Recurrent Neural Networks Design by Means of Multi-Objective Genetic Algorithm

Case study : Phoneme Recognition

Hanen Chihi[1] and Najet Arous[2]

Institut Supérieur d'Informatique, ISI
Département Génie Logiciels et Systèmes d'Information, GLSI
Université Tunis El Manar, Tunis Tunisie

Abstract
Evolutionary algorithms are considered more efficient for optimal system design because they can provide higher opportunity for obtaining the global optimal solution. This paper introduces a method for construct and train Recurrent Neural Networks (RNN) by means of Multi-Objective Genetic Algorithms (MOGA). The use of a multi-objective evolutionary algorithm allows the definition of many objectives in a natural way. The case study of the proposed model is the phoneme recognition. We have shown that the proposed model is able to achieve good results in recognition tasks.

Keywords: *Recurrent neural network, Genetic algorithm, Phonemes recognition, Multi-objective optimization.*

1. Introduction

Recurrent Neural Networks (RNN) represent a large and varied class of computational models that are designed by more or less detailed analogy with biological brain modules. In this paper we focus on the use a particular network : Elman-type recurrent networks in witch the hidden layer is returned to the input layer [7].

In recent years, gradient-based RNN solved many tasks [26]. The Back-propagation, however, has two major limitations: a very long training process, with problems such as local minima and network design. The back-propagation algorithm adjusts exclusively the connection weights for particular network architecture, but the algorithm does not adjust the network architecture to define the optimum Neural Network (NN) for a particular problem [3], [25]. To overcome these restrictions, various methods for auto-design NN have been proposed [2], [10].

Genetic Algorithms (GA) are a search heuristic that mimics the process of natural evolution. They maintain a population of solution candidates and evaluate the fitness of each solution according to a specific fitness function. Even though, GA are not guaranteed to find the global optimum, they can find an acceptable solution relatively in a wide range of problems [4].

Various combinations of GA and NN have been investigated [3], [10], [24]. Much research concentrates on the acquisition of parameters for a fixed network architecture [6], [9]. Other work allows a variable topology, but disassociates structure acquisition from acquisition of weight values by interweaving a GA search for network topology with a traditional parametric training algorithm over weights [2], [10]. Some studies attempt to co-evolve both the topology and weight values within a GA framework, but the network architectures are restricted [15].

Many researches exist, describing multitude applications for GA [4]. A substantial proportion of these applications involve the evolution of solutions to problems with more than one objective [13], [22], [27]. More specifically, such problems consist of several separate objectives, with the required solution being one where some or all of these objectives are satisfied to a greater or lesser degree.

Multi-objective genetic algorithms (MOGA) have been widely used for the evolution of NN. Dehuri and Cho [11] propose a multi-criterion pareto GA used to train NN for classification problems. Delgado and Pegalajar [12] propose a MOGA for obtaining the optimal size of RNN for grammatical inference.

In this study, we combine RNN with MOGA to provide an alternative way for optimizing both RNN structure and weights. An important aspect of our work is the use of multi-objective optimization to evaluate the ability of new RNN [20]. The use of different objectives for each network allows a more accurate estimation of the goodness of a network.

This paper is organized as follows. Section 2 explains the application of multi-objective optimization to the problem of fitness estimation. Section 3 describes the proposed constructive multi-objective RNN. Section 4 presents the experimental results obtained on the classification of the TIMIT vowels.

2. Multi-objective optimization

In this section, we briefly present the formulation of a multi-objective optimization problem (MOO) such as some required notions about Pareto based multi-objective optimization and some concepts relating to Pareto optimality [2], [12].

The scenario considered in this paper involves an arbitrary optimization problem with k objectives, which are, without loss of generality, all to be minimized and all equally important, i.e., no additional knowledge about the problem is available. We assume that a solution to this problem can be described in terms of a decision vector denoted by:

$$x = (x_1, x_2, ..., x_n) \quad (1)$$

where $x_1, x_2, ..., x_n$ are the variables of the problem.

Mathematically, the multi-objective optimization problem is stated by :

$$MOO : \begin{cases} \min F(x) = (f_1(x), f_2(x), ..., f_k(x)), \\ s.c.\ x \in C. \end{cases} \quad (2)$$

where f_i are the decision criteria and k is the number of objective function.

An optimization problem searches the action x^* where the constraints C are satisfied and the objective function F(x) is optimized.

