Physicochem. Probl. Miner. Process., 56(6), 2020, 194-205

http://www.journalssystem.com/ppmp

Physicochemical Problems of Mineral Processing

ISSN 1643-1049

Received September 01, 2020; reviewed; accepted October 15, 2020

Radial Basis Function Neural Network based on Growing Neural Gas Network applied for evaluation of oil agglomeration process efficiency

Marcin Kamiński [1], Radosław Stanisławski [1], Anna Bastrzyk [2]

[1] Faculty of Electrical Engineering, Wroclaw University of Science and Technology, Janiszewskiego 8, 50-377 Wroclaw, Poland

[2] Faculty of Chemistry, Wroclaw University of Science and Technology, Norwida 4/6, 50-373 Wroclaw, Poland 2

Corresponding author: anna.bastrzyk@pwr.edu.pl (Anna Bastrzyk)

Abstract: In this study, the neural model for modeling of oil agglomeration of dolomite in the presence of anionic and cationic surfactants (sodium oleate and dodecylammonium hydrochloride) was implemented. The effect of surfactants concentration, oil dosage, time of mixing, pH, and mixing speed of the impeller in the process recovery were investigated using Radial Basis Function Neural Network (RBFNN). A significant problem in this modeling, was the selection of the structure of the neural network. In algorithms based on the RBFNN, the issue mentioned relates to the number of nodes in the determination of the hidden layer. Also, the distribution of functions in data space is significant. In the proposed solution, at this stage of the neural model design, the Growing Neural Gas Network (GNGN) was implemented. Such a procedure introduced automation of the calculation process. The centers were obtained from the GNGN and the structure (number of radial neurons) can be approximated based on a simple searching algorithm. The idea of the data calculations was implemented as an original algorithm that can be easily transferred to Matlab, Python, or Octave software. The values predicted from the neural networks model were in good agreement with the experimental data. Thus, the RBFNN-GNGN model used in this study, can be employed as a reliable and accurate method to predict, and in the future to optimize the performance of oil agglomeration process.

Keywords: oil agglomeration modeling, dolomite, Radial Basis Function Neural Network, Growing Neural Gas Network

1. Introduction

In mineral processing to reveal a valuable mineral, very often large quantities of fine particles (below 45 μm) are produced. Traditional methods such as froth flotation or gravity-sedimentation processes are not sufficient in the removal of such fine particles. Oil agglomeration is over these methods due to the possibility of the fine particles (below 5 μm) agglomeration with high selectivity and using simple equipment (Huan and Berg, 2003; Pietsch, 2005). Generally speaking, oil agglomeration is one of the size-enlargement methods, in which oil (a bridging liquid) is added to the mineral suspension under the vigorous mixing condition (Drzymała, 2007). As a result, the dispersed oil droplets collide with the particles at the given condition, and hence, spherical agglomerates are formed. Then, the agglomerates settle down and can be easily separated from the suspension by screening or sedimentation. On the other hand, not all particles can form stable agglomerates. The efficiency of the process mostly depends on the properties of the surface of minerals. Only the particles well wetted by oil (hydrophobic) can agglomerate. The hydrophilic ones to be agglomerated require the addition of proper surface active agents that adsorb onto the mineral surface and change the surface properties of minerals (Sadowski, 2000; Sönmez and Cebeci, 2003; Duzyol and Ozkan, 2010). For this reason, the process can be used for selective separation of one mineral from the mineral waste, e.g. coal purification from inorganic components (Yadav et al., 2018; Guan et al., 2018; Kaya and Ari, 2019; van Netten et al., 2020), coal

DOI: 10.37190/ppmp/128533

recovery form tailings (Yasar and Uslu, 2019) or barite separation from carbonaceous minerals (Sadowski, 1995). Despite these advantages of this process, it is not commercially used due to the high costs associated with the use of large amounts of oil (Pietsch, 2005). For many years, researchers have been trying to reduce the cost by using the oil wastes or emulsion as bridging liquid (Laskowski and Zu, 2000; Bastrzyk et al., 2011; Polowczyk et al., 2018; Yadav et al., 2018; Shukla and Venugopal, 2019; Chakladar et al., 2019). The results from these studies showed that the agglomeration process could be an alternative to traditional methods used in mineral processing. Moreover, the course of the oil agglomeration process depends on many parameters presented in Fig. 1, and modeling which is very difficult and time-consuming. This requires solving complex mathematical equations that are problematic in real applications. To overcome this problem the neural network model has been used.

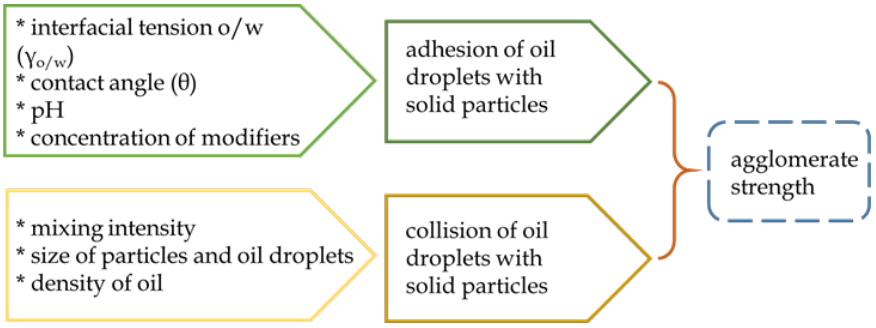

Fig. 1. The parameters affecting the final structure of agglomerates

Neural networks are useful tools in applications for problems that are difficult to describe with mathematical equations, in a plant or processes which contain nonlinear elements. These models can also be useful when the exact identification of parameters is problematic, in case of systems with time-varying parameters, the uncertainty of input data, etc. It deals with the following properties: data approximation, pattern classification, prediction, automatic signal processing, robustness against disturbances (Fausett, 1994; Dias and Pooliyadda, 2001; Shokry and Espuña, 2018; Amsolov and Galin, 2019). The above-mentioned features are related to the construction of the neural network models. Overall, the adaptation of the model for a given task of the most popular structures is based on the proper calculation of the weights connecting several nodes. The most effective methods are to use the gradient-based techniques. It can be done in a training process that can be realized using the appropriate software. However, the neural model, in most applications, tends to the separation of data according to specific features.

