Available online at www.sciencedirect.com

SCIENCE DIRECT®

Chemometrics and Intelligent Laboratory Systems 75 (2005) 149–162

Chemometrics and intelligent laboratory systems

www.elsevier.com/locate/chemolab

A two-stage evolutionary algorithm for variable selection in the development of RBF neural network models

Alex Alexandridis[a], Panagiotis Patrinos[a], Haralambos Sarimveis[a,*], George Tsekouras[b,1]

[a]School of Chemical Engineering, National Technical University of Athens, 9, Heroon Polytechniou str., Zografou Campus, Athens 15780, Greece
[b]Laboratory of Multimedia Applications, Department of Cultural Technology and Communication, University of the Aegean, Faonos & Harilaou Trikoupi Str., GR-81100, Mytilene, Greece

Received 15 October 2003; received in revised form 13 June 2004; accepted 14 June 2004
Available online 17 August 2004

Abstract

In many modeling problems that are based on input–output data, information about a plethora of variables is available. In these cases, the proper selection of explanatory variables is very critical for the success of the produced model, since it eliminates noisy variables and possible correlations, reduces the size of the model and accomplishes more accurate predictions. Many variable selection procedures have been proposed in the literature, but most of them consider only linear models. In this work, we present a novel methodology for variable selection in nonlinear modeling, which combines the advantages of several artificial intelligence technologies. More specifically, the Radial Basis Function (RBF) neural network architecture serves as the nonlinear modeling tool, by exploiting the simplicity of its topology and the fast fuzzy means training algorithm. The proper variables are selected in two stages using a multi-objective optimization approach: in the first stage, a specially designed genetic algorithm minimizes the prediction error over a monitoring data set, while in the second stage a simulated annealing technique aims at the reduction of the number of explanatory variables. The efficiency of the proposed method is illustrated through its application to a number of benchmark problems.
© 2004 Elsevier B.V. All rights reserved.

Keywords: Variable selection; Radial basis functions; Neural networks; Evolutionary computation; Genetic algorithms; Simulated annealing

1. Introduction

Due to the always increasing computational power and the reducing cost for collecting and storing information, many modeling problems nowadays depend on the analysis and proper manipulation of available data. The most typical examples are the so-called Distributed Control Systems (DCS) that are utilized by most modern industries, which enable them to store frequent measurements of hundreds or even thousands of variables that are involved in the various manufacturing processes. Information about large numbers of explanatory variables is also available in many other scientific areas, such as chemometrics, Quantitative Structure Activity Relationship (QSAR) analysis, decision sciences, pattern recognition, etc. A common feature of all the aforementioned paradigms is that usually only a small subset of the recorded variables carries essential information for building a mathematical model of the system. It is feasible to build models including all the variables but there are many reasons to avoid this:

- Irrelevant variables may add extra noise which deteriorates the accuracy of the model.
- They can cloud meaningful relationships that exist between important variables.
- They may provide unreliable parameter estimates for the model since the number of observations should be greater than the number of variables.
- They increase the complexity and number of parameters of the model.

* Corresponding author. Tel.: +30 210 7723237; fax: +30 210 7723138.
E-mail addresses: hsarimv@chemeng.ntua.gr (H. Sarimveis), gtsek@ct.aegean.gr (G. Tsekouras).
[1] Tel.: +30 2251 036631; fax: +30 2251 036609.

- Due to the possible correlation between some variables, redundant information may pass through the model. Investigations have clearly shown the degradation of the prediction results when correlation is present [1].

Therefore, in order to build a model that can predict a specific process output, it is desirable to select a subset of variables that are truly relevant to this output. This procedure is typically called variable selection, and it corresponds to finding a subset of the full set of recorded variables that exhibits good predictive abilities.

A solution to the variable selection problem could be the utilization of prior knowledge in order to screen out the irrelevant variables. However, in most cases, the large number of variables and the complexity of the process dictate the application of more sophisticated methods. A more advanced approach is to consider the variable selection problem as an optimization procedure, where the objective is to minimize the error between the true values and the model predictions of the explained (output) variables, by selecting the proper explanatory (input) variables. Thus, a variable selection methodology is composed of three ingredients [2]:

(1) A search algorithm.
(2) An objective function.
(3) An algorithm that models the relationship between the input and output variables.

The first ingredient defines the procedure that is utilized to search the space of possible solutions, which is in essence an optimization algorithm. This category of problems is classified as NP-hard, because the complexity increases exponentially with the number of variables. For problems involving a small number of candidate variables, it is possible to evaluate all possible combinations. However, this approach is unattainable for problems where the number of variables is large, given that the possible combinations are 2^N-1, where N is the number of variables. The NP-hardness of the problem indicates the need of employing approximation algorithms for its solution.

Several approaches for variable selection have been introduced in the literature. These techniques can be grouped in three categories [3], according to the search method they employ. The first category is called dimension-wise selection and the distinctive feature of these algorithms is that they add each predictor progressively according to some criterion. These methods include forward selection [4], interactive variable selection (IVS-PLS) [5], automatic variable selection (AVS-PLS) [6], cyclic subspace regression [7], the successive projections algorithm [8], etc.

The second category includes the model-wise elimination algorithms which perform the inverse procedure: The regression model initially takes into account all the variables, and then, unimportant variables are pruned according to some criterion. Algorithms that belong to this category are backward elimination [4], uninformative variable elimination in PLS modeling (UVE-PLS) [9], iterative predictor weighting PLS (IPW-PLS) [3], etc.

Though the most frequently applied variable selection techniques so far are forward selection and backward elimination, these methods share a common disadvantage: they fail to select those input variables that seem to be of no value when applied alone, but offer useful information combined together. The forward selection method adds variables until either a specific criterion has been minimized or all the variables have been selected, but once a variable has been included in the model, it may not be removed. Hence, the search space is not sufficiently explored and this method cannot guarantee the optimality of the subset of variables. The same drawback characterizes the backward elimination method, where once a variable is removed from the model, it cannot be reinserted.

This serious disadvantage is eliminated by the third category of algorithms, called subset selection algorithms. The general procedure followed by these algorithms is to create different subsets of variables and evaluate their performance. Based on this evaluation, new subsets are generated from the existing ones. Most techniques that belong to this class [10–13] employ genetic algorithms (GAs) [14,15] or simulated annealing (SA) [16] as a search algorithm.

As far as the second ingredient (objective function) of different methodologies is concerned, a variable selection problem inherently involves two conflicting objectives: the minimization of the prediction error and the minimization of the number of selected variables. This means that the optimization algorithm must compromise between the model accuracy and parsimony. The simplest approach is to include two terms that express in some way each one of the aforementioned objectives, in the same objective function [17,18]. However, the strategy of combining multiple conflicting objectives into a single objective function may not yield the optimum results. Thus, specifically tailored multi-objective optimization techniques have been developed in order to identify the solution to such problems [19].

The last component of a variable selection methodology is the algorithm that models the correlation between the input and output variables. The model parameters are fitted to a set of training data and the model is usually evaluated on a different set. Given that this procedure is repeated several times, the speed of the modeling algorithm is very crucial for the efficiency and performance of the variable selection methodology.

Multiple linear regression (MLR), partial least squares (PLS) and principal component regression (PCR) [20] are some typical modeling techniques. They have been widely applied in chemometrics, especially in spectroscopic analysis (UV–Visible or near-infrared spectrometry) and QSAR studies. However, these are linear methods and thus are unable to model possible nonlinearities of the system under investigation. The fact that the majority of the industrial processes is nonlinear has led many researchers to propose some other methods, like nonlinear PLS [21,22] and nonlinear-PCA [23–25]. However, in most conventional nonlinear modeling techniques, the main disadvantage is that the type of nonlinearity must be predefined. This problem has been faced successfully by artificial neural networks (ANNs), which are generic frameworks that have shown great capabilities in modeling highly nonlinear systems.

ANNs are considered to be global nonlinear approximators. This means that provided that the network structure is sufficiently large, any continuous function can be approximated within an arbitrary accuracy by carefully choosing the parameters of the network. Feedforward neural networks (FNNs) constitute the most popular ANN architecture and have been used in variable selection methodologies [26,27]. In Ref. [26], model-wise elimination methods like modified version of Hinton diagrams (magnitude approach), weight saliencies estimation, variance propagation and partial modeling approach were applied using feedforward neural networks as process models. In Ref. [27], a special genetic algorithm is developed for the selection of time points for quantitative analysis of ternary mixtures. The authors employ multilayer feedforward neural networks as a modeling technique. The structure of the network (number of hidden nodes) is assumed to be fixed and the scaled conjugate gradient (SCG) is employed as a training method.

