2018 15th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)
Mexico City, Mexico. September 5-7, 2018

Experimental Spiking Neural Network: Solving the XOR Paradigm with Metaheuristics

J. Enríquez-Gaytán[1], F. Gómez-Castañeda[1], J.A. Moreno-Cadenas[1], L.M. Flores-Nava[1].
Electrical Engineering Department, Cinvestav-IPN, Mexico City, Mexico
phone (52) 55 5747 3800 Ext. 6261
e-mail: {jenriquezg, fgomez, jmoreno, lmflores}@cinvestav.mx

Abstract— **This work presents a supervised training strategy applied to a biorealistic Spiking Neural Network (SNN) with feedforward 2-2-1 architecture. This network uses Izhikevich neurons with regular-spiking behavior. The input layer, which has 2 nodes, generates temporal pulse trains that pass through synaptic conductances. These conductances transform voltages into currents. The receiving currents by 2 hidden-neurons also generate voltage pulses into synaptic conductances towards the output neuron. Each synaptic conductance has 2-parallel Alpha functions, whose weighting factors are found by the Efficient Artificial Bee Colony Algorithm (EABC Algorithm). This is a variant of the Artificial Bee Colony Algorithm (ABC Algorithm). The efficacy of the EABC algorithm in this SNN is shown solving the XOR paradigm.**

Keywords— Spiking neural network; Efficient artificial bee colony algorithm; Synapic conductance;

I. Introduction

The onset of the Artificial Neural Network (ANN) contains an extremely simple abstraction of biological neurons, but it does not prevent them to have the ability to learn, creating its own representation of the information (self-organization), fault tolerance, flexibility and response at real time.

The ANNs have three basic components namely, the cell body, the axon and a set of dendrites with synapsis. The axon is able to send electrical signals or spikes, which are also called action potentials, through long nerve fibers; these electrical signals come from the cell body. The cell body integrates all the inputs in the manner an electrical capacitor works. The dendrites are the responsible for receiving the electrical signals among neurons. There are synapses in the inter-spaces between dendrites that behave as weighting factors and physiologically behave as a variable conductance. The SNN is considered the third generation of ANNs by modeling biological brain functions with temporal spike train events. Alan Lloyd Hodgkin and Andrew Huxley made different experiments with a squirt axon deducing the model that bears their name. At present, the Hodgkin-Huxley model is computationally unsuitable to realize simulations for large SNN. Alternatively, there exists the spiking Integrate-and-Fire model, but it deviates from real representations [1]. In this work, we use the Izhikevich model [2], which generates the temporal spiking patterns in the Hodgkin-Huxley model but with reduced numerical computation. The Izhikevich model can reproduce spiking behavior of known types of cortical neurons depending of four parameters; in particular for this work, we select those that generate regular spiking (RS). Even though, the SNN models in experimental neuroscience explain the Spike-Time Dependent Plasticity algorithm taking place in synapses as a process for adjusting their weighting factors, in this paper we use the EABC algorithm, which is suitable as a numerical program. The XOR non-linear classification problem is a vehicle to demonstrate that this metaheuristic approach might deal with training SNNs of any complexity, successfully.

This paper is organized as follows. In Section II, this work reviews the manner SNNs have been trained. Section III introduces the concepts that support the existing metaheuristic algorithms as optimizers. The EABC algorithm is presented in brief in Section IV with its variant used for the XOR problem. In Section V, the synaptic conductance and the Izhikevich neuron models are shown. The architecture of the SNN used in this work is shown in Section VI. The training scheme of the SNN using the EABC algorithm with representative numerical results, are presented in Section VII. The Conclusion Section ends this paper.

II. Training SNNs

In order to have a suitable SNN solving a specific task in complex engineering problems, there should be a previous training stage supported with an optimizer that fixes its weighting factors. The optimizer is responsible for being efficient and fairly accurate.

