2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

Evolutionary Features and Parameter Optimization of Spiking Neural Networks for Unsupervised Learning

Marco Silva, Adriano Koshiyama, Marley Vellasco, and Edson Cataldo

Abstract—**This paper introduces two new hybrid models for clustering problems in which the input features and parameters of a spiking neural network (SNN) are optimized using evolutionary algorithms. We used two novel evolutionary approaches, the quantum-inspired evolutionary algorithm (QIEA) and the optimization by genetic programming (OGP) methods, to develop the quantum binary-real evolving SNN (QbrSNN) and the SNN optimized by genetic programming (SNN-OGP) neuro-evolutionary models, respectively. The proposed models are applied to 8 benchmark datasets, and a significantly higher clustering accuracy compared to a standard SNN without feature and parameter optimization is achieved with fewer iterations. When comparing QbrSNN and SNN-OGP, the former performed slightly better but at the expense of increased computational effort.**

I. Introduction

CLUSTERING plays an important role in the machine learning literature [1,2] and also in real application domains, including market research [3] and medical analysis [4]. As new and more complex applications are developed (high-dimensional and scalable problems), classical techniques, such as k-means, hierarchical clustering, and k-medoids, tend to perform poorly [5]. Therefore, new unsupervised techniques are needed to generate more homogeneous and reliable groups.

Spiking neural networks (SNNs) are novel candidates for unsupervised learning [6]. SNNs are considered the third generation of neural networks and simulate how a neuron sends and receives information based on spikes. A neural spike is a discrete event within a continuous time frame with spatiotemporal properties. Previous studies have theoretically demonstrated that spiking neurons are computationally more powerful than neurons with sigmoidal activation functions [7, 8].

SNNs have been mostly applied to supervised learning [9] but some papers related to clustering problems can also be found [10, 11]. In [12], an evolving spiking neural network (eSNN) was proposed and applied to audio-visual pattern recognition [13]. Other applications include neural-based word recognition using liquid states [14], neural associative memory [15], and function approximation [16]. Previous studies [12-25] have used an evolutionary algorithm to fine-tuning the features and parameters of SNN for supervised learning, more specific for classification tasks.

In fact, fine-tuning the parameters of SNNs manually can be a challenging task due to the large number of adjustable parameter settings. Similar to other neural network models, the parameter refinement has a significant influence on the final performance of the SNN. Feature selection is also an important preprocessing tool to remove information that does not contribute to the unsupervised learning.

However, to the best of our knowledge, no previous work has applied evolving algorithms for feature selection and parameters tuning for unsupervised learning in SNN. In this paper, we present two different methods, based on the SNN presented in [10, 11], that evolve the SNN parameters and accomplish feature selection for clustering problems.

Therefore, this paper presents the evaluation of two new neuro-evolutionary approaches in which the SNN parameters are optimized and input features are automatically chosen to solve clustering problems. The objective is to evaluate the performance of both optimization methods, to verify which one is more suitable for this application.

The proposed models, named the quantum binary-real evolving SNN (QbrSNN) and the SNN optimized by genetic programming (SNN-OGP), are based on two novel evolutionary algorithms: the quantum evolutionary algorithm with binary-real representation (QIEA-BR) [26] and the optimization by genetic programming (OGP) method [27], respectively. The evolutionary algorithms present, for some problems, a slower performance than the one that could be expected. The quantum-inspired evolutionary algorithms have been used in combinatorial optimization problems using a binary-based representation and have presented a better performance than the conventional algorithms. Besides many other important properties, this model has the ability to find a good solution faster using fewer individuals. This feature reduces dramatically the number of evaluations needed and is an important performance factor when the model is being used in problems where each evaluation takes too much time to be completed.

The former evolutionary algorithm is a type of distribution algorithm, and the latter is a more classical approach based on Darwin's theory of evolution. We evaluated the two proposed models by applying them to 8 benchmark datasets.

The paper is organized as follows. Section II discusses the application of SNNs to unsupervised learning, and Section III introduces the QIEA-BR and OGP evolutionary algorithms. The framework of the proposed neuro-evolutionary models is presented in Section IV. Section V provides the details of our

M. Silva, M.M.B.R. Vellasco, and A. Koshiyama are with the Department of Electrical Engineering, Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Rua Marquês de São Vicente, 225, Rio de Janeiro, RJ Brazil, 22451-900 (phone: +55 21 3527-1630; fax: +55 21 3527-1232; email: {mabs21, marley, adriano}@ele.puc-rio.br).

E. Cataldo is with the Applied Mathematics Department, Graduate Program in Telecommunications Engineering, Federal Fluminense University (UFF), Rua Passos da Pátria, 156, São Domingos, Niterói, Rio de Janeiro, Brazil (email: ecataldo@im.uff.br).

This work was supported by CNPq and FAPERJ.

experiments and presents the results, and Section VI draws the conclusions of this work.

II. Unsupervised Learning with Spiking Neurons

A model of spiking neurons for clustering was proposed in [28] and extended in [29]. This model encodes the input patterns in the delays across its synapses and has been shown to identify the centers of high-dimensional clusters with high accuracy.