In practical applications, there is no solution that can minimize all of the k objectives. As a result, MOO problems tend to be characterized by a family of alternatives solutions.

The approach most used is to weight and sum the separate fitness values in order to produce just a single fitness value for every solution, thus allowing the GA to determine which solutions are fittest as usual. However, as noted by Goldberg [14], the separate objectives may be difficult or impossible to manually weight because of unknowns in the problem. Additionally, weighting and summing could have a detrimental effect upon the evolution of acceptable solutions by the GA (just a single incorrect weight can cause convergence to an unacceptable solution).

The concept of Pareto-optimality helps to overcome this problem of comparing solutions with multiple fitness values. A solution is Pareto optimal if it is not dominated by any other solutions. A Pareto optimal solution is defined as follows: a decision vector x is said to dominate a decision vector y if and only if $\forall i \in \{1,...,k\}$: $f_i(x) \le f_i(y) \wedge \exists j \in \{1,...,k\} : f_j(x) < f_j(y)$. The decision vector x is Pareto optimal if and only if x is non-dominated [5].

The Pareto approach is based on two aspects: the ranking and the selection. The ranking methods are the following:

- NDS (Non Dominated Sorting) : In this method, the rank of an individual is the number of solutions dominating this individual plus one [12].
- WAR (Weighted Average Ranking) : In this method, population members are ranked separately according to each objective function. Fitness equal to the sum of the ranks in each objective is assigned [2].
- NSGA (Non-dominated Sorting Genetic Algorithm) [21]: In this method, all non-dominated individuals of the population have rank 1. Then, these individuals are removed and the next set of non-dominated individuals are identified and assigned next rank [21].

Several methods of selection based on the concept of dominance are:

- Tournament based selection [2]: at each tournament, two individuals A and B fall in competition against a set of t_{dom} individuals in the population. If the competitor A dominates all individuals and all the other competitor B is dominated by at least one individual, then individual A is selected.
- Pareto reservation strategy [5]: in this method, the non-dominated individuals are always saved to the next generation.
- Ranking method [2]: the cost associated with a new individual is determined by the relative distance in objective space with respect to individuals not dominated of the current population.

3. Recurrent neural networks design by means of multi-objective genetic algorithm

We shall now tackle the problem of finding RNN having the smallest recognition error and the least number of hidden units. For this reason, we formulate the problem as optimisation algorithm, more specifically, as a matter of MOO. In order to solve it we shall use an algorithm based on Pareto optimality that his goal is to optimize three objectives. A performance goal minimizes the recognition error (to maximize the successes in the testing set). Tow goals of diversity to increase diversity in the population : mutual information and internal diversity.

In this paper we propose a model called Recurrent Neural Networks Design by means of Multi-Objective Genetic Algorithm (RNND-MOGA). It reflects the types of networks that arise from a RNN performing both structural and weight learning. The general architecture of RNND-MOGA is straightforward. Input and output units are considered to be provided by the task and they are immutable by the algorithm; thus each network for a given task always has m_{in} input units and m_{out} output units. The number of hidden units and bias varies from 0 to a user supplied maximum $h_{\max}$.

The proposed hybrid learning process is the following (see Fig. 1). In each generation, networks are first evaluated using Pareto optimisation algorithm. The best *P*% RNN are selected for the next generation; all other networks are discarded and replaced by mutated copies of networks selected by proportional selection. Generating an offspring is done using only tow types of mutation operators : the parametric mutation and the structural mutation. The parametric mutation alters the value of parameters (link weights) currently in the network, whereas structural mutation alters the number of the hidden units, thus altering the space of parameters.

