SYSTEM DESIGNING THE NEURAL NETWORK STRUCTURE IN PARALLEL ENVIRONMENT

Sebastian Cińcio, Halina Kwaśnicka, Urszula Markowska-Kaczmar

Department of Computer Science, Wroclaw University of Technology,
Wyb. Wyspiańskiego 27 50-370 Wroclaw, Poland. Tel. (48 71) 320 23 97, Fax: (48 71) 21 10 18,
E-mail: sc@z.pl;{kwasnicka, kaczmar}@ci.pwr.wroc.pl

Abstract: The paper presents an experience in designing the system searching for the optimal neural network structures. The investigation with the system are described as well. The system acts on the base of cellular model of Genetic Algorithm and works in parallel environment. Assumed fitness function and the code schema of neural network structure is described. The influence of mutation and crossover probabilities onto algorithm activity is tested. Additionally, the two schema of the instances replacement is compared and the influence of the neighbourhood radius on algorithm efficiency is investigated and described.

Keywords: Genetic Algorithm, Neural Network, Parallel Model, designing of the neural network structure

Introduction

Artificial neural networks are widely used in many domains. But the process of their design is not simple (Ossowski 1994). The method of the appropriate choice of the neural network architecture does not exist. The design process is based on the try and test method. Recently the trials of the automatisation arrive. They apply Genetic Algorithm (GA) which acts on a code of the neural network structure and allows to the best choice of a neural network in accordance with criteria given in a fitness function. The problems solved by designed neural networks are usually very simple. The real problems are so complex that they meet the computational possibilities' barriers. One way to overcome this limitation is using parallel environment. Hence, the aim of the presented work is the creation of a system designing the neural network structure by a GA in a parallel environment. As an assumption this system (we call it SNAG) searches in feedforward neural network structures, because to train a neural network the backpropagation method is applied. As an environment for our system we have chosen the Parallel Environment (PE) of supercomputer IBM SP2.

1. Designing of the neural network structure by Genetic Algorithms

Figure 1 shows the general schema of neural network structures designed by a GA. At the beginning the choice of an initial population is made. Then each code string is decoded and coming into being neural network is evaluated. To do this it is necessary to train and test neural network. (Schafer at all, 1992). On the base of this process the fitness function is computed. If we are not satisfied, genetic operators are applied and offsprings population is created. From this description one can see that to design the system which apply a GA to automatically search neural network structures, it is necessary to solve following problems: choose the way of the genotype coding, design the fitness function, designate parameters of GA and the last but not least – choose this kind of parallel Genetic Algorithm, which fits the best to parallel environment where it works.

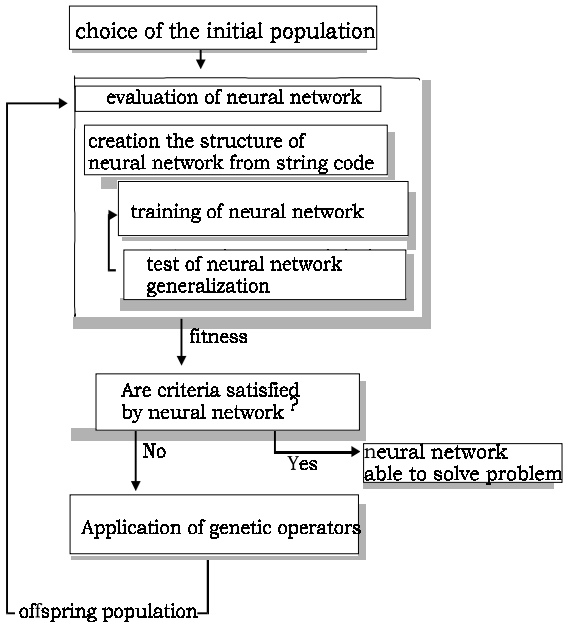

Figure 1. Neural network structure designed by a Genetic Algorithm

2.Parallel Genetic Algorithm

A genetic algorithm can be parallelized in different way (Cantu-Paz, 1997). The first solution is one *global population* of instances, but all of the genetic operation and fitness function are computed in a parallel way. This method is easy for implementation and to analyse its performance, because it does not change the schema of traditional GA. In the *island model* the population is divided on separate subpopulations and each of them evolves. The only one connection between them consists in the individuals exchange (the best as a rule).

