Hindawi Publishing Corporation
Advances in Artificial Neural Systems
Volume 2015, Article ID 931379, 16 pages
http://dx.doi.org/10.1155/2015/931379

Research Article

Stochastic Search Algorithms for Identification, Optimization, and Training of Artificial Neural Networks

Kostantin P. Nikolic

Faculty of Management, 21000 Novi Sad, Serbia

Correspondence should be addressed to Kostantin P. Nikolic; kpnikolic@gmail.com

Received 6 July 2014; Revised 19 November 2014; Accepted 19 November 2014

Academic Editor: Ozgur Kisi

This paper presents certain stochastic search algorithms (SSA) suitable for effective identification, optimization, and training of artificial neural networks (ANN). The modified algorithm of nonlinear stochastic search (MN-SDS) has been introduced by the author. Its basic objectives are to improve convergence property of the source defined nonlinear stochastic search (N-SDS) method as per Professor Rastrigin. Having in mind vast range of possible algorithms and procedures a so-called method of stochastic direct search (SDS) has been practiced (in the literature is called stochastic local search-SLS). The MN-SDS convergence property is rather advancing over N-SDS; namely it has even better convergence over range of gradient procedures of optimization. The SDS, that is, SLS, has not been practiced enough in the process of identification, optimization, and training of ANN. Their efficiency in some cases of pure nonlinear systems makes them suitable for optimization and training of ANN. The presented examples illustrate only partially operatively end efficiency of SDS, that is, MN-SDS. For comparative method backpropagation error (BPE) method was used.

1. Introduction

The main target of this paper is a presentation of a specific option of direct SS and its application in identification and optimisation of linear and nonlinear objects or processes. The method of stochastic search was introduced by Ashby [1] related to gomeostat. Till 60th of last century the said gomeostat of Ashby's was adopted mostly as philosophic concept in cybernetics trying to explain the question of stability of rather complex systems having impacts of stochastic nature [2].

The stochastic direct search (SDS) had not been noticed as advanced concurrent option for quite a long time. The researches and developments works of Professor Rastrigin and his associates promoted the SS to be competing method for solving various problems of identification and optimization of complex systems [3].

It has been shown that SDS algorithms besides being competing are even advancing over well-known methods. Parameter for comparing is a property of convergence during solving the set task. For comparing purposes gradient methods were used in reference [4]. The SDS method showed remarkable advance. For systems with noise certain numerical options offer the method of stochastic approximation (MSA) [5]. In some cases procedures of SDS are more efficient than MSA [6].

During the last 20 years, vast interests have been shown for advanced SDS, especially on the case where classical deterministic techniques do not apply. Direct SS algorithms are one part of the SSA family. The important subjects of random search were being made: theorems of global optimization, convergence theorems, and applications on complex control systems [7–9].

The author has been using SDS algorithms (in several of his published papers) regarding identification of complex control systems [10], as well as synthesis and training of artificial neural networks [11–13].

Through experience in application of certain SDS basic definition the author was motivated to introduce the so-called modified nonlinear SDS (MN-SDS) applicable as numerical method for identification and optimization of substantial nonlinear systems. The main reason is rather slow convergence of N-SDS of basic definition and this deficiency has been overcome.

The application of SDS is efficient for both determined and stochastic description of systems.

The SDS algorithm is characterized by introduction of random variables. An applicable option is generator of random numbers [14, 15].

The previously said is enhanced by well-developed modern computer hardware and software providing suitable ambient conditions for creation and implementation of SDS methods and procedures.

2. Method and Materials

2.1. Definition of SDS Method. The solving of theoretical and/or practical problems usually requests firstly an identification task followed by final stage, that is, a system optimization. The analysis and synthesis of systems always consider the previously said [16, 17].

Methods of SDS are ones of competing options for numerical procedures providing solution for identification and optimization of complex control systems [18], but so ANN. Let us start with an internal system description in general form [19]:

$$\frac{dx}{dt} = f(x, u, s, a, t); \quad t_0 \geq 0, \; x(t_0) = x_0, \tag{1}$$

$$y = g(x, \overline{n}, b, t), \tag{2}$$

$$h_1(x, \overline{n}, c) = 0, \tag{3}$$

$$h_2(x, \overline{n}, d) \geq 0, \tag{4}$$

where $f(\cdot)$ and $g(\cdot)$ are nonlinear vector functions; $h_1(\cdot)$ and $h_2(\cdot)$ are vector functions of constrains (variables and parameters); x is system state vector, u is control vector, and s is vector of disturbance; a, b, c, d are parameters describing the system structure such as constants, matrices, and vectors; t is real time; $\overline{n}$ is noise usually added in (2).

A parameters identification of the above system anticipates certain measurements of the system variables observing the criteria function:

$$Q = Q(x, u, y). \tag{5}$$

The criteria function Q in (5) is without involvement of constrains $h_1(\cdot)$ and $h_2(\cdot)$; in case that constrains are not possible to avoid, Q_h is introduced [20]:

$$Q_h = Q + \sum_{i=1}^{i=m} \lambda_i h_{1,i}(\cdot) + \sum_{j=1}^{j=l} \lambda_j h_{2,j}(\cdot)$$
$$i = 1, 2, \ldots, m; \; j = 1, 2, 3, \ldots, l, \tag{6}$$

where λ_i and λ_j are Langrage multiplicators and $\lambda_i > 0$, a λ_j {0 for $h_{2,j} \geq 0$; Λ_j for $h_{2,j} < 0$, $\Lambda_j > 0$} [20].

When λ_i and λ_j are rather large and tend to ∞, then both Q_h and Q tend to the same value for variables x, u, s, that is, corresponding optimal parameters.

Further for the purpose of simplicity of this presentation a function Q from (5) is to be used and to be the function of one vector variable x, so $Q = Q(x)$.

Methods of optimization start with iterative form where from current state $\mathbf{x}_i$ system transfers into $\mathbf{x}_{i+1}$ by the following rule:

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha F(Q). \tag{7}$$

So, $\alpha F(Q)$ is a function stepping into new state where $\alpha > 0$ is a step and $F(Q)$ is vector function of guiding search. For iterative optimization by using gradient method [21]:

$$F(Q) = -\operatorname{grad} Q, \qquad \operatorname{grad} Q = \nabla Q,$$
$$\mathbf{x}_{i+1} = \mathbf{x}_i - \alpha \operatorname{grad} Q, \quad 0 < \alpha < 1. \tag{8}$$

In case of SDS the relation (7) gets the next form:

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha F(\xi, Q), \tag{9}$$

where the direction of search is function of Q and random vector ξ (Figure 1).

If it is introduced terms

$$\Delta \mathbf{x}_{i+1} = \mathbf{x}_{i+1} - \mathbf{x}_i,$$
$$\Delta Q_i = Q_i - Q_{i-1} \tag{10}$$

the general expression (9) gives some of basic algorithm of SDS.

2.1.1. Nonlinear SDS. Consider

$$\Delta \mathbf{x}_{i+1} \triangleq \begin{cases} -\Delta \mathbf{x}_i, & \Delta Q_i \geq 0; \; \Delta Q_i = Q_i - Q_{i-1} \\ \alpha \zeta_i, & \Delta Q_i < 0; \; \alpha > 0, \; \zeta_i = \operatorname{ort} \xi_i. \end{cases} \tag{11}$$

2.1.2. Linear SDS. Consider

$$\Delta \mathbf{x}_{i+1} \triangleq \begin{cases} \Delta \mathbf{x}_i, & \Delta Q_i < 0 \\ \alpha \zeta_i, & \Delta Q_i \geq 0; \; \alpha > 0, \; \zeta_i = \operatorname{ort} \xi_i. \end{cases} \tag{12}$$

Some of the more complex forms of SDS are as follows.

2.1.3. SDS Stochastic Gradient. Consider

$$\mathbf{x}_{i+1} \triangleq \mathbf{x}_i - \alpha \zeta_i, \quad \alpha \geq 0,$$
$$\zeta_i \triangleq \operatorname{dir} \sum_{P=1}^{P} \xi_i \operatorname{sign}(\Delta Q_P); \quad |\zeta_i| = 1,$$
$$\Delta Q_P \triangleq Q_P(\mathbf{x}_i + \sigma \zeta_P) - Q_P(\mathbf{x}_i - \sigma \zeta_P)$$
$$p = 1, 2, 3, \ldots, P, \tag{13}$$

where p is number of tests before effective step forward.

2.1.4. SDS Fastest Descent. Iterative procedure for a SDS fastest descent is

$$\Delta \mathbf{x}_{i+1} \triangleq \begin{cases} \Delta \mathbf{x}_i, & Q_{i-1} < 0, \\ -\Delta \mathbf{x}_{i-1} + \alpha \zeta_{i-1}; & \Delta Q \geq 0 \\ -\alpha \zeta_i + \alpha \zeta_{i+1}; & \Delta \mathbf{x}_i \neq \Delta \mathbf{x}_{i-1}, \; \Delta Q \geq 0. \end{cases} \tag{14}$$