The main drawback of the FNN architecture is the lack of fast training algorithms, since variations of classical nonlinear optimization techniques (such as the backpropagation method) are usually utilized to train these networks. Radial Basis Function (RBF) neural networks form a different class of ANNs which has certain advantages, including better approximation capabilities, simpler network structures and faster learning algorithms. Surprisingly, very few publications can be found in the literature that apply RBF networks on the variable selection problem [28].

This paper presents a new variable selection methodology that employs the RBF neural network architecture and the fuzzy means training algorithm [29] in order to model nonlinear correlations between the input and output variables. The ability of the fuzzy means algorithm to determine both the structure and the parameters of the network, combined with the inherent speed of the method, enhances the variable selection procedure, which is completed in two steps: in the first step, the prediction error is optimized using a specially designed genetic algorithm, and in the second, the number of variables is minimized using the simulated annealing concept. Thus, the problem of compromising two different goals in the same objective function is avoided. The methodology, which is recommended especially for large-scale problems, is illustrated by the application to three real world benchmark variable selection problems found in the literature [30] and to a simulated variable selection example where the true input–output relationship is known.

The rest of this paper is organized as follows: In the next section, we present some theoretical aspects of the RBF network architecture, the fuzzy means training algorithm, genetic algorithms and simulated annealing, which are all ingredients of the variable selection method that is proposed in this work. The complete variable selection framework is presented in details in Section 3, followed by the application of the methodology to benchmark variable selection problems. The paper concludes by outlining the advantages of the proposed approach.

2. Theoretical background

2.1. The RBF neural network architecture

An RBF network can be considered as a special three-layer neural network, which is linear with respect to the output parameters after fixing all the radial basis function centers and nonlinearities in the hidden layer. The typical structure of an RBF network is shown in Fig. 1.

The input layer distributes the inputs to the L nodes of the hidden layer. Each node in the hidden layer is associated with a center, equal in dimension with the number of input variables. Thus, the hidden layer performs a nonlinear transformation and maps the input space onto a new higher-dimensional space. The output of the RBF network is produced by a linear combination of the hidden node responses, after adjusting the weights of the network appropriately.

The standard algorithm decomposes the training problem in two steps: in the first step, the parameters of the basis functions of the nodes are obtained using the k-means algorithm [31], which is an unsupervised clustering method; the second step involves the determination of the output-layer weights by linear least squares regression.

However, the k-means algorithm has three major drawbacks:

- Several passes of all training examples are required. This iterative procedure increases the computational effort, especially when a large database is available.
- The number of hidden nodes must be predefined by the user. This means that a trial and error procedure must be followed in order to obtain the optimum number of hidden nodes.
- It depends on an initial random selection of center locations, so that different sets of centers are obtained for different runs of the same network structure.

It is obvious that this technique lacks two basic features that are needed in order to incorporate it in a variable selection procedure: It is neither fast enough nor repetitive. In order to overcome these problems, we employed the

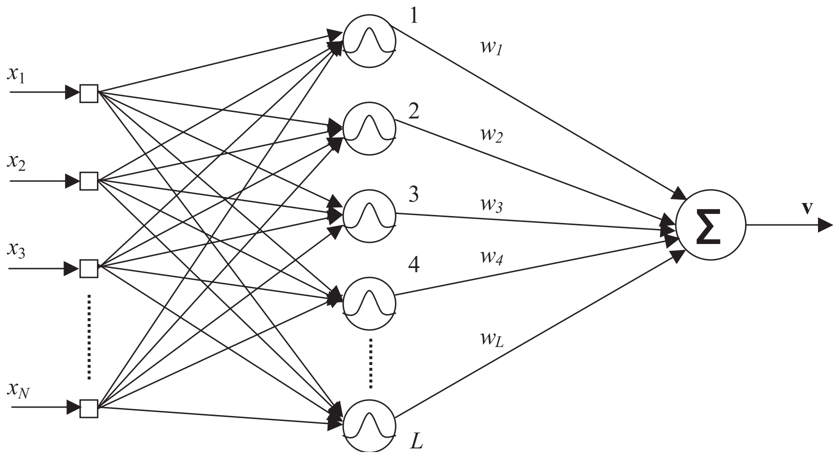


Fig. 1. Standard topology of an RBF neural network.

novel fuzzy means training algorithm [29], which is presented in the next subsection.

2.2. The fuzzy means algorithm

The fuzzy means algorithm is in essence an alternative to the *k*-means algorithm for determining the structure and the parameters of the hidden layer in an RBF network. Based on a fuzzy partition of the input space, the number and the locations of the hidden layer centers are automatically obtained in a single step, using only one pass of the training examples. Thus, the fuzzy means algorithm requires very short computational times, while at the same time it always produces the same results since it does not depend on an initial random selection of the center locations. A short description of the algorithm is given below, while the interested reader is referred to Ref. [29] for more details.

Assuming a system with N input variables, the universe of discourse (domain) of each variable x_i, $i\in 1,\ldots, N$ is partitioned into l_i triangular fuzzy sets:

$$\left\{A_{i,1}, A_{i,2}\ldots, A_{i,l_i}\right\}, 1\leq i\leq N \tag{1}$$

Supposing that $a_{i,j}$ is the center element of the fuzzy set $A_{i,j}$, and δa_i is half of the respective width, the membership function $\mu_{A_{i,j}}(x_i(k))$ of the input $x_i(k)$ to the fuzzy set $A_{i,j}$ is defined as:

$$\mu_{A_{i,j}}(x_i(k))=\begin{cases} 1-\dfrac{|x_i(k)-a_{i,j}|}{\delta a_i}, & \text{if } x_i(k)\in\left[a_{i,j}-\delta a_i, a_{i,j}+\delta a_i\right] \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

By employing the principle of fuzzy partitioning to the entire input space, we can dismember it into M fuzzy subspaces, where:

$$M=\prod_{i=1}^{N} l_i \tag{3}$$

Each fuzzy subspace $\mathbf{A}^m$ ($1\leq m\leq M$) can then be defined as a combination of N particular fuzzy sets and is represented as:

$$\begin{aligned}\mathbf{A}^m &= \left[A^m_{1,j_1}, A^m_{2,j_2},\ldots, A^m_{N,j_N}\right] \\ &= \left\{\left[a^m_{1,j_1}, a^m_{2,j_2},\ldots, a^m_{N,j_N}\right], \left[\delta a_1, \delta a_2,\ldots, \delta a_N\right]\right\}\Rightarrow\end{aligned} \tag{4}$$

$$\mathbf{A}^m=\{\mathbf{a}^m, \delta\mathbf{a}\} \tag{5}$$

An example is shown in Fig. 2, which represents a fuzzy partition in the two-dimensional input space that defines 25 fuzzy subspaces. The particular fuzzy subspace $\boldsymbol{A}$ is defined by the fuzzy sets $A_{1,3}$, $A_{2,3}$.

The idea of the membership function can be expanded similarly to multiple dimensions, by introducing the notion of the multidimensional membership function $\mu_{\mathbf{A}^m}(x(k))$ of an input vector $\mathbf{x}(k)$ into $\mathbf{A}^m$:

$$\mu_{A^m}(\mathbf{x}(k))=\begin{cases} 1-rd^m(\mathbf{x}(k)), & \text{if } rd^m(\mathbf{x}(k))\leq 1 \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

where $rd^m(\mathbf{x}(k))$ is the Euclidean relative distance between $\mathbf{A}^m$ and the input data vector $\mathbf{x}(k)$:

$$rd^m(\mathbf{x}(k))=\frac{\left[\sum_{i=1}^{N}\left(a^m_{i,j_i}-x_i(k)\right)^2\right]^{1/2}}{\left[\sum_{i=1}^{N}(\delta a_i)^2\right]^{1/2}} \tag{7}$$

The value produced by the membership function $\mu_{\mathbf{A}^m}(\mathbf{x}(k))$ for a given input vector $\mathbf{x}(k)$ is called multidimensional membership degree of $\mathbf{x}(k)$ into $\mathbf{A}^m$. The notion of the multidimensional membership function can be used to determine the closest fuzzy subspace to the vector $\mathbf{x}(k)$. It can be easily proved that this fuzzy subspace is defined as the combination of fuzzy sets in each dimension that assign