The classical optimizer in layered architectures of SNNs has numerical algorithms [3], [4]that find the weighting factors based on the principle of backpropagating the error. This training procedure observes the minimization of the Mean Square Error using gradient operators for temporal series of spikes. In other algorithms that take into account means, variances and correlations in spike patterns, training is also possible for SNNs [5]. Alternatively, SNNs were early trained with genetic and evolutionary algorithms that avoid using numerical derivatives with acceptable results for classification of nonlinear separable data [6].This optimization approach, which is inspired in darwinian principles, establishes strategies proper in metaheuristics, discarding and keeping candidate-solutions, until it arrives to the optimal one.

III. Metaheuristic Algorithms

In general, the metaheuristic algorithms evaluate an objective function $f(x)$, with or without restrictions and whose minima might be found following the collective

978-1-5386-7033-0/18/$31.00 ©2018 IEEE

organization in living beings, for example by bees and ants when they forage or explore, respectively. However, more optimal-value search strategies are possible inspired in biology or nature. The use of memory for important states in the course of search is crucial. The metaheuristic algorithms define firstly a space of search, in which the mechanisms of Exploration and Exploitation should be addressed with good balance for providing acceptable algorithm performance [7]. These features are kept by the Artificial Bee Colony algorithm; also one of its extensions used here for the XOR problem.

IV. ABC and EABC algorithms

The ABC algorithm mimics the intelligent behavior of a group of bees when they forage (There are three kinds of bees: *employee* bee, *onlooker* bee and *scout* bee). The ABC algorithm presents an excellent performance. However, in this work we have used the Efficient ABC algorithm or EABC algorithm [8], to improve the behavior of the bees, whose main differences are evident below.

A global optimization problem is solved when a parameter vector x minimizes the objective function $f(x)$.

$$where: x = (x_1, x_2, \dots, x_i, .. x_{n-1}, x_n,) \in \mathbb{R}^n \qquad (1)$$
$$l_i \le x_i \le u_i, \qquad i = 1, \dots, n$$

The function $f(x)$ is defined as the search space with dimension n. The variables x are constrained by l_i and u_i that limit the domain. Initially, a colony with D elements is divided in two groups of bees, one half is the employee bees and the other half is the onlooker bees (N= colony size/2). Each source of food is a vector $x = \{x_1, x_2, \dots, x_D\}$ that represents a candidate solution, these are the parameter to be optimized and are initialized randomly as follows.

$$x_{i,j} = u_{i,j} + \phi_{i,j}(u_{i,j} - l_{i,j}) \qquad (2)$$

Where: $i = 1,2, \dots N,\ j = 1,2, \dots, D,\ \phi_{ij}$ is a random number between [0,1]. After initialization, the objective function is evaluated in each food source.

We define the working phase of the algorithm associating the kind of bee that realizes it.

The **employee bee phase** is controlled by (3).

$$V_{i,j} = X_{N1j} + \theta_{i,j}(X_{N2,j} - X_{i,j}) \qquad (3)$$

Where: N1, N2 and i are exclusive each other; with $N1, N2 \in \{1,2,3, \dots, N\}$, $j \in \{1,2,3, \dots, D\}$ and are selected randomly. Θ_{ij} is a random number between [-0.25, 0.25]. In the phase of employee bee, the ABC algorithm searches around its own food source; on the other hand, in the EABC algorithm the search is around other food source. Next, a greedy selection is applied between the new food source (V_{ij}) and the original food source (X_{ij}).