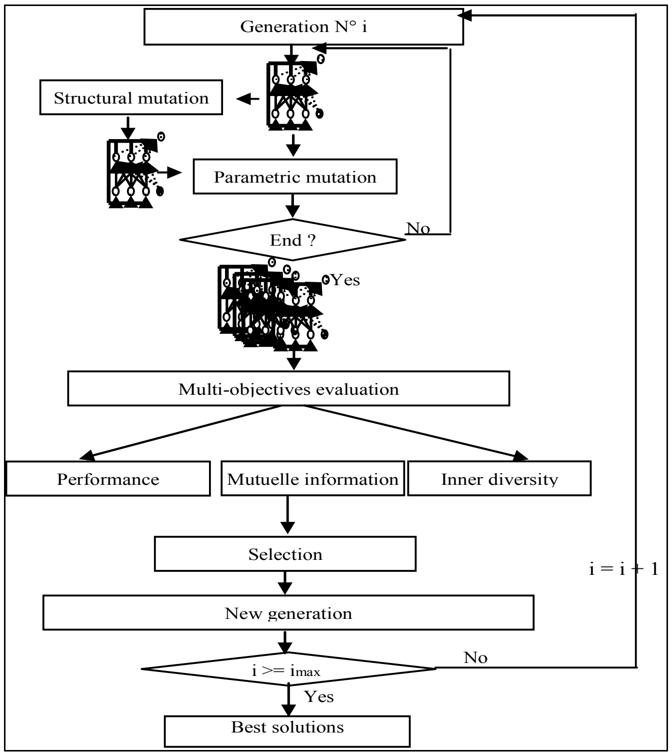

Fig. 1. Proposed evolution strategy

3.1 Encoding

The proposed chromosome representation is a structure encoding the learning parameters, the weights and the bias. The chromosome structure is a record composed of these attributes :

- IW: matrix of the RNN input weights ;
- LW: matrix of the RNN connection weights ;
- b1: vector of RNN bais ;
- trainPrm: save learning rate and epochs number ;
- learnFcn: save the learning function of an RNN.

3.2 Initialization

The proposed algorithm initializes the population with randomly generated RNN. The number of hidden units for each one is chosen from a uniform distribution in a user defined range ($0 \le h \le h_{\max}$ | $h_{\max}$ is the maximum number of hidden units in the network). Once a topology has been chosen, all links are assigned weights random initialized.

3.3 Genetic operators

GA used here is a modified algorithm. The main differences compared to the standard GA are that there is no crossover and structural mutations are added. Both of the mutation operators will be described in detail below. The crossover operator, which combines genes from two individuals, is rarely useful when evolving NN and is therefore not used here.

The parametric mutation changes the weights of a network without changing its topology. In this work, we use the back-propagation algorithm as a parametric mutation operator. It is run using a low learning rate for few epochs. In our model, this epochs number is randomly chosen within a user defined rang. The network is allowed to draw lessons from the training set, but it is also prevented from being too similar to the rest of networks. The parametric mutation is always performed after the structural one, because it does not alter the structure of a network and it is used to adapt mated networks.

Fig. 2 describe two types of structural mutation :

- *Add hidden units:* Generating an offspring using structural mutation involves three steps: copying the parent, determining the severity of the mutations to be performed, and finally mutating the copy. The severity of a mutation of a given parent is dictated by its score. It defines the number of hidden units to be added. Networks with a low score suffer a severe mutation, and those with a high score are undergoing a slight transformation. Equations (3) and (4) calculate, respectively, the severity of mutation and the number of hidden units to add.

$$T(i) = 1 - \frac{Score(i)}{\sum_{k=1}^{N} Score(k)} \quad (3)$$

$$HU(i) = \lceil \Delta_{\min} + \alpha T(i)(\Delta_{\max} - \Delta_{\min}) \rceil \quad (4)$$

where *Score(i)* represents the score of the i^{th} individual, $\Delta_{\min}$ an $\Delta_{\max}$ are respectively the minimum and maximum number of hidden units to be add, α is a random value between 0 and 1.

Once the number of units to be added is determined, we modify the network structure under the new constraints and the connections' weights of these units are randomly initialized.

- *Remove hidden units:* This type of mutation is used to remove the hidden units that do not contribute to improve recognition of the network. The process of deleting a hidden unit occurs as follows. In the first step, we seek the inactive unit among hidden units of the network. This is done by calculating the score of each hidden unit using equation (5). It calculates the difference in score of RNN with and without the hidden unit. The unit having the lowest fitness is eliminated.

$$S_u(i) = Score(i) - Score_u(i) \quad (5)$$

where *Score(i)* is the generalisation rate of the i^{th} RNN, $Score_u(i)$ is the generalization rate of the i^{th} RNN without the u^{th} unit.