One of the criteria for dividing neural networks is the direction of data processing. The first group is the feedforward neural networks. Others - recurrent models - contain additional internal connections. Depending on the construction, the most frequently used networks are Elman (additional context layer - feedback connections in the hidden layer) and Jordan (additional connections from the output to the model input). Recurrent neural networks contain memory elements, therefore they are most often used for processing of dynamic signals. In the application described in this study, this feature is not required. Thus, feedforward neural networks were used. Among them, two types of models dominate, which are distinguished according to the type of activation functions used and the method of performing calculations – Multilayer Perceptron (MLP) and Radial Basis Function (RBF) Neural Network. The significant problem appearing at the stage of adjusting the neural network to a specific issue is the suitable selection of the structure complexity (Kamiński and Bastrzyk, 2018). The goal of neural network training is not to achieve an exact fit to the training data, but to create a model capable of approximating data based on input values, which were not presented during the training process. In other words, the aim of parameter optimization (weights and structure) is to attain an acceptable level of generalization.

In Fig. 2 main procedures used for this important feature improvement are shown. One of them is the application of extended cost function (using weights or modified definition of error). Another method works similarly, and relies on noise inserting to the training data (Molina and Zerubia, 2000). In the literature, this method is often called jittering. Another mode of action (early stopping) depends on specifying a relatively large number of training iterations, then stopping the calculations when

generalization error begins to rise (Wu and Liu, 2009). Network structure modifications (during the training process) algorithms are also preferable (it can be significant in hardware implementation) and analyzed in literature. One way is to first choose a comparatively big network structure and then eliminate unimportant weights. Good results are yielded through the use of sensitivity methods such as Optimal Brain Damage or Optimal Brain Surgeon (Hassibi et al., 1993). Besides structure pruning, structure growing algorithms are also implemented. This group of algorithms includes Cascade Correlation method (Fahlman and Lebiere, 1997) and Growing Neural Gas Network (GNGN) based models (Vachkov, 2004).

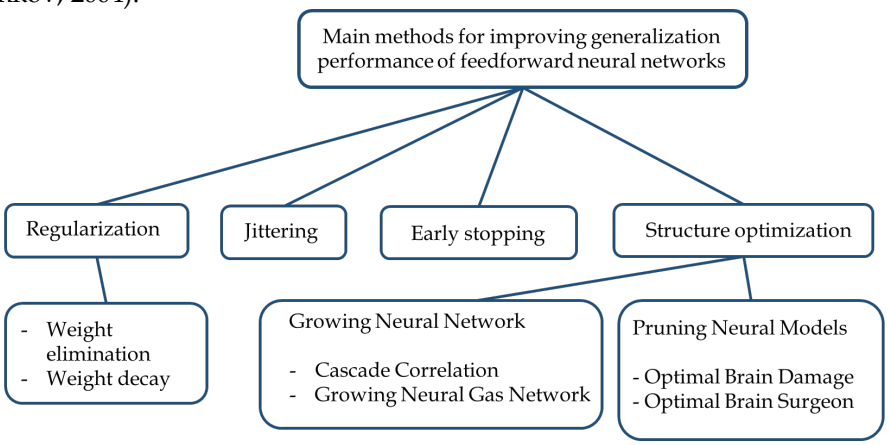

Fig. 2. Main methods used for obtaining better generalization performance

Nowadays, one of the areas of research neural modeling has been often applied to is minerals processing. Mineral systems and processes are hard to measure (Cisternas et al., 2020), therefore, neural models can be used for their better representation and analysis. Numerous examples of the usage of neural modeling have been presented in the literature. For instance, neural networks were used to model the process of metal removal from mining wastewaters through the use of electrocoagulation (Ribeiro et al., 2019). It has also been used to estimate the performance of the industrial floatation process. Authors point out an important element in the process of training neural networks - the selection of initial values for weights. A genetic algorithm was used in these tasks (Allahkarami et al., 2017; Jamróz et al., 2020). Thus, neural models can be created to perform tasks of predicting the behavior of a selected chemical process under certain conditions. Compared to conveying traditional experiments, research involving neural modeling gives results faster, is more cost–efficient, and allows optimization of the conditions of future experiments (Cisternas et al., 2020). However, the studies regarding the application of neural networks for oil agglomeration prediction is limited. For this purpose, the MLPs were only used (Kamiński and Bastrzyk, 2018; Yadav et al., 2018). Relying on the motivations and trends presented above, in this study a combination of the RBFNN and the GNGN implemented for structure optimization was applied. Details of the networks and training methods are described later.

This paper presents a proposition and tests of modeling of oil agglomeration of dolomite in the presence of anionic and cationic surfactants. The implemented algorithm uses the radial neural network structure. Therefore, after the general problems of the neural modeling presentation, the details of the analyzed process, conditions, and measurement activities are described. Then, the fundamentals of the RBFNN calculations are shortly presented. In the following section, the GNGN is shown. The next part of the article was focused on the idea of the training algorithm. After the theoretical background, the main results were analyzed. The article was closed by remarks that highlight the achievements, observations, and conclusions.

2. Materials and methods

2.1. Materials

As mineral, dolomite (d_{50} = 17 μm) collected form quarry 'Kletno' (Poland) was used. The X-ray diffraction analysis showed that the mineral is relatively pure with a small quantity of silica (c.a. 2 %).

The anionic surfactant for the hydrophobization of the mineral surface was sodium oleate (NaOL) (J.T. Baker Chemical Co., USA), and the cationic surfactant for the emulsion preparation was dodecylammonium hydrochloride (DDAHCl) (99 %, Alfa Aesar, USA). Kerosene with density of 0.81 g cm^{-3} (Synpeko, Poland) was used as a bridging liquid.

2.2. Methods

2.2.1. Oil agglomeration procedure

For the preparation of the neural networks model, the previously performed experimental data (set of 48 experiments) were used (Bastrzyk et al., 2011). The four experiments were used for a model verification (these experiments were not included in a training process). The detail of the experiments is shown in Fig. 3. Firstly, the mineral suspension (5 g of dolomite in 100 cm^3 of distilled water) with desired pH was prepared and conditioned for 24 hours. Then, the process agglomeration started by pouring 100 cm^3 of the emulsion to the suspension. After the agglomeration process had been completed, agglomerates were screened on a 125 μm sieve and dried in the oven. During the experiment, the pH (6-12), oil dosage (0.04-0.16 cm^3 g^{-1}_{solid}), stirring rate (200-500 rpm), and time of mixing (6-1500 s), and surfactant concentration (NaOL: 2.43-24.36 mg g^{-1}_{solid}; DDAHCl: 1.8-26.6 mg g^{-1}_{solid}) were changed and taken for modeling as inputs. The output was the process recovery calculated as the ratio of mass agglomerate to the total mass of the feed.