Thanks of this genetic algorithm proceeds in different way in different subpopulations. It guarantees a big diversity of solutions but the common direction of search exists thanks the leader exchange. From theoretical point of view, the island algorithm activity can be described as a searching in subpopulations different solutions and their junction in the individuals exchange time. But combination of different individuals with high fitness values does not ensure as well good individual. This fact is a major fault of this model. The population partition onto larger and larger number smaller and smaller subpopulations gives us next model of parallel genetic algorithms environment called *cellular model.* In boundary each subpopulation consists of one individual. Each individual is treated independently. The population of individuals creates a space, where the neighbourhood relation between individuals exists (Figure 2). Individuals which are neighbours exchange between them string codes. Each individual realizes individual genetic operations on own code and code structures delivered by its neighbours. The code strings of individuals attainable for one cell is called logical subpopulation. Each individual is autonomous and assigned to one process.

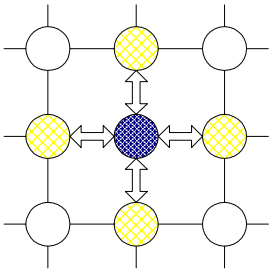

Figure 2. Neighbourhoods relation between individuals

In this algorithm the global information about population does not exist. Selection is not made on the global population, but each individual realizes it on the neighbour codes attainable for it (logical subpopulation). The main property of this model is the choice from relatively small number possible solutions. The crossover exists between selected solution and other string code in the logical subpopulation.

The island model is distinguishable by isolated (less or more) subpopulations. It is very useful property of this model. The cellular model does not give so distinct separation of individuals, but another property exists – isolation by distance. It is connected with structure of the cell neighbourhood. Individuals can migrate only to the nearest cells, what prevents them the quick migration in global population. In addition, the chance on the individual migration in unchanged form is very small because of genetic operation waited on it in every cell. Good solutions are promoted by quicker propagation in the global population, while poorly in natural way are eliminated. Assumed neighbourhood schema is very important in the cellular genetic algorithm on account of the gene migrations. In his investigations Schwehm (Cantu 1997) shows that connection in torus shape is better than others (ring, hipercube).

2. SNAG – an example of a system searching for optimal structure of neural network

Below we describe the assumptions of the SNAG system. They refer to the neural network structure proceeded by SNAG, its representation ,fitness function and architecture of the system.

2.1 Architecture

Figure 3. shows the data flow in the SNAG system.

As an input it receives:

- ✦parameters describing the structure of a neural network (the number of inputs and outputs, maximum number of neurons in the designed neural network)
- ✦parameters of the neural network learning (learning coefficient, momentum, maximum number of learning epoch)
- ✦parameters refer to genetic algorithm (size of population, probabilities of genetic operators),
- ✦training and test sets for neural networks.

As output data it produces:

- ✦optimal neural network structure found by the system,
- ✦report about the system activity.

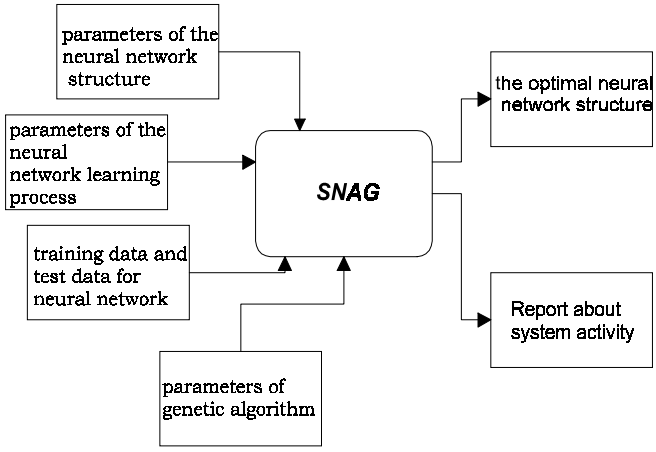

Figure 3. The data proceeded in SNAG system

As we said before, the neural network simulation ,beginning from its creation by learning and testing processes has a very high computational cost and is very time consuming. Sequential work of the system when succeeding individuals are evaluated is the simplest solution but worst one. We can treat modules as the independently working process with communication between them. Having a parallel system we can make the next step to develop a concept of the system.