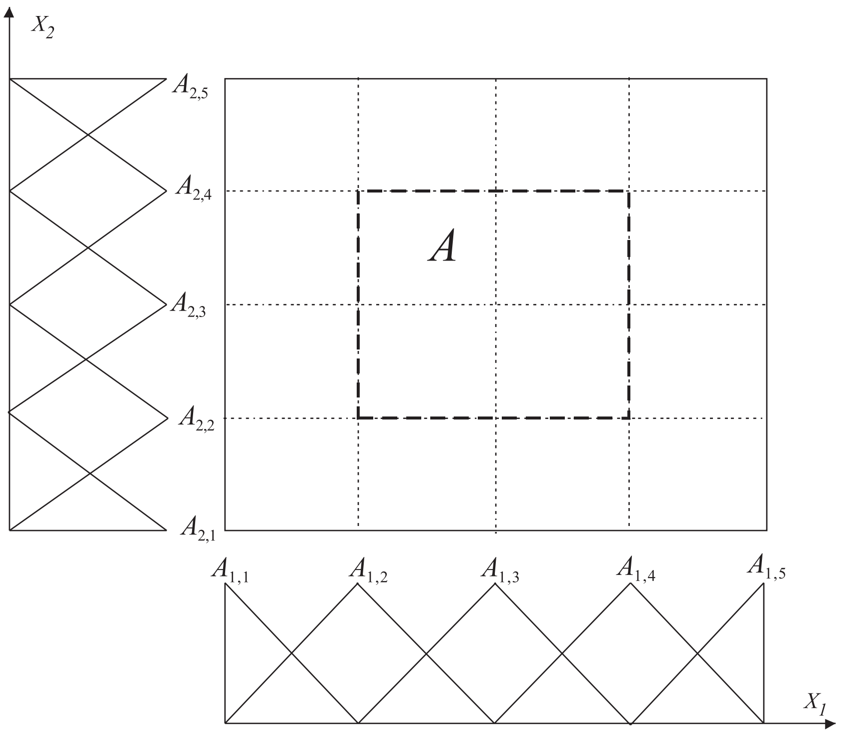


Fig. 2. Fuzzy partition and definition of a fuzzy subspace in the two-dimensional input space.

the maximum membership degrees to the particular inputs $x_1(k)$, $x_2(k)$,. . ., $x_N(k)$.

After the fuzzy partitioning of the input space has been accomplished, the fuzzy means algorithm considers all the fuzzy subspaces as potential hidden node centers in the produced RBF network. The final hidden node centers are selected so that for each input training vector, there is at least one fuzzy subspace that assigns a nonzero multi-dimensional membership degree.

2.3. Genetic algorithms

Genetic algorithms are probabilistic optimization techniques which are based on the principles of genetics and natural selection [14,15]. They form a class of general purpose (domain independent) search methods, which strike a remarkable balance between exploration and exploitation—two apparently conflicting objectives in any optimization method. There are certain differences between GAs and conventional optimization methods: GAs manipulate coded versions of the problem parameters instead of the parameters themselves; they do not use any auxiliary information about the objective function such as derivatives; they operate in parallel on a population of solutions instead of the classical approach of searching from a single point. Hence, GAs are considered to be efficient and robust optimization methods that are able to escape from local optima. Not surprisingly, GAs have been used successfully for the variable subset selection problem, where the search domain is quite large and many local optima exist.

GAs comprise of the following five components:

- A genetic representation of potential solutions to the problem
- A formula for building the initial population of candidate solutions
- An objective function that measures the fitness of the chromosomes
- Genetic operators that guide the algorithm into new areas of the search space
- Parameter values that need to be determined by the user (population size, probabilities of applying genetic operators, etc.)

2.4. Simulated annealing

Simulated annealing is a threshold accepting optimization method that imitates the physical annealing process of a solid. The basic ideas on which simulated annealing is based were founded by Metropolis et al. in 1953 [32]. In order to simulate the annealing process of metals, he considered the material as a system of particles. The probability of a particle being in a specific energy level is expressed by the Boltzmann probability distribution. Virtually, Metropolis's algorithm simulates the changes in the energy of the system during the cooling process, until it converges to a steady state, where it is frozen.

Kirkpatrick et al. [16] first incorporated these ideas into an optimization method. They created a variation of the classical local search method with an important difference: detrimental moves are accepted according to some probability function p, which decreases as the algorithm progresses. As a probability function, Kirkpatrick et al. used the Boltzmann probability distribution.

Initially almost all detrimental moves are accepted so that the algorithm performs a random search. However, as the temperature decreases, less detrimental moves are accepted but the algorithm still has the ability to escape from local optima. During the last iterations, where the temperature approaches zero, the solution is 'frozen' to its final state. This modification allows the algorithm to reach global optima. However, the performance of SA strongly depends on the selection of a proper cooling schedule. Some empirical techniques have been developed for the determination of the cooling schedule, but they are subjective to the nature of the problem and require a lot of trial and error procedures.

Generalized simulated annealing (GSA) [33] is a modification of the conventional simulated annealing algorithm, where the probability function is given by:

$$p = \exp\left(-\beta\frac{f_{\text{new}} - f_{\text{cur}}}{f_{\text{bsf}} - f_{\text{cons}}}\right) \tag{8}$$

In this equation, the temperature is substituted by the term (f_{bsf}–f_{cons}) where f_{new}, f_{cur} and f_{bsf} are the new (not yet approved), current and best so far values of the objective function, respectively, and f_{cons} represents a guess of the expected global optimum. The only parameter that needs to be defined is the constant β. The authors in Ref. [34] suggest that β should be selected so that, at the initial stages of the method, the ratio $R_{\text{a/d}}$ of accepted detrimental moves to rejected detrimental moves lies between 0.5 and 0.7.

3. The proposed algorithm

Variable selection problems can be classified according to their size. To be more specific, a variable selection problem is small scale, medium scale or large scale if the total number of input variables belongs to [0, 19], [20, 49] or [50, ∞], respectively [35]. Most of the variable selection problems that appear in the process industry, in chemometrics and in QSAR analysis are medium- or large-scale problems. The proposed GASA-RBF algorithm (acronym of the words Genetic Algorithm Simulated Annealing Radial Basis Function) is specifically designed in order to take care of the large number of variables that appear in these special classes of variable selection problems. To be more specific, the proposed approach decomposes the variable selection procedure in two stages:

- The first stage employs a specially designed GA in order to find the combination of input variables that produces the best model, as far as the prediction error is concerned. During this stage, different combinations of input variables are coded as chromosomes. In each generation, an RBF network is trained for each chromosome using the fuzzy means algorithm. The networks are then validated on a different set of data, in terms of their prediction error. The combinations of input variables that correspond to the networks with the smallest prediction error are more likely to survive in the next generation. Then, a number of genetic operators are applied, including crossover and mutation. The final solution, which represents the optimum combination of input variables, together with the prediction error of the respective network is carried over to the second stage.
- At the second stage, GSA is used to minimize the number of input variables. The algorithm is initialized with the solution that resulted from the first stage and tries to reduce the number of variables. However, during the different moves of the GSA algorithm, the error of the produced model is not allowed to climb significantly above the threshold imposed by the outcome of the first stage.

Following this qualitative presentation of the method, a more detailed description of the two stages, which are graphically represented in Fig. 3, is given next. It is assumed that a set of input–output examples [**X**, **y**] is available. The dimensionality of the input matrix **X** is $K\times N$, where K denotes the number of the available examples and N the total number of input variables. **y** is a $K\times 1$ vector which stores the values of the output y. Before the algorithm proceeds with the first stage, the set of input–output data must be split into a training set of K_1 examples, a monitoring set of K_2 examples and a validation set of K_3 examples.

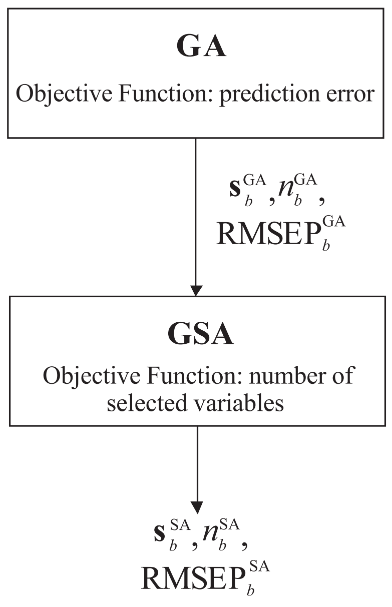

GA
Objective Function: prediction error
$\mathbf{s}_b^{\text{GA}}, n_b^{\text{GA}}$, $\text{RMSEP}_b^{\text{GA}}$
GSA
Objective Function: number of selected variables
$\mathbf{s}_b^{\text{SA}}, n_b^{\text{SA}}$, $\text{RMSEP}_b^{\text{SA}}$

Fig. 3. An overview of the GASA-RBF algorithm.