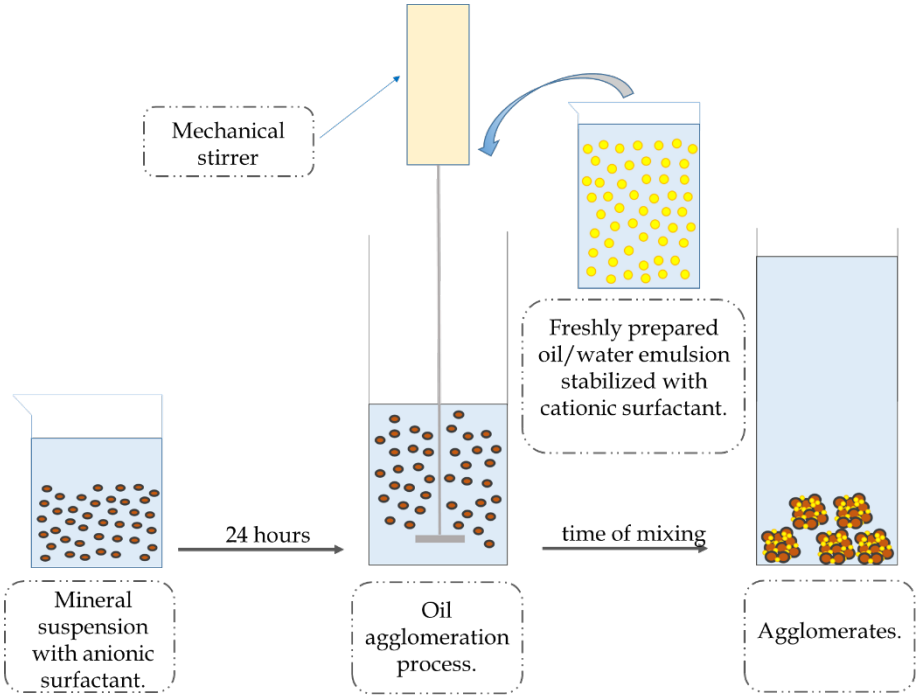

Fig. 3. The procedure of oil agglomeration of dolomite in the presence of anionic and cationic surfactants

2.2.2. Neural model implemented for oil agglomeration process

2.2.2.1. Mathematica background of Radial Basis Function Neural Network

As mentioned in the introduction, the effective operation of the neural model is possible if the data space is divided according to the properties of the samples. Neural networks based on perceptrons – typically with activation functions defined as a hyperbolic tangent – perform data separation globally – through appropriate combinations of shifted sigmoid functions (it creates the Gaussian functions). In the case of radial networks, such data analysis is performed directly (Fig. 4a). Each neuron of the hidden layer represents a cluster of the input data space. Global mapping of the data space is achieved through the superposition of the hidden layer output signals in the output layer neuron (Zhang et al., 2017; Meng et al., 2018).

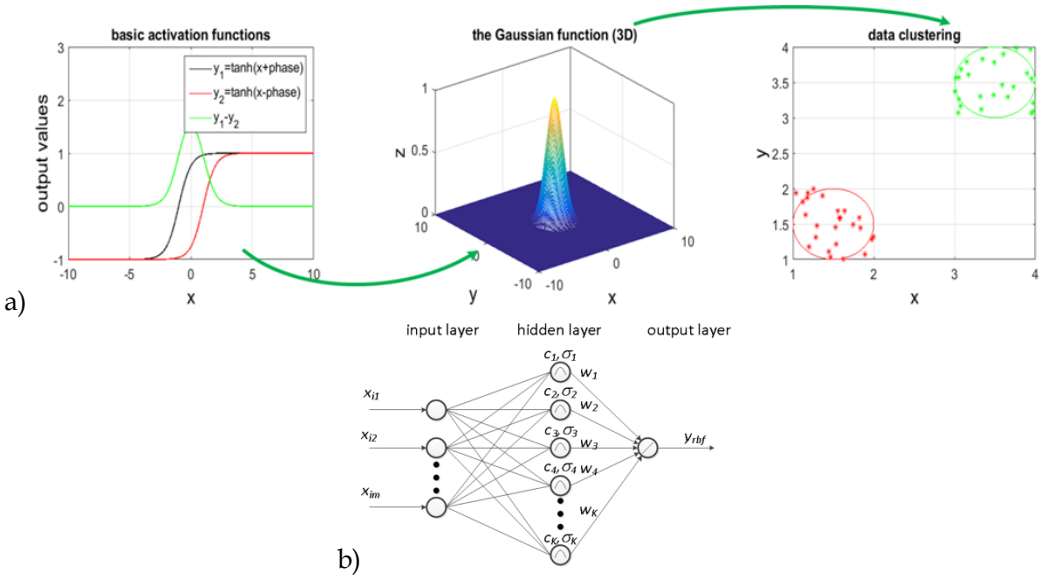

Fig. 4. The concept of data classification (a) and the structure of the RBFNN (b)

Radial Basis Function Neural Networks are a special case of feedforward neural networks. They consist of three layers: an input layer, a hidden layer, and an output layer. The general structure of such a network is shown in Fig. 4b. The output layer nodes have linear activation functions. Radial activation functions of the hidden layer neurons are characterized by their centers c and widths σ. Most commonly the Gaussian functions are used. The output of such neurons can be calculated using Eq. 1 (Yu et al. 2014):

$$\varphi_k(X_{in}) = exp\left(-\frac{\|X_{rbf}-c_k\|^2}{\sigma_k}\right) \quad (1)$$

where φ_k is the output of k-th radial node, $X_{rbf} = [x_{11}, x_{12}, x_{13}, \ldots, x_{im}]$ is the input vector assuming m inputs and i number of samples, $\|X_{rbf} - c_k\|$ is the Euclidean distance.

The output neuron implements the weighted sum of the outputs of the hidden layer neurons. The output y_{rbf} of the RBFNN with K neurons in the hidden layer is described with the expression:

$$y_{rbf} = \sum_{k=1}^{K} w_k \varphi_k + b_0 = \sum_{k=1}^{K} w_k \, exp\left(-\frac{\|X_{rbf}-c_k\|^2}{\sigma_k}\right) + b_0 \quad (2)$$

During the adaptation of the model to the assumed task, the number of radial neurons and their parameters should be determined, and then the calculation of coefficients between the hidden layer and the output should be carried out.