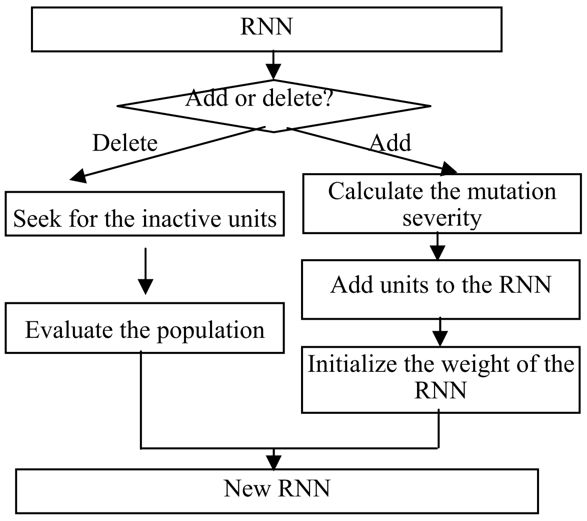
RNN
Add or delete?
Delete
Add
Seek for the inactive units
Calculate the mutation severity
Add units to the RNN
Evaluate the population
Initialize the weight of the RNN
New RNN

Fig. 2. Structural mutation strategy

3.4 Multi-objective optimization

A promising approach for performing optimization problems is the MOGA aiming at producing Pareto optimal solutions [11]. The key concept here is dominance. However, the success of a Pareto optimal GA depends largely on its ability to maintain diversity. Usually, this is achieved by employing niching techniques such as fitness sharing [5] and the inclusion of some useful measures applied to other models, such as negative correlation or mutual information [17]. The MOGA employed in this work can be described as a niched Pareto GA with NSGA [21] and tournament selection [2]. The algorithm uses a specialised tournament selection approach, based on the concept of dominance.

The proposed algorithm is based on the concept of Pareto optimality [19]. We consider a population of networks where the i^{th} individual characterised by a vector of objectives values. In fact, the population has *N* individuals and *M* objectives are considered. In our study, tree objectives are considered.

In this paper, we define the following four objectives:

- *Objective of performance:* The performance of RNN is given by its generalization rate.
- *Mutual information:* The mutual information between RNN f_i and f_j is given by equation (6) :

$$O_{MI}(f_i, f_j) = -\frac{1}{2}\log(1 - \rho_{ij}^2) \quad (6)$$

where ρ_{ij} is the correlation coefficient between the networks. The objective is the average of mutual information between each pair of networks [18].

- *Internal diversity:* The internal diversity of a RNN measures the difference between the outputs of the networks [16]. The internal diversity of the k^{th} RNN is given by equation (7) :

$$O_{ID}(i) = \frac{1}{N-1}\sum_{j=1, j\neq i}^{N} O_{FD}(f_i, f_j) \quad (7)$$

where *N* is the size of the RNN population, *P* is the number of training vectors and O_{FD} is the functional diversity between the i^{th} and the j^{th} network :

$$O_{FD}(f_i, f_j) = \frac{1}{P}\sum_{k=1}^{P}\left\| f_i(x_k) - f_j(x_k) \right\| \quad (8)$$

where x_k is the k^{th} training vectors.

4. Experimental results

In this section, we evaluate and compare the described and the proposed evolutionary constructive RNN for continuous speech recognition on the maco-class of vowels of TIMIT speech corpus [1].

4.1 Database description

The third component is a phoneme recognition module. The speech database used is the DARPA TIMIT acoustic-phonetic continuous speech corpus which contains: /iy/, /ih/, /eh/, /ey/, /ae/, /aa/, /aw/, /ay/, /ah/, /ao/, /oy/, /ow/, /uh/, /uw/,/ux/, /er/, /ax/, /ix/, /axr/ and /ax-h/. The corpus contains 13 699 phonetic unit for training and 4041 phonemes for testing.