3.1. First stage: Minimization of the prediction error using a GA

The objective of the first stage is to optimize the prediction error, regardless of the number of selected input variables. This is accomplished by employing a specifically designed GA, which uses a hybrid coding of genes containing both binary and integer values. More specifically, each potential input variable is coded by a binary gene, while the last gene of the chromosomes is an integer and represents an essential parameter for training the RBF network. Thus, the length of each chromosome is equal to the number of total variables N plus one. The binary coding is used to denote whether a variable is present in the model (the gene has the value 1) or not (the gene has the value 0), while the integer coding of the last gene of the chromosome denotes the number of fuzzy sets used by the fuzzy means training algorithm. Following this hybrid coding, each chromosome j represents a potential solution of the variable selection problem and it can be written as:

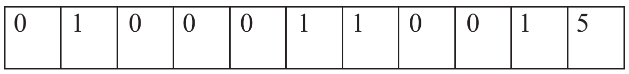
$$\mathbf{s}_j^{GA} = \left[\mathbf{v}_j^{GA}, l_j^{GA}\right] \tag{9}$$

where $\mathbf{v}_j^{\mathrm{GA}}$ is a binary vector of length N denoting the input variables that are selected in the chromosome and l_j^{GA} is the respective number of fuzzy sets. Fig. 4 depicts an example of a chromosome with 11 genes: The first 10 genes correspond to 10 potential input variables while the last one denotes the number of fuzzy sets used in fuzzy partitioning of the input space. For the particular example, variables 2, 6, 7 and 10 are selected and the input space of each one of them is partitioned into 5 fuzzy sets.

Before the execution of the GA, a number of operational parameters must be defined, which are:

- The size of the population, P, which represents the total number of chromosomes
- The total number of generations, G
- The probability of crossover, p_{c}
- The probabilities of uniform mutation, p_{um}, and nonuniform mutation, p_{num}
- The parameters $l_{\min}^{\mathrm{GA}}$, $l_{\max}^{\mathrm{GA}}$ and b^{GA}, which are associated with the nonuniform mutation operator and will be explained later in this section

The first step is to select an initial population of chromosomes. This is randomly created, giving an equal probability for each gene to be coded as '1' or '0'. Then the algorithm proceeds with calculating the fitness of the



0	1	0	0	0	1	1	0	0	1	5

Fig. 4. Representation of a chromosome.

initial generation of chromosomes. For each chromosome that contains a specific sequence of input variables and an integer number, the algorithm creates an RBF network. The fuzzy means algorithm is employed to train each network by assigning the integer gene of the associated chromosome to the number of fuzzy sets. It should be noted that only the training data set is used by the fuzzy means algorithm in order to calculate the parameters of the RBF networks. When this is complete, the performance of each network is evaluated using the monitoring data set. To be more specific, the root mean square error (RMSEP) is used as a measure of the fitness of the chromosomes:

$$RMSEP_j^{GA} = \sqrt{\frac{\sum_{i=1}^{K_2} (\hat{y}_{i,j} - y_i)^2}{K_2}}, j = 1, 2, .., P \tag{10}$$

where $\hat{y}_{i,j}$ is the prediction of the jth network for the ith example. The fitness E_j of each chromosome j is then expressed as:

$$E_j = \frac{1}{RMSEP_j}, \; j = 1, 2, .., P \tag{11}$$

since the GA is tailored to maximize the fitness function.

Based on the calculated fitness of the chromosomes, the algorithm proceeds with the natural selection process, which must assure the reproduction of the fittest chromosomes in the next generation. In the present study, the reproduction is implemented as a linear search through a roulette wheel. Each chromosome is allocated a slot on the roulette, with size proportional to its fitness. A random number is generated and a copy of a chromosome passes to the mating pool only if the random number falls in the slot corresponding to the particular chromosome. This procedure is repeated P times in order to select P chromosomes for the next generation. It is possible that some chromosomes may be selected more than once in accordance to the schema theorem: the best chromosomes get more copies, the average stay even and the worst die off [14].

The genetic operators that are applied after the creation of the new generation are crossover and mutation. The crossover operator is employed to exchange genes between two chromosomes. For the specific GA, a one-point crossover scheme is utilized, where after some pairs of chromosomes are randomly selected based on the probability of crossover p_{c}, they exchange strings of genes. During the crossover operation, the last integer string is treated in the same manner as the binary strings and it is exchanged as well.

However, the different nature of the last string makes it necessary to treat it specially during the mutation operation.

Thus, uniform flip bit mutation is applied to the binary genes and nonuniform mutation is used for altering the integer genes that represent the number of fuzzy sets. The probability for each binary gene to be selected for mutation is p_{um}, while the probability for the integer genes to undergo the nonuniform mutation operator is p_{num}. The flip bit mutation simply inverts the value of the chosen gene (0 becomes 1 and vice versa). Nonuniform mutation is implemented through the use of the following equations:

$$l_{\mathrm{new}}^{GA} = \begin{cases} l_{\mathrm{old}}^{GA} + \left(l_{\max}^{GA} - l_{\mathrm{old}}^{GA}\right)\left(1 - r^{(1-g/G)^{b^{GA}}}\right) & \text{if a random digit is 0} \\ l_{\mathrm{old}}^{GA} + \left(l_{\mathrm{old}}^{GA} - l_{\min}^{GA}\right)\left(1 - r^{(1-g/G)^{b^{GA}}}\right) & \text{if a random digit is 1} \end{cases} \tag{12}$$

where $l_{\mathrm{old}}^{\mathrm{GA}}$ is the old number of the fuzzy sets, $l_{\mathrm{new}}^{\mathrm{GA}}$ is the number of the fuzzy sets that results from the nonuniform mutation, $l_{\min}^{\mathrm{GA}}$ and $l_{\max}^{\mathrm{GA}}$ are lower and upper bounds on the number of fuzzy sets, r is a random number between 0 and 1, g is the number of the current generation, and b^{GA} is a parameter which determines the degree of dependency on the generation number. In Eq. (12), the result is rounded to the nearest integer. Nonuniform mutation has been selected for altering the number of fuzzy sets, since it has the ability to search the space uniformly initially, while as the number of iterations approximates the maximum number of generations, it is forced to perform a more local search.

After the genetic operators have been implemented, the algorithm evaluates the fitness of the new chromosomes and produces a new generation, until the maximum number of generations has been achieved. The chromosome $\mathbf{s}_b^{\mathrm{GA}}$ that produces the smallest prediction error $\mathrm{RMSEP}_b^{\mathrm{GA}}$ during the whole procedure is finally selected. The combination of input variables $\mathbf{v}_b^{\mathrm{GA}}$ and the number of fuzzy sets l_b^{GA} contained in this chromosome, together with $\mathrm{RMSEP}_b^{\mathrm{GA}}$ and the number of selected input variables n_b^{GA}, are carried over to the second stage. A detailed schematic description of the GA is shown in Fig. 5.