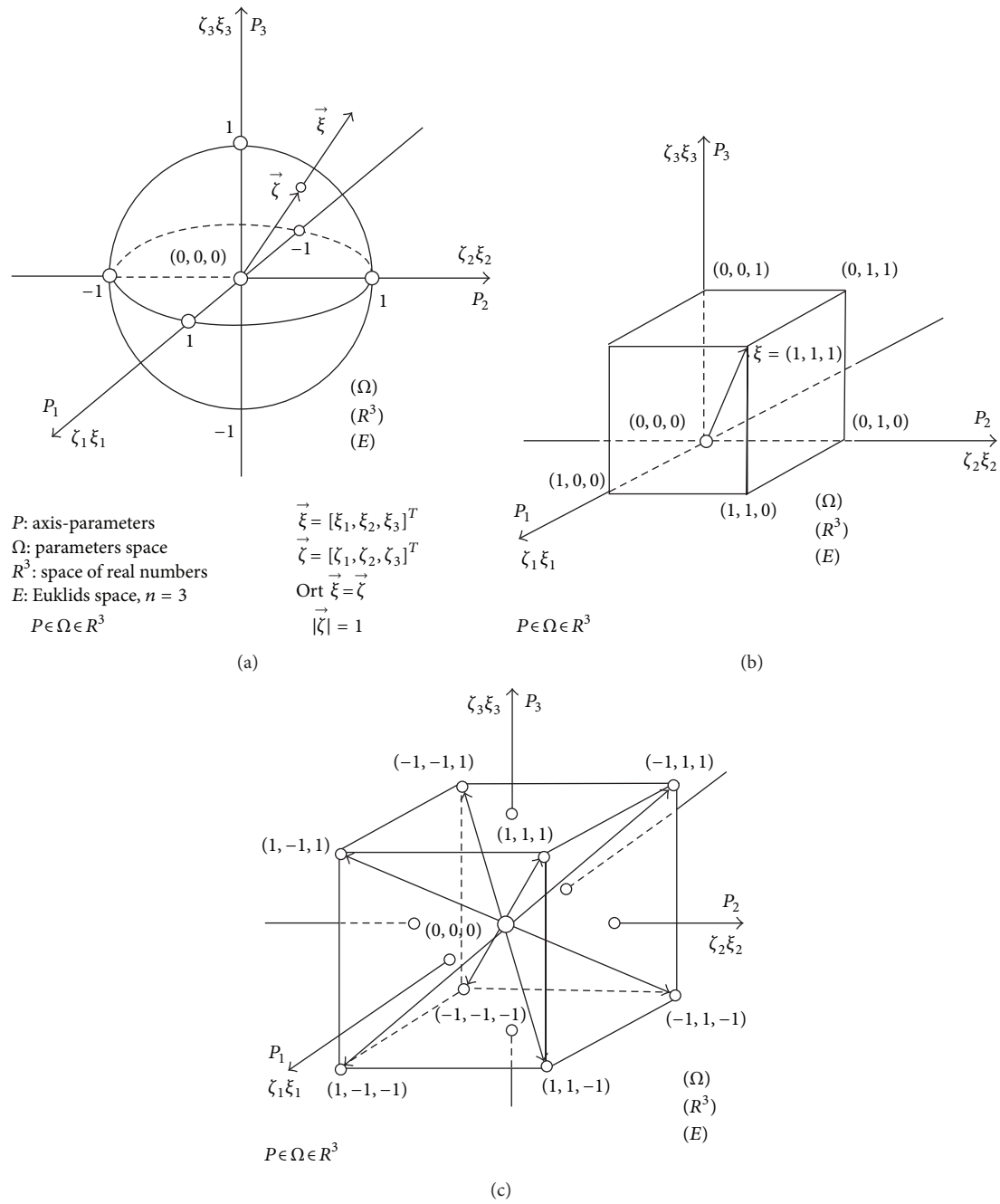


FIGURE 1: A random variable ξ; (a) on sphere radius $|r| = 1$, (b) on cube edge $|a| = 1$, and (c) on cube edge $|a| = 2$; dimension $n = 3$.

The graphic presentation of SDS behavior in the space of gradient Q is given in Figure 2.

Gradient is starting from point P_1, linear SDS starts from point P_2, nonlinear SDS starts from point P_3, and SDS-statistic gradient starts from point P_5 (vector pairs marked dash opposes failure tests). The gradient-fastest descend is presented from starting point P_1. SDS fastest descend is not presented in Figure 2 (it is similar to linear SDS from P_2).

The random vector ξ_n is created on n-dimensional sphere r_n; $|r_n| = 1$, or on n-dimensional cube; n is system dimension in Euclide space E. A presentation is shown in Figures 1(a), 1(b), and 1(c) for $n = 3$.

The SSA properties used for ranking with other competitive methods (gradient, fastest descend, Gaus-Zeidel, and scanning and others) are

(i) *local*: describing algorithm in i-step,

(ii) *nonlocal*: describing algorithm from start to the final stage of optimization.

The main property is dissipation, that is, losses in one step of search and displacement on hipper-surface of the function Q. The said properties describe the algorithm convergence, that is, the method itself.

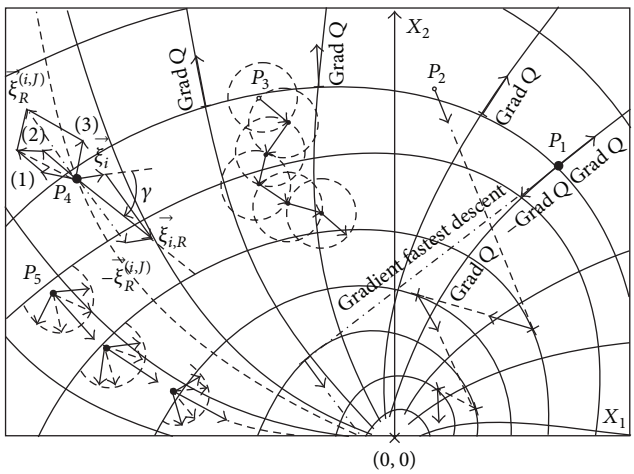


Figure 2: SDS algorithms in gradient field.

The convergence K_n is defined by expression:

$$K_n = \frac{M(N_i)}{\left|M\left(\Delta\overline{Q}_i\right)\right|}, \tag{15}$$

that is, ratio of math expectation of test number N_i in i-iteration over relative change of $\Delta\overline{Q_i}$ (criteria function Q) [22].

Reciprocal value of relation (15) gives an average value q of Q in i-step of searching.

The local SDS property includes probability of failing step of searching:

$$p = P\{Q(x) < Q(x + \Delta x)\}, \tag{16}$$

where x is vector of initial state and Δx is working step of state change in $x + \Delta x$. The probability of failing step is rather important regarding choosing of algorithm for real process optimisation.

So, the choice of optimal algorithm from local property point of view guides to compromise of choice of three:

$$(K, q, p). \tag{17}$$

Besides local properties the relevant ones are properties describing the search in whole [23] which are

(i) total number of steps, that is, number of effective steps N,

(ii) accuracy, that is, allowed error the procedure is ending $\boldsymbol{\varepsilon}$ or relative error δ in %.

The SDS option choice is guided by the above indicated local characteristics properties (K, q, p) as well as nonlocal ones (N, ε).

It is necessary also to observe that the dispersion $D(\Delta\overline{Q})$ of $\Delta\overline{Q}$ could be useful in some statistical estimation of searching procedure properties. Also, it is necessary to notice that dispersion depends on how the vector ξ is generated (hipper sphere or cube) [23].

Finally, it is worthwhile to mention that the choice of algorithm is subject to request to have procedures (identification and optimization) effective, having in minds that system model or criteria function is nonlinear. Substantially nonlinear process models so, described by (1) not having linear approximation which simulates original behavior (1) within narrow range of variables and parameters change against defined error limits. SDS methods are effective in this case so [22, 23].

2.2. Modified Nonlinear SDS: The Definition. Good properties of SDS as per (11) are simplicity, reliability, and applicability over nonlinear systems but slow convergence during the optimization procedure shows weak point. A rough explanation of nonlinear SDS of basic definition within gradient field Q is shown in Figure 2, with starting point $P3$. Comparing with regular gradient method this SDS becomes competitive when the system dimension is over $n > 12$ [24].

By increasing the system dimension the probability for acceptable tests decrease indicating that the target is reached after rather numerous tests and effective steps N. The stochastic gradient is SDS having competitive performance during optimisation of nonlinear systems; however, the algorithm itself accumulates information and becomes fed up in numerical proceeding.

The idea to have failed tests converted onto useful accumulated information guides toward the so-called modified nonlinear SDS. The previously said is shown in Figure 2. For the 3 failed tests from starting point P_4 it effects vector $\xi_{i,R}$ to turn over toward the target under angle γ. So, if the accumulation of failed tests is J, between i and $i+1$ successful steps on hyper area of Q then:

$$\sum_{j=1}^{J} \xi_j^{(i)} = \xi_R^{(i,J)}; \qquad \operatorname{ort} \xi_R^{(i,J)} = \zeta_R^{(i,J)}. \tag{18}$$

Now it is possible to form the next form of SSA search:

$$\Delta\mathbf{x}_{i+1} = \begin{cases} \alpha\zeta_i, & \Delta Q_i < 0, \\ -\alpha\zeta_R^{(i,J)} + \alpha\zeta_{i+1}, & \Delta Q_{i+1} < 0, \end{cases} \tag{19}$$

where $\zeta_{i+1} = \operatorname{ort} \xi_{i+1}$ is successful test after J failed, $+\alpha\zeta_R^{(i,J)}$, $\Delta Q_j = Q_j - Q_{j-1} \geq 0$; $j = 1, 2, 3, \ldots, J$, generate of accumulation information in MN-SDS algorithm and possibly being used so:

$$-\alpha\xi_j^{(i,\max)} + \alpha\xi_{i+1}, \tag{20}$$

where $\zeta_i^{(j,\max)}$ corresponds to $\max |Q_i^{(j)}|$.

A modification defined by (18) and (19) is naturally linked on basically defined nonlinear SDS and further on will be referred to as MN-SDS. At certain extent MN-SSA possesses property of nonlinear algorithm, linear SDS with accumulation and extrapolation and stochastic gradient.

Having in mind that MN-SDS explore accumulated information some sort of self-learning features are obtained (specifically when all data are memorized) leading to the

conclusion that stochastic probability of testing depends on accumulated information:

$$P' = p'\left(\frac{\mathbf{x}}{\mathbf{w}}\right), \tag{21}$$

where, memory vector, could be defined as

$$\mathbf{w} = \int p'\left(\frac{\mathbf{x}}{\mathbf{w}}\right) dx. \tag{22}$$

This brings that searching is guided within the vicinity of the best probe or test [23]; a memory vector indicates adaptation features and can be calculated for l step like

$$\mathbf{w}_i^{(l)} = k\mathbf{w}_{i-1}^{(l-1)} + \chi\boldsymbol{\zeta}_i \Delta Q_i^{(l-1)}, \tag{23}$$

where

$$\Delta Q_i^{(l-1)} = Q_{i+1}^{(l-1)} - Q_i^{(l-1)}, \tag{24}$$

where k is coefficient of erasing and χ is coefficient of learning intensity or self-learning.

Now the vector guidance toward the target is

$$\boldsymbol{\zeta}_l^{(\mathrm{sl})} = \mathrm{ort}\,(\mathbf{w} + \boldsymbol{\xi}) = \frac{\mathbf{w} + \boldsymbol{\xi}}{|\mathbf{w} + \boldsymbol{\xi}|}, \tag{25}$$

where $\zeta_i^{(\mathrm{sl})}$ is corrected direction by self-learning (sl) on l-step; $l = 1, 2, 3, \ldots, L$, $L \leq J$, "steps of accumulation" (J failed test after i step).