The **onlooker bee phase** is controlled by (4).

$$V_{i,j} = X_{best,j} + \Phi_{i,j}(X_{N1,j} - x_{N2,j}) \qquad (4)$$

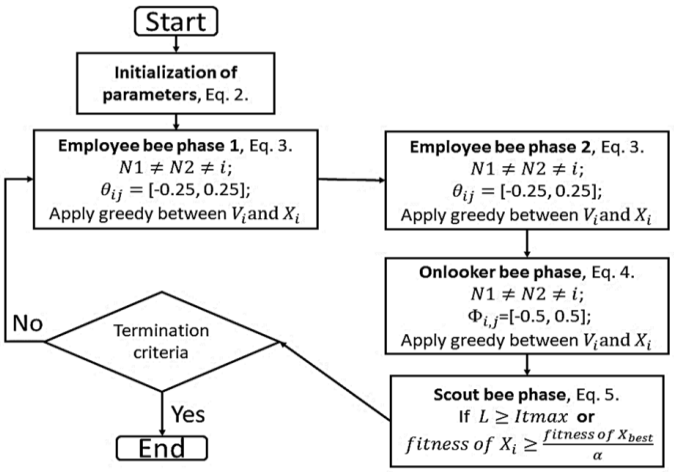
Where: N1, N2 and i are exclusive each other; with $N1, N2 \in \{1,2,3, \dots, N\}$, $j \in \{1,2,3, \dots, D\}$ and selected randomly. Φ_{ij} is a random number between [-0.5, 0.5]. The onlooker bee takes care of the exploitation. There is a difference with the ABC algorithm, where the onlooker bee selects a food source depending on the fitness that is defined by the employee bee and uses a probability process for selecting a food source.

The **scout bee phase** begins when the food source cannot be improved in a determined number of trials (L) or whose fitness is very less in comparison with the best food source found, so far. The scout bee proposes a new food source by Eq. (5).

$$U_i = X_{best,j} + \theta_i(X_{N1} - x_{N2}) \qquad (5)$$
$$X_{ij} = \begin{cases} X_{i,j} & if\ Random[0,1] > p \\ U_{i,j} & if\ Random[0,1] < p \end{cases}$$

Where: $N1$ and N2 are different each other; with $N1, N2 \in \{1,2,3, \dots, N\}$, and selected randomly. θ_i is a random number between [-0.25, 0.25] and p is a constant between 0 and 1.

The flow diagram of the EABC algorithm is shown below.



The EABC performs twice the phase of employee bee. This fact enhances the exploitation mechanism. The onlooker bee phase performs the exploration mechanism. The scout bee phase searches the food sources.

V. Synaptic conductance & Izhikevich models

There are three fundamental and biologically supported synaptic models, which can be taken into account for SNNs [1]. They approach stochastic properties due to noise in the nervous system. They model the way neurotransmitters pass through them, observing the change of value of the conductance. We choose for this work that one characterized by the Eq. 6, which is called Alpha function.

$$g_{syn}(t) = K_{syn} t e^{-t/\tau} \qquad (6)$$

Where: $g_{syn}(t)$ is the synaptic conductance, τ is a time constant and it controls the rise and decay times and, K_{syn} establishes the peak conductance value. The Alpha function is valid from the time when a single spike appears.

Fig. 1a shows one Alpha function that is stimulated with consecutive spikes, see Fig. 1b, coming from a single neuron, which is modeled by Izhikevich and its response in Fig. 1c.