2.2.2.2. Growing Neural Gas Network

The basic assumptions related to data representation using the neural gas algorithm were presented in the literature (Martinetz and Schulten, 1991). Then, based on this theory and a competitive Hebbian learning method, the neural network was proposed (Fritzke, 1995). It allows achieving a varying node topology a representation of data. The training is realized in unsupervised mode. This means that the output reference data are not needed. Additionally, the cost function and the calculations of derivatives are omitted. The proposed neural network is a model with a varying structure. It starts with a small connection of two elements, and is being expanded in subsequent iterations. The topology is adapted to accurately map the training data.

Each node in the network is described by its position h_k. Edges are used to define the neighborhoods of points in the net. The data analysis process is initialized by inserting a few, usually two elements at random positions. After presenting a certain input vector x, the position update is only applied to the closest node and the points in its direct topological neighborhood. Then adaptation of h_k is performed and an update of the present state of the network connection is realized. The processing is implemented in iterative mode (Sun et al., 2017). As a stopping criterion, following arguments can be assumed:

number of iterations, additional error calculated for the analyzed signals, number of nodes, etc. Details of the GNGN adaptation are presented below.

1. Define the calculation conditions (number of iterations, fixed training coefficients, level beyond which the link is removed, etc.).
2. Initialize two nodes *a* and *b*, set their positions randomly to h_a and h_b so that $h_a, h_b \in R^{in}$.
3. Analysis of the input vector x, $x \in R^{in}$.
4. Find the closest neuron h_1 and the second closest neuron h_2. This can be done using the Euclidean distance (Eqs. 3, 4 and 5):

$$\text{dist}_l = \|x - h_l\|^2 \quad \text{for} \quad l = 1{:}H \tag{3}$$

$$h_1 = \arg\min(\text{dist}_l) \quad \text{for} \quad l = 1{:}H \tag{4}$$

$$h_2 = \arg\min(\text{dist}_l / h_l) \quad \text{for} \quad l = 1{:}H \tag{5}$$

where H is number of all actual neurons, l is currently analysed node.

5. Update the error value for the best neuron according to Eq. 6:

$$err_{h_1} = err_{h_1} + \Delta err_{h_1} = err_{h_1} + dist_1 \tag{6}$$

6. Adaptation of the winner neuron and elements connected with Eqs. 7 and 8:

$$h_1(k_{gngn} + 1) = h_1(k_{gngn}) + \Delta h_1(k_{gngn}) = h_1(k_{gngn}) + \varsigma_1(x - h_i) \tag{7}$$

$$h_n(k_{gngn} + 1) = h_n(k_{gngn}) + \Delta h_n(k_{gngn}) = h_n(k_{gngn}) + \varsigma_n(x - h_n) \tag{8}$$

where n means n neurons connected with h_1, k_{gngn} is number of iteration in the GNGN processing.

7. Increment the age of all the edges in the network.
8. Set the edge between h_1 and h_2 to zero. If they are not connected by the edge, create one.
9. Remove any edges, which age exceeds the maximal age. Delete any nodes that are not connected by any edge.
10. Check if the current iteration is an integer multiple of initially defined, insert an additional node with the rules described below.

10.1. An additional neuron h_{new} should be implemented between the node with the maximal value of the error h_{max} and other connected node with $h_{c\,max}$ the highest error as seen in Eq. 9:

$$h_{new} = 0.5(h_{max} + h_{c\,max}) \tag{9}$$

10.2. Create an edge between h_{new}, h_{max} and $h_{c\,max}$, moreover the connection between h_{max} and $h_{c\,max}$ has to be removed.

10.3. Decrease err_{hmax} and err_{hcmax} errors by multiplying them by a constant value γ, set the error $err_{hnew} = err_{hmax}$ (value after actualization).

11. Decrease all error values by multiplying them by a constant value β.
12. If one of the stopping criterions is not fulfilled, return to step 2.

In Fig. 5 several stages of data calculations using the Growing neural gas network are presented. The State variables of the Lorenz attractor are used for the test (Kaminski, 2019). Whole training takes 120 iterations. After each 30 of them, transients are shown. The final number of neurons is 100, however the number of samples inserted in input vector was 186. As it was explained above, the structure starts with two nodes, in the first iteration another one is added. Apart from changes in the number of neurons, corrections in connections are also visible. It should be highlighted that the achieved result at the end of calculation is not an ideal representation of the input samples, the network is active in the areas where data with similar properties are collected.

2.2.2.3. Concept of the RBFNN model training

The training of the analyzed neural model is composed of two main stages. The first of them is related to the structure selection – the number of radial nodes and the placement in the numerical space. The second task deals with the output weights determination. For the centers selection: randomization, optimization using gradient methods, or nature-inspired algorithms can be applied. Moreover, analyzing the theory and implementations of the RBFNN, it can be concluded that these algorithms have elements that can be interpreted as data separation. Thus, the above observation is often used for the selection of some methods used in the training process of the radial network. According to this,