In practice the optimisation starts without self-learning option. In that sense MN-SDS regardless of the range of accumulated information of failed tests not memorized enables sampling of those most useful ones.

3. Theoretical Importance for Application of MN-SDS on ANN

3.1. Basic Theoretical Notes of MN-SDS. The main result achieved by MN-SDS is improved convergence compared to that of nonlinear SSA of basic definition (Section 2.1, relation (11)).

For the SSA purposes uniform distribution of random numbers ξ_i within chosen interval $[a, b]$ of real axes is used. Most often it is interval $[-1, +1]$, Figure 1. More complex distribution produces difficulties in numerical calculation [25].

By increasing of system dimension the probability $P(\xi_i)$ of each testing is decreased; $\xi_i \in \boldsymbol{\xi}$, $\boldsymbol{\xi} = [\xi_1, \xi_2, \ldots, \xi_i, \ldots, \xi_n]^T$. The consequence of the previously said is less and less successful testings producing more steps toward the target.

The idea to use failed SDS tests for increase of efficiency of nonlinear SDS initiated creation of the so-called modified nonlinear SDS (MN-SDS). In fact failed tests give accumulated information for increasing of N-SDS convergence. Rearranged iterative procedure for minimization of criteria function Q mentioned in Section 2.2 brought rather effective procedure, that is, MN-SDS.

The MN-SDS procedure provides acceptable choice referring to the set of three characteristics (K, q, p), number of steps N, and set-up error $\boldsymbol{\varepsilon}$ during aiming the target.

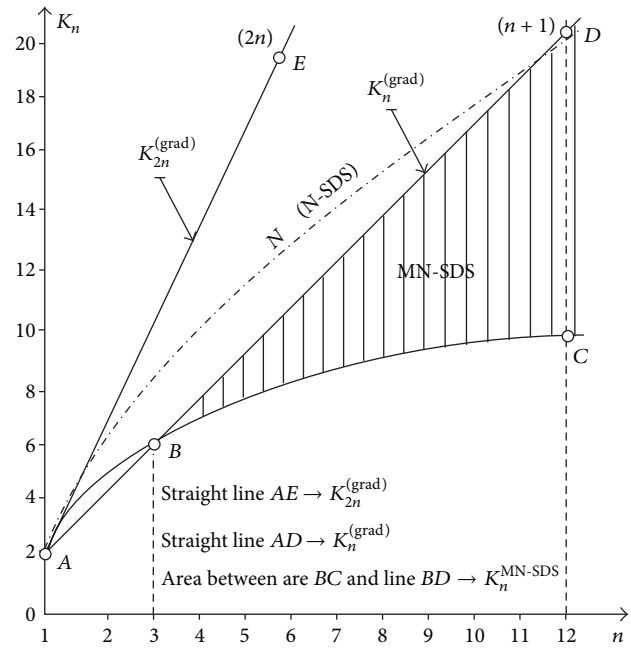


Figure 3: Comparability of convergence MN SDS and N-SDS with gradient.

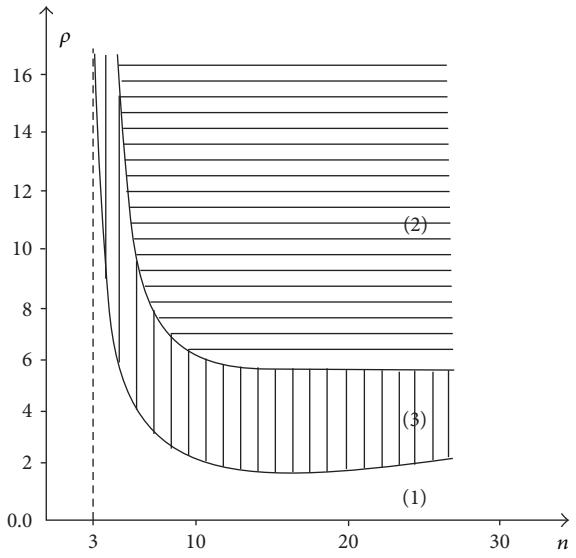
The convergence K_n for N-SDS algorithm with prediction [23, 26] for system dimension n is given by the following relations:

$$K_n = \frac{\sqrt{\pi}\,(n-1)\,\Gamma\left((n-1)/2\right)}{\Gamma\,(n/2)}. \tag{26}$$

This is shown like curve ABC in Figure 3. The curve ABC is the boundary case for search with MN-SDS algorithm when after i-step exist only one failed test [24]. In the same Figure 3 it is given K_n for gradient method with single testing (AD) and in case pair $(2n)$ testing (AE). Sum feathers in Figure 3 has been taken over reference [24].

Nonlinear SSA (N-SDS) of basic definition (the curve AND in Figure 3) has better local convergence than gradient method when system dimension is $n > 12$. MN-SDS confirms that SS algorithms overcome gradient method in efficiency whenever system dimension rapidly increases; $n > 3$. SDS numerical procedures are simple, specifically when $P(\xi)$ is uniform distribution by system dimension increase. In case very large number of dimensions the concerns in term of efficiency between SDS methods and the gradient method changed in the favor of the gradient [24]. MN-SDS its advantages over gradient procedures retains much longer than most SDS algorithms.

It is worthwhile to recognise when to use SDS or some other optimisation method. In that sense in Figure 4 is shown that gradient procedure is more effective within the vicinity of the optimum (Figure 4 has been taken over on [26]). The MN-SDS in certain situation incorporate features of nonlinear SSA and stochastic gradient (SG-SDS). The stochastic gradient is based on accumulation of numerous information and as such it targets optimum with the same efficiency as regular gradient method. Regular-determined



(1) Area of gradient advance
(2) Area of SSA advance
(3) Area of application of both methods

ρ = distance from target
α-searching step
$\rho = r/\alpha$; $r = |x - x_{\text{opt}}|$

Figure 4: The applicability areas of gradient and SDS methods.

gradient method could be considered as specific case of stochastic gradient [26].

It is necessary to mention that random numbers generator should pass more strict tests whenever the system dimension is large [25]. Figure 5 shows diagram of MN-SDS numerical procedure. The random vector generator ξ is shown as outside device just to point necessity of such generator. The said generator is SW-software solution within the computer system in form an application package [15].

3.2. SDS Implementation for ANN Training. Hereinafter an implementation of MN-SDS over multilayer ANN with feedforward information flow through network will be considered (FANN).

The FANN properties (well adopted by researchers and designers) enable wide range of ANN to be transformed into FANN.

The SDS, that is, MN-SDS, can be applied on both FANN model forms, oriented graph and matrix form.

For MN-SDS the first mentioned form is more practical having in mind available heuristic options offering a more efficient MN-SDS.

In this part of the paper and onward a multilayer FANN will be observed as shown in Figure 6.

After adjustment of symbols (in expression (19)) for an ANN, the following form is obtained for MN-SDS:

$$\Delta\omega_{i+1} = \begin{cases} \alpha\zeta_i, & \Delta Q_i = Q_i - Q_{i-1} < 0, \\ \alpha\zeta_j^{(i)}, & \Delta Q_j = Q_j - Q_{j-1} \geq 0, \\ -\alpha\zeta_R^{(i,J)} + \alpha\zeta_{i+1}, & \Delta Q_{i+1} = Q_{i+1} - Q_i < 0, \end{cases} \quad (27)$$

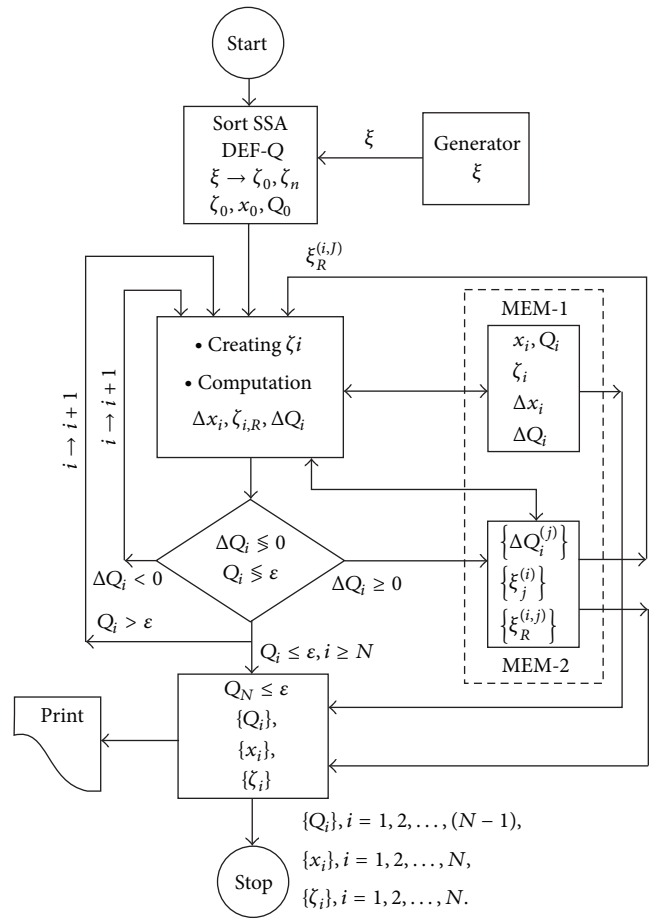


Figure 5: Diagram of procedure for computer processing of MN-SSA.

where $\Delta\omega_{i+1}$ areincrement vector ω_i in optimization process iterationand ζ_i are random vectors of uniform distribution. The cost function will stay Q observing that now $Q = Q(t, y^{(L)})$.