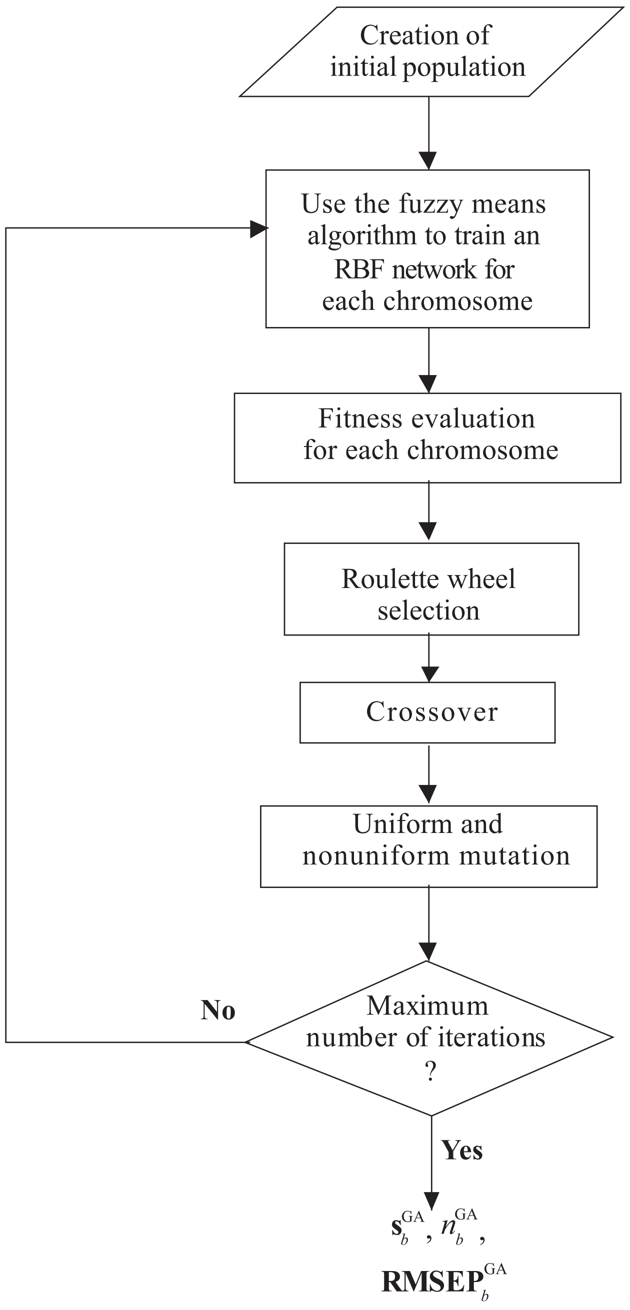

Creation of initial population
Use the fuzzy means algorithm to train an RBF network for each chromosome
Fitness evaluation for each chromosome
Roulette wheel selection
Crossover
Uniform and nonuniform mutation
Maximum number of iterations ?
No
Yes
$\mathbf{s}_b^{GA}$, n_b^{GA}, $\mathbf{RMSEP}_b^{GA}$

Fig. 5. Schematic description of the first stage.

3.2. Second stage: Minimization of the number of input variables using GSA

The outcome of the first stage is a subset of input variables that produces an RBF model with minimum prediction error. However, it is possible that an equal or almost equal prediction error could be achieved with a smaller number of variables. The objective of the second stage is to investigate this possibility, using the GSA algorithm. GSA has also been used for variable selection in [34], where the size of the variable subset was predefined and an integer coding was used for the representation of each variable. However, in the proposed approach, we employ the same coding as the one used in the implementation of the first stage. Thus, the solution at iteration j is represented as:

$$\mathbf{s}_j^{SA} = \left[\mathbf{v}_j^{SA}, l_j^{SA}\right] \tag{13}$$

where $\mathbf{v}_j^{\mathrm{SA}}$ denotes the subset of selected variables and l_j^{SA} is the number of fuzzy sets. The objective function of the GSA algorithm is the number of selected variables n^{SA}. The operational parameters that are associated with the second stage and must be defined before its execution are:

- The total number of iterations, T
- A guess of the expected number of variables, n_{cons}

- The GSA parameter, β
- The maximum number of binary digits that can be perturbed in each iteration, d_c
- The parameters a, $l_{\text{low}}{}^{\text{SA}}$, $l_{\text{high}}{}^{\text{SA}}$ and b^{SA}, which will be explained later in this section.

The subset of variables $\mathbf{v}_b^{\text{GA}}$ and the number of fuzzy sets l_b^{GA} that resulted from the first stage are used as an initial solution in the GSA. Then, during each iteration, the algorithm executes a perturbation in the subset of variables $\mathbf{v}_{\text{cur}}^{\text{SA}}$ of the current solution $\mathbf{s}_{\text{cur}}^{\text{SA}}$, by flipping some of the binary values in the current solution. Thus, a new subset of variables $\mathbf{v}_{\text{new}}^{\text{SA}}$ is created. At the first iterations the maximum number d_c of digits that can change is given relatively large values, but as the algorithm progresses this number reduces, so that in the last iterations only one flip is allowed. It should be noted that the perturbation is biased, as there is greater chance for 1's to become 0's than the opposite. This is reasonable since the objective is to minimize the number of variables.

A different kind of perturbation is applied to the number of the fuzzy sets, which is represented by the last digit of the solution. At each iteration the algorithm uses the following equations to calculate a lower bound $l_{\text{min}}^{\text{SA}}$ and an upper bound $l_{\text{max}}^{\text{SA}}$ for the number of fuzzy sets, with respect to its current value $l_{\text{cur}}^{\text{SA}}$:

$$l_{\text{min}}^{SA} = l_{\text{cur}}^{SA} - \left(l_{\text{cur}}^{SA} - l_{\text{low}}^{SA}\right)\left(1 - r^{(1-t/T)^{b^{SA}}}\right)$$

$$l_{\text{max}}^{SA} = l_{\text{cur}}^{SA} + \left(l_{\text{high}}^{SA} - l_{\text{cur}}^{SA}\right)\left(1 - r^{(1-t/T)^{b^{SA}}}\right) \tag{14}$$

where T is the maximum number of iterations, t is the number of current iteration, $l_{\text{low}}^{\text{SA}}$ and $l_{\text{high}}^{\text{SA}}$ are the predefined minimum and maximum allowed values for the number of fuzzy sets, b^{SA} is a parameter which determines the degree of dependency on the iteration number and r is a random number between 0 and 1. In the above equations, the results are rounded to the nearest integers.

Using the new subset of variables $\mathbf{v}_{\text{new}}^{\text{SA}}$, the proposed methodology tests the different numbers of fuzzy sets that belong to the limits defined by Eq. (14), by creating an RBF network for each one of them with the fuzzy means algorithm. The number of fuzzy sets $l_{\text{new}}^{\text{SA}}$ which gives the smallest RMSEP in the validation data set is selected and incorporated in the new solution $\mathbf{s}_{\text{new}}^{\text{SA}}$.

In order to evaluate the new solution, the algorithm checks whether its RMSEP is equal or smaller than a threshold value RMSEP_t. The threshold value is a function of $\text{RMSEP}_b^{\text{GA}}$ which is transferred from the first stage. Since $\text{RMSEP}_b^{\text{GA}}$ approximates the global optimum, there is a great chance that GSA will not be able to find a subset of fewer variables that produces equal or smaller RMSEP. Hence, $\text{RMSEP}_b^{\text{GA}}$ is multiplied by a number a that is greater than unity:

$$RMSEP_t = aRMSEP_b^{GA} \tag{15}$$

In other words, we sacrifice some of the accuracy of the network predictions in order to achieve a considerable reduction of the number of selected input variables. In the case that RMSEP is equal or smaller than the threshold value, then the new solution $\mathbf{s}_{\text{new}}^{\text{SA}}$ is forwarded for further processing. Otherwise, it is rejected and a different random perturbation of the current solution $\mathbf{s}_{\text{cur}}^{\text{SA}}$ is performed. If the new solution $\mathbf{s}_{\text{new}}^{\text{SA}}$ has passed the threshold accepting test, the algorithm checks the associated number of utilized input variables $n_{\text{new}}^{\text{SA}}$. In case this number is smaller compared to the number of variables $n_{\text{cur}}^{\text{SA}}$ involved in the current solution, then $\mathbf{s}_{\text{new}}^{\text{SA}}$ substitutes $\mathbf{s}_{\text{cur}}^{\text{SA}}$. However, this is allowed to happen even in the opposite case, using the probability function (Eq. (8)), which for the specific problem can be expressed as follows:

$$p = \exp\left(-\beta\frac{n_{\text{new}}^{SA} - n_{\text{cur}}^{SA}}{n_{\text{bsf}}^{SA} - n_{\text{cons}}^{SA}}\right) \tag{16}$$

More precisely, the new solution $\mathbf{s}_{\text{new}}^{\text{SA}}$ is accepted if p is greater than or equal to a random number between 0 and 1. In the above equation, $n_{\text{bsf}}^{\text{SA}}$ represents the number of variables in the best so far solution and $n_{\text{cons}}^{\text{SA}}$ is a guess for the number of variables in the final solution. For this last parameter, the algorithm initially uses a conservative guess, but if during the execution of the algorithm a better solution (with less number of variables) is obtained, the value of $n_{\text{cons}}^{\text{SA}}$ is reduced accordingly [34]. Obviously a higher value of the probability function gives more chances of accepting a non-improving solution. Finally, in case the current solution $\mathbf{s}_{\text{cur}}^{\text{SA}}$ has changed, the algorithm checks if the new accepted solution is the best solution obtained so far and continues with the next iteration.

The final outcome of the second stage is the solution $\mathbf{s}_b^{\text{SA}}$ that is close to the global optimum as far as the prediction error is concerned, while at the same time it contains a minimum number of input variables. A descriptive flow-chart of the second stage is given in Fig. 6.

The neural network model associated with the solution $\mathbf{s}_b^{\text{SA}}$ is finally tested on the validation data set, which has not been utilized during the entire training procedure.