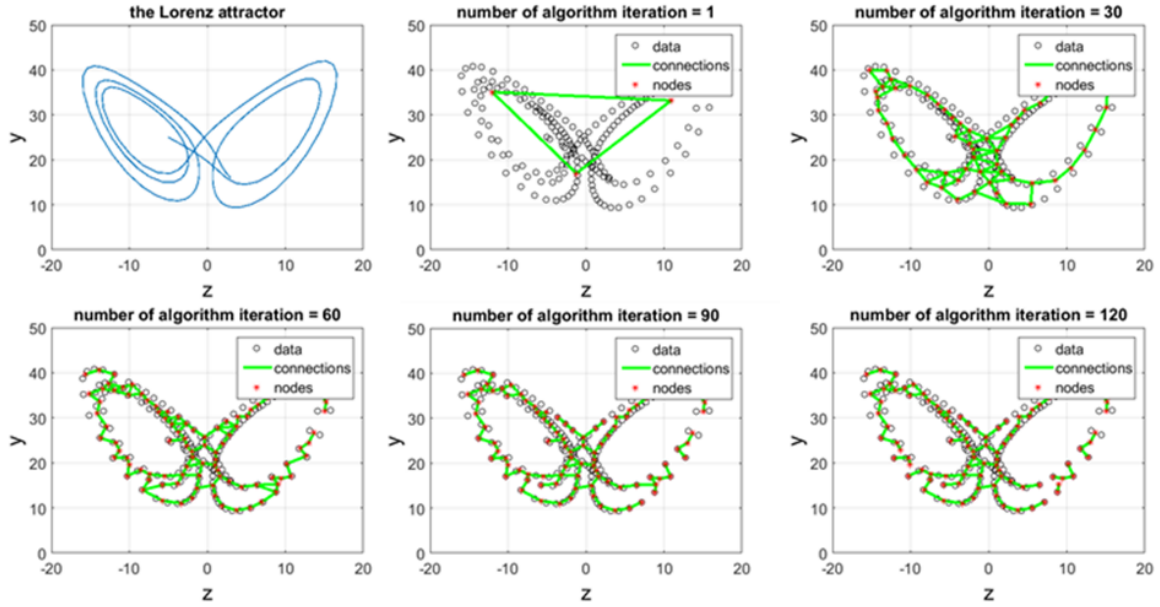

Fig. 5. Example of processing using the GNGN

clustering techniques are useful methods often implemented for centers values or also the calculation of widths (Sarimveis et al., 2003; Yang et al., 2009). The described above issue is a significant difference between the training of the RBFNN and the MLPs.

In neural networks applied in the work described in this study, the Growing Neural Gas Network was used for the RBFNN adaptation to minerals process simulation. These models (GNGN) have high data reproductive abilities. They can modify the structure according to the analyzed values. Due to the regular deployment of nodes after training, it can be also base on a data clustering (Qin and Suganthan, 2004). The data collected in the laboratory experiment was analyzed using the GNGN in the first part of the RBFNN training. The values of nodes h_l have been directly assigned to the centers c_k of the radial network. It should be noted that the number of nodes was not defined. The algorithm automatically selects this quantity using its internal stopping criteria. It was achieved after a slight modification. Additional verification of error was performed (Fritzke, 1994). The level is defined by the user. However, the assignation of a certain value is not problematic due to the fast stabilization of this transient (error in subsequent iterations).

If the calculation of the neural network centers is correctly carried out, the superposition of values through the output neuron can lead to the effective operation of the neural model. For the RBFNN, two solutions are dominant. Calculations of cost function derivatives according to weights, including the second order methods, are commonly implemented (Cecati et al., 2015). Another way is based on calculation of the output layer matrices (Bishop, 1994). For simplicity of the analysis, the equation describing the output of the neural network should be presented as matrix formula seen in Eq. 10.

$$y_{rbf} = \sum_{k=1}^{K} w_k \varphi_k + b_0 = \sum_{k=1}^{K} w_k \, exp\left(-\frac{\|X_{rbf} - c_k\|^2}{\sigma_k}\right) + b_0 = W^T \Phi \tag{10}$$

where W is a weight matrix, Φ is matrix with the values of the hidden layer.

Adaptation assumes modification of the weights at each step k of calculations (Eq. 11):

$$W(k+1) = W(k) + \Delta W(k) \tag{11}$$

It should be realized for the minimization of the error. Thus, the cost function is defined as Eq. 12.

$$E = \frac{1}{2}(y_{ref} - y_{rbf})^2 \tag{12}$$

where y_{ref} is the reference value for the neural network.

Updates should be done in opposite to the gradient (Eq. 13):

$$\Delta W(k) = -g(k) \tag{13}$$

described using the expression presented below as Eq. 14:

$$g = \frac{\partial E}{\partial W} = -\left(y_{ref} - y_{rbf}\right)\frac{\partial (y_{rbf})}{\partial W} = -e_{rbf}\Phi \tag{14}$$

Combining the above equations, Eq. 15 presents the adaptation of weights in the output layer is achieved:

$$W(k+1) = W(k) - a_{rbf}\,g(k) = W(k) + a_{rbf}\,e_{rbf}\Phi(k) \tag{15}$$

where y_{rbf} is the learning rate.

One can assume that after a suitable time the error tends to zero ($k \to optimum$, $e_{rbf} \to 0$). Then, it can also be concluded that the weight adjustments, for such conditions, are slight ($g \to 0$). Therefore, Eq. 16 can be formed:

$$\frac{\partial E}{\partial W} = \left(y_{ref} - y_{rbf}\right)\Phi^T = (W^T\Phi - y_{ref})\Phi^T \tag{16}$$

after a simple transformation, Eqs. 17 and 18 describing the changes of matrix W is obtained:

$$W^T(\Phi^T\Phi) = \Phi^T y_{ref} \tag{17}$$

$$W = (\Phi^T\Phi)^{-1}\Phi^T y_{ref} \tag{18}$$

Eqs. 15 and 18 were applied for training of the neural network used for the model of the oil agglomeration process.

3. Results and discussion

The solution proposed by (Bastrzyk et al., 2011), where kerosene, a bridging liquid, was introduced to the system as an emulsion can significantly lower the cost of the process of oil agglomeration of hydrophilic particles. The amount of kerosene was reduced twice compared to the literature data (Ozkan et al., 2005). In such a system, large and mechanically strong agglomerates were formed with less kerosene. To predict the behavior of the oil agglomeration of dolomite particles in such a system in terms of the process recovery, the combination of the RBFNN and GNGN was used.

Based on the rules of calculations described earlier in this article, the software (simple application) has been developed. The code was prepared without additional external libraries. Moreover, only basic commands were used. This approach enables quick conversion of the algorithm between different programming languages. It was successfully tested using Python, Matlab, and Octave. The whole calculations, from start with neural training, tests, and closing with plots, took 19.63 sec.

For the correct processing of the neural network, the operation of the signals should be computed in terms of the most variable range of activation function. Otherwise, the variables in the network can get stuck as almost constant numbers. The output level, in this case, cannot achieve the desired state (Fig. 6a). This problem appears often in the calculation of real data. Samples have to be scaled before training. In this work, both groups (Fig. 6b) or only the input values (Fig. 6c) were divided by the maximal number of several sets. It was an effective operation, the results for neural network calculations of the normalized data are similar, and the results are repeatable.