Several types of information-encoding methods exist in SNNs. The implementation based on population encoding, as proposed in [11], allows continuous values to be encoded by using a collection of neurons with overlapping sensitivity profiles. For a specific variable in an interval $\left[I^n_{min}, I^n_{max}\right]$, M neurons are used with Gaussian receptive fields. For the ith coding neuron of variable x_n, the center (μ) and width (σ) of each Gaussian receptive field are calculated by (1). Parameter γ directly controls the width of each Gaussian receptive field.

$$\mu = I^n_{min} + (2*i-3)/2*\left(I^n_{max} - I^n_{min}\right)/(M-2)$$
$$\sigma = 1/\gamma\left(I^n_{max} - I^n_{min}\right)/(M-2),\ 1 \le \gamma \le 2 \tag{1}$$

The proposed architecture is similar to a traditional neural network, with the neurons of the first layer fully connected to the neurons of the following layer. The first layer is composed of receptive field neurons, and the second layer is composed of RBF neurons [11].

Spiking neurons generate spikes when the membrane potential crosses a given threshold. The relationship between the spikes and membrane potential is described by the spike response model (SRM), as introduced in [30].

The first difference between SNNs and traditional neural networks is the use of multiple synapses [29]. Instead of a single synapse with a specific delay, the neuron of the first layer is connected to the output layer by m synapse terminals, where each terminal serves as a sub-connection associated with different delays and weights. The delay d^k of synaptic terminal k is defined by the difference between the firing time of the pre-synaptic neuron and the time when the post-synaptic (PSP) potential begins to rise.

Figure 1 presents an example of the encoding of multiple delayed synapses between layers *I* and *J*.

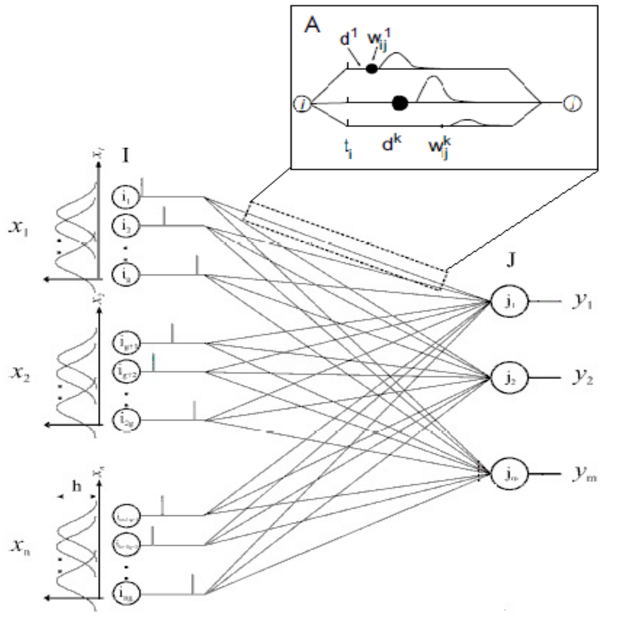

Fig. 1. SNN architecture – A single connection composed of multiple synapses. Each input dimension of a dataset is encoded separately and translated into trains of spikes.

In a clustering task, the learning process consists mainly of adapting the time delays so that each output neuron represents an RBF center. A winner-takes-all learning rule modifies the weights between the input neurons i and the first neuron j to fire in the output layer using a time variant of the Hebbian learning rule. For a weight with delay d^k from neuron i to neuron j, the learning rule is expressed as

$$\Delta w^k_{ij} = \eta L(\Delta t) = \eta(1-b)e^{-\left((\Delta t - c)^2/\beta^2\right)} - b \tag{2}$$

where parameter b determines the amount by which the weights will be reduced, β sets the width of the positive part of the learning window, and c determines the position of the peak. The value of Δt denotes the time difference between the onset of a PSP at a synaptic terminal and the time at which the spike is generated in the winning output neuron [11]. Figure 2 presents the graph of the learning function $L(\Delta t)$.

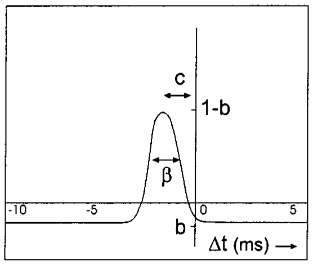

Fig. 2. Graph of the learning function.

III. Evolutionary Algorithms

Evolutionary algorithms (EAs) are important tools for solving complex problems in several areas. These optimization methods present a high degree of parallelism during the search process, displaying good performance when applied to non-continuous, non-differentiable, or multimodal problems. For this reason, two new EAs were established: QIEA-BR [26] and OGP [27]. The next two sections describe the QIEA-BR approach and OGP algorithm, respectively.

A. Quantum-Inspired Evolutionary Algorithm with Binary-Real (QIEA-BR)

QEIA-BR is a model in which numerical and binary parameters must be optimized; these values determine the final topology of a SNN and identify the relevant features. The parameters and features will be represented in the hybrid chromosome representation of the QIEA-BR model.