Speech utterance was sampled at a sampling rate of 16 KHz using 16 bits quantization. Speech frames are filtered by a first order filter. After the pre-emphasis, speech data consists of a large amount of samples that present the original utterance. Windowing is introduced to effectively process these samples. This is done by regrouping speech data into several frames. A 256 sample window that could capture 16 ms of speech information is used. To prevent information lost during the process, an overlapping factor of 50% is introduced between adjacent frames.
Thereafter, mel frequency cepstral analysis was applied to extract 12 mel cepstrum coefficients (MFCC) [8].
Among all parameterization methods, the cepstrum has been shown to be favourable in speech recognition and is widely used in many automatic speech recognition systems [23]. The cepstrum is defined as the inverse Fourier transform of the logarithm of the short-term power spectrum of the signal. The use of a logarithmic function permits us to deconvolve the vocal tract transfer function and the voice source. Consequently, the pulse sequence originating from the periodic voice source reappears in the cepstrum as a strong peak in the 'quefrency' domain. The derived cepstral coefficients are commonly used to describe the short-term spectral envelope of a speech signal. The advantage of using such coefficients is that they induce a data compression of each speech spectral vector while maintaining the pertinent information it contains. The mel-scale is a mapping from a linear to a nonlinear frequency scale based on human auditory perception. It is proved that such a scale increases significantly the performance of speech recognition systems in comparison with the traditional linear scale. The computation of MFCC requires the selection of M critical bandpass filters. To obtain the MFCC, a discrete cosine transform, is applied to the output of M filters. These filters are triangular and cover the 156 − 6844 Hz frequency range; they are spaced on the mel-frequency scale. This scale is logarithmic above 1 kHz and linear below this frequency. These filters are applied to the log of the magnitude spectrum of the signal, which is estimated on a short-time basis.

4.2 Discussion

In the experiments below, the number of hidden units for networks of the initial population was selected uniformly between 1 and 5. Each network has 12 input units representing the 12 MFCC coefficients and 20 output units representing the TIMIT vowels. Table 1 represents the parameter setting.

In this section, results produced by the proposed model will be presented and compared with results produced by the Elman model using 30 hidden units the GA and the Elman model using 16 hidden units (the best topology given by the proposed model).

Table 1: Learning parameters of the proposed model

Parameter name	**Value**
Learning rate for the training of the Elman model	0.5
Epochs number for the trainig of the Elman model	100
Mutation rate for the standard GA	0.8
Crossover rate for the standard GA	0.4
Structural mutation rate	0.2
Parametric mutation rate	0.3
Generation number of the population of networks	20

The learning process of the GA used for comparison is the following. First, a population of chromosomes is created and initialised randomly. Then, a roulette selection is used to select individuals to be reproduced. Thereafter, a one-point crossover operator is used to produce new individuals. During crossover process, pairs of genomes are mated by taking a randomly selected string of bits from one and inserting it into the corresponding place in the other, and vice versa. After that, a classic mutation operator is used to mate these individuals. The classic mutation operator exchanges a random selected gene with a random value within the range of the gene's minimum value and the gene's maximum value. 40% of the best individuals are guaranteed a place in the new generation. This process is repeated for 100 generations.

The best structure of RNN provided by the proposed model is composed of 16 hidden units. We use the back-propagation algorithm to train a RNN using this structure. We note that, using this network, recognition rates and run time are greatly improved than those given by the RNN using 30 hidden units (see tables 2 and 3). We conclude that the proposed constructive evolutionary process improves the objective of defining the best structure of a RNN.

Tables 2 and 3 present a comparison of training rates, generalization rates and run time of the studied models. The Elman model using 30 hidden neurons provides the lowest recognition rate and the greater runtime of about 10 hours. GA gives best recognition rates than those given by the Elman model using 30 hidden units and it requires only 3 hours 30 minutes.

Furthermore, we note that the proposed model provides the best training rate of about 58.79% and the best generalisation rate of about 58.38%. In addition, it ameliorates the recognition rate of most of the phonemes such as /ey/ having 18% rather than 2% and /ay/ having 39% rather than 8%. We conclude, then, that the proposed multi-objective constructive model improves the objective of training of RNN. Furthermore, it should be noted that the proposed model takes 7 hours for training. This is justified by the fact that we use several objectives.

Table 2: Training rates of the Elman model usig 30 hidden units, the GA, the Elman model using 16 hidden units and the RNND-MOGA model

Vowels	**Samples**	**Elman (30 hidden units)**	**GA**	**Elman (16 hidden units)**	**RNND-MOGA**
iy	1552	77.83	**85.5**	77.19	84.99
ih	1103	11.6	18.04	17.32	**41.52**
eh	946	28.43	25.58	28.65	**57.19**
ey	572	2.27	1.40	0.35	17.83
ae	1038	77.84	**86.71**	74.95	84.49
aa	762	71.39	72.57	66.01	**80.18**
aw	180	0.00	0.56	0.00	**5.00**
ay	600	7.67	17.33	1	**38.83**
ah	580	7.07	7.41	**23.79**	12.41
ao	665	64.36	72.03	62.86	**83.16**
oy	192	0.00	0.00	0.00	0.00
ow	549	14.39	29.87	**41.71**	28.05
uh	141	0.00	0.00	0.00	0.00
uw	198	47.98	20.71	50.51	**66.67**
ux	400	2.25	1.00	2.00	**11.25**
er	392	8.42	16.58	8.67	**37.24**
ax	871	38.35	47.19	38.12	**57.41**
ix	2103	71.85	70.28	66.14	**84.31**
axr	739	52.23	63.46	54.26	**64.68**
axh	86	37.21	34.88	**62.79**	38.37
Global rate	13966	43.63	47.68	44.29	**58.79**
Runtime		10h20mn	**3h30mn**	4h	7h

