Multilayer Perceptron Neural Networks Training Through Charged System Search and its Application for Non-Technical Losses Detection

Luis A. M. Pereira, Luis C. S. Afonso, João P. Papa, Zita A. Vale,
Caio C. O. Ramos, Danillo S. Gastaldello and André N. Souza

***Abstract*— The non-technical loss is not a problem with trivial solution or regional character and its minimization represents the guarantee of investments in product quality and maintenance of power systems, introduced by a competitive environment after the period of privatization in the national scene. In this paper, we show how to improve the training phase of a neural network-based classifier using a recently proposed meta-heuristic technique called Charged System Search, which is based on the interactions between electrically charged particles. The experiments were carried out in the context of non-technical loss in power distribution systems in a dataset obtained from a Brazilian electrical power company, and have demonstrated the robustness of the proposed technique against with several others nature-inspired optimization techniques for training neural networks. Thus, it is possible to improve some applications on Smart Grids.**

***Index Terms*— Charged System Search, Neural Networks, Non-technical Losses.**

I. Introduction

In the recent years, the problem of detecting non-technical losses in distribution systems has been paramount. Theft and adulteration of power meters with the purpose of modifying the measurement of the energy consumption are the main causes that lead to non-technical losses in power companies [1]. Since to perform periodic inspections to minimize such frauds may be very expensive, it is a hard task to calculate or measure the amount of that losses, and in most part of the cases it is almost impossible to know where they occur [2].

Smart grid is a new concept of power grid-based systems, which employs the information of technology in power systems, integrating the communication system with a infrastructure of an automated network. The electric energy is supplied to consumer with reliability, security and efficiency. Thus, it is possible to reduce frauds, energy thefts or problems with power meters, helping non-technical losses detection through information about the operation and performance, as the voltage, the current and others parameters [3].

Currently, some advances in this area can be observed with the use of various artificial intelligence techniques in order to automatically identify non-technical losses, which is a real application in smart grids. Among all neural networks (NN)-based classification techniques, ANN with Multilayer Perceptron (MLP) is the widely employed one, which consists of three main layers of neurons (input, hidden and output), whose number of neurons varies according to the application domain [4]. In ANN-MLP, the neurons of the hidden layer are connected with all neurons from the previous and latter layers, and their connections are properly weighted.

Generally, the weights are computed by a learning process that adjusts them by reducing an error (*i.e.*, the difference between the actual NN's output and the desired output). The commonest method for NN training is Backpropagation (BP), which updates the weights in order to minimize the mean squared error over a given number of iterations or until a minimum error is reached, whichever it comes first. However, the main drawbacks of such training method are its extensive computations, relatively slow convergence speed and possible divergence for certain conditions [5].

Since the NN training step can be modeled as an optimization problem, several works have proposed to improve the NN training by employing meta-heuristics algorithms based on nature-social behavior. In such context, Gudise and Venayagamoorthy [5] have compared the BP with the well-known Particle Swarm Optimization (PSO) for weights computation, and Karaboga et al. [6] used the Artificial Bee Colony (ABC) optimization algorithm for training feed-forward neural networks. Further, Al-Hadi et al. [7] have employed an optimization algorithm based on Bacterial Foraging for neural network learning. Recently, Kulluk et al. [8] have introduced the Harmony Search algorithm (HS) for the same context. They have shown that Self-adaptive Global Harmony Search (SGHS) achieved better results compared against with four different variants of HS.

Recently, Kaveh and Talatahari [9] proposed an optimization algorithm called Charged System Search (CSS), which is based on the interactions between electrically charged particles. The idea is that an electrical field of one particle generates an attracting or repelling force over other particles. This

L.A.M. Pereira, L.C.S. Afonso and J.P. Papa are with Department of Computing, Faculty of Science, São Paulo State University - UNESP, Bauru, Brazil. Email: papa@fc.unesp.br.

Z.A. Vale is with Knowledge Engineering and Decision Support Research Center - GECAD, Polytechnic Institute of Porto - IPP, Porto, Portugal. Email: zav@isep.ipp.pt.

A.N. Souza, C.C.O. Ramos, D.S. Gastaldello are with Department of Electrical Engineering, Polytechnic School, University of São Paulo - USP, São Paulo, Brazil. Emails: andrejau@feb.unesp.br, caioramos@gmail.com.