According to classical computing concepts, information is represented in bits, where each bit must hold either “0” or “1”. However, in quantum computing, information is instead represented by a q-bit, where the value of a single q-bit could be “0”, “1”, or a superposition of both. Superposition allows the possible states to represent both “0” and “1” simultaneously based on the states’ probabilities.

As noted in [26], the state of a q-bit can be represented as

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{3}$$

where α and β are complex numbers that determine the probability of observing the corresponding state and satisfy the following normalization condition:

$$|\alpha|^2 + |\beta|^2 = 1 \tag{4}$$

Thus, $|\alpha|^2$ and $|\beta|^2$ are the probability that the q-bit is in the OFF (0) state and ON (1) state, respectively. Through probabilistic observation, each q-bit can be rendered as one binary bit [26].

A binary quantum chromosome is represented as a sequence of pairs of numbers that can be observed to generate classical individuals. This observation is made by choosing a random number; if this number is larger than the observed state, the bit will be “1”; otherwise, it will be “0”.

Other important SNN configuration parameters will be represented as numerical genes in the QIEA-BR chromosome. The representation of the numerical portion of the chromosome requires a population of individuals to represent the superposition of possible states that might be observed for classical individuals. The quantum population *Q(t)* at any instant *t* of the evolutionary process is formed by a set of *N* quantum individuals q_i (*i=1, 2, 3 ... N*) such that each q_i is formed by *G* genes g_{ij} (*i=1, 2, 3... N*) that are formed by functions that represent probability density functions (PDFs). The quantum individuals can be expressed as

$$q_i = [g_{i1} = p_{i1}(x), g_{i2} = p_{i2}(x), \ldots, g_{iG} = p_{iG}(x)] \tag{5}$$

where the functions $p_{ij}(x)$ represent the PDFs. For a simple PDF, a pair of values (μ, σ) consisting of the mean and width of a square pulse was proposed to represent a gene. A PDF and cumulative distribution function (CDF) of a train of square pulses were calculated to connect a pulse representation with real-value variables.

The next step of the process consists of transforming the PDF into a CDF by integration and then using this CDF to generate random observations of the quantum individual as classical individuals. After classical individuals are generated, one can choose to use traditional recombination and mutation operators over these values. The mutation operator might not be useful because the classical individuals are generated through a random process, but applying the crossover operator between individuals from the previous and current generations might help the algorithm to reach better results.

The classical individuals that were generated can then be evaluated as they would in a conventional genetic algorithm. Then, the quantum population must be updated [31].

QIEA-BR defines a chromosome that incorporates both types of quantum gene.

Thus, at instant *t*, a quantum individual *j* representing mixed numeric and binary features can be defined as

$$q_j^t = \left[\left(q_j^t\right)_b \left(q_j^t\right)_r \right] = \left[\begin{pmatrix} \alpha_{j1}^t & \alpha_{j2}^t & \cdots & \alpha_{jk}^t \\ \beta_{j1}^t & \beta_{j2}^t & \cdots & \beta_{jk}^t \end{pmatrix}_b \begin{pmatrix} \mu_{j1}^t & \mu_{j2}^t & \cdots & \mu_{jm}^t \\ \sigma_{j1}^t & \sigma_{j2}^t & \cdots & \sigma_{jm}^t \end{pmatrix}_r \right] \tag{6}$$

Where the first part of the chromosome represents the α and β values for each quantum binary gene, and the second part of the chromosome represents the μ and σ of each quantum real gene.

Thus, a population *Q(t)* in generation *t* with *n* possible solutions q_j^t can be given a

$$Q(t) = \{q_1^t, q_2^t, \ldots, q_n^t\} \tag{7}$$

A detailed description of each step of QIEA-BR is provided in [26, 31]. The parameters of QbrSNN are provided in Table I.

TABLE I
PARAMETERS OF QBRSNN

QbrSNN	Parameters
NQ	*Number of quantum individuals*
NC	*Number of classic individuals*
T	*Number of generations*
Ccb	*Real classic crossover rate*
Ccr	*Binary classic crossover rate*
Δ□	*Update the binary part of quantum*
Cq	*Update the real part of quantum*
updatesT	*Execution before using operators*

B. *Optimization by Genetic Programming Method (OGP)*

OGP is a tree-based optimization algorithm based on genetic programming (GP) principles [27]. In OGP, each individual represents a set of *k* functions (trees), where *k* denotes the number of objective function parameters. For instance, suppose an objective function (fitness function) $F_6(x,y)$ attains its maximum value: $F_6(x,y)=1$ at $x=y=0$. This optimization problem is characterized by two parameters (*x* and *y*). Then, using OGP, we set $k=2$ for each individual in the population. An example of solution is presented in Figure 3.

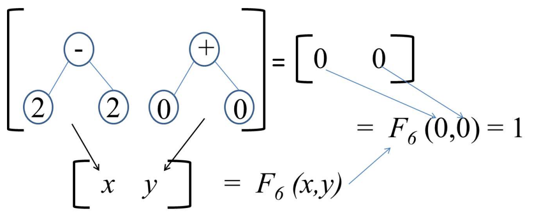

Fig. 3. Example of an OGP individual and the solution for problem F_6.