Remark. It should be noted that the number of fuzzy sets, which is a parameter that ultimately controls the size of the produced RBF network, is included in the optimization procedure as a free variable, in both stages of the algorithm. Though the objective of the employed optimization algorithms is only to find a suitable set of input variables, the necessity for including the number of fuzzy sets can be justified considering the nature of the fuzzy means algorithm: The fuzzy means algorithm picks the RBF centers from a grid that is defined in the input space. As

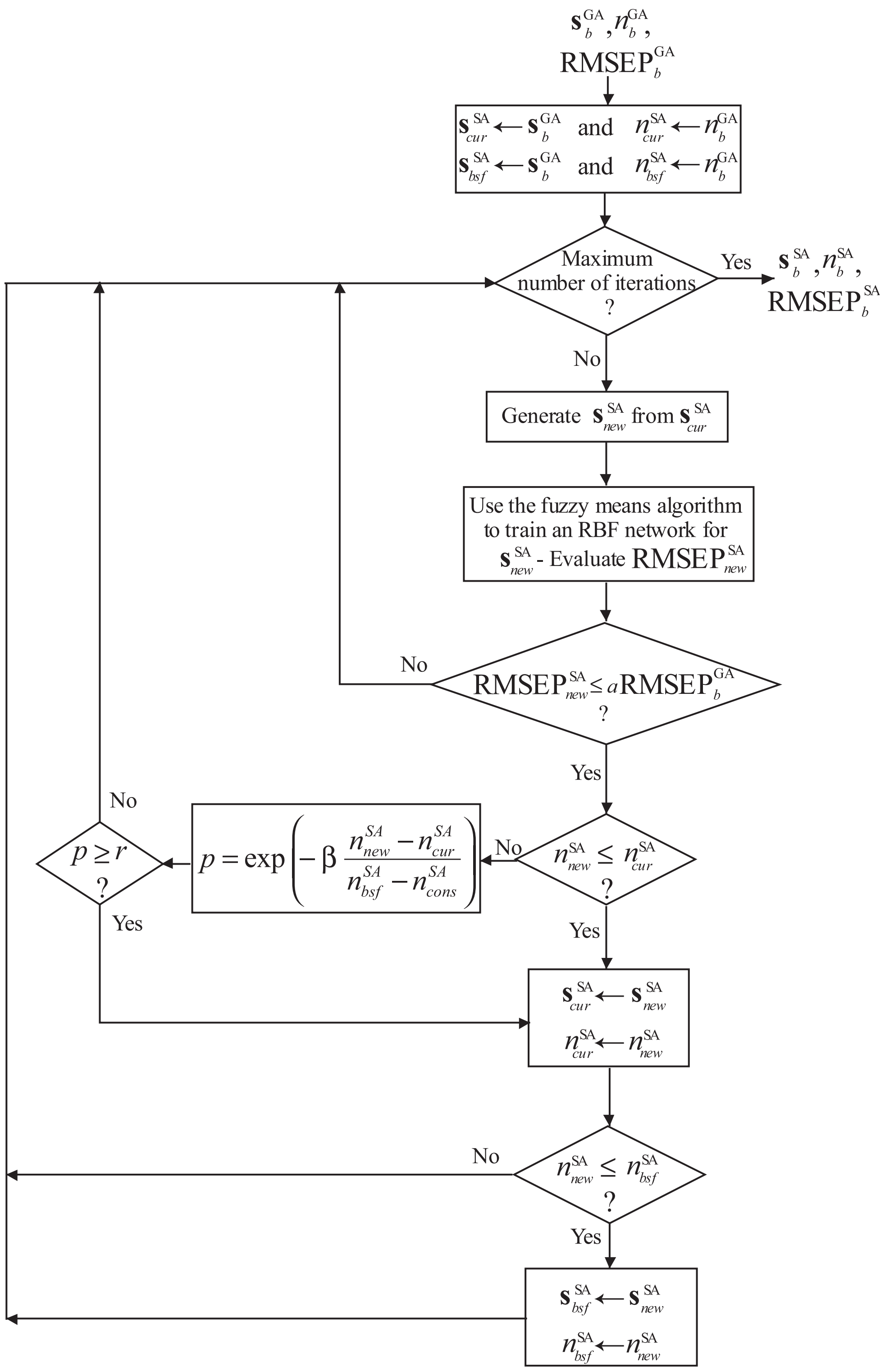

$\mathbf{s}_b^{GA}, n_b^{GA}$, $RMSEP_b^{GA}$
$\mathbf{s}_{cur}^{SA} \leftarrow \mathbf{s}_b^{GA}$ and $n_{cur}^{SA} \leftarrow n_b^{GA}$
$\mathbf{s}_{bsf}^{SA} \leftarrow \mathbf{s}_b^{GA}$ and $n_{bsf}^{SA} \leftarrow n_b^{GA}$
Maximum number of iterations ?
Yes
$\mathbf{s}_b^{SA}, n_b^{SA}$, $RMSEP_b^{SA}$
No
Generate $\mathbf{s}_{new}^{SA}$ from $\mathbf{s}_{cur}^{SA}$
Use the fuzzy means algorithm to train an RBF network for $\mathbf{s}_{new}^{SA}$ - Evaluate $RMSEP_{new}^{SA}$
No
$RMSEP_{new}^{SA} \le a RMSEP_b^{GA}$?
Yes
$n_{new}^{SA} \le n_{cur}^{SA}$?
No
$p = \exp\left(-\beta \frac{n_{new}^{SA} - n_{cur}^{SA}}{n_{bsf}^{SA} - n_{cons}^{SA}}\right)$
$p \ge r$?
No
Yes
Yes
$\mathbf{s}_{cur}^{SA} \leftarrow \mathbf{s}_{new}^{SA}$
$n_{cur}^{SA} \leftarrow n_{new}^{SA}$
No
$n_{new}^{SA} \le n_{bsf}^{SA}$?
Yes
$\mathbf{s}_{bsf}^{SA} \leftarrow \mathbf{s}_{new}^{SA}$
$n_{bsf}^{SA} \leftarrow n_{new}^{SA}$

Fig. 6. Schematic description of the second stage.

the number of input variables increases, the number of nodes in the grid increases as well, leading to more potential RBF centers and consequently more complex RBF network structures. On the other hand, it is known that more RBF centers are needed in order to describe complex and nonlinear systems. In case of such a system, the optimization procedure would tend to create models with more RBF centers, in order to reduce the prediction error. However, if the number of fuzzy sets was fixed during the optimization procedure, the only way to increase the number of RBF centers would be to increase the number of selected input variables. Thus, the algorithm would be biased to select models with more input variables. In order to avoid this, the number of fuzzy sets has been added as a free variable, so that the algorithm can manipulate it for determining the proper number of RBF centers, without increasing unnecessarily the number of input variables.

4. Results and discussion

The first three data sets (ADPN, LATEX, SPIRA) used in this study for the evaluation of the GASA-RBF variable selection methodology are taken from Ref. [30] and they offer useful benchmarks for comparing the method with other variable selection techniques. For a more thorough examination of the performance of the proposed method as far as successful variable selection and accurate prediction are concerned, a fourth simulated data set was constructed from an intrinsically nonlinear relationship. Further description of the first three data sets is given in the original paper of Gauchi and Chagnon [30]. For the construction of the artificial data set, 600 input examples of dimension 100 were generated, where each component was selected as a uniformly distributed number ranging between 1 and 6. The respective output examples were calculated using a deterministic nonlinear function of a subset of the input variables $f(x)$=$f(x_3, x_5, x_{16}, x_{19}, x_{30}, x_{33}, x_{44}, x_{50}, x_{57}, x_{88}, x_{90})$ where normally distributed noise was added. More specifically, for each input example, the output response was calculated as follows:

$$y = (20x_{19}^2/x_{88} + 50\log(x_{90}/x_{33}) + 2x_{30}x_5^2 + 4x_3^3/x_{16} + 200e^{-\frac{(x_{44}-4.5)^2}{2}} + 400e^{-\frac{(x_{50}-3.5)^2}{2}} + 500e^{-\frac{(x_{57}-4.5)^2}{2}})(1+\eta) \qquad (17)$$

where η is a normally distributed random variable with 0 mean and 0.05 standard deviation.

The four data sets together with the total number of variables, the total number of observations, the number of observations used for training the model and the number of observations used for evaluation can be found in Table 1. Obviously, all four variable selection problems belong to the class of large-scale problems, which makes them suitable for testing the proposed methodology.