The vector of parameter $\boldsymbol{\omega}$ is changed in pace with iterative optimization procedure as follows:

$$\omega_{i+1} = \omega_i + \Delta\omega_i. \quad (28)$$

The vector ω dimension is determined by the ANN level complexity and also complexity of an optimization procedure, that is, training:

$$\dim\omega = \dim\xi = N_\omega; \qquad \operatorname{ort}\xi = \zeta, \quad |\zeta| = 1, \quad (29)$$

where ξ vector has coordinates as random vector i

$$\xi = (\xi_1, \xi_2, \ldots, \xi_{N\omega})^T, \quad (30)$$

where N_ω is set of all parameters in parametric space, while T means transposition into column matrix. Gradient method and so the backpropagation error (BEP) method use this iterative relation:

$$\omega_{i+1} = \omega_i - \alpha \cdot \nabla Q(\omega_i). \quad (31)$$

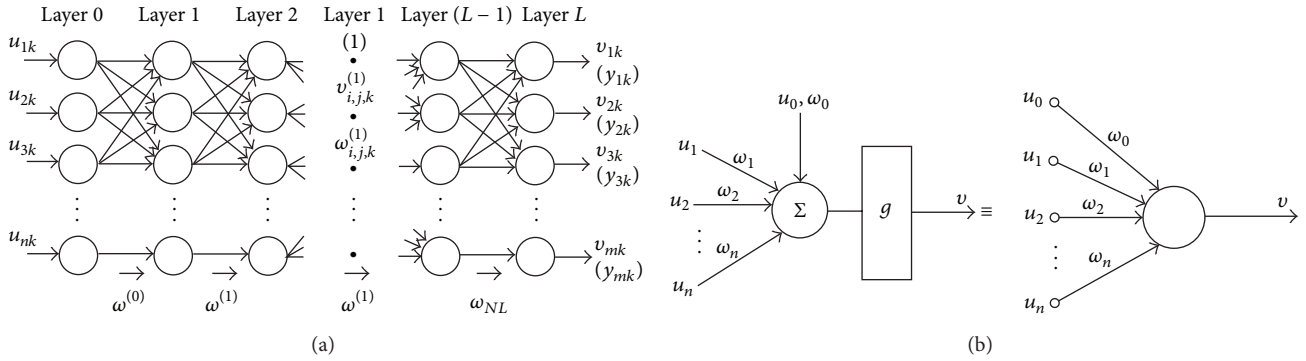


FIGURE 6: The FANN a general type (a) and used model of neuron (perceptron) (b) where $u_0 = 1$.

Stochastic methods (also including MN-SDS) instead of $\nabla Q(\omega_i)$ use random vector ξ which is heuristically chosen enabling iterative parametric optimization with certain efficiency measured by convergence range. Previously the rank of MN-SDS compared to SSA of SDS type as well as other gradient procedures has been set up.

An application of MN-SDS on the FANN (Figure 6) is used known linear interaction function $\text{net}_j^{(\ell)}$ and corresponding output $y_j^{(\ell)}$:

$$\text{net}_j^{(\ell)} = \sum_i \omega_{ij}^{(l)} y_j^{(\ell-1)},$$
$$y_j^{(\ell)} = g_j\left(\sum_i \omega_{ij}^{(l)} y_j^{(\ell-1)}\right), \tag{32}$$

where is:

(i) $\omega_{ij}^{(l)}$—components of weights of vector ω,

(ii) $i = 1, 2, 3, \ldots, N_\nu^{(l-1)}$—neurons in layer $(l-1)$,

(iii) $j = 1, 2, 3, \ldots, N_\nu^{(l)}$—neurons in layer l,

(iv) $\ell = 0, 1, 2, \ldots, L$—all layers in network,

(v) $N_\nu^{(l-1)}$ and $N_\nu^{(l)}$—number of neurons for adjacent layers.

Application MN-SDS algorithm of the training FAAN involve the introduction of random vector ξ of the same size as vector ω:

$$\dim \omega = \dim \xi = \sum_l \sum_\nu N_\nu^{(l-1)} N_\nu^{(l)} + \sum_l N_\nu^{(l)}. \tag{33}$$

The correspondents between components $\omega_{ij}^{(l)}$ and $\xi_{ij}^{(l)}$ must be make before of feedward numerical procedure:

$$\omega_{ij}^{(l)} \longrightarrow \xi_{ij}^{(l)}; \qquad \Delta\omega_{ij}^{(l)} \longrightarrow \alpha \text{ort}\, \xi_{ij}^{(l)}. \tag{34}$$

For the each training pair $p_k = (u_k, t_k)$ is made training; that is, minimization of criteria function Q_k. In the set of training there is K pairs; $p_k \in P$, $k = 1, 2, 3, \ldots, K$. If the training of network performed for entire set P, then it is achieved an epoch of training.

If in the FANN there are more outputs than only one, previously it must be from an error for one output:

$$\varepsilon_{sk}^{(L)} = t_{sk} - y_{sk}^{(L)}. \tag{35}$$

After this it can to form criteria function Q_k:

$$Q_k = \frac{1}{2}\sum_s \left(\varepsilon_{sk}^{(L)}\right)^2 = \frac{1}{2}\sum \left(t_{sk} - y_{sk}^{(L)}\right)^2, \tag{36}$$

where it is: $s = 1, 2, 3, \ldots, m$ and $k = 1, 2, 3, \ldots, K$.

The increment of $\Delta\omega_{ij}^{(l)}$, by weights of the layer l, for one step iteration can be presented as follows:

$$\Delta\omega_{ij}^{(l)} = \alpha \cdot \text{ort}\, \xi_{ij}^{(l)}; \text{ in MN-SDS method,} \tag{37a}$$

$$\Delta\omega_{ij}^{(l)} = -\alpha \frac{\partial Q_k}{\partial \omega_{ij}^{(l)}} = -\alpha \frac{\partial Q_k}{\partial \text{net}_j^{(\ell)}} \frac{\partial \text{net}_j^{(\ell)}}{\partial \omega_{ij}^{(l)}}, \text{ in BPE method} \tag{37b}$$

$$\delta_j^{(l)} = \frac{\partial Q_k}{\partial \text{net}_j^{(\ell)}}, \tag{37c}$$

the indexes denoted the same in the previous expressions.

$\delta_j^{(l)}$ is local gradient, an important characteristic in BPE method [27, 44].

$\delta_j^{(l)}$ can be calculated via SDS procedures (with application the relations (37a) and (37b)) but only through MN-SDS or SDS gradient which gives the BPE primal version [27].

3.3. SDS Algorithms and FANN Synthesis. Synthesis of ANN is engineering design. An ANN design starts with set-up of an initial architecture based on experience and intuition of a designer. In this Section 3.3 it was presented the formally recommendations which are relatively good orientation in design of FANN.

An ANN architecture is determined by the number of inputs and outputs, neurons, and interlinks between them and biases a perceptron is a neuron with adjustable inputs and activation function not necessary to be of step type-threshold.

Experience shows, that is quite clear that for solution of complex problem it is necessary to create a complex ANN architecture.

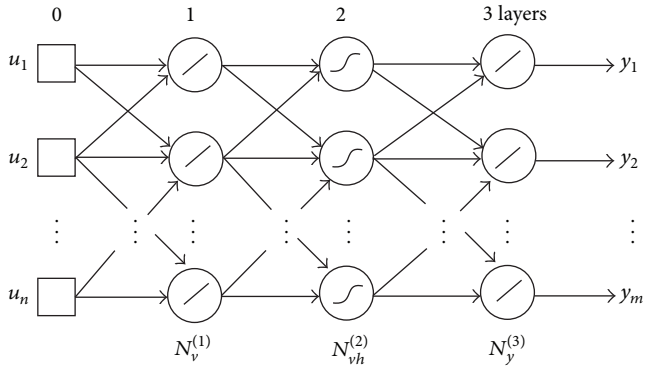


Figure 7: Basic FANN architecture of universal approximation.

Working with SDS various experiences have confirmed the previously noticed. The SDS methods are more efficient comparing to numerous known ones specifically when complex optimization problems are in question specifically when complexity is measured by system dimensions. Having in mind the significance of multilayer of an FANN hereinafter the structure shown in Figure 6 will be considered.

It is worthwhile to mention that successful optimization process of an FANN does not mean that it would have necessary features: first of all required capacity and properties for generalization [29].

Capacity features (C) is one of ANN properties to memorize certain information during the process of training, that is, learning.

An FANN property of generalization is basic paradigm of its performance. In fact it is an essential property approving that "network has learnt" and should provide valid answers to inputs not within training pairs. In other words testing data should have the same distribution as a set of training pairs.

The synthesis issue is an open problem. In [29, 30] some theoretical results have been shown mostly for three layer networks processing binary data. Some researches were working to attempt and expand implementation on three-layer (and multilayer) processing analogue information what have brought the so-called universal approximator [31–33].

The name universal approximator is linked to a three-layer network having perceptrons in hidden layer with nonlinear activation function (type sigmoid) and perceptrons at outputs with linear activation function (Figure 7).

By introducing the following designations, for hidden neurons N_{vh}, for other neurons N_v, for interlinks-synapses N_ω, for threshold N_θ, then by simplifying the theoretical results of [29, 31–33] certain indicators are obtained as relatively good orientation for creation of a FANN architecture.

When a starting network structure has been set up, then its dimension is

$$NN_d = N_\omega + N_\theta. \tag{38}$$

The range of training pairs samples for G level of generalization above 90% (expressions (39), (40) and (41) represent compress (in simple form) of ideas in references [29, 31–33]) is

$$N_p = NN_d \times 10 = \left(N_\omega + N_\theta\right) 10. \tag{39}$$

The FANN ability to memorize and capacity C are determined by relation

$$C = \frac{N_\omega}{m} \tag{40}$$

if the following condition is fulfilled

$$\left(n + \ N_{vh}\right) \gg m; \tag{41}$$

n and m are number of network inputs and outputs respectively (Figure 7).

In case of collision between ANN dimension and required training samples N_p changing of NN_d is required. It is point out that $N_v + N_{vh}$ is in collision with generalization G. For training under the same conditions better generalization is got for networks with less neurons number. The aforesaid indicates that ANN capacity is reduced [34].

Previous consideration with fix configuration during training procedures is characterized as static.

There are some methods approaching to optimization dynamically; during training procedures network structures are changed such as cascade correlation algorithm [35], tiling algorithm [36], dynamic learning by simulated annealing [37], and others. Most complex one is dynamic training by simulated annealing [37, 38]. The aforesaid method resolves numerous FANN synthesis problems.

Training with MN-SDS is processed through forward phase. Numerical procedure is simple and gives enough information SDS shows on dynamically approach in training, optimization and syntheses of artificial neural networks.

3.4. Examples

Example 1 (searching by matrix type). This example is concerned with the theory and system control [39]. Here is presented to show when the SS model system works in matrix form as well as differences in the efficiency of algorithms N_SDS and MN-SDS.