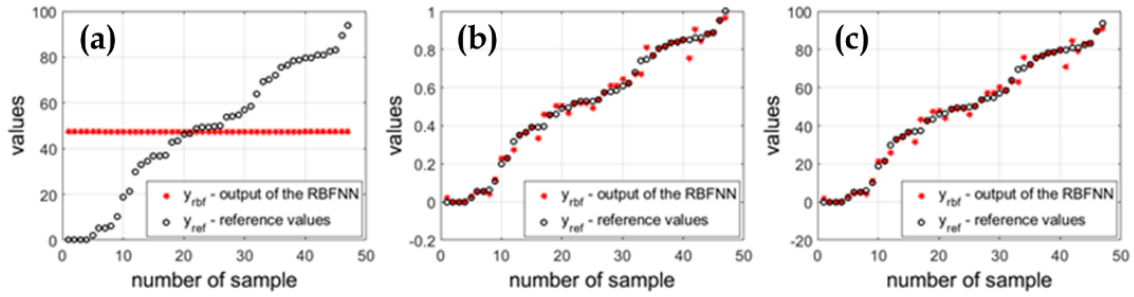

Fig. 6. Effect of data normalization – direct values from measurement (a), normalization of input data (b) and outcome for all scaled samples (c)

Both methods (Eqs. 15 and 18) of weights optimization were considered in this application. Comparing the gradient method (Fig. 7a) and the matrix transformations (Fig. 7b), the second one, in this case, allows achieving higher precision. The error was calculated using Eq. 12. For the first method, the value was 0.1301, and for the results presented in Fig. 7b, it was equal to 0.0194. For a relatively small number of analyzed samples, a direct mathematical transformation of the equation describing the

output of the neural network is useful (if the size of matrices is high, the problem with determining the inverse matrices can be found). An additional selection of elements like training constant or momentum is not necessary. The influence of the learning rate is significant for the time of calculation and the final results (Fig. 8a). It determines the step of optimization. A low value can lead to long-time calculations, but a high coefficient may make it impossible to obtain the optimum of the cost function. In the extreme case, it is possible to lose the convergence of the training algorithm (Kaminski and Orlowska-Kowalska, 2015).

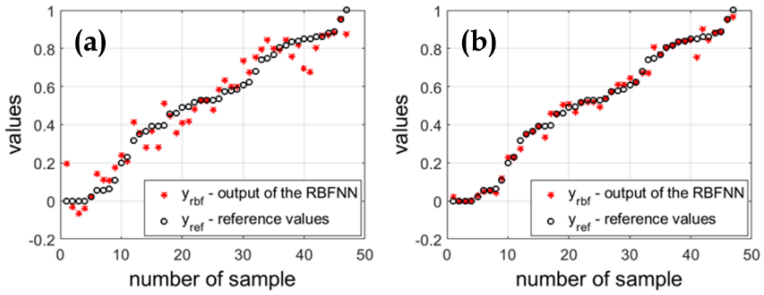

Fig. 7. Influence of method applied for weight optimization on final results

The sigma can be treated as a scaling factor for the argument of the Gaussian function. In the classical version, this element multiplies the input. According to Eq. 2 it is in the denominator. This parameter shapes the width of the activation function (Fig. 8b). It influences the work of the whole network. In this study, it was not optimized, but selected arbitrarily. However, the dependencies between the error, the number of nodes, and the σ parameter were analyzed.

Simplifying, for each value of a constant number of nodes the transient has the shape of a parabola (Fig. 9). It means that there is an optimal value of the width. If the value is too small, the function performs data separation in a narrow area. This can introduce precision to the data processing, but the entire data set cannot be classified. If the function is too wide, the algorithm does not calculate precisely. Increasing the number of neurons (in tested range) increased the quality of calculations. However, if the appropriate number is exceeded, the generalization properties can be low. Then, the net only works correctly for the training data.

The repeatability of the optimization algorithm is an important element when evaluating the results. In the tested model, the initial values in the GNGN model were selected as randomly chosen elements from the dataset. This introduces a different starting point for subsequent calculations. At this point in the study, the impact of the above action on the final results obtained using the RBFNN model was analyzed. In order to clear the presentation of the results, the selected test values are presented in Table 1. Only minor differences can be noticed.

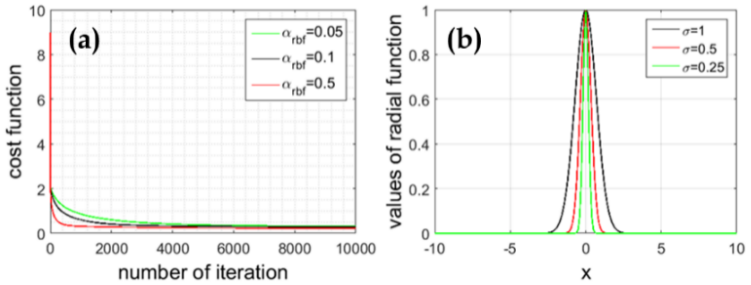

Fig. 8. Transient of error values during the training of neural networks – the influence of the learning rate (a). The activation function of the hidden layer in the RBFNN (b)

Table 1. Results achieved for three starting points of the GNGN

y_{ref}	0.3670	0.5729	0.6077	0.8486	0.7507
y_{rbf} (test 1)	0.3667	0.5217	0.6462	0.8485	0.7543
y_{rbf} (test 2)	0.3688	0.5217	0.6470	0.8485	0.7563
y_{rbf} (test 3)	0.3688	0.5217	0.6467	0.8486	0.7542

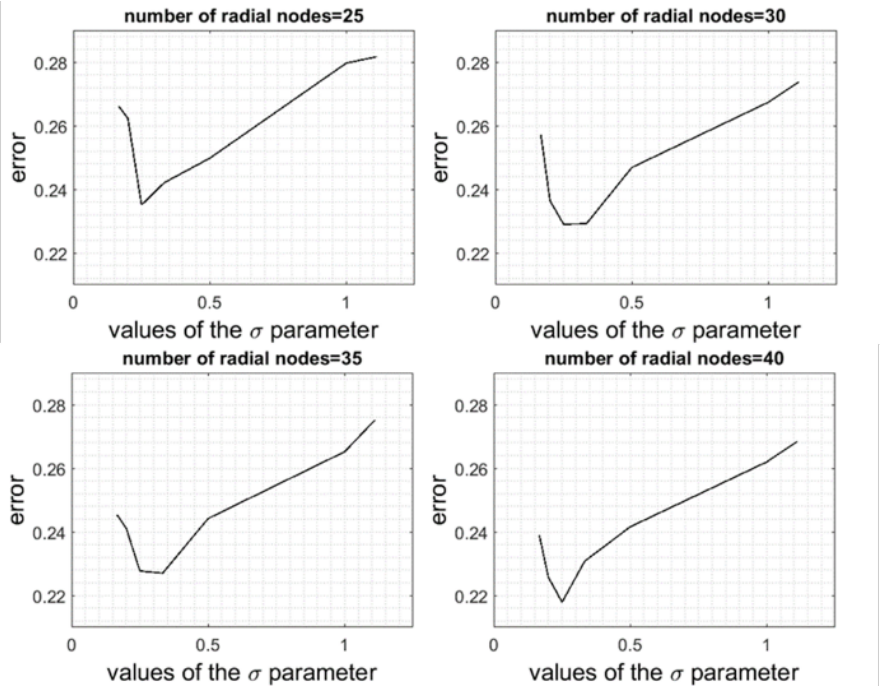

Fig. 9. Influence of width applied for the Gauss function and number of nodes on quality of calculations