The authors would like to thank FAPESP grants #2009/16206-1, #2011/14094-1 and #2012/14158-2, CAPES, CNPq grant #303182/2011-3 and also People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme FP7/2007-2013/ under project ELECON - Electricity Consumption Analysis to Promote Energy Efficiency Considering Demand Response and Non-technical Losses, REA grant agreement No 318912.

978-1-4673-5274-1/13/$31.00 ©2013 IEEE

interaction is defined by physical principles such as Coulomb, Gauss and Newtonian laws. Such approach has demonstrated to outperform some well-known nature-inspired optimization techniques, and it has never been employed for ANN-MLP training phase. Therefore, this work aims to introduce CSS for ANN-MLP's weights computation. The proposed approach is then compared over two datasets obtained from a Brazilian power electric company in order to identify non-technical losses. In addition, we have proposed a modification of the traditional CSS in order to position the charged particles that fall outside the search space using the Global-best Harmony Search (GHS) [10], instead of HS.

The remainder of this paper is organized as follows. Section II and III present the background theory of Artificial Neural Networks with Multi-Layer Perceptron and the Evolutionary-based Techniques, respectively, highlighting Charged System Search method. Section IV describes the methodology and Section V presents the experiments conducted in order to evaluate robustness of the proposed work. Conclusions are stated in Section VI.

II. Multi-Layer Perceptron

Multilayer Perceptron is a combination of Perceptron layers aiming to solve multiclass problems [4]. The neural network architecture is composed of neuron layers, such that each output feeds the input neurons at the following layer. The first layer, denoted by A, has N_A neurons, where N_A has the same dimensionality of the feature vector, while the last layer, denoted by Q, has N_Q neurons, which stands for the number of the classes. This neural network assigns a pattern vector $\vec{p}$ to a class ω_m if the m-th output neuron achieves the highest value, $m = 1, 2, \ldots, N_Q$.

Each input layer corresponds to a weighted sum of the previous layer. Let $J-1$ be the previous layer of J, such that each input I_j^J in J is given by:

$$I_j^J = \sum_{k=1}^{N_{J-1}} w_{jk} O_k^{J-1} \tag{1}$$

and

$$O_k^{J-1} = \phi(I_k^{J-1}), \tag{2}$$

where $j = 1, 2, \ldots, N_J$, being N_J and N_{J-1} the number of neurons at the layer J and $J-1$, respectively, w_{jk} stands for the weights that modify the k^{th} output of layer $J-1$, i.e., O_k^{J-1}, and $\phi(\cdot)$ corresponds to the activation function.

The Backpropagation algorithm is usually employed to train the ANN-MLP approach [11]. This algorithm minimizes the mean squared error between the desired outputs r_q and the obtained outputs Φ_q of each node of the output layer Q. Therefore, the idea is to minimize the equation bellow:

$$E_Q = \frac{1}{N_Q} \sum_{q=1}^{N_Q} (r_q - \Phi_q)^2, \tag{3}$$

in which N_Q is the number of neurons at the output layer Q.

III. Evolutionary-based Techniques

In this paper, we address three approaches to this problem: Charged System Search, Particle Swarm Optimization and Self-adative Global best Harmony Search. The next sections discuss these techniques in the context of feature selection.

A. Charged System Search

The governing Coulomb's law is a physics law used to describe the interactions between electrically charged particles. Let a charge be a solid sphere with radius r and uniform density volume. The attraction force F_{ij} between two spheres i and j with total charges q_i and q_j is defined by:

$$F_{ij} = \frac{k_e q_i q_j}{d_{ij}^2}, \tag{4}$$

where k_e is a constant called the Coulomb constant and d_{ij} is the distance between the charges.