The linearized multivariable system described in internal presentation (see relation (1) and (2) (in Section 2.1)) is as follows:

$$\begin{aligned} \frac{dx}{dt} &= Ax + Bu = 0, \\ y &= Cx + Du, \end{aligned} \tag{42}$$

where A, B, C, D are matrix of parameters of system; x, y, u are vectors with corresponding dimensions.

If the static of the real system is described by matrix form,

In the expression (42) $dx/dt = 0$, then the reduced equations describing the steady-static behavior of the system is:

$$y = C'v + D'u, \tag{43}$$

Table 1: Collection data of variables $\{y_i\}$, $\{v_j\}$, $\{u_k\}$.

v	1	2	3	4	5	6	7
$\{y_i\}$	1,3	2,1	2,9	2,0	2,3	−0,8	−0,9
	1,1	3,9	6,0	2,8	2,1	−0,6	−1,0
	7,0	7,0	2,7	2,6	7,0	+0,8	−3,0
	7,0	1,9	1,1	3,0	3,0	+2,0	−4,0
$\{v_j\}$	1,0	−1,0	2,0	1,0	−1,0	1,0	0,0
	1,0	−1,0	2,0	1,0	1,0	0,0	0,0
	0,0	−1,0	2,0	1,0	2,0	−1,0	0,0
	0,0	1,0	2,0	1,0	1,0	0,0	−1,0
	−1,0	0,0	2,0	1,0	0,0	1,0	−1,0
$\{u_k\}$	−1,0	+1,0	−1,0	+1,0	0,0	−1,0	0,0
	+1,0	0,0	+1,0	0,0	+1,0	−1,0	0,0

where C' and D' are the matrix of parameters corresponding to the static of the observed system.

The relation (43) in the developed form can represent energy network with passive consumers:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} & c_{15} \\ c_{21} & c_{22} & c_{23} & c_{24} & c_{25} \\ c_{31} & c_{32} & c_{33} & c_{34} & c_{35} \\ c_{42} & c_{43} & c_{44} & c_{45} & c_{41} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} + \begin{bmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \\ d_{31} & d_{32} \\ d_{41} & d_{42} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, \tag{44}$$

where $[y]$, $[v]$, and $[u]$ are sizes that can be traced through the check in checkpoint; $[v]$ includes certain set of all measurable sizes in the system (42).

Linear forms type (44) are more complex in numerical processes of form type:

$$y = Cv; \text{ matrix } C \text{ is type } m \times m. \tag{45}$$

Checks consistency and livelihoods solutions [40]:

$$\det C_{m\times m} \neq 0 \tag{46}$$

is inapplicable to form (44).

Having in mind the previously indicated, numerical experiment for coefficients $\{c_{ij}\}$ and $\{d_{ik}\}$; $i = 1, 2, 3, 4$; $j = 1, 2, 3, 4, 5$; $k = 1, 2$ identification has been created. In fact the procedures with SSA algorithm have been used. The data of $\{y_i\}$ were collected by changing variables $\{v_j\}$ and $\{u_k\}$ that are in Table 1.

For identification of $\{c_{ij}\}$ and $\{d_{ik}\}$ matrix searching is used with random matrix $[\xi,]$, generated on hipper sphere in E-space with $n = 28$ [25].

The current values of the sizes y, v, and u monitored through the checkpoint. At certain intervals performs registration thereof. With a certain number of the same is formed by a collection of required data (Table 1). A series of 28 randomly selected numbers filled matrix C' and D'; in each iteration of optimization process. The use of any of the algorithms N-SDS or MN-SDS requires the formation of error ε: $\varepsilon_{v,i} = y_{i,v}^{(\xi)} - y_{i,v}^{(M)}$, $i = 1, 2, 3, 4$; $v = 1, 2, 3, \ldots, N$; N is number of last iteration v, and then coresponding function criteria Q_v is:

$$Q_v = \frac{1}{2}\sum_i \varepsilon_{v,i}^2 = \frac{1}{2}\sum_i \left(y_{i,v}^{(\xi)} - y_{i,v}^{(M)}\right)^2, \tag{47}$$

where $y_{i,v}^{(\xi)}$ are components $[y]$ for the random selection parameters and $y_{i,v}^{(M)}$ required measurement values.

For step iteration used $\alpha = 0.1$; 0,001; and 0,0001 respectively. The initial random parameters of this procedure are

$$\left[c_{0,ij}\right]_{4\times5} = \begin{bmatrix} 0{,}100 & 0{,}201 & 0{,}302 & 0{,}403 & 0{,}500 \\ 0{,}500 & 0{,}503 & 0{,}501 & 0{,}500 & 0{,}504 \\ 1{,}010 & 1{,}005 & 1{,}009 & 1{,}010 & 2{,}000 \\ 1{,}012 & 1{,}011 & 1{,}006 & 2{,}018 & 2{,}001 \end{bmatrix};$$
$$\left[d_{0,ik}\right]_{4\times2} = \begin{bmatrix} 0{,}100 & 1{,}012 \\ 1{,}013 & 0{,}100 \\ 0{,}101 & 1{,}008 \\ 0{,}001 & 0{,}000 \end{bmatrix}. \tag{48}$$

The final results after random procedure with N-SSA are

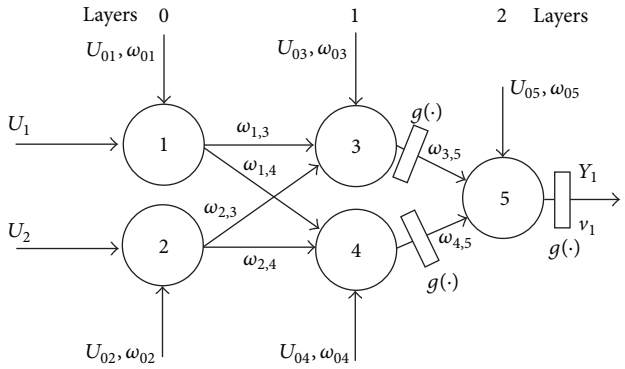
$$\left[c_{N,ij}\right]_{4\times5} = \begin{bmatrix} 0{,}213 & 0{,}051 & 0{,}524 & 0{,}911 & 1{,}004 \\ 0{,}301 & 0{,}401 & 0{,}512 & 0{,}701 & 0{,}816 \\ 0{,}931 & 1{,}631 & 0{,}622 & 1{,}743 & 1{,}320 \\ 1{,}520 & 0{,}721 & 0{,}831 & 0{,}997 & 0{,}802 \end{bmatrix};$$
$$\left[d_{N,ik}\right]_{4\times2} = \begin{bmatrix} 0{,}301 & 0{,}834 \\ 0{,}824 & 0{,}501 \\ 0{,}313 & 0{,}542 \\ 0{,}116 & 0{,}311 \end{bmatrix}. \tag{49}$$

The accuracy of $\delta = 1\%$ calls to have $\alpha = 0,0001$ and number of steps l $N \geq 8000$. There is no noise in system.

Implementation of MN-SDS is possible after transforming of equation system (44) into matrix form:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = v_1 \begin{bmatrix} c_{11} \\ c_{21} \\ c_{31} \\ c_{41} \end{bmatrix} + v_2 \begin{bmatrix} c_{12} \\ c_{22} \\ c_{32} \\ c_{42} \end{bmatrix} + v_3 \begin{bmatrix} c_{13} \\ c_{23} \\ c_{33} \\ c_{43} \end{bmatrix} + v_4 \begin{bmatrix} c_{14} \\ c_{24} \\ c_{34} \\ c_{44} \end{bmatrix} + v_5 \begin{bmatrix} c_{15} \\ c_{25} \\ c_{35} \\ c_{45} \end{bmatrix} + u_1 \begin{bmatrix} d_{11} \\ d_{21} \\ d_{31} \\ d_{41} \end{bmatrix} + u_2 \begin{bmatrix} d_{12} \\ d_{22} \\ d_{32} \\ d_{42} \end{bmatrix}. \tag{50}$$

By application of MN-SDS method some N= 2,156 steps are needed indicating that some 4 times less steps are required for the same accuracy of $\delta = 1\%$.



U_1	U_2	Y_1
0	0	0
0	1	1
1	0	1
1	1	0

(b1)

U_1	U_2	Y_1
−1	−1	−1
−1	1	1
1	−1	1
1	1	−1

(b2)

(b)

Figure 8: Optimization, for example, training of multilayer ANN; XOR problem.

Example 2 (training of multilayer perceptron). This example is linked to treatment of training of perceptron related to "XOR" logic circuit by using of SDS procedures. The said example had an important role in R&D works in the field of artificial neural network. In fact Minsky and Pappert (1969) confirmed that perceptron cannot "learn" to simulate XOR logic circuit and not to expect much of ANN [41].

If from XOR true table training pairs are formed (000, 011, 101, 110) then it is possible to obtain similar conclusion as Minsky and Pappert. If we observe the definition of neuron (perceptron) as per McCulloch and Pitts [42] for neuron with two inputs,

(1) $o \cdot \omega_1 + o \cdot \omega_2 < \theta$; or $0 < \theta$ (t-o is 0),

(2) $o \cdot \omega_1 + 1 \cdot \omega_2 > \theta$; or $\omega_2 > \theta$ (t-o is 1),

(3) $1 \cdot \omega_1 + o \cdot \omega_2 > \theta$; or $\omega_1 > \theta$ (t-o is 1),

(4) $1 \cdot \omega_1 + 1 \cdot \omega_2 < 0$; or $\omega_1 + \omega_2 < 0$ (t-o is 0),

where t-o are training outputs: 0, 1, 1, 0.

It is obvious that relations (2), (3), and (4) are excluding each other. A perceptron cannot be trained to simulate XOR logic circuit.

The aforesaid is understood as obstacle "*in principum*".

The previously indicated somehow interrupted further development of ANN more than 20 years up to works of Hopfield [43] and others.

The problem has been overcome by studying of multilayer and recurrent ANN properties as well as creation of advance numerical procedures for training of the same [27].

This example incorporates the results of training having at least one hidden layer of neurons which could be trained to simulate XOR logic circuit.

Figure 8(a) shows an FANN configuration having one hidden layer of two neurons and one neuron at the ANN output. That ANN can learn to simulate XOR logic circuit.