The GASA-RBF method was applied to the four data sets with the operational parameters shown in Tables 2 and 3, for the first and second stage, respectively. The results are presented in Tables 4–7, which depict separately for each stage of the algorithm the number of selected variables, the RMSEP for the training, monitoring and validation data sets and the parameters of the produced RBF network.

Table 2
Values for the operational parameters of the first stage of the GASA-RBF method

Parameter	Value
Population size, P	20
Maximum number of generations, G	300
Crossover probability, p_c	0.75
Mutation probability, p_{um}	0.01
Nonuniform mutation probability, p_{num}	0.1
Lower bound of the number of fuzzy sets, l_{min}^{GA}	3
Upper bound of the number of fuzzy sets, l_{max}^{GA}	21
Freezing parameter, b^{GA}	2

In order to test the efficiency of the proposed approach, a comparison was performed for the first three problems, between the results of the GASA-RBF method and the results produced by the Backward Q_{cum}^2 (BQ) method, which is selected by Gauchi and Chagnon [30] as the best among the 20 algorithms they tried. Q_{cum}^2 is a cross-validated prediction criterion, which is defined precisely in Ref. [30]. The comparison is made using the $RT_{o,p}^2$ criterion (the squared correlation coefficient of the regression line of the observed–predicted pairs of the validation set, multiplied by 100). The comparative results are summarized in Table 8, which shows the values of the $RT_{o,p}^2$ coefficient and the number of selected variables for the two methods.

For the last variable selection problem, the proposed approach was compared to a well established method, the Forward Selection PLS (FS-PLS). The FS-PLS scheme employs PLS as a modeling technique, forward selection as a search algorithm and cross-validated RMSE as the selection criterion. More specifically, multiple out cross-validation was used, by considering the first 500 data and performing 50 random splits into 250 training data and 250 monitoring data. The proposed methodology was compared to the FS-PLS method on the same external validation data set comprised of 100 objects. Comparative results for the last problem are shown in Table 8.

Tables 9 and 10 depict the indices of the selected variables for all four data sets, where indices in bold indicate the common selections. Based on these results the following remarks can be made:

(1) The GASA-RBF is significantly more accurate compared to the BQ and FS-PLS methods; as in all test

Table 1
Description of the four data sets

Data Set	Total number of variables	Total number of observations	Size of training set	Size of monitoring set	Size of validation set
ADPN	100	71	29	28	14
LATEX	117	262	105	105	52
SPIRA	96	145	58	57	30
Artificial	100	600	400	100	100

Table 3
Values for the operational parameters of the second stage of the GASA-RBF method

Parameter	Value
Total number of iterations, T	5000
GSA parameter, β	2
Initial estimate of global optimum, $n_{\mathrm{cons}}^{\mathrm{SA}}$	5
Multiplier for the threshold value, a	1.05
Maximum number of binary digits that can be perturbed, d_c	5
Minimum allowed value for the number of fuzzy sets, $l_{\mathrm{low}}^{\mathrm{SA}}$	3
Maximum allowed value for the number of fuzzy sets, $l_{\mathrm{high}}^{\mathrm{SA}}$	21
Freezing parameter, b^{SA}	2

Table 4
Results for the ADPN data set

Parameter	First stage	Second stage
Number of total variables	100	45
Number of selected variables	45	28
Number of fuzzy sets	5	5
Number of centers	20	18
RMSE for the training set	1.024	1.2570
RMSE for the monitoring set	1.2099	1.2552
RMSE for the validation set	4.3046	2.6523

cases it produces a higher $RT_{\mathrm{o,p}}^2$value. This is probably due to the nonlinear relations that are present between the input and the output data in the four data sets. Nonlinearities are easily described by an RBF network, but cannot be captured by the BQ and the FS-PLS methods, which are based on the linear PLS methodology.

(2) As far as the number of selected variables is concerned, the GASA-RBF selects more variables for the ADPN data set while it selects significantly less variables for the LATEX, SPIRA and the simulated data set. For the simulated data set, GASA-RBF selects only five true variables and makes no wrong choices, while FS-PLS selects—among 12 variables—9 out of the 11 true variables. It should be mentioned here that in this particular example, an RBF network that uses all the correct input variables increases the $RT_{\mathrm{o,p}}^2$value by only 2%, showing that the inclusion of the rest of the true parameters does not considerably improve the accuracy of the produced nonlinear model. The results show that the GASA-RBF is superior to the BQ method. The results also illustrate the good performance of the FS-PLS method as far as the selection of the correct variables is concerned in the highly nonlinear fourth case.

(3) The most surprising result is that there are a lot of different variables selected by the different methods, especially in the first three data sets. This can be explained in two ways. The first is concerned with the nature of the algorithms (linear vs. nonlinear). Alternatively, this disagreement can be explained by assuming that there exist strong correlations between the explanatory variables.

Since there are several parameters involved in both stages of the method, it would be interesting to examine the sensitivity of the method with respect to the values of these parameters. A first indication of the robustness of the method is given by the fact that exactly the same set of parameters was used in all four variable selection problems. In order to further examine the issue of robustness, we performed a set of experiments on the simulated data set by changing two parameters, one in each stage, namely the crossover probability in the first stage and the multiplier for

Table 5
Results for the LATEX data set

Parameter	First stage	Second stage
Number of total variables	117	53
Number of selected variables	53	12
Number of fuzzy sets	8	11
Number of centers	56	88
RMSE for the training set	0.4174	0.2031
RMSE for the monitoring set	0.4456	0.4618
RMSE for the validation set	0.4985	0.5798

Table 6
Results for the SPIRA data set

Parameter	First stage	Second stage
Number of total variables	96	36
Number of selected variables	36	5
Number of fuzzy sets	5	10
Number of centers	6	34
RMSE for the training set	0.2281	0.1222
RMSE for the monitoring set	0.1666	0.1697
RMSE for the validation set	0.1600	0.2455

Table 7
Results for the simulated data set

Parameter	First stage	Second stage
Number of total variables	100	33
Number of selected variables	33	5
Number of fuzzy sets	11	9
Number of centers	400	200
RMSE for the training set	20.12	105.01
RMSE for the monitoring set	172.28	170.95
RMSE for the validation set	203.75	140.59

Table 8
Comparison of the GASA-RBF and BQ, FS-PLS methods in terms of the number of selected variables and the root mean square error for the validation set

Data set	Method	Number of selected variables	$RT_{\mathrm{o,p}}^2$
ADPN	GASA–RBF	28	76
	BQ	13	71
LATEX	GASA-RBF	12	67
	BQ	24	60
SPIRA	GASA–RBF	5	54
	BQ	15	45
Artificial	GASA-RBF	5	77
	FS-PLS	12	49

Table 9
Indices of selected variables for the GASA-RBF and BQ methods

Data set	GASA-RBF						BQ					
ADPN (71×100)	7	8	**11**	13	26	33	**11**	29	31	**42**	**44**	**46**
	42	**44**	**46**	48	52	58	**68**	69	72	86	90	**95**
	59	62	66	**68**	71	78	**100**					
	79	80	81	83	88	89						
	93	94	**95**	**100**								
LATEX (262×117)	8	**34**	**51**	**53**	58	**90**	2	4	5	26	32	**34**
	95	**96**	**98**	104	109	111	**51**	**53**	54	55	57	64
							74	83	84	**90**	**95**	**96**
							97	**98**	101	110	115	116
SPIRA (145×96)	19	**49**	**71**	74	77		26	31	32	**49**	56	61
							65	67	**71**	72	75	76
							77	78	80			

the threshold value α in the second stage. These are the most critical parameters, since probability of crossover determines to a great extent the variability of each generation related to its preceding generation, while the value a controls the deterioration of accuracy in favor of variable elimination. The results are summarized in Table 11, where the importance of the parameter a can be observed. When the value of a is close to 1, the importance of the second stage of the algorithm is obviously limited. The result is a very accurate model as far as the training and monitoring sets are concerned, but the prediction performance of the model deteriorates when it is tested on the validation data set. Table 11 shows that a good choice for a is 1.05, i.e., the value we used in all four data sets. On the other hand, the performance of the method does not seem sensitive with respect to the parameter $p_{c.}$ It is interesting to note that in two experiments, the prediction accuracies of the produced models were better, compared to the nominal case.

Another interesting issue on the proposed method concerns the execution times. The most time-consuming part is the inversion of the matrix containing the responses of the hidden layer nodes, in order to calculate the weights between the hidden layer and the output layer of each network. This means that there is a great variation on the computational times depending on the size of the training data. For the four particular examples, training times ranged from 1.5 h for the ADPN data set to 5 h for the simulated set in a Pentium IV 1400 MHz processor running Matlab. These times are much larger compared to the ones needed by linear variable selection techniques, but this is the price we pay for improving the accuracy of the produced model. It should be mentioned, however, that if a different neural network architecture or training methodology was applied, the computational times would be orders of magnitude larger.