Based on aforementioned definition, Kaveh and Talatahari [9] have proposed a new metaheuristic algorithm called Charged System Search (CSS). In this algorithm, each Charged Particle (CP) on system is affected by the electrical fields of the others, generating a resultant force over each CP, which is determined by using the electrostatics laws. The CP interaction movement is determined using Newtonian mechanics laws. Therefore, Kaveh and Talatahari [9] have sumarized CSS over the following definitions:

- *Definition 1*: The magnitude of charge q_i, with $i = 1, 2, ..., n$, is defined considering the quality of its solution, i.e. objective function value $fit(i)$:

$$q_i = \frac{fit(i) - fitworst}{fitbest - fitworst}, \tag{5}$$

  where $fitbest$ and $fitworst$ denote, respectively, the so far best and the worst fitness of all particles. The distance d_{ij} between two CPs is given by the following equation:

$$d_{ij} = \frac{\|\vec{x}_i - \vec{x}_j\|}{\|\frac{\vec{x}_i - \vec{x}_j}{2} - \vec{x}_{best}\| + \epsilon}, \tag{6}$$

  in which $\vec{x}_i$, $\vec{x}_j$ and $\vec{x}_{best}$ denote the positions of the ith, jth and the best current CP respectively, and ϵ is a small positive number to avoid singularities.
- *Definition 2*: The initial position $x_{ij}(0)$ and velocity $v_{ij}(0)$, for each jth variable of the ith CP, with $j = 1, 2, \ldots, m$, is given by:

$$x_{ij}(0) = x_{i,min} + \theta(x_{i,max} - x_{i,min}) \tag{7}$$

  and

$$v_{ij}(0) = 0, \tag{8}$$

  where $x_{i,max}$ and $x_{i,min}$ represents the upper and low bounds respectively, and $\theta \sim U(0, 1)$.
- *Definition 3*: For maximization problem, the probability of each CP moves toward others CPs is given as follow:

$$p_{ij} = \begin{cases} 1 & \text{if } \frac{fit(j) - fitworst}{fit(i) - fit(j)} > \theta \vee fit(i) > fit(j) \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

- *Definition 4*: The value of the resultant force acting on a CP j is defined as:

$$r = 0.1\text{max}(x_{i,max} - x_{i,min}) \tag{10}$$

$$F_j = q_j \sum_{j,i \neq j} \left(\frac{q_i}{r^3} \cdot d_{ij} \cdot c_1 + \frac{q_i}{d_{ij}^2} \cdot c_2 \right) p_{ij}(\vec{x_i} - \vec{x_j}), \tag{11}$$

where $c_1 = 1$ and $c_2 = 0$ if $d_{ij} < r$, otherwise $c_1 = 0$ and $c_2 = 1$.
- *Definition 5*: The new position and velocity of each CP is given by

$$\vec{x}_j(t) = \theta_{j1} \cdot k_a \cdot F_j + \theta_{j2} \cdot k_v \cdot \vec{v}_j(t-1) + \vec{x}_j(t-1) \tag{12}$$

and

$$\vec{v}_j(t) = \vec{x}_j(t) - \vec{x}_j(t-1), \tag{13}$$

where $k_a = 0.5(1 + \frac{t}{T})$ and $k_v = 0.5(1 - \frac{t}{T})$ are the acceleration and the velocity coefficients respectively, being t the actual iterations and T the maximum number of iterations.
- *Definition 6*: A number of the best so far solutions is saved using a Charged Memory (CM). The worst solutions are excluded from CM, and better new ones are included to the CM.

B. Particle Swarm Optimization

Recently, several applications have used evolutionary techniques as heuristic methods to find optimal or quasi-optimal solutions. A particular attention has been devoted to Particle Swarm Optimization (PSO) due to its simplicity and effectiveness. Basically, PSO is an algorithm modeled on swarm intelligence that finds a solution in a search space based on the social behavior dynamics [12]. Each possible solution of the problem is modeled as a particle in the swarm that imitates its neighborhood based on a fitness function.

Other definitions consider PSO as a stochastic and population-based search algorithm, in which the social behavior learning allows each possible solution (particle) to "fly" onto this space (swarm) looking for other particles that have better characteristics, i.e., the ones that maximize a fitness function. Each particle has a memory that stores its best local solution (local maxima) and the best global solution (global maxima). Taking into account this information, each particle has the ability to imitate the other ones that give to it the best local and global maxima. This process simulates the social interaction between humans looking for the same objective or bird flocks looking for food, for instance.

