent and thus in the rate controlling mechanism. It was presented by Oeters *et al.* [39] that in hot metal desulfurization with calcium oxide (CaO), the reaction controlling mechanism can be divided in two parts; solid-state and boundary-layer diffusion controlled phases. The solid-state control, i.e. the control mechanism of the reaction product layer, determines the rate of reaction if the sulfur content is relatively high.[39] Consequently, introducing a fresh reaction surface into the hot metal improves the rate and efficiency of the reagent with the result that the mass flowrate of the reagent explains the variance in the process. However, with low sulfur contents, the rate of reaction is increasingly controlled by the boundary-layer diffusion mechanism, which practically means that the rate is to some extent independent of the available solid reaction surface. This reasoning could explain why changes in the mass flowrate do not result in observable changes in the sulfur content in the hot metal. In the case of calcium carbide (CaC_2) the solid-state diffusion controlled phase does not control the reaction, as the reaction product (CaS) is soluble in CaC_2. [1] The boundary-layer diffusion controlled mechanism was confirmed in the study of Chiang et al. [2].

The effect of the activity of oxygen on the rate and efficiency of hot metal desulfurization was comprehensively studied by Zhao and Irons [40]. In the study, the effect of the activity of oxygen in the hot metal on the rate and equilibrium content of sulfur was observed to be relatively high. [40] However, the activity of oxygen cannot be determined if the activities of the metal and slag species are not known. As no information on the slag composition is available, this attribute cannot be indirectly deduced by making use of computational thermodynamics-based feature engineering. Thus in industrial conditions, this attribute is still not easily measurable, but it may improve the modeling accuracy. Consequently, more research needs to be carried out on the determination of these attributes.

5. Conclusions

In this work, an algorithm for the selection of a well-generalizing neural network model using a Genetic Algorithm was presented. The repeatability and the robustness were found to be very good, which make it well-suited for process identification and control purposes. The selection results of the algorithm suggest that the sulfur end-point can be predicted with a high degree of accuracy by making use of the initial sulfur content and amount of reagent injected as the main predictor variables. The figures of merit of the model presented, R^2 = 0.91 and MAE = 5.46 ppm, were determined for a sufficiently representative external validation set of 100 treatments. The results also indicate that the injection rate and the activities of some of the dissolved elements in the hot metal, namely Si, Mn, Ti and V, have only a minor effect on the end content of sulfur. This study emphasizes the applicability of data-driven techniques and genetic algorithms in model selection tasks based on a noisy industrial data set. However, the selection has to be performed by making use of repeated cross-validation in order to avoid the selection of overfitted and coincidental models.

Acknowledgements

This work was conducted within the Symbiosis of Metal Production and Nature (SYMMET) research program funded by Business Finland. The financial support of Technology Industries of the Finland Centennial Foundation, the Tauno Tönning foundation, the Finnish Foundation of Technology Promotion and Walter Ahlström foundation are also acknowledged.

Conflict of interest

The authors declare no conflict of interest.

724 **References**

725

726 1. Oeters, F., Metallurgy of Steelmaking, ISBN 3-514-00465-X ed., Düsseldorf: Verlag
727 Stahlheisen, 1994.
728 2. Chiang, L., Lu, W., Irons, G., Cameron, I., I & SM 1990, 17, pp. 35–52.
729 3. Visuri, V.-V., Vuolio, T., Fabritius, T., Unpublished Manuscript.
730 4. Vuolio, T., Visuri, V.-V., Tuomikoski, S., Paananen, T., Fabritius, T., Metall. Mater.
731 Trans. B 2018, 49, 2692–2708.
732 5. Vuolio, T., Visuri, V.-V., Sorsa, A., Paananen, T., Fabritius, T., Steel Res. Int, 2019, 90,
733 Article number 1900090.
734 6. Vinoo, D., Mazumdar, D., Gupta, S., Ironmak. Steelmak. 2007, 34, pp. 471–476.
735 7. Deo, B., Datta, A., Kukreja, B., Rastogi, R., Deb, K., Steel Res. 1994, 65, pp. 528–533.
736 8. Datta, A., Hareesh, M., Kalra, P., Deo, B., Boom, R., Steel Res. 1994, 65, pp. 466–471.
737 9. Rastogi, R., Deb, K., Deo, B., Boom, R., Steel Res. 1994, 65, pp. 472–478.
738 10. Pettersson, F., Hinnelä, J., Saxén, H., Materials and Manufacturing Processes 2003, 18,
739 pp. 385–399.
740 11. Saxén, H., Petterson, F., ISIJ Int. 2007, 47, pp. 1732–1737.
741 12. Pettersson, F., Chakraborti, N., Saxén, H., Applied Soft Computing 2007, 7, pp. 387-
742 397
743 13. Saxén, H., Petterson, F., 2009, Lecture Notes in Computer Science (including subseries
744 Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 5495
745 LNCS, pp. 72–78.
746 14. Wang, Z., Chang, J., Ju, Q., Xie, F., Wang, B., Li, H., Wang, B., Lu, X., Fu, G., Liu, Q.,
747 ISIJ Int. 2012, 52, pp. 1585–1590.
748 15. Wang, W., Liu, J., Liu, X., 2015, The Open Automation and Control Systems Journal
749 2015, 7, pp. 966–973.
750 16. Wang, X., Han, M., Wang, J., Engineering Applications of Artificial Intelligence 2010,
751 23, pp. 1012–1018.
752 17. B. Mahanta, N. Chakraborti, Steel Res. Int. 2018, 89, Article number 1800121.
753 18. Guyon, I., Ellisseef, A., An Introduction to Variable and Feature Selection Methods,
754 Journal of Machine Learning Research 2003, 3, pp. 1157–1182.
755 19. Back, A., Trappenberg, T., IEEE Transactions on Neural Networks, 12, 612–617.
756 20. Goldberg, D., Genetic Algorithms in Search, Optimization, and Machine Learning,
757 Addison-Wesley, Boston, MA, USA, 1989.