Table 2. Comparison of the results (lab – the data obtained in the laboratory, NN – the data obtained from the RBFNN-GNGN model)

Mixing intensity [rpm]	Time of mixing [s]	NaOL [mg/g]	DDAHCL [mg/g]	Kerosene [cm³/g]	pH	R_{lab} [%]	R_{NN} [%]
200	600	11.3	14.6	0.1	10	0	0.2
2000	600	11.3	14.6	0.07	10	9.4	14.1
1500	600	11.3	14.6	0.1	10	42.5	45.1
2500	600	11.3	14.6	0.1	10	80.0	78.6
2000	600	11.3	19.5	0.1	10	80.9	72.7

In the previous test, the testing data was generated using disturbances in training samples (uniformly distributed random numbers were added). Moreover, the created model RBFNN-GNGN was validated using four new experiments, which were not used in the training of the neural networks. The data are presented in Table 2; and confirm the convergence of the values obtained from the RBFNN-GNGN model with that obtained in the laboratory. The difference between the values was changing from 0.2 to 8.2%. This means that the considered neural network can be successfully used to model the agglomeration process.

4. Conclusions

This study aimed to find the effective implementation of the Radial Basis Function Neural Network as a model presenting the features of a chemical process. The application of neural modeling, using the RBFNN, to a simulation of the oil agglomeration of dolomite can be treated as original. The results in this study present the procedure of creating a model that does not directly require any mathematical equations (the parameters between input and output are not described by any formula). The properties are estimated from the measurement data. Thus, the analyzed tools seem to be competitive for the classic methods. The prepared application can be easily used for modeling of other processes (only new data need to be applied). Implementation of the GNGN at the design stage enables the activation function centers to be determined. It also selects the number of nodes. It is a solution to a common problem related to neural network applications. During the tests, the relationships known from the theory of

neural networks were analyzed on a practical example (influence of training parameters on the convergence of adaptation, data normalization, etc.). The recovery was the output of the model. High precision of calculation was obtained. The difference between the experimental data and the predicted one was less than 10%. The obtained results showed that the proposed RBFNN-GNGN model can be effectively used to predict, and in the future to optimize the performance of mineral processing, such as oil agglomeration or flotation.

Acknowledgments

The work was funded by subsidy from the Polish Ministry of Science and Higher Education for the Faculty of Chemistry (A.B.) and Faculty of Electrical Engineering (M.K.) of Wroclaw University of Science and Technology.

References

ALLAHKARAMI, E., SALMANI NURI, O., ABDOLLAHZADEH, A., REZAI, B., MAGHSOUDI, B., 2017. *Improving estimation accuracy of metallurgical performance of industrial flotation process by using hybrid genetic algorithm – artificial neural network (GA-ANN)*. Physicochem. Probl. Miner. Process. 53, 366−378.

ASMOLOV, T., GALIN, R., *Study of the effectiveness of combinatorial protection algorithms based on the hardware and software of the electronic storage of corporate information systems*. IOP Conf. Ser.: Mater. Sci. Eng. 537, 052027, 1-6.

BASTRZYK, A., POLOWCZYK, I., SADOWSKI, Z., SIKORA, A., 2011. *Relationship between properties of oil/water emulsion and agglomeration of carbonate minerals*. Sep. Purif. Technol. 77, 325-330.

BISHOP, C.M., 1994. *Neural networks and their applications*. Rev. Sci. Instrum. 65, 1803-1832.

CECATI, C., KOLBUSZ, J., RÓŻYCKI, P., SIANO P., WILAMOWSKI, B. M., 2015. *A novel RBF training algorithm for short-term electric load forecasting and comparative studies*. IEEE Trans. Ind. Electron. 62, 6519-6529.

CHAKLADAR, S., BANERJEE, R., MOHANTY, A., CHAKRAVATRY, S., KUMAR PATAR, P., 2019. *Turpentine oil: a novel and natural bridging liquid for oil agglomeration of coal fines of high ash coal*. Int. J. Coal Prep. Util. DOI:10.1080/19392699.2020.1789976 (in press).

CISTERNAS, L.A., LUCAY, F.A., BOTERO, Y.L., 2020. *Trends in modeling, design, and optimization of multiphase systems in minerals processing*, Minerals 10, 1-28.

DIAS, W.P.S., POOLIYADDA, S.P., 2001. *Neural networks for predicting properties of concretes with admixtures*. Constr. Build. Mater. 15, 371-379.

DRZYMALA, J., 2007. *Mineral processing. Foundations of theory and practice of minerallurgy*. Ofic. Wyd. PWr, Wroclaw, Poland.

DUZYOL, S., OZKAN, A., 2010. *Role of hydrophobicity and surface tension on shear flocculation and oil agglomeration of magnesite*. Sep. Purif. Technol. 72, 7-12.

FAHLMAN, S., LEBIERE, C., 1997. *The cascade-correlation learning architecture*. Adv. Neur. Inf. Process. Syst. 2, 524-532.

FAUSETT, L.V., 1994. *Fundamentals of neural networks: architectures, algorithms and applications*. Prentice-Hall, United States.

FRITZKE, B., 1995. *A growing neural gas network learns topologies*. Adv. Neur. Inf. Process. Syst. 7, 625-632.

FRITZKE, B., 1994. *Fast learning with incremental RBF networks*. Neural Process. Lett. 1, 2-5.

GUAN, W., SHA, J., LIU, P., PENG, Y., XIE, G., 2018. *Effect of stirring time on oil agglomeration of fine coal*. J. S. Afr. I. Min. Metall. 8, 89-94.