The entire swarm is modeled in a multidimensional space $\Re^m$, in which each particle $p_i = (x_i, v_i) \in \Re^m$ has two main features: (i) position (x_i) and (ii) velocity (v_i). The local (best current position $\widehat{x}_i$) and global solution $\widehat{g}$ are also known. After defining the swarm size, i.e., the number of particles, each one of them is initialized with random values of both velocity and position. Each individual is then evaluated with respect to some fitness function and its local maximum is updated. At the end, the global maximum is updated with the particle that achieved the best position in the swarm. This process is repeated until some convergence criterion be reached. The updated position and velocity equations of the particle p_i in the simplest form that govern the PSO are, respectively, given by

$$v_i = wv_i + c_1 r_1(\widehat{x}_i - x_i) + c_2 r_2(\widehat{g} - x_i) \tag{14}$$

and

$$x_i = x_i + v_i, \tag{15}$$

where w is the inertia weight that controls the interaction power between particles, and $r_1, r_2 \in [0, 1]$ are random variables that give the idea of stochasticity to PSO. Constants c_1 and c_2 are used to guide particles onto good directions.

C. Self-adaptive Global best Harmony Search

In the last years, several research have focused to develop variants from traditional Harmony Search (HS) [13]. Some of them propose ways to dynamically set HS parameters, others propose a new improvisation schemes etc. Mahdavi et al. [14], for instance, proposed an Improved Harmony Search (IHS), which differs from traditional HS by updating the PAR and bandwidth values dynamically.

Later, Omran and Mahdavi [10] proposed Global-best Harmony Search (GHS), which has exactly the same steps as the IHS, except by employing a new pitch adjustment rule, generating a new solution by making use of the best harmony vector. Zou et al. [15] proposed the Novel Global Harmony Search (NGHS), which differs from traditional HS in three aspects: in the NGHS the HMCR and PAR parameters are excluded, and a mutation probability p_m is then used. NGHS has an improvisation scheme modifyed, and it always replaces the worst harmony with the new one.

Finally, Pan [16] proposed Self-adaptive Global best Harmony Search (SGHS), which has a new improvisation scheme based on GHS. In addition, HMCR and PAR are self-adaptive parameters.

1) Traditional Harmony Search: Harmony Search (HS) is an evolutionary algorithm inspired in the improvisation process of music players [17]. HS is simple in concept, has few parameters, and it is easy to implement, with theoretical background for a stochastic derivative.

The main idea is to use the same process adopted by musicians to create new songs to obtain a near-optimal solution for some optimization process. Basically, any possible solution is modeled as a harmony and each parameter to be optimized can be seen as a musical note. The best harmony (solution) is chosen as the one that maximizes some optimization criteria.

The algorithm is composed by few steps, as described below:

- Step 1: Initialize the optimization problem and algorithm parameters;
- Step 2: Initialize a Harmony Memory (HM);
- Step 3: Improvise a new harmony from HM;
- Step 4: Update the HM if the new harmony is better than the worst harmony in the HM, include the new harmony in HM, and remove the worst one from HM; and

- Step 5: If the stopping criterion is not satisfied, go to Step 3.

Below, we discuss each one of the aforementioned steps.

a) The Optimization Problem and Algorithm Parameters: In order to describe how HS works, an optimization problem is specified in Step 1 as follows:

$$\max \ f(x) \quad \text{subject to } x^j \in X_j, \forall j = 1, 2, \ldots, m, \qquad (16)$$

where $f(x)$ is the objective function, x^j and X_j, mean, respectively, the design variable j and its set of possible values, and m is the number of design variables. Notice that $X_j \in \{0, 1\}$ in the case of feature selection, and this extends to all variables $j = \{1, 2, \ldots, m\}$. Thus, in this case, we can generalize X_j to X.