For training pairs two options can be used (Figures 8(b1) and 8(b2)):

$$P_k(u_k, t_k) \longrightarrow P_1(0,0;0), P_2(0,1;1), P_3(1,0;1), P_4(1,1;0),$$
$$P'_k(u'_k, t'_k) \longrightarrow P'_1(-1,-1,-1), P'_2(-1,1;1), P'_3(1,-1;1),$$
$$P'_4(1,1;-1). \tag{51}$$

Further on results realized with training pairs $P_k(u_k, t_k)$ shown in Figure 8(b1) shall be used. It is necessary to observe that for some of variables should have fixed values since do not contribute in solving the problem:

$$U_{01} = U_{02} = 0, \qquad \omega_{01} = \omega_{02} = 0,$$
$$U_{03} = U_{04} = U_{05} = -1. \tag{52}$$

At the time of training it was shown that $\omega_{12}, \omega_{13}, \omega_{23}$ and ω_{24} have very little variations near number 1 and it is possible to used

$$\omega_{12} = \omega_{13} = \omega_{23} = \omega_{24} = +1. \tag{53}$$

Values of all other $\omega_{0j}^{(l)}$ and $\omega_{ij}^{(l)}$ are changeable parameters where l is indication of neuron layer. By that the dimension of the random vector ξ is decreased. For the first training pair and activation function of logistic type

$$g_1(x) = \frac{1}{[1 + \exp(-x)]} \tag{54}$$

training will be performed through BPE method [27, 28] and MN-SDS; x presents a linear function interaction of neuron inputs.

The criteria function for both cases is

$$Q_k = \varepsilon_k = \frac{1}{2}\sum_{s=1}^{m}\left(y_{sk}^{N_L} - t_{sk}\right)^2. \tag{55}$$

The SSA iterative procedures were used to minimize Q_k; the results are presented in Figure 9(a). The cost functions Q_k

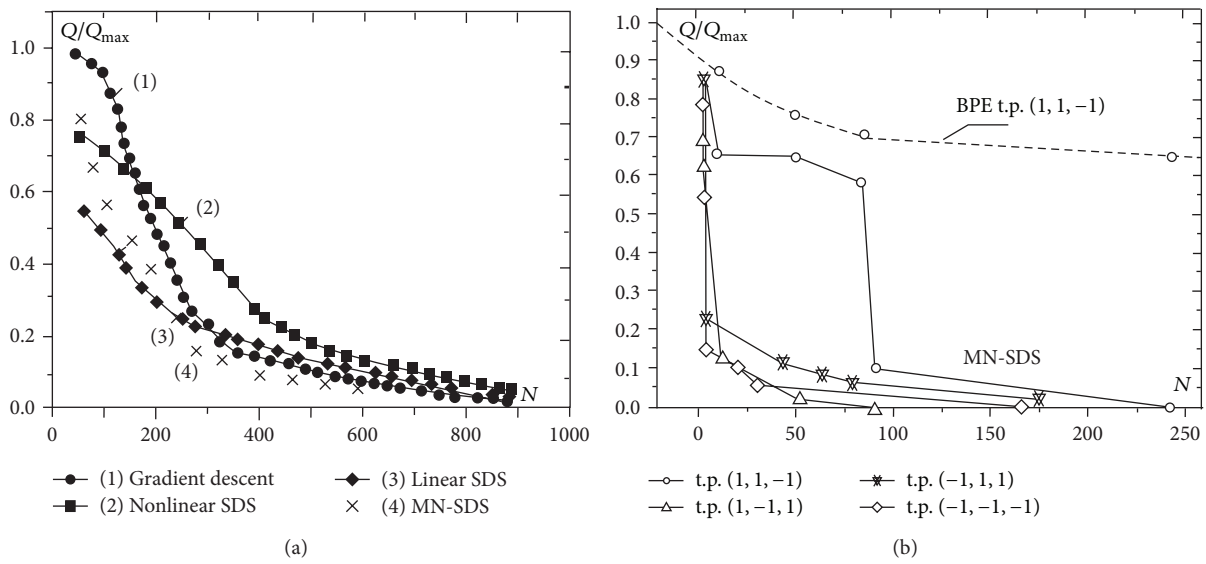


FIGURE 9: Cost function; (a) for $g_1(x) = 1/(1 + \exp(-x))$, (b) for $g_2(x)$ step function; BPI optimization is finished after 600 iterations.

are formed for the same training pairs and initial values, and ponderation is done against the highest value of one of Q_k.

In Figure 9(a) diagram has been singed with number 4, is criteria function Q of training by NN-SDS method.

The results of training are shown so for a step activation function (threshold):

$$g_2(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0. \end{cases} \tag{56}$$

The method BPE is disabled since $g_2(x)$ is not differentiable. For application BPE in this case it must be to approximate $g_2(x)$ with logistic function $S(x) = 1/(1 + \exp(cx))$; $c \geq 100$. The process optimization is finished after more than 600 iterations (Figure 9(b)). The final results for MN-SDS of the ponderation Q as shown in Figure 9(b); training was done with set. SS procedure with MN-SDS has been initiated with random parameters (weights): $\omega_{0,03} = 1.01$; $\omega_{0,04} = 0.55$; $\omega_{0,05} = 0.74$; $\omega_{0,35} = 0.19$; $\omega_{0,45} = 0.57$.

Finally results after $N > 300$ iterations are $\omega_{03,N} = 1.50$, $\omega_{04,N} = 0.99$, $\omega_{05,N} = 0.51$, $\omega_{35,N} = -1.98$, $\omega_{45,N} = 0.98$. with relative error of 2%.

The random vector of forward propagation is with dimension $n = 5$ is:

$$\xi \equiv (\xi_1, \xi_2, \xi_3, \xi_4, \xi_5)^T. \tag{57}$$

Let us refer to an example when an activation function $g_2(x)$ is given by relation (56), with training pairs $P'(u'_k, t'_k)$. Then it shows that a training procedure depends on choice of training pairs as well. The minimization process of such case is done rather faster (Figure 9(b)).

BPE methods implementation was made with known procedures [27, 28]. BPE is used as comparative method.

In this paper MN-SDS and BPE use ECR (procedures error correction rules) which are more convenient for "online" ANN training [45].

Whenever the training coefficient (α) is small then ECR and GDR procedures provide the same results for parameters estimation within set-up range of an acceptable error [44, 45]. GDR (gradient rules) is called "batch" optimization procedure.

Example 3 (synthesis FANN to approximate model FSR). In this example some of theoretical R&D results of [29–33] are applied.

The real problem is approximated model of an ANN training program related on technical term for process in reactor in which the concentrate heated to powder so that will behave like a fluid [46, 47].

When the said FSR is to be tempered either the first time or after service attempts, there is a program of tempering to temperature T as per diagram (see Figure 10). After reaching working temperature it is maintained by control system. The tempering to $T = 600 \pm 25^{\circ}$C is to be completed within 130 min (range AB). In Figure 10 the whole cycle is within 240 min. The FSR was under operation 40 minutes (range BC). Due to some maintenance problem the FSR is often shut down (point C). Before next campaign the FSR should be cooled down (range CD) for maintenance purposes and afterwards the tempering operation is repeated.

The mapping in real conditions have not enough data to make FANN (Figure 11, [46]). There is not enough, to model approximations over ANN that learns. Overall number of pairs (t_k, T_k) collected from the said diagram (Figure 10) is 80 pairs; sampling period is 3 min. More frequent sampling is not realistic. The task is to determine a FANN, which is to give the best approximate model of the FSR within required accuracy. The relative error of correct response should be more than 3% of obtained measuring data.

On Figure 12 is given a starting architecture of FANN NN_1 = (1-10-10-1): with one input and one output, with 10 neurons in the first and in second layers. Only neurons in the

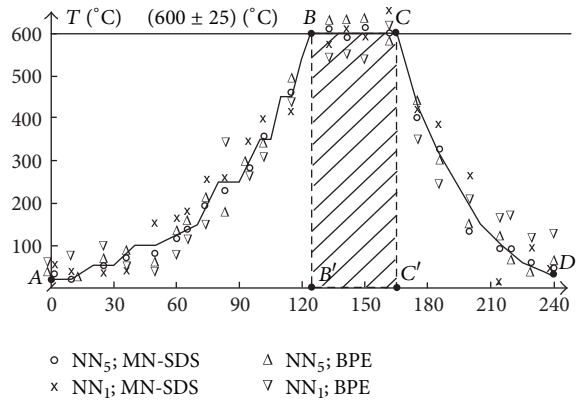


Figure 10: Plan introducing FSR in operative mode (AB), work (BC), possibly down time (CD).

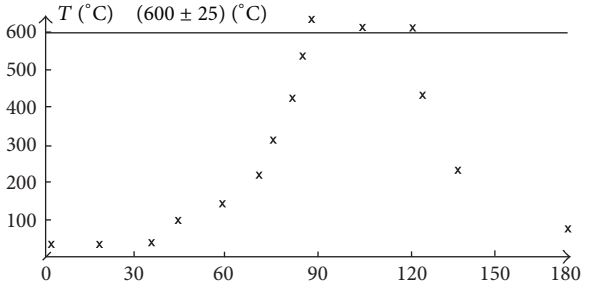


Figure 11: The real mapping data of FSR at operative environment.

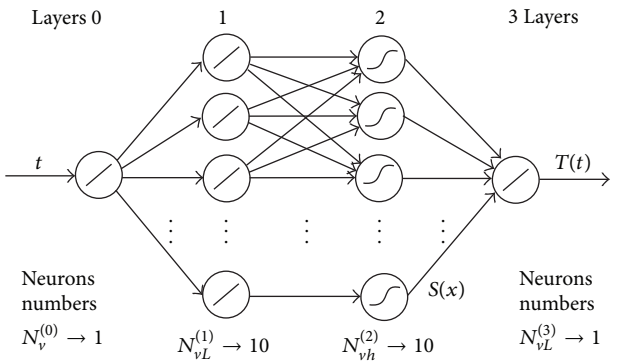


Figure 12: The initial structure of FANN for approximation model's tempering of FSR.

second layer has non-linear activation function. All neurons has a linear interaction function.

Application of the relations ((39)–(41)) of Section 3.2 on NN_1 gives:

$NN_d = 120 + 21 = 141$ interlinks between unknown parameters and biases.

$N_p = 141 \times 10 = 1410$ required training pairs for generalization $G > 90\%$!,

$C = 120$ memorized information under condition that $n + N_{uh} \gg m$, $11 \gg 1$; condition is satisfied.