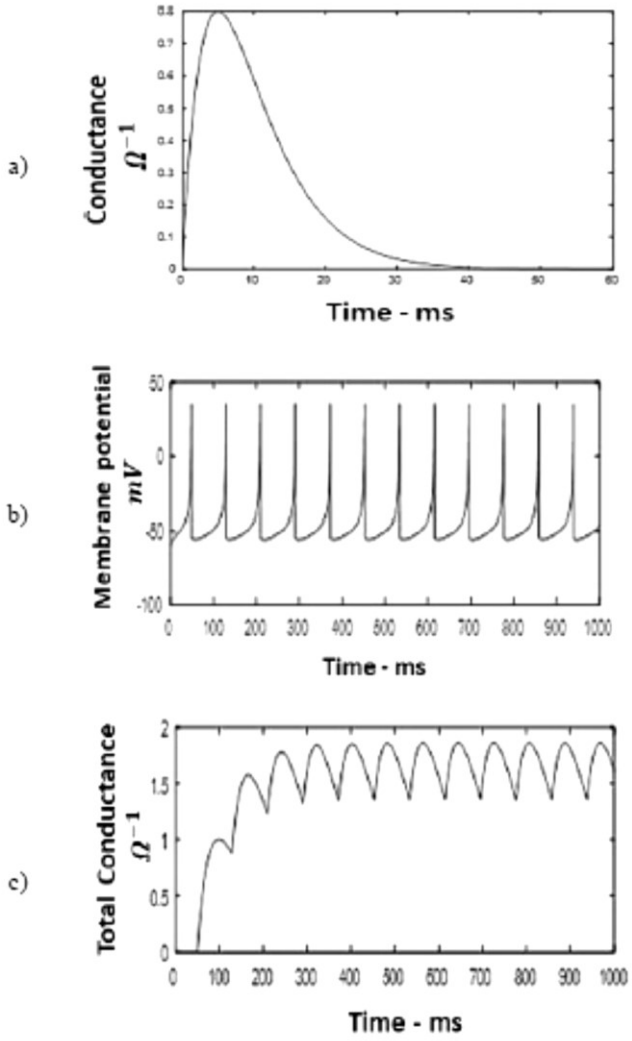


Fig. 1 a) Alpha function, b) Stimulating spikes and c) Synaptic current response

The second model used for the SNN in this work, is Izhikevich's neuron model, whose analytical representation is given by Eq. (7), where a set of two coupled ordinary differential equations represent the membrane potential and the recovery current, respectively.

$$C\frac{dV}{dt} = k(V - V_{rest})(V - V_{th}) - U + I_{syn} \quad (7a)$$

$$\frac{dU}{dt} = a[b(V - V_{rest}) - U] \quad (7b)$$

$$if\ V \geq V_{peak}\ then$$

$$V = c\ and\ U = U + d$$

Where: V is the membrane voltage, U is the recovery current, C is the membrane capacitance, V_{rest} is the resting membrane potential and V_{th} is the instantaneous threshold potential. The constant a is the recovery time and, the sign of b, which is another constant, determines if U is an amplifying or resonant variable.

The input current to the spiking neuron is: I_{syn} and denotes the sum of all input currents from neighboring neurons. This current is evaluated by Eq. (5).

$$I_{syn}(t) = \sum g(t)(E - V) \quad (8)$$

Where: $g(t)$ is the conductance between the reference neuron and the neighboring neurons from which current comes. Finally, E is a reference voltage.

VI. Architecture of the SNN

Fig. 2 shows the proposed architecture of the SNN that solves the XOR paradigm, which is a two-input non-linear classification problem. In neuroscience, the solution is assisted by the spike-time dependent plasticity algorithm [1]. Likewise, this is a training algorithm in the field of intelligent systems, which should precede to determine the weighting factors in SNNs. In Fig. 2, the weighting factors are the set of labels W01, W02, and so on, which represent the set $\{K_{syn}\}$ in Eq. (6). The inputs in any SNN are analog and might be coded as continuous-time or discrete-time variables; likewise for the output. We have chosen to code the inputs in frequency with pulse generators, marked as F1 and F2.

The algorithms that are used for training neural networks work as optimizers finding the optimal set of weighting factors. For the SNN in Fig. 2, the optimizer is the EABC algorithm. Table I summarizes the description of the elements in Fig. 2.