The HS parameters required to solve the optimization problem (16) are also specified in this step. They are: the harmony memory size (HMS), the harmony memory considering rate (HMCR), the pitch adjusting rate (PAR), and the stopping criterion. HMCR and PAR are parameters used to improve the solution vector, i.e., they can help the algorithm to find globally and locally improved solutions in the harmony search process (Step 3).

b) Harmony Memory (HM): In Step 2, the HM matrix (17) is initialized with randomly generated solution vectors with their respective values for the objective function:

$$HM = \left[\begin{array}{cccc|c} x_1^1 & x_1^2 & \ldots & x_1^m & f(x_1) \\ x_2^1 & x_2^2 & \ldots & x_2^m & f(x_2) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{HMS}^1 & x_{HMS}^2 & \ldots & x_{HMS}^m & f(x_{HMS}) \end{array}\right], \qquad (17)$$

where x_i^j denotes the decision variable j from harmony i.

c) Generating a New Harmony from HM: In Step 3, a new harmony vector $\widehat{x}$=($\widehat{x}^1$,$\widehat{x}^2$,..., $\widehat{x}^m$) is generated from the HM based on memory considerations, pitch adjustments, and randomization (music improvisation). It is also possible to choose the new value using the HMCR parameter, which varies between 0 and 1 as follows:

$$\widehat{x}^j \leftarrow \begin{cases} \widehat{x}^j \in \left\{x_1^j, x_2^j, \ldots, x_{HMS}^j\right\} & \text{with probability HCMR,} \\ \widehat{x}^j \in X & \text{with probability (1-HCMR).} \end{cases} \qquad (18)$$

The HMCR is the probability of choosing one value from the historic values stored in the HM, and (1- HMCR) is the probability of randomly choosing one feasible value not limited to those stored in the HM.

Further, every component j of the new harmony vector $\widehat{x}$ is examined to determine whether it should be pitch-adjusted:

$$\text{Pitching adjusting decision for } \widehat{x}^j \leftarrow \begin{cases} \textbf{Yes} \text{ with probability PAR,} \\ \textbf{No} \text{ with probability (1-PAR).} \end{cases} \qquad (19)$$

The pitch adjustment for each instrument is often used to improve solutions and to escape from local optima. This mechanism concerns shifting the neighboring values of some decision variable in the harmony.

In such a way, if the pitch adjustment decision for the decision variable $\widehat{x}^j$ is Yes, $\widehat{x}^j$ is replaced as follows:

$$\widehat{x}^j \leftarrow \widehat{x}^j + rb, \qquad (20)$$

where b is an arbitrary distance bandwidth for the continuous design variable, and r is a uniform distribution between 0 and 1. In case of binary optimization problems, this computation is not used.

d) Update HM: In Step 4, if the new harmony vector is better than the worst harmony in the HM, the latter is replaced by this new harmony.

e) Stopping Criterion: In Step 5, the HS algorithm finishes when it satisfies the stopping criterion. Otherwise, Steps 3 and 4 are repeated in order to improvise a new harmony again.

IV. Methodology

In this work, we propose to train the ANN-MLP using evolutionary algorithms where the main idea is use CSS, PSO and SGHS to perform a global search and update the weights and biases of the neural network. The next sections describe the proposed methodology as well as the employed datasets.

A. ANN-MLP training through Charged System Search

As aforementioned, the idea is use CSS to perform a global search and update the weights and biases of the neural network. Firstly, the weight vectors (Charged Particles) are randomly initialized with values within the interval $[-1, 1]$. In order to handle with particles that violate the limits of the search space, i.e., values outside the interval $[-1, 1]$, Kaveh and Talatahari [9] have proposed to use a Harmony Search-based approach. Therefore, a new solution generated by Equation 12 that exceeds the boundaries can be regenerated according to the HS mechanism [18]. Variants of the traditional HS can be employed in this task. Thereby, we have chosen the Global Harmony Search (GHS) [10], which employ a new pitch adjustment rule, generating a new solution by making use of the best harmony vector, in our case, it uses the best CP in Charged Memory.

The Mean Squared Error (MSE), already presented in Section II (Equation 3), is used as the fitness function to guide the CPs toward the best solutions. The stop criterion is the number of iterations of CSS, and the best CSS global solution is used as the weight configuration in the ANN-MLP testing phase.