Table 10
Indices of selected variables for the GASA-RBF and FS-PLS methods

Data set	GASA-RBF				FS-PLS			
Artificial	**3**	**5**	**19**	50	**3**	**5**	16	**19**
(600×100)	**57**				30	33	44	**57**
					60	67	80	88

5. Conclusions

This work presents a novel variable selection method that has the ability to produce nonlinear models between the input and output variables. The method decomposes the optimization problem into two subproblems: in the first subproblem, the objective is to minimize the prediction error, while in the second the objective is to minimize the number of input variables. In both stages, stochastic search techniques (a specially designed genetic algorithm and simulated annealing) are utilized to perform the optimization task. The correlation between the input and output data is modeled based on the fuzzy means algorithm, which is utilized to train RBF neural network models. The particular training methodology exhibits small computational times compared to utilizing other neural network architectures and makes feasible the combination of neural networks with stochastic search techniques. It should be mentioned, however, that the computational effort is larger and the produced model is more complicated compared to standard linear techniques, indicating that the method is particularly valuable when nonlinear relationships exist between the input and output data.

Table 11
Sensitivity of the GASA-RBF method with respect to the design variables (simulated data set)

p_c	a	$RT^2_{o,p}$	Number of selected variables	Number of selected true variables
0.55	1.15	74	8	6
0.75	1.15	78	8	7
0.95	1.15	77	9	7
0.55	1.05	76	6	5
0.75	1.05	77	5	5
0.95	1.05	78	7	6
0.55	1.01	73	15	6
0.75	1.01	75	11	7
0.95	1.01	72	15	9

The GASA-RBF methodology was tested in four variable selection benchmark problems and produced very satisfactory results. Comparisons with other algorithms found in the literature demonstrated the efficiency of the proposed method.

References

[1] M.B. Seasholtz, B. Kowalski, The parsimony principle applied to multivariate calibration, Anal. Chim. Acta 277 (1993) 165–177.

[2] K. Baumann, M. von Korff, H. Albert, A systematic evaluation of the benefits and hazards of variable selection in latent variable regression. Part I. Search Algorithm, theory and simulations, J. Chemom. 16 (2002) 339–350.

[3] M. Forina, C. Casolino, C. Pizarro Millan, Iterative Predictor Weighting (IPW) PLS: A technique for the elimination of useless predictors in regression problems, J. Chemom. 13 (1999) 165–184.

[4] J.M. Sutter, J.H. Kalivas, Comparison of Forward selection, Backward elimination and Generalised Simulated Annealing for variable selection, Microchem. J. 47 (1993) 60–66.

[5] F. Lindgren, P. Geladi, S. Rannar, S. Wold, Interactive Variable Selection (IVS) for PLS. Part 1: theory and algorithms, J. Chemom. 8 (1994) 349–363.

[6] M. Forina, C. Drava, C. De La Pezuela, VI CAC (Chemometrics in Analytical Chemistry Conference), Tarragona, June 1986, Abstract PII-29.

[7] G.A. Bakken, T.P. Houghton, J.H. Kalivas, Cyclic subspace regression with analysis of wavelength-selection criteria, Chemometr. Intell. Lab. Syst. 45 (1999) 225–239.

[8] M.C. Araujo, T.C. Saldanha, R.K. Galvao, T. Yoneyama, H.C. Chame, V. Visani, The successive projections algorithm for variable selection in spectroscopic multicomponent analysis, Chemometr. Intell. Lab. Syst. 97 (2001) 65–73.

[9] V. Centner, D.L. Massart, O.E. de Noord, S. de Jong, B.M. Vandeginste, C. Sterna, Elimination of uninformative variables for multivariate calibration, Anal. Chem. 68 (1996) 3851–3858.

[10] A.S. Barros, D.N. Rutledge, Genetic algorithm applied to the selection of principal components, Chemometr. Intell. Lab. Syst. 40 (1998) 65–81.

[11] U. Depczynski, V.J. Frost, K. Molt, Genetic algorithms applied to the selection of factors in principal component regression, Anal. Chim. Acta 420 (2000) 217–227.

[12] R. Leardi, M.B. Seasholtz, R.J. Pell, Variable Selection for Multivariate Calibration using a Genetic Algorithm: Prediction of Additive Concentrations in Polymer Films from Fourier Transform-Infrared Spectral Data, Anal. Chim. Acta 461 (2002) 189–200.

[13] L.H. Chiang, R.J. Pell, Genetic algorithms combined with Discriminant Analysis for Key Variable Identification, J. Process Control 14 (2004) 143–155.

[14] Z. Michalewitz, Genetic Algorithms+Data Structures=Evolution Programs, 3rd ed., Springer-Verlag, Berlin, 1996.

[15] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, Reading, MA, 1989.

[16] S. Kirkpatrick, C.D. Gelati, M.P. Vechi, Optimization by simulated annealing, Science 20 (1983) 671–680.

[17] H. Akaike, A new look at a statistical model identification, IEEE Trans. Automat. Contr. 19 (1974) 716–722.

[18] I.J. Leontaritis, S.A. Billings, Model selection and validation methods for non-linear systems, Int. J. Control 45 (1987) 311–341.

[19] C. Emmanouilidis, A. Hunter, J. MacIntyre, A multiobjective evolutionary setting for feature selection and a commonality-based crossover operator, Proceedings of CEC'2000, San Diego, California (2000 July).

[20] P. Geladi, B.R. Kowalski, Partial least squares regression: a tutorial, Anal. Chim. Acta 185 (1986) 1–17.

[21] F. Yacoub, J.F. MacGregor, Product Optimization and Control in the Latent Variable Space of Nonlinear PLS Models, Chemometrics and Intelligent Laboratory Systems 70 (2004) 63–74.

[22] Y.H. Bang, C.K. Yoo, I.B. Lee, Nonlinear PLS Modeling with Fuzzy Inference System, Chemometr. Intell. Lab. Syst. 64 (2002) 137–155.

[23] M. Daszykowski, B. Walczak, D.L. Massart, Projection methods in chemistry, Chemometr. Intell. Lab. Syst. 65 (2003) 97–112.

[24] F. Doymaz, J. Chen, J.A. Romagnoli, A. Palazoglu, A robust strategy for real-time process monitoring, J. Process Control 11 (2001) 343–359.

[25] Y. Huang, J. Gertler, T.J. McAvoy, Sensor and actuator fault isolation by structured partial PCA with nonlinear extensions, J. Process Control 10 (2000) 459–469.

[26] F. Despagne, D.L. Massart, Variable selection for neural networks in multivariate calibration, Chemometr. Intell. Lab. Syst. 40 (1998) 145–163.

[27] F. Dieterle, B. Kieser, G. Gauglitz, Genetic algorithms and neural networks for the quantitative analysis of ternary mixtures using surface plasmon resonance, Chemometr. Intell. Lab. Syst. 65 (2003) 67–81.

[28] C. Lu, T. Van Gestel, J.A.K. Suykens, S. Van Huffel, I. Vergote, D. Timmerman, Preoperative prediction of malignancy of ovarian tumors using least squares support vector machines, Artif. Intell. Med. 28 (2003) 281–306.

[29] H. Sarimveis, A. Alexandridis, G. Tsekouras, G. Bafas, A fast and efficient algorithm for training radial basis function neural networks based on a fuzzy partition of the input space, Ind. Eng. Chem. Res. 41 (2002) 751–759.

[30] J.P. Gauchi, P. Chagnon, Comparison of selection methods of explanatory variables in PLS regression with application to manufacturing process data, Chemometr. Intell. Lab. Syst. 58 (2001) 171–193.

[31] J. Moody, C. Darken, Fast learning in networks of locally-tuned processing units, Neural Comput. 1 (1989) 281–294.

[32] M. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller, Equation of state calculations by fast computing machines, J. Chem. Phys. 21 (1953) 1087–1092.

[33] J.H. Kalivas, Optimization using variations of simulated annealing, Chemometr. Intell. Lab. Syst. 15 (1992) 1–12.

[34] U. Horchner, J.H. Kalivas, Further investigation on a comparative study of simulated annealing and genetic algorithm for wavelength selection, Anal. Chim. Acta 311 (1995) 1–13.

[35] M. Kudo, J. Sklansky, Comparison of algorithms that select features for pattern classifiers, Pattern Recogn. 33 (2000) 25–41.