In our implementation we chose cellular model of a GA, where a set of cells constitutes the population. The algorithm works on each single cell as follows:

1. Calculation of the individual fitness in the cell,
2. Sending of this individual (its string code) with its fitness value to the neighbor cells,
3. Receiving individuals from the neighbors,
4. Processing of genetic operation on own and received string codes and producing new individual residing in cell.

In each cell a communication module, a GA module and module of the neural network simulator can be distinguished (Figure 4). The communication module is responsible for the individual exchange between cells, a GA module performs genetic operations and the neural network simulator computes the fitness function of a new solution.

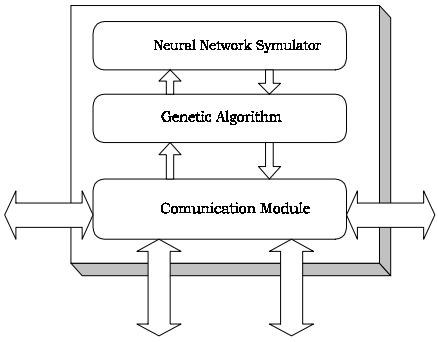

Figure 4. Modules of single cell

The designing of parallel asynchronous cellular GA needs to define: the neighbourhood structure, the structure of local populations and the way of individuals exchange, the communication schema – waiting or not for a change in a local population, the method of selection and the individuals replacement schema.

In the SNAG system structure of neighbourhood has a form of wraparound mesh (Figure 5) and two sizes of radius were tested: radius equal 1 (four nearest cells connected with given cell create its neighbourhood) and radius equal 2 (eight neighbour cells – immediately connected and moved two position in each direction).

2.2.The neural network structure and its representation

As it was mentioned above, we assume that the GA works to search optimal structure between feedforward neural networks, mostly because of simplicity of this approach. Feeding connections are not allowed. In this case backpropagation algorithm as a learning method can be used. If we mark all neurons creating neural network with numbers from *1* to *N* and we take into account the restriction about feedforward structure of the network, the following condition must be satisfied:

if exits connection between neuron a to neuron b then a < b

In addition, if we assume that first *Nin* neurons are the input neurons and the last create output neurons *(Nout*) the network connection (if connection between *a* and *b* exists), must satisfy the following conditions:

if $a <= Nin$ then $b > Nin$

if $a > N–NOut$ then $b <= N–NOut$

As a result we obtain neural network structure with distinguish input and output neuron layer, which are integral part of all structures. The next step is an appropriate representation of neural network for GA. A lot of approach exist (Boers, 1992, Dasgupta & McGregor, 1992, Kwaśnicka, 1997). In the SNAG system the individuals are coded by the connection matrix – one of the blueprint method. We assumed that unequal 0 value of matrix on *i, j* position determines a connection from neuron *j* to neuron *i*. Because we delimited neural network structure to feedforward neural network only, the half of connection matrix is sufficient to code neural networks. We choose lower triangle of the connection matrix under the main diagonal, and joining them together by lines we receive code string.

2.3. Communication between cells

In the SNAG system the connection topology has form of torus. The shade (Figure 5) in the grey scale corresponds to the individuals fitness value. Each cell (individual) is assigned to the single, independent process. Processes are distributed between available processors, thanks which logical structure of the system (processes assign to the cells) are irrespective of environment where system acts. Virtual topology of processes corresponds to the connection schema between cells.

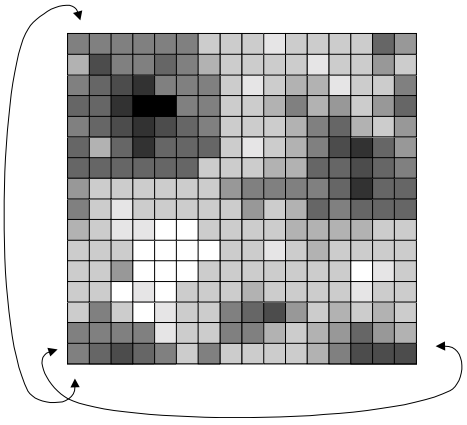
Figure 5. Population in cellular model of a GA

The individual residing in a given cell is called resident. Each cell acts in asynchronous way. It sends messages containing calculated solution just after the fitness value calculation of its resident. Then it receives messages sent by neighbours, but it does not wait for messages from all neighbours to make genetic operations. These operations can be proceed on the local code strings sent by neighbours in the previous stage. It induces growth request for memory but improves efficiency of computation. In this way a set of the string codes from the neighbour cells constitutes a local population inside the cell. This population is implemented as a simple buffer (Figure 6), with solutions sent by

neighbours and is updated after receiving a new solution (the number of individuals in a local population is equal to the number of neighbours). Characteristic feature of cellular model is the production by genetic operations in single cell only a single solution.