B. Datasets

We used two labeled datasets obtained from a Brazilian electric power company, B_i and B_c. The former is a dataset composed of 3486 industrial profiles, and the latter contains 5645 commercial profiles. Each industrial and commercial profile is represented by eight features, as follows:

- Demand Billed (DB): demand value of the active power considered for billing purposes, in kilowatts (kW);

- Demand Contracted (DC): the value of the demand for continuous availability requested from the energy company, which must be paid whether the electric power is used by the consumer or not, in kilowatts (kW);
- Demand Measured or Maximum Demand (D_{max}): the maximum actual demand for active power, verified by measurement at fifteen-minute intervals during the billing period, in kilowatts (kW);
- Reactive Energy (RE): energy that flows through the electric and magnetic fields of an AC system, in kilovolt-amperes reactive hours (kVArh);
- Power Transformer (PT): the power transformer installed for the consumers, in kilovolt-amperes (kVA);
- Power Factor (PF): the ratio between the consumed active and apparent power in a circuit. The PF indicates the efficiency of a power distribution system;
- Installed Power (P_{inst}): the sum of the nominal power of all electrical equipment installed and ready to operate at the consumer unit, in kilowatts (kW);
- Load Factor (LF): the ratio between the average demand ($D_{average}$) and maximum demand (D_{max}) of the consumer unit. The LF is an index that shows how the electric power is used in a rational way.

We have employed the following neural architecture: λ=8:ψ=15:χ=2, in which λ denotes the number of neurons of the input layer (number of features), and ψ and χ stand for the number of neurons of the hidden and output (number of classes: regular or irregular consumer profile) layers, respectively.

C. Parameters setting

We have compared our proposed approach with Particle Swarm Optimization (PSO), Self-adaptive Global Best Harmony Search (SGHS) and the well-known Backpropagation (BP) training algorithm. Table I presents the parameters employed for each evolutionary-based technique. Notice for all techniques we employed 50 agents with 10.000 iterations. These parameters have been empirically set.

TABLE I
PARAMETERS SETTING OF META-HEURISTIC ALGORITHMS.

Technique	Parameters
CSS	HMCR $= 0.9$, PARmin $= 0.01$, PARmax $= 0.98$
PSO	$c_1 = 2.0$, $c_2 = 2.0$, $w = 0.7$
SGHS	HMCRm $= 2.0$, BWmin $= 0.0005$, BWmax $= 0.2$
	PARm $= 2.0$, LP $= 100$, LB $= -1.0$, UB $= 1.0$

V. Experimental Results

In this section we discuss the experiments conducted in order to assess the robustness of CSS-based algorithms against with PSO, SGHS and BP for ANN-MLP training. It is worth nothing that the experiments were carried out using 10-fold cross validation and were executed in a computer with a Pentium Intel Core $i7$® 2.8 Ghz processor, 4 GB of memory RAM and Linux Ubuntu Desktop LTS 11.04 as the operational system.

Table II displays the MSE values achieved by each algorithm in the training phase over the datasets. We can see that the optimization techniques achieved lower error rates than BP for the two datasets. For the B_c dataset, PSO and CSS give the best results, respectively. In case of B_i, PSO achieved the lowest error rate.

TABLE II
AVERAGE MSE

	Dataset	
Technique	B_c	B_i
CSS	0.025±0.0003	0.027±0.0006
PSO	0.024±0.0004	0.023±0.0008
SGHS	0.029±0.002	0.031±0.0023
BP	0.105±0.00004	0.114±0.001

Table III shows the ANN-MLP accuracy rate over the test set using the best solutions achieved by all evaluated techniques. For B_c all optimization techniques achieved good results with very close similarity. For the B_i dataset, PSO and CSS presented the best accuracy rates followed by SGHS with about 1% less accurate. Therefore, according to mean testing accuracies, we can consider that CSS and PSO have similar accuracy rates related to non-technical losses detection.

TABLE III
AVERAGE ACCURACY.

	Dataset	
Technique	B_c	B_i
CSS	94.54±0.006	94.09±0.751
PSO	94.58±0.085	94.50±0.690
SGHS	94.54±0.006	93.81±0.572
BP	50.00±0.0	52.24±2.744

TABLE IV
AVERAGE OF EXECUTION TRAINING TIME IN SECONDS.

	Dataset	
Technique	B_c	B_i
CSS	2230.89±27.87	1446.71±30.73
PSO	2003.10±75.72	1299.87±40.88
SGHS	42.58±0.60	27.72±0.13
BP	621.88±151.16	394.25±80.0355

In regard to ANN-MLP training time (Table IV), one can see that SGHS was the fastest technique, followed by PSO and CSS, which have very similar computational load.