Therefore, based on mathematical operations (function set) and user-defined constant values (terminal set), OGP attempts to synthesize a suitable solution for the established criterion. For instance, in a set of benchmark optimization problems, OGP found significantly better results than a canonical genetic algorithm, particle swarm optimization, and differential evolution [27].

As noted in [27], a common OGP routine begins by setting the population parameters (e.g., population size, number of generations), genetic operators (e.g., crossover rate, mutation rate), objective function (fitness function), and number of optimization parameters (*k*). From these conditions, a population is randomly generated, comprising individuals with *k* functions. The solutions expressed by all of the individuals are evaluated and verified if the stopping criterion is met. If this criterion is met, the current population is returned; otherwise, the algorithm enters in a loop that is interrupted only when the stopping criterion is met.

Three operations are performed in the loop: the first is to select the entities of the next population based on some heuristic (e.g., roulette, tournament). Then, the low and high-level crossover (Figure 4 and 5) and mutation operators are applied to this new population. This process generates a new population that is again evaluated.

Additional details of OGP are provided in [27].

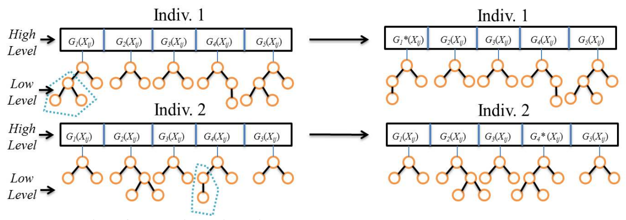

Fig. 4. Example of OGP low-level crossover.

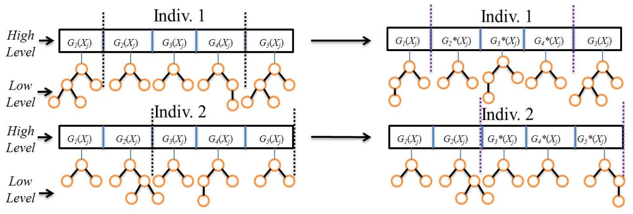

Fig. 5. Example of OGP high-level crossover.

IV. NEURO-EVOLUTIONARY FRAMEWORK

This section presents the proposed hybrid models for automatically tuning of SNNs for clustering problems. The two neuro-evolutionary models for unsupervised learning are denoted quantum binary-real evolving spike neural network (QbrSNN) and spike neural network optimized by genetic programming (SNN-OGP) and are presented in the following sub-sections. Let us now consider a database with *d* features and *n* patterns ($p=1,\ldots,n$).

A. *Quantum Binary-Real Evolving Spike Neural Network*

The QbrSNN model represents the connection between QIEA-BR and a SNN for unsupervised learning. The first step is to set the real and binary length of each individual in the QIEA-BR population. For example, suppose we wish to optimize the following parameters: number of Gaussian functions (*nG*), delay (Δ), learning rate (η), and number of neurons in the output layer (number of clusters – *nj*). Then, we define the real part of the chromosome with 4 genes, where each gene represents one of these 4 parameters. The binary length is based on the number of input features (*d*) in the database. Thus, for a database with $d=5$, the number of binary genes is equal to 5, where the value of the gene is 1 if the feature is active (will be included in the clustering process) and 0 otherwise. Figure 6 depicts the complete chromosome.

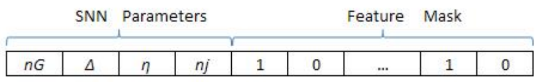

Fig. 6. Chromosome used to simultaneously optimize the parameters and feature space; *nG* - number of Gaussian, Δ - delay, η - learning rate, and *nj* - neuron in the output layer.

With these definitions, QIEA-BR randomly initializes the population with the pre-specified codification. For every individual, the real parameters are used to configure the structure of the SNN and the binary values define the features selected to supply the information to the SNN. Then, the SNN is trained based on this configuration, determining which pattern best fits each group. The next step is to evaluate the grouping quality. The fitness function plays a crucial role for a successful application. Several methods that evaluate the clustering quality could be applied, such as Ward's method and Silhouette (Si) [1, 32]. When the class label is available, one could use the Corrected Rand (cR) to evaluate the clustering accuracy [33]. Based on the user-defined fitness function, each individual is evaluated, and this procedure is repeated for all individuals in the population. With these fitness values, the individuals are ranked and selected, and recombination operators are applied. This process is repeated until a stopping criterion is met. Figure 7 presents the QbrSNN framework.

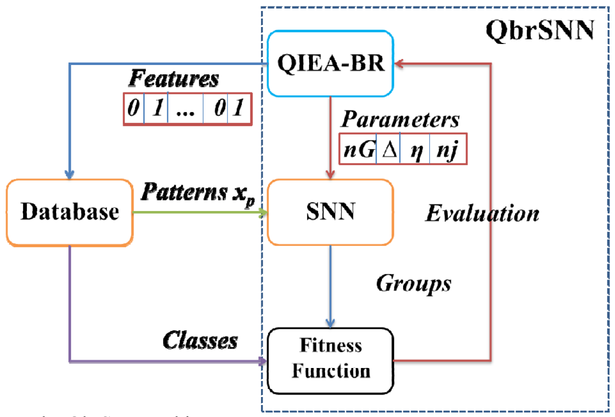

Fig. 7. The QbrSNN architecture.