Table 2: Data defining program warming FSR (t, T).

t [min]	T [°C]
0	20
10	20
20	50
30	50
40	100
50	100
70	150
80	250
90	250
100	350
105	350
110	450
115	450
120	550
125	600
130	600
140	600
160	600
170	600
175	425
185	325
205	150
215	110
225	75
235	60
240	50

Based on the aforesaid there is overnumbered required training pairs ($1410 \gg 80$).

Basic moments in conjunction approximation model FSR through the use of MN-SDS in the training of FANN NN_1 = (1-10-10-1) are that

(i) assignment training NN_1 does of 80 pairs of training is achieved through the diagrams on Figure 10 that define Table 2,

(ii) random vector ξ replaces ω_{ij} and ω_{0j} at expression (32), so $\dim \xi = 141$, unknown parameters,

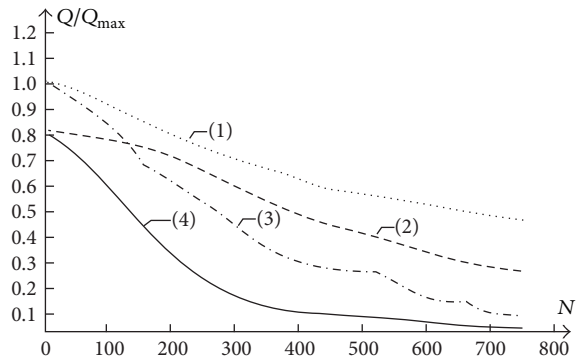
(iii) feedforward phase in layers for each training pair achieved numerical value for $\text{net}_j^{(\ell)}$ and $y_j^{(\ell)}$, $l = 1, 2, 3$, to the expression (32),

(iv) cost function for sequence for each training par Q_k, $k = 1, 2, 3, \ldots, 80$, was calculated with the expression (36). Figure 13 presents the trend of Q_k for NN_1,

(v) the procedure optimization, that is, training for FANN NN_1 takes more than 1000 iterations.

Due to the big volume of numerical data in the paper, we will not list all the details.

Out of the two ANN under training with same training samples more strong generalization has network with less number of neurons.



(1) BPE of NN_1 = (1-10-10-1)
(2) MN-SDS of NN_1 = (1-10-10-1)
(3) BPE of NN_5 = (1-3-1)
(4) MN-SDS of NN_5 = (1-3-1)

Figure 13: Trend cost functions multilayers FANN's NN_1 and NN_5.

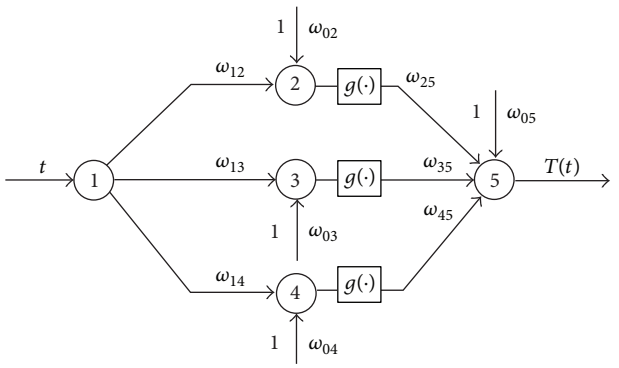


Figure 14: Oriented graph of NN_5 = (1-3-1).

That is the reason to refer to less complex network structures:

NN_2 = (1-5-5-1), NN_3 = (1-3-3-1) and NN_4 = (1-2-2-1).

Structure NN_4 eventually could be trained and have certain generalization possibility since for NN_4

NN_d = 13 interlink, that is, unknown parameters,

N_p = 130 required training pairs for $G > 90\%$; $130 > 80$!,

$C = N_\omega/m = 8$, condition $n + N_{vh} \gg m$; $3 > 1$, barely accepted.

More acceptable solution is NN_5 = (1-3-1), although it presents a bared architecture FANN, since for NN_5

NN_d = 10, dimension of NN_5,

N_p = 100; there are not 100 but only 80 training pairs,

C = 6, condition $n + N_{vh} \gg m$; $4 \gg 1$, acceptable.

It has been possible to continue to a minimized architecture NN_6 = (1-2-1), but choice of NN_5 = (1-3-1) provides better performance of a hidden layer.

The closest solution is the FANN NN_5 = (1-3-1), Figure 14. In hidden layer this network has 3 perceptrons with sigmoid type for activation function.

Having in mind that determining of an ANN, that is, FANN architecture is always open problem then estimation of an adopted structure is suitable after the final numerical result including validity test of a network training.

Theoretical results for universal approximator are derived for nonlinear activation, function of hidden neurons of type $S(x) = 1/(1 + \exp(-x))$.

Since $2S(x) - 1 = \tanh x/2$ bring an option to use tanh function for a FANN model approximation.

Application the relation of $\text{net}_j^{(\ell)}$ and $y_j^{(\ell)}$ in the expression (32), for structure on Figure 14, NN_5 = (1-3-1) in general form:

$$T(t) = \omega_{25} \tanh\left(\omega_{12} t + \omega_{02}\right) + \omega_{35} \tanh\left(\omega_{13} t + \omega_{03}\right) + \omega_{45} \tanh\left(\omega_{14} + \omega_{04}\right) + \omega_{05}; \tag{58}$$

represents an approximation model of the temperature regime of FSR.

Here will be presented numerical data and the results of network training for NN_5 by MN-SDS only.

On the beginning of the numerical procedure for practical reasons t and T should be reduced 100 times.

The symbolic presentation of the vector unknow parameters $\boldsymbol{\omega}$, in the network NN_5, at the beginning ($\boldsymbol{\omega}_0$) and end of training procedure ($\boldsymbol{\omega}_N$) is given by:

$$\begin{aligned} \boldsymbol{\omega}_0 &= \left(\omega_{12}^{(0)}, \omega_{13}^{(0)}, \omega_{14}^{(0)}, \omega_{25}^{(0)}, \omega_{35}^{(0)}, \omega_{45}^{(0)}, \omega_{02}^{(0)}, \omega_{03}^{(0)}, \omega_{04}^{(0)}, \omega_{05}^{(0)}\right)^T \\ \omega_N &= \left(\omega_{12}^{(N)}, \omega_{13}^{(N)}, \omega_{14}^{(N)}, \omega_{25}^{(N)}, \omega_{35}^{(N)}, \omega_{45}^{(N)}, \omega_{02}^{(N)}, \omega_{03}^{(N)}, \omega_{04}^{(N)}, \omega_{05}^{(N)}\right)^T. \end{aligned} \tag{59}$$

The initial random value of the parameters ω_0 is:

$$\omega_0 = (0.1576, 0.9572, 0.4218, 0.8003, 0.7922, 0.9706, 0.9706, 0.4854, 0.1419, 0.9157)^T. \tag{60}$$

Random vector $\boldsymbol{\xi}$ in this case is

$$\boldsymbol{\xi} = (\xi_1, \xi_2, \xi_3, \ldots, \xi_{10})^T. \tag{61}$$

Behind the training network NN_5 by algorithm MN-SDS after 700 iterations ($N > 700; Q_{k,N} < \varepsilon,\ \varepsilon = 10^{-4}$) vector ω_N of unknown parameters is given by

$$\omega_N = (1.705, 3.847, 11.61, 3.159, -3.597, 0.731, -1.670, -7.021, -13.23, 02939)^T. \quad (62)$$

Previous data characterize the approximation process model tempering temperature of FSR (58), overtraining FANN NN_5 by the MN-SDS algorithm, with 5% of the relative error. Trend of Q_k for NN_5 is given on Figure 13.

Some responses to the test inputs for checking the validity of the obtained model deviate a large error. Counting these to interrelate (sharing) the total number received relatively rough estimate of generalization capabilities appropriate network. Based Figure 10 test values have special graphic symbols (O, X for MN-SDS and Δ and ∇ for BPE). For a training set of 80 pairs generalization G ability of the network NN_5 is about 70%. For the network NN_1 it is about 20%. Previous values obtained training using MN-SDS.

Application BPE method gave the following values of generalization: about 70% for network NN_5 and below 20% for the network NN_1.

The previous presented FSR model could be used in more sophisticated option as an intelligent process monitoring.

4. Discussion

Why someone should go to the application of stochastic search methods (SSMs) to solve problems that arise in the optimization and training of ANN? Our answer to this question is based on the demonstration that the SSMs, including SDS (stochastic direct search), have proved to be very productive in solving the problems of complex systems of different nature.

In particular, previous experience with ANN and relatively simple architecture suggest that they can exhibit quite a complex behavior, which can be traced to (i) a large number of neuron-perceptron elements in ANN system, (ii) the complexity of the nonlinearity in the activation function of neurons within ANN, (iii) the complexity of the neuron activation function model (i.e., higher level models), (iv) complexity in the optimization procedure due to the large volume of data in the training set, and (v) the complexity of the specification of internal architecture of particular types of ANN.

The features listed above require competitive methods which can deal efficiently with such complexity. The SDS represent a combinatorial approach offering great potential via certain heuristics and algorithms they provide for numerical procedures.

For example, various methods based on the notion of gradient and which are considered competitive when applied to complex systems cannot avoid the linear scaling of convergence in numerical implementations. In SDS, the trend is roughly speaking proportional to $\sqrt{n}$ where n represents the dimension of the vector of parameters in the parameter space of a complex system. This indicates that, with increasing complexity of the system, the relative advantage of SDS method increases when compared to the gradient scheme. That is the key conclusion of why code optimization and training of ANN should employ SDS.

The author has previously used some algorithms belonging to the SDS methodology, such as nonlinear SDS (N-SDS) and statistical gradient (SG-SDS). Both of these methods have exhibited poor convergence. By starting from N-SDS, we have designed MN-SDS which already for $n > 3$ is superior to gradient descent (with original N-SDS this is achieved only for $n > 12$).

In this paper, Section 2 overviewed the concept of SDS and SSA and then introduced MN-SDS. Section 3 examined the possibilities of MN-SDS algorithm and its application on FANN as the target architecture. Section 3 also presents steps in synthesis of FANN in order to emphasize that performed optimization of FANN does not guarantee that the network will achieve the required level of generalization (i.e., ability to learn). Generally speaking the problem of syntheses ANN remains open.