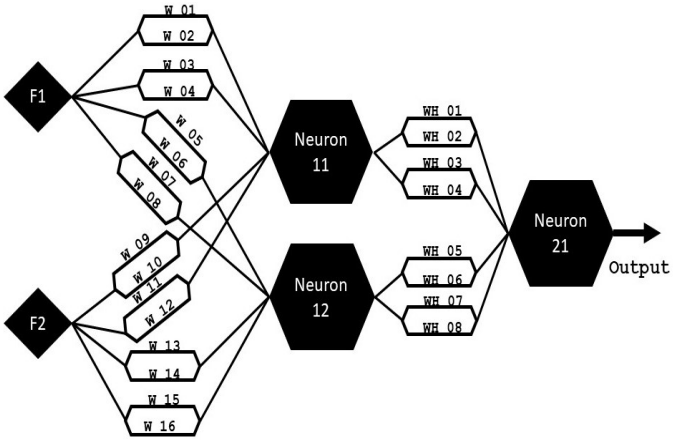


Fig. 2 Architecture of the SNN

Table I

Element	Characteristic
F1, Input F2, Input	Pulse generator F1 => fo = 50 Hz for "**0**" F2 => f1 = 100 Hz for "**1**"
Neuron 11 Neuron 12 Neuron 21	Izhikevich´s model, Eq. (7) V_{rest} -60mV a 0.03 V_{th} -40mV b -2 C 35mV c -50 K 0.7 d 100 T 1000ms τ 0.2ms
Synaptic Conductance Input layer *Excitatory synapsis* {W01, W03, W05, W07, W09, W11, W13, W15} *Inhibitory synapsis* {W02, W04, W06, W08, W10, W12, W14, W16} Hidden layer *Excitatory synapsis* {WH01, WH03, WH05, WH07} *Inhibitory synapsis* {WH02, WH04, WH06, WH08}	Alpha function, Eq. (6)

The weighting factors are arrayed in pairs; for example W01 and W02, which are excitatory and inhibitory synapses, respectively. This was done for convergence purposes in the training process with the EABC algorithm. Otherwise, the numerical solutions would be outside the dynamic behavior established by the condition if (.) then (.) for the Eq. (7).

VI. Training the SNN with EABC

In this section, we show the outcome of the SNN in Fig. 2, that was trained with the EABC algorithm. The training patterns, representing the symbols "**0**" and "**1**" are coded as uniform pulses with 50 Hz and 100 Hz, respectively. Likewise, the universe of values where the XOR problem is valid is inside the interval [45 Hz, 105 Hz], which is read as an analog variable at the output. The EABC algorithm observes an objective function, which is evaluated with the Mean Square Error operator or MSE given by Eq. (6). Where: y_i is the output of the SNN in Hz, y_{ob} is the the target in Hz and Nr is the the number of experiments realized.

The training process with EABC can find acceptable results when the MSE value is lower than 5 as can bee seen in graph MSE vs number of iterations, in Fig. 3. In each iteration, the best MSE is chosen from all the present food sources.

$$MSE = \frac{1}{Nr}\sum_{i=1}^{Nr}(y_i - y_{ob})^2 \qquad (9)$$

Table II presents the set of values used in the EABC algorithm.

Table II

Efficient ABC algorithm	
Parameter	**Value**
N	100
L	100
Max. Iteration	1000
u_i	0
l_i	100