The number of individuals in a local population has big influence on the memory demand of the system. When cell desires to receive messages from neighbours but they are busy, the two possibilities can be consider:

- cell waits on receiving at least one new string code to make genetic operation,
- genetic operators are performed assuming that they do not create identical individual.

In our system we apply the last one approach.

2.4. Genetic operators, selection and substitution schema

As genetic operators the two classical operators are implemented: one point crossover and mutation. In the SNAG system selection is made on a local population. Selected individual is crossed over with assumed probability with resident. It is a tournament selection, where two individuals are chosen and the best (with higher fitness value) is taken out. When new individual is evaluated it substitutes previous resident. This substitution can be unconditional, but it can depend from relation between the old and new solutions. In SNAG the two schema are implemented. In both of them when crossover takes place a new individual certainly substitutes resident. If this condition is not satisfied then:

- individual chosen in selection substitutes resident certainly in first schema,
- individual chosen in selection substitutes resident only if it has better fitness function.

2.5. Fitness function

An integral part of a GA is a function, which assigns each code string evaluation of its fitness. It is called fitness function. The knowledge of individuals fitness is necessary to perform a selection. Because of the selection method implemented in SNAG system, the fitness function should give the necessary information to compare two solutions. According to our assumption an optimisation task is the optimisation of the neural network structure. This structure must be useful of course. It means, it has to be able to act properly.

The first and fundamental condition for neural network performance is its ability to learning given task to perform. Here the only one requirement is the sufficient big structure of neural network. The next important factor is its ability to generalization. This is inversely proportional to the size of neural network. With a big structure the neural network memorizes patterns instead of learning their common features. From this point of view the neural network structure should be minimized. It has a profitable influence on a speed of the neural network learning. Summarizing, the optimal neural network structure is the minimal structure enables the neural network to learn given task and it assures the maximal generalization ability.

In the SNAG system the fitness function is discontinuous and that is given by following equation:

$$Fit = \begin{cases} -(MAXN - n) & for\ error\ structure \\ -ns - learnerror & for\ not\ trained\ structure\ with\ an\ assumed\ error \\ vrf * GRN + ns * STR & for\ trained\ structure \end{cases}$$

where:

MAXN – the maximal number of connections allowed in a neural network,
n – a number of connection in the evaluated neural network,
learnerror – a learning error of a neural network,
vrf – a fraction of the correct generalized patterns from the test set,
GRN – a factor – the weight of importance of the neural network generalization,
ns – a parameter describing the size of the neural network. It has a maximal value for minimal size of the neural network and is defined as follows:

$$ns = 1 - \frac{N}{MAXN}$$

STR – a factor – the weight of importance of the neural network size.

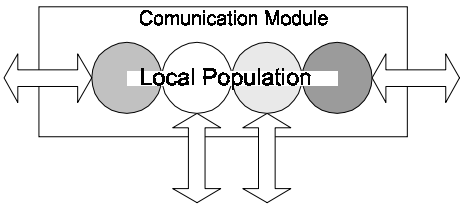

Figure 5. Local population in cellular model of GA as a buffer

The first part of this equation (for the error structure) is applied to the network where does not exist any (not necessary immediately) connection between input and output neurons. In this case we prefer structure with bigger number of connection in intend to obtain connections between inputs and outputs.

The second part of the fitness function describes the error of the neural network and the size of its structure. This

part of fitness function is calculated for the network with the correct structure, which is not able to learn patterns correctly. The value of the fitness function is equal to the output error in the last epoch of learning with the minus sign reduced by parameter describing the size of the neural network structure.

The last part of fitness function is used to the trained neural network with correct architecture. The two parameters are included :

– the size of the neural network structure (*ns*). It is the value calculated on the base of connection number in the neural network after eliminating the connections and neurons, which are not active.
– generalization (*vrf*) – a fraction of the correct generalized patterns from the test set.