The present synthesis elements are simplified theoretical results of the recent studies. This is illustrated by Example 3, which is directly connected to practical applications. Example 2 should give an insight about the relationship between N-SDS and MN- SDS, as well as connection between MN-SDS and BPE methods, where the latter was used as a reference. Example 1 confirms efficiency of MN-SDS methods for problems outside of ANN.

Let us finally mention that there is an increasing interest in using SSM, both from academia and industry. This is due to the fact that SSM, and in a particular SDS, can find increasing applications in economics, bioinformatics, and artificial intelligence, where the last area is intrinsically linked to ANN.

5. Conclusion

The central goal of this study is the presentation of stochastic search approach applied to identification, optimization, and training of artificial neural networks. Based on the author's extensive experience in using SDS approach to the problems of identification and optimization of complex automatic control system, a new algorithm based on nonlinear SDS (N-SDS), which is termed MN-SDS, is proposed here. The MN-SDS offers significant improvement in convergence properties compared to nonlinear N-SDS and some other SSA.

MN-SDS maintains all the other good features of the existing SDS: a relatively easy adaptation to problem solving; simple mathematical construction of algorithmic steps; low sensitivity to noise.

The convergence properties of MN-SDS make it superior to majority of standard algorithms based on gradient scheme. Note that convergence is the most suitable characteristics for comparing the efficiency of algorithms for systems with the same number of optimization parameters. For example, already for more than three parameters the MN-SDS exhibits better convergence properties than most other algorithms, including those based on the gradient. This means that, in certain optimization procedures and training, MN-SDS is superior to widely used BPE method for ANN and in the development of artificial intelligence.

The MN-SDS in optimization and training of ANN employs only feedforward phase flow of information in FANN. The parameters that are used in optimization within MN-SDS are changed using random number generator. The efficiency of MN-SDS in numerical experiments suggests that it can be applied to very complex ANN. This study has shown its application to feedforward ANN (FANN). The obtained results were compared with results obtained with BPE method, of course, when applied to the same problems.

Numerical experiments performed here can be implemented even on simple multicore PC using MATLAB package.

Conflict of Interests

The author declares that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The author has been using the achievements of Professor Rastrigin courses and expresses his gratitude. Unfortunately there are no possibilities for direct thanks for using some elements for the figures out of the Professor's books.

References

[1] W. R. Ashby, "Cybernetics today and its adjustment with technical sciences in the future," in *Computer Machine and Logical Thinking*, Compman and Hall, 1952.

[2] L. A. Rastrigin, "Ashby's Gomeostat," in *Stochastics Search Methods*, pp. 50–58, Science, Moscow, Russia, 1965.

[3] L. A. Rastrigin, *The Theory and Application Stochastics Search Methods*, Zinatne, Riga, Latvia, 1969.

[4] K. K. Ripa, "Comparison of steepest deescent and stochastics search self-learning methods," in *Stochastic Search Optimization Problems*, Zinatne, Riga, Latvia, 1968.

[5] A. Dvorezky, "On stochastisc approximation," in *Proceedings of the 3rd Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, University of California Press, Berkeley, Calif, USA, 1956.

[6] L. A. Rastrigin and L. S. Rubinshtain, "Comparison of stochastic search and stochastic approximation method," in *The Theory and Application Stochastics Search Methods*, pp. 149–156, Zinatne, Riga, Latvia, 1968.

[7] A. A. Zhigljavsky, *Theory of Global Random Search*, Kluwer Academic, Boston, Mass, USA, 1991.

[8] N. Baba, T. Shoman, and Y. Sawaragi, "A modified convergence theorem for a random optimization method," *Information Sciences*, vol. 13, no. 2, pp. 159–166, 1977.

[9] J. C. Spall, "Introduction to stochastic search and optimization: estimation, simulation and control," *Automation and Remote Control*, vol. 26, pp. 224–251, 2003.

[10] K. P. Nikolic, "An identification of complex industrial systems by stochastic search method," in *Proceeding of the ETAN '79*, vol. 3, pp. 179–186, 1979.

[11] K. P. Nikolic, "Neural networks in complex control systems and stochastic search algorithms," in *Proceeding of the ETRAN '09 Conference*, vol. 3, pp. 170–173, Bukovička Banja, Aranđelovac, Serbia, 2009.

[12] K. P. Nikolic and B. Abramovic, "Neural networks synthesis by using of stochastic search methods," in *Proceeding of the ETRAN '04*, pp. 115–118, Čačak, Serbia, 2004.

[13] K. P. Nikolic, B. Abramovic, and I. Scepanovic, "An approach to synthesis and analysis of complex recurrent neural network," in *Proceedings of the 8th Seminar on Neural Network Applications in Electrical Engineering (NEUREL '06)*, Belgrade, Serbia, 2006.

[14] J. A. Gentle, *Random Number Generation and Monte Carlo Method*, Springer, New York, NY, USA, 2nd edition, 2003.

[15] C. B. Moler, *Numerical Computing with MATLAB*, SIAM, Philadelphia, Pa, USA, 2003.

[16] P. Eykhoof, "Some fundamental aspects of process-parameter estimation," *IEEE Transactions on Automatic Control*, vol. 8, no. 4, pp. 347–357, 1963.

[17] C. S. Beighlar, *Fundamental of Optimization*, 1967.

[18] L. A. Rastrigin, *Stochastic Model of Optimization of Multiple Parameters Objects*, Zinatne, 1965.

[19] J. T. Tou, *Modren Control Theory*, McGraw-Hill, New York, NY, USA, 1964.

[20] J. Stanic, "Langrage's method of multiplicators," in *Book Introduction in Techno—Economic Theory of Process Optimization*, pp. 35–40, Faculty of Mechanical Engineering, Belgrade, Serbia, 1983.

[21] G. A. Korn, "Derivation operators," in *Mathematical Handbook for Scientists and Engineers*, pp. 166–170, McGraw-Hill, New York, NY, USA, 1961.

[22] L. A. Rastrigin, "Stochastic local search algorithms," in *Book Stochastics Search Methods*, pp. 64–102, Science, Moscow, Russia, 1968.

[23] L. A. Rastrigin, "Characteristics of effectiveness of stochastic search method," in *Stochastics Search Methods*, pp. 32–41, Science Publishing, Moscow, Russia, 1986.

[24] L. A. Rastrigin, "Comparison of methods of gradient and stochastics search methods," in *Book Stochastics Search Methods*, pp. 102–108, Science, Moscow, Russia, 1968.

[25] K. P. Nikolic, "An approach of random variables generation for an adaptive stochastic search," in *Proceeding of the ETRAN '96*, pp. 358–361, Zlatibor, Serbia, 1996.

[26] L. A. Rastrigin, "Multistep algorithms in the central field," in *Book Stochastics Search Methods*, pp. 95–103, Science, Moscow, Russia, 1968.

[27] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[28] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representation by error propagation," in *Parallel Distributed Processing Explorations in the Microstructures of Cognition*, D. E. Rumelhart and J. L. Mc Clelland, Eds., vol. 1, pp. 318–362, MIT Press, Cambridge, Mass, USA, 1986.

[29] E. E. Baum and D. Haussler, "What size net gives valid generalization?" *Neural Computation*, vol. 1, no. 1, pp. 151–160, 1989.

[30] E. B. Baum, "On the capabilities of multilayer perceptrons," *Journal of Complexity*, vol. 4, no. 3, pp. 193–215, 1988.

[31] K. Hornik, M. Stinchcombe, and H. White, "Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks," *Neural Networks*, vol. 3, no. 5, pp. 551–560, 1990.

[32] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1991.

[33] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function," *Neural Networks*, vol. 6, no. 6, pp. 861–867, 1993.

[34] J. Flacher and Z. Obradović, "Constructively learning a near-minimal neural network architecture," in *Proceedings of the International Conference on Neural Networks*, pp. 204–208, Orlando, Fla, USA, 1994.

[35] S. E. Fahlman and C. Lobiere, "The Cascade-corellation learning architecture," in *Advances in Neural Information Processing Systems*, D. Touretzky, Ed., vol. 2, pp. 524–532, Morgan Kaufmann, San Mat, Calif, USA, 1990.

[36] M. Mezard and J.-P. Nadal, "Learning in feedforward layered networks: the tiling algorithm," *Journal of Physics A: Mathematical and General*, vol. 22, no. 12, pp. 2191–2203, 1989.

[37] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[38] S. Milenkovic, "The idea of adaptive selection of type preturbations in the algorithm of simulated annealing," in *Proceedings of the XXXVI YU Conference for ETRAN*, vol. 2, pp. 67–74, Kopaonik, Serbia, 1992.

[39] K. P. Nikolic, "An implementation of stochastic search for complex systems identification and optimization," in *Proceedings of the ETRAN '82*, vol. 3, pp. 221–227, Subotica, Serbia, 1982.

[40] G. A. Korn and T. M. Korn, *Mathematical Handbook for Scientists and Engineers*, McGraw-Hill, New York, NY, USA, 1961.

[41] M. Minsky and S. Pappert, "Perceptrons," in *An Introduction to Computational Geometry*, MIT Press, Cambridge, Mass, USA, 1969.

[42] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, pp. 115–133, 1943.

[43] J. J. Hopfield, "Neural network and physical systems with emergent collective computational abilites," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 79, pp. 2554–2558, 1992.

[44] S. Haykin, "Summary of the back-propgation algorithm," in *Book Neural Networks (A Comprehensive Foundation)*, pp. 153–156, Macmillan College Publishing, New York, NY, USA, 1994.

[45] S. Milenkovic, "Algorithms for artificial neuron networks training," in *Ph.D dissssertation: Annealing Based Dynamic Learning in Second—Order Neuron Networks*, ("Artificial Neuro Networks" Library Disertatio—Andrejevic, Belgrad, 1997), pp. 29–44, Univecity of Nish, ETF, 1996.

[46] K. P. Nikolić, "An identification of non-linear objects of complex industrial systems," in *Proceedings of ETRAN '98—XLII Conference for Electronics, Computers, Automation, and Nuclear Engineering*, pp. 359–362, Vrnjacka Banja, Yugoslavia, 1998.

[47] G. M. Ostrovsky and Yu. M. Volin, "The mathematical description of process in fluo-solid reactors," in *Methods of Optimization of Chemical Reactors*, pp. 30–47, Chemie, Moscow, Russia, 1967.