VI. Conclusion

Non-technical losses in power distribution systems are related to the energy consumed and that is not billed, representing a problem for power companies, since these losses are strongly related to fraud and theft of energy. In this work, we have introduced the Charged System Search and Particle Swarm Optimization for Multi-Layer Perceptron network training related to non-technical losses detection, which has never been employed for such purpose before. In regard to

the experimental results, CSS and PSO have similar accuracy rates related to non-technical losses detection and we can consider using evolutionary techniques instead of the traditional Backpropagation algorithm to improve the accuracy rates in Smart Grid applications.

REFERENCES

[1] R. Alves, P. Casanova, E. Quirogas, O. Ravelo, and W. Gimenez, "Reduction of non-technical losses by modernization and updating of measurement systems," in *Proceedings of the IEEE/PES Transmission and Distribution Conference and Exposition: Latin America*, 2006, pp. 1–5.

[2] C. C. O. Ramos, A. N. Souza, J. P. Papa, and A. X. Falcão, "Fast non-technical losses identification through optimum-path forest," in *Proceedings of The 15th International Conference on Intelligent System Applications to Power Systems (ISAP)*, 2009, pp. 1–5.

[3] X. Fang, S. Misra, G. Xue, and D. Yang, "Smart grid the new and improved power grid: A survey," *IEEE Communications Surveys and Tutorials*, vol. PP, pp. 1–37, 2011.

[4] S. Haykin, *Neural Networks: A Comprehensive Foundation (3rd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2007.

[5] *Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks*, 2003.

[6] D. Karaboga, B. Akay, and C. Ozturk, "Artificial bee colony (abc) optimization algorithm for training feed-forward neural networks," in *Proceedings of the 4th international conference on Modeling Decisions for Artificial Intelligence*, ser. MDAI '07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 318–329.

[7] I. AL-Hadi, S. Hashim, and S. Shamsuddin, "Bacterial foraging optimization algorithm for neural network learning enhancement," in *Hybrid Intelligent Systems (HIS), 2011 11th International Conference on*, 2011, pp. 200 –205.

[8] S. Kulluk, L. Ozbakir, and A. Baykasoglu, "Training neural networks with harmony search algorithms for classification problems," *Eng. Appl. Artif. Intell.*, vol. 25, no. 1, pp. 11–19, Feb. 2012.

[9] A. Kaveh and S. Talatahari, "A novel heuristic optimization method: charged system search," *Acta Mechanica*, vol. 213, no. 3, pp. 267–289, 2010.

[10] M. G. Omran and M. Mahdavi, "Global-best harmony search," *Applied Mathematics and Computation*, vol. 198, no. 2, pp. 643 – 656, 2008.

[11] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach (3rd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, 2009.

[12] J. Kennedy and R. Eberhart, *Swarm Intelligence*. M. Kaufman, 2001.

[13] O. Alia and R. Mandava, "The variants of the harmony search algorithm: an overview," *Artificial Intelligence Review*, vol. 36, pp. 49–68, 2011.

[14] M. Mahdavi, M. Fesanghary, and E. Damangir, "An improved harmony search algorithm for solving optimization problems," *Applied Mathematics and Computation*, vol. 188, no. 2, pp. 1567 – 1579, 2007.

[15] D. Zou, L. Gao, J. Wu, and S. Li, "Novel global harmony search algorithm for unconstrained problems," *Neurocomputing*, vol. 73, no. 1618, pp. 3308 – 3318, 2010.

[16] Q.-K. Pan, P. Suganthan, M. F. Tasgetiren, and J. Liang, "A self-adaptive global best harmony search algorithm for continuous optimization problems," *Applied Mathematics and Computation*, vol. 216, no. 3, pp. 830 – 848, 2010.

[17] Z. W. Geem, "Novel derivative of harmony search algorithm for discrete design variables," *Applied Mathematics and Computation*, vol. 199, pp. 223–230, 2008.

[18] ——, *Music-Inspired Harmony Search Algorithm: Theory and Applications*, 1st ed. Springer Publishing Company, Incorporated, 2009.