B. Spike Neural Network Optimized by Genetic Programming

The SNN-OGP model is created by assembling OGP with a SNN for unsupervised learning. Similar to QbrSNN, the first step is to set the real and binary length of each individual in the OGP population. However, in this case, each real parameter and binary value is a function represented in a tree format. For example, suppose the user wishes to optimize the same 4 SNN parameters previously presented and the database contains 5 features. Then, $k=9$, where k represents the number of functions in each individual. In the same manner, the value is 1 if the feature is active (will be included in the clustering process) and 0 otherwise. Figure 8 depicts the complete tree-based individual.

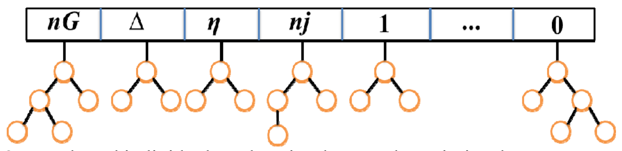

Fig. 8. Tree-based individual used to simultaneously optimize the parameters and features.

Because any real value could be generated when applying mathematical operations (e.g., plus, minus, times) in arbitrary values, we must constrain the output of each tree to make each solution feasible. Given a value x and feasible interval $[a, b]$, Equation (8) is applied to make the x domain feasible:

$$x^* = a + g(x)\ (a-b) \tag{8}$$

where x is the tree output (most likely unfeasible), a and b are the minimum and maximum values of the interval, respectively, g is a function that maps any real value to the interval [0, 1], typically the sigmoid function, and x^* is the new tree output (feasible). To convert the tree output into a binary value, one can define $a=-1$ and $b=1$, with $x^*=1$ if $x>0$ and 0 otherwise. With these transformations, feasible solutions can be generated using OGP.

After these definitions, OGP randomly initializes the population with the pre-specified codification and output transformation. Then, the SNN is trained based on the transformed outputs (parameter values and selected input features). Provided the groups, the next step is to evaluate each individual, and the population is ranked based on these values. Selection and recombination operators are also applied, and this procedure is repeated until a stopping criterion is achieved. Figure 9 illustrates the SNN-OGP framework.

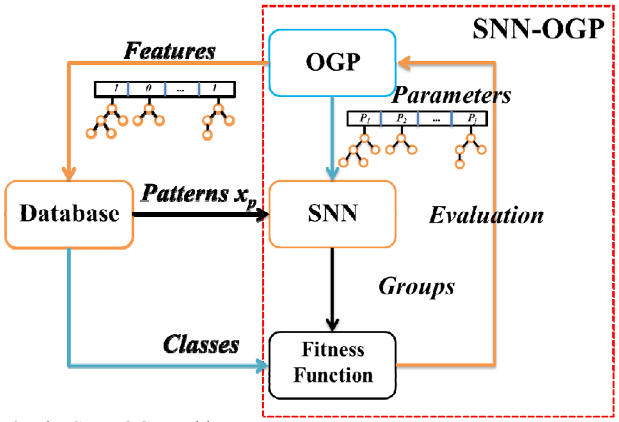

Fig. 9. The SNN-OGP architecture.

V. Experimental Results

A. Experiments

We used 8 benchmark datasets from the UCI repository to evaluate each proposed model [34]. Some of these datasets are common in the literature, such as the Iris and Wine datasets. These datasets were also selected due to the presence of a class label for each pattern. This information is used to compute the clustering quality measures.

All datasets are summarized in Table II, where n, d, and k denote the number of patterns, features, and class labels, respectively.

TABLE II
DATASETS

Datasets	n	d	k
Aggregation	788	2	7
Compound	399	2	6
Iris	150	4	3
Wine	178	13	3
Glass	214	9	7
Yeast	1484	8	10
Breast	699	10	2
Thyroid	215	5	2

For each dataset, QbrSNN and SNN-OGP were allowed to evolve a total of 3000 evaluations. We also trained a standard SNN for clustering using all features and setting the SNN parameters as suggested in [11].

Perform 3000 function evaluations entails the evaluation of almost 3000 different SNN configurations. This value was chosen to be computationally reliable and to demonstrate that when compared to other neuro-evolutionary approaches [35, 36], it is possible outperform a non-evolving SNN with relatively few evaluations.

To ensure statistical relevance, we executed 40 independent runs of the QbrSNN and SNN-OGP models for each dataset. These executions were performed in MATLAB R2010a [37]. Table III defines the parameter values of each evolutionary algorithm. These values were selected based on the recommendations of [26, 27]. The mathematical operations used in SNN-OGP were plus, minus, and times.