5. Conclusion

In this paper, we have presented a model based on multi-objective genetic algorithms in order to train and to design recurrent neural networks. This algorithm is able to reach a wider set of possible RNN structures. We have shown that this model is able to achieve good performance in the recognition of TIMIT vowels, outperforming other studied methods.

The main results are as follows:

- The best RNN structure produced by the proposed model gives a better recognition rate at a lower runtime.
- The proposed model improves the recognition rate of the TIMIT vowels macro-classes of about 15% compared with the Elman model.

We suggest extending the constructive method to determine the optimal number of hidden layer and the number of hidden units in each one.

Table 3: Generalization rates of the Elman model usig 30 hidden units, the GA, the Elman model using 16 hidden units and the RNND-MOGA model

Vowels	**Samples**	**Elman (30 hidden units)**	**GA**	**Elman (16 hidden units)**	**RNND-MOGA**
iy	522	72.22	83.33	72.41	**86.02**
ih	327	8.26	12.23	16.51	**34.86**
eh	279	30.83	22.94	24.01	**63.44**
ey	162	1.24	1.85	0.00	**20.99**
ae	237	73.1	**86.92**	73	81.43
aa	237	62.87	59.49	54.85	**74.68**
aw	30	0.00	0.00	0.00	0.00
ay	168	2.38	17.86	0.00	**41.07**
ah	183	8.74	9.84	**21.86**	12.02
ao	222	59.91	64.41	54.96	**82.88**
oy	51	0.00	0.00	0.00	0.00
ow	171	9.94	26.32	**35.09**	22.81
uh	59	0.00	0.00	0.00	0.00
uw	51	31.37	11.76	23.53	**39.22**
ux	104	2	2.88	2.88	**8.65**
er	141	3.55	16.31	4.96	**36.88**
ax	249	50.2	61.85	47.39	**67.07**
ix	610	67.21	69.18	60.98	**81.97**
axr	210	55.72	65.71	46.19	**69.05**
axh	28	32.14	39.29	39.29	28.57
Global rate	4042	41.28	46.57	40.68	**58.38**

Acknowledgments

The authors are grateful to the anonymous reviewers for their valuable comments which improved the presentation and contents of this paper considerably.

References

[1] http://www.ldc.upenn.edu/Catalog/readme_files/timit.readme.html

[2] P.J. Angeline, G.M. Saunders, and J.B. Pollack, *An evolutionary algorithm that constructs recurrent neural networks*, IEEE Transactions on Neural Networks (1993).

[3] N. Arous, *Hybridation des cartes de Kohonen par les algorithmes génétiques pour la classification phonémique*, Ph.D. thesis, Thèse de doctorat,ENIT, 2003.

[4] H. Azzag, F. Picarougne, C. Guinot, and G. Venturini, *Un survol des algorithmes biomimétiques pour la classification*, Revue des nouvelles technologies de l'information (RNTI) (2004), 13–24.

[5] D. Beasly and R. Martin, *A sequential niche technique for multimodel function operation*, Conference on evolutionary computation **1** (1993), 101–125.

[6] P.A. Castillo, J.J. Merelo, M.G. Arenas, and G. Romero, *Comparing evolutionary hybrid systems for design and optimization of multilayer perceptron structure along training parameters*, Information Sciences **177** (2007), 2884–2905.

[7] R. Chandra, M. Frean and M. Zhang, *Building Subcomponents in the Cooperative Coevolution Framework for Training Recurrent Neural Networks*, School of Engineering and Computer Science, Victoria University of Wellington, Wellington, New Zealand, 2009.