21. Sorsa, A., Leiviskä, K., Santa-aho, S., Vippola, M., Lepistö,T., 2013, J. Nondestruct. Eval. 32, pp. 341–349.
22. Barros, A., Rutledge, D., Chemometrics and Intelligent Laboratory Systems 1998, 40, pp. 65–81.
23. Whitley, D., Starkweather, T., Bogard, C., Parallel Computing 1990, 14, pp. 347–361.
24. Chyzhyk, D., Savio, A., Grana, M., Neurocomputing 2014, 128, pp. 73–80.
25. Baumann, K., Trends in Analytical Chemistry 2003, 22, pp. 395–406.
26. Harrell, F., Regression Modeling Strategies: With Applications to Linear Models. Logistic Regression and Survival Analysis, Springer, New York, 2001.
27. Anders, U., Korn, O., Neural Networks 1999, 12, pp. 309–323.
28. Baumann, K., QSAR Comb. Sci. 2005, 24, pp. 1033-1046.
29. Sorsa, A., Leiviskä, K., Feature Selection from Barkhausen Noise Data Using Genetic Algorithms with Cross-Validation, Adaptive and Natural Computing Algorithms, 9th International Conference, 2009, pp. 213-222.
30. Kepplinger, D., Filzmoser, P., Varmuza, K., Variable Selection with Genetic Algorithms Using Repeated Cross-Validation of PLS Regression Models as Fitness Measure. Unpublished work. https://arxiv.org/pdf/1711.06695.pdf
31. Bäck, T., Schütz, M., Proceedings of the 9th International Symposium on Methodologies for Intelligent Systems, 1996, pp. 158–167.
32. Deniz, A. and Kiziloz, H., Expert Systems With Applications 2019, 137, pp. 11–21.
33. Mäkinen, R., Periaux, J., Toivanen, J., Int. J. Numer. Meth. Fluids 1999, pp. 149–159.
34. Foresee, F., Hagan, M., Proceedings of International Joint Conference on Neural Networks, 1997, pp. 1930–1935.
35. Qin, S., Neural networks for intelligent sensors and control – practical issues and some solutions. In: Elliott, D., Neural Networks for Control, Chapter 8, Academic Press, 1996.
36. Huang, G., Zhu, Q., Siew, C., Neurocomputing 2006, 70, pp. 489–501.
37. Hagan, M., Menhaj M., IEEE Trans. Neural Netw 1994, 5, pp. 989–993.
38. Eisenheart, C., Science 1968, 160, pp. 1202–1204.
39. Oeters, F., Strohmenger, P., Pluschkell, W., Arch. Eisenhüttenwes. 1973, 44, pp. 727-733.
40. Zhao, Y., Irons, G., Ironmaking and Steelmaking 1994, 21, pp. 309–316.

CRediT author statement

Conceptualization – Tero Vuolio, Ville-Valtteri Visuri, Aki Sorsa

Methodology – Tero Vuolio, Ville-Valtteri Visuri, Aki Sorsa

Software – Tero Vuolio

Writing, Original draft – Tero Vuolio

Writing, Review and Editing – Tero Vuolio, Ville-Valtteri Visuri, Aki Sorsa

Validation – Tero Vuolio, Ville-Valtteri Visuri, Aki Sorsa

Resources – Seppo Ollila

Supervision – Timo Fabritius

Project Administration – Ville-Valtteri Visuri, Timo Fabritius

Conflict of interest

The authors declare no conflict of interest.