TABLE III
PARAMETER VALUES OF QBRSNN AND SNN-OGP

QbrSNN	Parameter Values	SNN-OGP	Parameter Values
NQ	15	*Population Size*	50
NC	50	*Generations*	60
T	60	*Tournament Size*	5
Ccb	95%	*High-level Crossover Rate*	50%
Ccr	5%	*Low-level Crossover Rate*	70%
Δ□	0.050*π	*Mutation Rate*	30%
Cq	10%	*Elitism Rate*	2%
updatesT	4	*Maximum Tree Depth*	3

We selected 4 SNN parameters for optimization: the number of Gaussian functions (feasible range: 8 – 20), delay (feasible range: 8 – 20), learning rate (feasible range: 0 – 1), and number of neurons in the output layer (feasible range: 2 – 10). The binary length depends on the number of features in each dataset.

Among many clustering quality measures, we selected the cR [33] as the fitness function. The cR index basically measures the relationship between the agreement and disagreement of two partitions (resultant clusters). Let $U = \{u_1, ..., u_R\}$ be the partition provided by the clustering algorithm and $V = \{v_1, ..., v_C\}$ the real partition based on a priori knowledge about the data. The corrected Rand (cR) index is defined by Equation (9):

$$cR = \frac{\sum_i^R \sum_j^C \binom{n_{ij}}{2} - \binom{n}{2}^{-1} \left[\sum_i^R \binom{n_i}{2} \sum_j^C \binom{n_j}{2} \right]}{\frac{1}{2}\left[\sum_i^R \binom{n_i}{2} \sum_j^C \binom{n_j}{2} \right] - \binom{n}{2}^{-1} \left[\sum_i^R \binom{n_i}{2} \sum_j^C \binom{n_j}{2} \right]} \qquad (9)$$

Where n_{ij} represents the number of common objects to groups u_i and v_j; n_i indicates the number of objects in the group u_i; n_j indicates the number of objects in group v_j; n is the total number of objects; and $\binom{a}{b}$ is the binomial coefficient:

$$\frac{a!}{b!(a-b)!} \qquad (10)$$

The cR index can take values between [-1, 1], where 1 denotes a perfect agreement between partitions and negative values or values close to 0 represent concordances found randomly. Therefore, we have to maximize the cR index.

We also compute the Si index based on the parameters provided by the best individual in the population. The Si index provides a succinct graphical representation of how well each object lies within a cluster [32], as indicated below.

$$Si = \frac{b(k) - a(k)}{\max\{a(k), b(k)\}}$$

(11)

Where, for each pattern k, let *a(k)* be the average dissimilarity of k between all other patterns within the same cluster; *b(k)* is the lowest average dissimilarity of k to any other cluster which k is not a member. This measure can take values $-1 \leq \text{Si} \leq 1$, where 1 (*a(k)<<b(k)*) denotes a perfect agreement between partitions and -1(*a(k)>>b(k)*) indicates clustering of the datum in its neighboring cluster.

B. Results

Table IV displays the cR and Si results obtained for each benchmark using a standard SNN, QbrSNN, and SNN-OGP. These values represent an average of 40 independent runs. The standard SNN approach performed poorly in datasets with a high number of features, such as Wine and Breast. Both neuro-evolutionary models obtained cR results approximately 25% and 38% higher in terms of the average and median, respectively, compared to a non-optimized SNN.

Applying the Wilcoxon rank test [38] to compare cR measures between the standard SNN, QbrSNN, and SNN-OGP, the QbrSNN and SNN-OGP models clearly outperformed the standard SNN in terms of the median (p-value < 0.01). Therefore, performing feature selection and parameter optimization with few evaluations can greatly enhance the clustering quality. However, when comparing the cR and Si results between SNN-OGP and QbrSNN, there was not a substantial difference (p-value > 0.05). The QbrSNN results in terms of cR and Si are typically higher than the OGP-SNN results, but this difference is small.

These similarities are also expressed in the final optimal parameter values for each dataset and the final number of input features selected for each SNN (Tables V and VI). On average, 45% of input features were not considered for clustering. Additionally, in most datasets, the number of output neurons (groups) obtained was very close to the number of class labels for each dataset.

TABLE IV
RESULTS OBTAINED FOR BENCHMARKS

Results- cR and Si					
	SNN	*QbrSNN*		*SNN-OGP*	
Datasets	*cR*	*cR*	*Si*	*cR*	*Si*
Aggregation	0.80	**0.93**	**0.63**	0.85	0.58
Compound	0.74	**0.82**	**0.63**	0.81	0.53
Iris	0.71	**0.92**	**0.79**	0.88	0.69
Wine	0.37	**0.81**	**0.57**	0.79	0.54
Glass	0.44	**0.55**	0.63	0.54	**0.67**
Yeast	0.18	0.29	**0.50**	0.29	0.44
Breast	0.32	**0.43**	**0.52**	0.40	0.50
Thyroid	0.78	**0.81**	**0.55**	0.78	0.45
Average	0.54	**0.70**	**0.60**	0.67	0.55
Median	0.58	**0.81**	**0.60**	0.79	0.54