[8] M. Chetouani, B. GAS, and J.L. Zarader, *Une architecture modulaire pour l'extraction de caractéristiques en reconnaissance de phonèmes*, Intenational conference on information processing (ICONIP'02) (2002).

[9] H. Chihi and N. Arous, *Adapted evolutionary recurrent neural network*, JTEA (2010).

[10] D. Dasgupta and D. R. McGregor, *Designing application-specific neural networks using the structured genetic algorithm*.

[11] S. Dehuri and S.-B. Cho, *Multi-criterion pareto based particle swarm optimized polynomial neural network for classification : A review and state-of-the-art*, Computer Science Review 3 (2009), 19–40.

[12] M. Delgado and M.C. Pegalajar, *A multiobjective genetic algorithm for obtaining the optimal size of a recurrent neural network for grammatical inference*, Pattern Recognition **38** (September 2005), 1444–1456.

[13] N. Garcia and C.J. Hervas, *Multi-objective cooperative coevolution of artificial neural networks*, Neural Networks **15** (2002), 1259–1278.

[14] D.E. Goldberg, *Algorithmes génétiques exploration optimisation et apprentissage automatique*, Kluwer Academic Publisher, 19 janvier 1996.

[15] J.R. Koza and J.P. Rice, *Genetic generation of both the weight and architecture for a neural network*, Proceedings of the International Joint Conference on Neural Networks (1991), 397–404.

[16] L. Kuncheva and C.J. Whitaker, *Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy*, Machine Learning **51** (2003), 181–207 51.

[17] Y. Liu and X. Yao, *Ensemble learning via negative correlation*, Neural Networks **12** (1999), 1399–1404.

[18] Y. Liu, X. Yao, Q. Zhao and T. Higuchi, *Evolving a cooperative population of neural networks by minimizing mutual information*, In Proceedings of the 2001 IEEE Congress on Evolutionary Computation (2001), 384–389.

[19] K. Maneeratana, K. Boonlong and N. Chaiyaratana, *Multi-objective Optimisation by Co-operative Co-evolution*, PPSN VIII : parallel problem solving from nature, 772-781, (2004).

[20] R.T. Marler and J.S. Arora, *Survey of multi-objective optimization methods for engineering*, Struct Multidisc Optim **26**, 369–395 (2004).

[21] N. Srinivas and K. Deb, *Multi-objective function optimization using non-dominated sorting genetic algorithms*, Evolution. Comput. 2 (1994), 221–248.

[22] E.G. Talbi, *Metaheuristiques pour l'optimisation combinatoire multi-objectif : Etat de l'art*, PM2O'1999 (1999).

[23] L. Tcheeko, *Un réseau de neurones pour la classification et la reconnaissance de la parole*, Ecole nationale supérieure polytechnique (1994), 277–280.

[24] R. Tlemsani, N.R. Tlemsani, N. Neggaz, and A. Benyettou, *Amélioration de l'apprentissage des réseaux neuronaux par les algorithmes evolutionnaires : application à la classification phonétique*, SETIT (2005).

[25] S. Kazarlis V.Petridis and A. Papaikonomou, *A genetic algorithm for training recurrent networks, Proceedings of IJCNN* .93 (1993), 2706–2709.

[26] M. Zhang and V. Ciesielski, *Using back propagation algorithm and genetic algorithms to train and refine neural networks for object detection*, Database and expert systems applications. International conference No10 1677 (1999), 626–635.

[27] A. Zinflou, *Système interactif d'aide à la décision basé sur des algorithmes génétiques pour l'optimisation multi-objectifs*, Master's thesis, UNIVERSITé DU QUEBEC, 2004.

Hanen Chihi received computer science engineering degree from Institut Supérieur d'Informatique (ISI), Tunis, Tunisia, the MS degree Software Engineering (Intelligent Imaging Systems and Artificial Vision) from ISI Tunisia. She is currently working towards the Ph.D degree, Tunisia. Her research interests include optimization, pattern classification and evolutionary neural networks.

Najet Arous received computer science engineering degree from Ecole Nationale des Sciences d'Informatique, Tunis, Tunisia, the MS degree in electrical engineering (signal processing) from Ecole Nationale d'IngTenieurs de Tunis (ENIT), Tunisia, the Ph.D. degree in electrical engineering (signal processing) from ENIT. She is currently a computer science assisting master in the computer science department at FSM, Tunisia. Her research interests include scheduling optimization, speech recognition and evolutionary neural networks.