HASSIBI, B., STORK, D.G., WOLF, G.J., 1993. *Optimal brain surgeon and general network pruning*. IEEE International Conference on Neural Networks, San Francisco, CA, USA, 1, 293-299.

HUAN, A.Y., BERG, J.C., 2003. *Gelation of liquids bridges in spherical agglomeration*. Colloid. Surf. A: Physicochem. Eng. Aspect. 215, 241-252.

KAMINSKI, M., 2019. *Neural network training using Particle Swarm Optimization - a case study*. 24th International Conference on Methods and Models in Automation and Robotics (MMAR), Międzyzdroje, Poland, 115-120.

KAMINSKI, M., BASTRZYK, A., 2018. *Modelling of oil agglomeration of dolomite by the use of an artificial neural network optimized using Optimal Brain Damage algorithm*. IOP Conf. Ser.: Mater. Sci. Eng. 427, 012012, 1-9.

KAMINSKI, M., ORLOWSKA-KOWALSKA, T., 2015. *Adaptive neural speed controllers applied for a drive system with an elastic mechanical coupling – a comparative study*. Eng. Appl. Artif. Intell. 45, 152-167.

KAYA, Ö., ARI, M., 2019. *The effect of novel bridging liquid mixtures on lignite enrichment using the oil agglomeration process*. Int. J. Coal. Prep. Util. 39(6), 293-301.

LASKOWSKI, J., ZHU, Z., 2000. *Oil agglomeration and its effect on beneficiation and filtration of low-rank/oxidized coals*. Int. J. Miner. Process. 58, 237-252.

MARTINETZ, T., SCHULTEN, K., 1991. *A "neural gas" network learns topologies*. Artificial Neural Networks, 397–402.

MENG, X., ROZYCKI, P., QIAO, J., WILAMOWSKI, B.M., 2018. *Nonlinear system modeling using RBF Networks for industrial application*. IEEE Trans. Industr. Inform. 14, 931-940.

MOLINA, C.G., ZERUBIA, J., 2000. *Regularisation by convolution in probability density estimation is equivalent to jittering.* Neural Networks for Signal Processing X. Proceedings of the 2000 IEEE Signal Processing Society Workshop (Cat. No.00TH8501), Sydney, NSW, Australia, 1, 204-210.

JAMRÓZ, D., NIEDOBA, T., PIĘTA, P., SUROWIAK, A., 2020. *The use of neural networks in combination with evolutionary algorithms to optimize the copper flotation enrichment process*. Appl. Sci. 10(9), 3119.

OZKAN, A., AYDOGAN, M., YEKELER, M., 2005. *Critical solution surface tension for oil agglomeration*. Int. J. Miner. Process. 76, 83-91.

PIETSCH, W., 2005. *Agglomeration in Industry. Occurrence and Application.* Willey-Vch Verlag GmbH&Co. KGaA, Weinhein.

POLOWCZYK, I., KRUSZELNICKI, M., KOWALCZUK, P.B., 2018. *Oil agglomeration of metal-bearing shale in the presence of mixed cationic-anionic surfactants*. Physicochem. Probl. Miner. Process. 54(4), 1052-1059.

QIN, A.K., SUGANTHAN, P.N., 2004. *Robust growing neural gas algorithm with application in cluster analysis*. Neural Networks 17, 1135–1148.

RIBEIRO, T.S., GROSSI, C.D., MERMA, A.G., DOS SANTOS, B.F., TOREM, M.L., 2019. *Removal of boron from mining wastewaters by electrocoagulation method: modelling experimental data using artificial neural networks*. Miner. Eng. 131, 8-13.

SADOWSKI, Z., 1995. *Selective spherical agglomeration of fine salt-type mineral particles in aqueous solution*. Colloid. Surf. A: Physicochem. Eng. Aspect. 96, 277-285.

SADOWSKI, Z., 2000. *The role of surfactant salts on the spherical agglomeration of hematite suspension*. Colloid. Surf. A: Physicochem. Eng. Aspect. 173, 211-217.

SARIMVEIS, H., ALEXANDRIDIS, A., BAFAS, G., 2003. *A fast training algorithm for RBF networks based on subtractive clustering*. Neurocomputing 51, 501-505.

SHOKRY, A., ESPUÑA, A., 2018. *The ordinary kriging in multivariate dynamic modelling and multistep-ahead prediction*. Comput. Aided Chem. Eng. 43, 265-270.

SHUKLA, D.; VENUGOPAL, R., 2019. *Optimization of the process parameters for fine coal-oil agglomeration process using waste mustard oil*. Powder. Technol. 346, 316-325.

SUN, Q., LIU, H., HARADAC, T., 2017. *Online growing neural gas for anomaly detection in changing surveillance scenes*. Pattern Recognit. 64, 187-201.

SÖNMEZ, İ., CEBECI, Y., 2003. *A study on spherical oil agglomeration of barite suspensions*. Int. J. Miner. Process. 71, 219-232.

VACHKOV, G., 2004. *Growing model algorithm for process identification based on neural-gas learning and local linear mapping*. Fourth International Conference on Hybrid Intelligent Systems (HIS'04), Kitakyushu, Japan, 222-227.

VAN NETTEN, K., BORROW, D.J., GALVIN, K.P., 2020. *Ultrafast plug flow agglomeration – exploiting hydrophobic interactions via a concentrated water-in-oil-emulsion binder*. Minerals 10, 506.

WU, X., LIU, J., 2009. *A new early stopping algorithm for improving neural network generalization*. 2009 Second International Conference on Intelligent Computation Technology and Automation, Changsha, Hunan, 15-18.

YANG, P., ZHU, Q., ZHONG, X., 2009. S*ubtractive clustering based RBF neural network model for outlier detection*. J. Comput. 4, 755-762.

YASAR, O., USLU, T., 2019. *Use of agglomeration for fine coal recovery from washery tailings*. Int. J. Coal Prep. Util. DOI:10.1080/19392699.2019.1705287 (in press).

YU, H., REINER, P. D., XIE, T., BARTCZA, T., WILAMOWSKI, B.M., 2014. *An incremental design of radial basis function networks*. IEEE Trans. Neural Netw. Learn. Syst 25, 1793-1803.

ZHANG, P., ZHOU, X., PELLICCIONE, P., LEUNG, H., 2017. *RBF-MLMR: a multi-label metamorphic relation prediction approach using RBF neural network*. IEEE Access 5, 21791-21805.