TABLE V
AVERAGE PARAMETER VALUES

Parameter values (average-rounded)								
	QbrSNN				*SNN-OGP*			
Datasets	*nG*	Δ	*η*	*nj*	*nG*	Δ	*η*	*nj*
Aggregation	11	14	0.002	7	10	12	0.055	7
Compound	11	16	0.023	6	12	15	0.094	6
Iris	9	16	0.017	3	8	20	0.056	3
Wine	15	15	0.045	3	14	18	0.003	3
Glass	11	16	0.467	7	10	20	0.071	7
Yeast	12	18	0.633	8	10	12	0.077	6
Breast	15	14	0.029	6	8	16	0.016	2
Thyroid	16	14	0.019	2	12	18	0.066	2
Median	12	16	0.026	-	10	17	0.061	-

TABLE VI
AVERAGE NUMBER OF FEATURES SELECTED

Number of Features (average-rounded)		
Datasets	*QbrSNN*	*SNN-OGP*
Aggregation	2	2
Compound	2	2
Iris	3	3
Wine	8	7
Glass	4	4
Yeast	4	6
Breast	6	8
Thyroid	3	4
Average	4	5

VI. CONCLUSIONS

This work proposed two new neuro-evolutionary models for parameter and feature selection for clustering problems, termed SNN-OGP and QbrSNN. The characteristics of both SNN-OGP and QbrSNN were described, including the chromosome codification of each individual (real and binary values) and the evaluation function, selection, and recombination procedures. We used a set of 8 benchmarks from the UCI repository to evaluate both models.

The results demonstrate that SNN-OGP and QbrSNN are feasible in that they significantly outperformed a standard SNN with fewer evaluations. When comparing both approaches, QbrSNN yielded a slightly higher-quality clustering than SNN-OGP in most cases. However, QbrSNN required an average of 40% more computational effort than SNN-OGP. Therefore, the user must consider this trade-off when applying QbrSNN for feature selection and parameter optimization.

Future works can extend both models for supervised learning, such as classification and forecasting problems. Other research can perform a behavioral analysis on the evolutionary algorithms' parameters (e.g., population size, recombination rates) and evaluate their influence on the final clustering quality. Finally, both models can be evaluating with a larger set of benchmarks using other evolutionary approaches to find the most suitable evolutionary algorithm for feature selection and SNN parameter selection.

REFERENCES

[1] B. S. Everitt, S. Landau, and M. Leese, "Cluster Analysis," 4th Edition, Oxford University Press, Inc., New York; Arnold, London, 2001.

[2] I. H. Witten, E. Frank, and M. A. Hall, "Data Mining: Practical Machine Learning Tools and Techniques," Elsevier, 2001.

[3] A. Seret, T. Verbraken, S. Versailles, and B. Baesens, "A new SOM-based method for profile generation: Theory and an application in direct marketing," European Journal of Operational Research, vol. 220, pp. 199-209, 2012.

[4] Z. Ma, J. M. R. Tavares, R. N. Jorge, and T. Mascarenhas, "A review of algorithms for medical image segmentation and their applications to the female pelvic cavity," Computer Methods in Biomechanics and Biomedical Engineering, vol. 13, pp. 235-246, 2010.

[5] J. Liang, L. Bai, C. Dang, and F. Cao, "The K-Means-Type Algorithms Versus Imbalanced Data Distributions," *IEEE Transactions on Fuzzy Systems*, vol. 20, pp. 728-745, 2012.

[6] W. Mass, and C. M. Bishop, "Pulsed Neural Networks," MIT Press, Cambridge, MA, 1999.

[7] W. Gerstner, and W. M. Kistler, "Spiking Neuron Models: Single Neurons, Populations, Plasticity," Cambridge University Press, Cambridge, UK, 2002.

[8] W. Mass, "Networks of Spiking Neurons: The Third Generation of Neural Network Models," *Neural Networks*, vol.10, no.9. pp. 1659-1671, 1997.

[9] S. M. Bohte, H. L. Poutré, and, J. N. kok, "SpikeProp: Error-Backpropagation for Networks of Spiking Neurons," The proceedings of ESANN, pp. 419-425, 2000.

[10] B. Ruf, and, M. Schmitt, "Self-organization of spiking neurons using action potential timing*," IEEE-Transactions Neural Networks*, vol.9, no.3, pp. 575-578, 1998.

[11] S. M. Bohte, H. L. Poutré, and, J. N. kok, "Unsupervised Clustering with Spiking Neurons by Sparse Temporal Coding and Multilayer RBF Networks," *IEEE-Transactions Neural Networks*, vol.13, no.2, pp. 426-435, 2002.

[12] N. Kasabov, "Evolving Connectionist Systems: The Knowledge Engineering Approach," 2nd ed. Secaucus, USA: Springer-Verlag, New York, 2007.

[13] S. Soltic, S. Wysoski, and N. Kasabov, "Evolving Spiking Neural Networks for taste recognition," in *IEEE World Congress on Computational Intelligent (WCCI)*, Hon Kong, 2008.