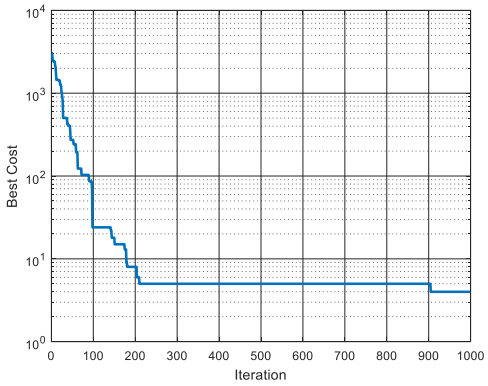


Fig. 3 Progress of MSE with iterations

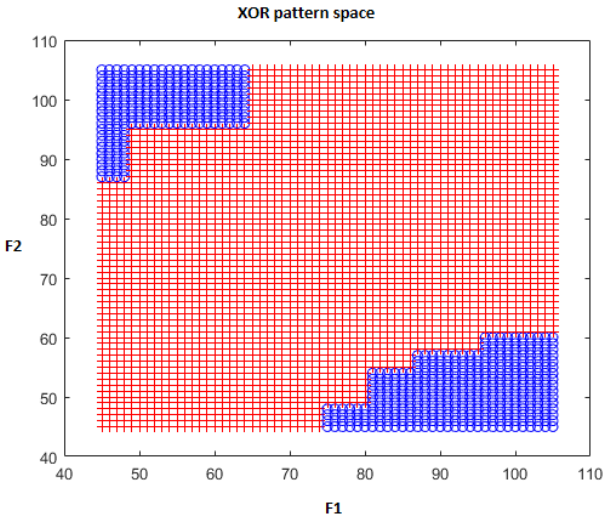


Fig. 4 Result of test of the trained SNN

Fig. 4 shows the result of the SNN after training, where the recognition of the symbols "**0**" and "**1**" have an extended definition of their domain. Therefore, the symbol "**0**" is defined in the interval [45 Hz, 75Hz] and the symbol "**1**" in the interval [76 Hz, 105 Hz]. Color keys: red for "**0**" and blue for "**1**".

Finally, Table III presents the values of the weighting factors found by the EABC algorithm that solves satisfactorily the XOR paradigm. We should mention that this set of values belongs to a set a possible solutions, due to the nature of the learning method namely, metaheuristic one.

Table III

W01	55.878	W09	44.004	WH01	46.695
W02	28.809	W10	12.885	WH02	40.143
W03	40.492	W11	31.403	WH03	86.579
W04	23.493	W12	3.166	WH04	31.266
W05	70.871	W13	29.423	WH05	6.560
W06	54.694	W14	22.096	WH06	2.046
W07	88.665	W15	9.995	WH07	20.613
W08	11.335	W16	85.331	WH08	100

VII. Conclusion

This work presented experimental results of training a small SNN for the two-input XOR classification problem, where the inputs are analog values represented as pulse frequency, in Hz. The key point in this work is choosing a pair of synapses: one excitatory and one inhibitory in parallel, where both represent a connection. This kind of connectivity allows numerical convergence when the EABC algorithm searches by metaheuristics the weighting factors of the SNN, which by itself is dificult to train due to its temporal nature according to the biorealistic model by Izhikevich.

Further research in the area of optimization with other metaheuristic algorithms deserves attention as long as new opportunities of using spiking neural networks become evident.

References

[1] P. Dayan and L. F. Abbott, Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems, MIT Press, 2005.

[2] E. M. Izhikevich, Dynamical Systems in Neuroscience: The Geometry of Excitability and Bursting, Cambridge, Massachusetts: The MIT Press, 2007.

[3] S. M. Bohte, J. N. Kok and H. Poutré, "Error-backpropagation in temporally encoded networks of spiking neurons," *Neurocomputing,* vol. 48, no. Issues 1–4, pp. 17-37, 2002.

[4] X. Xie, H. Qu, G. Liu, M. Zhang and J. Kurths, "An Efficient Supervised Training Algorithm for Multilayer Spiking Neural Networks," *PLoS One; 11(4): e0150329. doi: 10.1371/journal.pone.0150329,* 2016.

[5] P. Rowcliffe and J. Feng, "Training Spiking Neuronal Networks with Applications in Engineering Tasks," *IEEE TRANSACTIONS ON NEURAL NETWORKS,* vol. 19, no. 9, pp. 1626-1640, 2008.

[6] A. Belatreche, L. P. Maguire and M. McGinnity, "Advances in Design and Application of Spiking Neural Networks," *Soft Computing,* vol. 11, no. Issue 3, p. 239–248, 2007.

[7] P. Siarry, Metaheuristics, Switzerland: Springer International Publishing, 2016.

[8] P. Subhash and T. Rajesh, "An Efficient Artificial Bee Colony Algorithm and Analog Circuit Design Environment," *WSEAS Transactions on Circuits and Systems,* vol. 16, pp. 108-122, 2017.