[14] D. Verstraeten, B. Schrauwen, and D. Stroobandt, "Isolated word recognition using a Liquid State Machine," in *ESANN*, pp. 534-440, 2005.

[15] A. Knoblauch, "Neural associative memory for brain modeling and information retrieval," *Inf. Process Lett*, vol. 95, no. 6, pp. 537-544, 2005.

[16] N. Iannella and L. Kindermann, "Finding iterative roots with a spiking neural network," *Information Processing Letters*, vol. 95, no. 6, pp. 545-551, 2005.

[17] M. D. Platel, S. Schliebs, and N. Kasabov, "A Versatile Quantum-inspired Evolutionary Algorithm," in *IEEE Congress on Evolutionary Computation*, pp. 423-430, Singapore, 2007.

[18] S. G. Wysoski, L. Benuskova, and N. Kasabov, "Evolving spiking neural networks for audiovisual information processing," Neural Networks 23, pp. 819-835, 2010.

[19] H. N. A. Hamed, N. Kasabov, and S. Shamsuddin, "Integrated feature selection and parameters optimization for evolving spiking neural networks using quantum inspired particle swarm optimization," International Conference on Soft Computing and Pattern Recognition, pp. 695-698, *IEEE Press*, 2009.

[20] H. N. A. Hamed, S. Shamsuddin, and N. Kasabov, "Quantum-inspired particle swarm optimization for feature selection and parameter optimization in evolving spiking neural networks for classification tasks," In: Numerical Analisys and Scientific Computing, pp. 133-148, 2011.

[21] S. Schliebs, M. Defoin-Platel, and N. Kasabov, "Integrated feature and parameter optimization for an evolving spiking neural network," *Lecture Notes in Computer Science*, vol. 5506, pp. 1229-1236, Springer, Germany, 2009.

[22] S. Schliebs, M. Defoin-Platel, and N. Kasabov, "Integrated feature and parameter optimization for an evolving spiking neural network: Exploring heterogeneous probabilistic models," Neural Networks, vol. 22, Issues: 5-6, pp. 623-632, 2009.

[23] S. Schliebs, M. Defoin-Platel, and N. Kasabov, "Quantum-inspired feature and parameter optimization of evolving spiking neural networks with a case study from ecological modeling," In: International joint conference on Neural Networks, *IEEE Computer Society (IJCNN)*, pp. 2833-2840, CA – USA, 2009.

[24] S. Schliebs, H. N. A. Hamed, and N. Kasabov, "A reservoir-based evolving spiking neural networks for on-line spatio-temporal pattern learning and recognition," In: 18th International Conference on Neural Information Processing, no. 7063, pp. 160-168, Springer, Shanghai – China, 2011.

[25] S. G. Wysoski, L. Benuskova, and N. Kasabov, "Evolving spiking neural networks for audiovisual information processing," Neural Networks, vol. 23, pp. 819-835, 2010.

[26] A. G. de Pinho, M. Vellasco, and A. V. Abs da Cruz, "A New Model for Credit Approval Problems: A Quantum-Inspired Neuro-Evolutionary Algorithm with Binary-Real Representation," *World Congress on Nature & Biologically Inspired Computing (NaBIC)*, pp. 445-450, 2009.

[27] A. S. Koshiyama, D. M. Dias, A. V. Abs da Cruz, and M. A. C. Pacheco, "Numerical Optimization by Multi-Genetic Programming," *GECCO*, pp. 145-146, 2013.

[28] J. J. Hopfield, "Pattern recognition computing using action potential timing for stimulus representation," *Nature*, vol. 376, pp. 33-36, 1995.

[29] T. Natschlãger and B. Ruff, "Spatial and temporal pattern analysis via spiking neurons," Network Comp. Neural Syst., vol. 9, no.3, pp. 319-332, 1998.

[30] W. Gerstner, "Time structure of the activity in neural network models," *Phys. Rev. E*, vol. 51, pp. 738-758, 1995

[31] A. V. Abs Cruz, M. B. R. Vellasco, and M. A. C. Pacheco, "Quantum-inspired evolutionary algorithms for numerical optimization," *IEEE*, pp. 2630-2637, 2006.

[32] L. Kaufman and P. J. Rousseeuw, "Finding groups in data: An introduction to cluster analysis", John Wiley & Sons, New York, 1990.

[33] L. I. Kuncheva,"Combining Pattern Classifiers: Methods and Algorithms", Wiley-Interscience, 2004.

[34] K, Bache and M. Lichman, "UCI Machine Learning Repository", [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[35] R. Chandra, and M. Zhang, "Cooperative coevolution of Elman recurrent neural networks for chaotic time series prediction," *Neurocomputing*, vol. 86, pp. 116-123, 2012.

[36] K. O. Stanley, and R. Miikkulainen, "Evolving neural networks through augmenting topologies," Evolutionary Computation, vol. 10, pp. 99–127, 2002.

[37] MATLAB R2010a. MATLAB R2010a. Mathworks, Natick, 2010.

[38] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," Swarm and Evolutionary Computation, vol. 1,pp. 3-18, 2011.