To the both of these parameters the weights are assigned. They allow to influence on desired direction of the neural network design.

2.6. The neural network simulator

The simulation of the neural network activity consists of the three stages:

- the construction of the neural network on the base of the code structure received from GA,
- the neural network learning using the learning set,
- the investigation of the neural network generalization ability on the base of the test set.

In the first step the structure of the neural network is refined. It means, the following connections and neurons are eliminated: within input neurons and within output neurons, not connected to the output neurons, not connected to the input neurons. In the SNAG system the upper limit on the number of neurons is given as a parameter. The network without connection between any input and output is discarded. It is not necessary to connect all inputs to the neural network structure, because some of them do not carry essential information.

Next neural network is trained and tested. In the SNAG system the neural network is trained by using back propagation algorithm with momentum. The weights change after each pattern is made. Trained neural network is then tested and its generalization ability is calculated as a fraction of properly recognized patterns from the test set.

3. Experiments and received results

We have tested our system on different problems. Taking into account the calculation time to find influence on the system behaviour we have investigated simple problems. We have used classical problems applied in examination of the classification algorithms efficiency such as: the parity problem investigation, the iris classification, the binary code change on the thermometer scale. The final task of our system is to find neural network recognizing human faces. It is clear this task is not trivial.

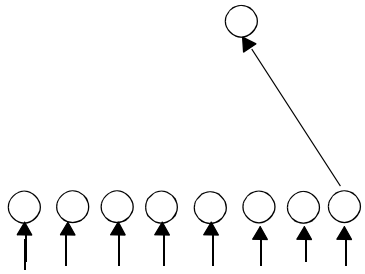

Figure 7. The neural network structure found for the parity problem

Table 1. Conversion of binary codes on thermometer scale

inputs	outputs
0 0 0	0 0 0 0 0 0 0 0
0 0 1	1 0 0 0 0 0 0 0
0 1 0	1 1 0 0 0 0 0 0
. . .	. . .
1 1 1	1 1 1 1 1 1 1 0

The parity problem

In this tasks we test our system if it is able to optimise the neural network structure which realize the parity recognition of numbers written in binary code. As an assumption our network has eight inputs and one output. For odd numbers the target output is set to one and for even numbers is set to zero. The population size was equal 16 individuals. The optimalization process in Figure 7. is presented. In resulted network all input neurons besides the first one do not connect with output. The signals form these inputs do not carry an important information for parity problem.

The binary code change on thermometer scale

In this experiments the neural network converts binary code on thermometer scale as in Table 1. The input vector of neural network consists of three inputs from Table 1.and two inputs with noisy information. On this example we try to find the influence of basic parameters on optimization process. In our experiments we tested the three values of probability mutation: 0.001,0.01, 0.1. The probability of crossover was set to 0.8.

Comparing the results we can ascertain that increasing of the mutation probabilities gives greater diversity of individuals in population, but it does not cause significant improvement of results. Additionally, very high probability of mutation makes a genetic algorithm alike a pure random.

Classification of iris

In this case we try to investigate the influence of the population size on the genetic algorithm activity. The task for neural network lies in systematization of iris into three classes on the base of the four features: length of petal, width of petal, length of sepal, width of sepal. We had tested the activity of genetic algorithm for three sizes of population: 25, 49 and 64 individuals. For little population (25 individuals) the quick loss of diversity can be observed. However

in the first period the average fitness function in a population distinctly grows up what gives next best solutions, but after the average fitness falls down and many times the value of the best fitness of population is near the worst solution. Comparing these results bigger neighbourhoods radius gives better evolution of population measured by average fitness value. Observed results are not consistent with conception saying about worsening of the algorithm efficiency with enlargement of the neighbourhood cells number. Here we had tested the problem in a little range of the neighbourhood radius change. If we make the experiments with the bigger range of changes obtained results probably confirm theoretical reduction of the algorithm efficiency.

The face recognition

The final task for our investigation is searching for a structure of neural network which correctly recognizes faces. The set of faces was divided onto two groups one with known faces and the second one with unfamiliar faces. The potential application of a such neural network are all identification systems. It can be used to find searched person in the archives or in entry gates to secure objects.

As an input data the 17 characteristic features of face were chosen. The training set consists of 60 training vectors where 30 vectors represent unfamiliar faces and next 30 vectors familiar faces. The test set consists of 50 patterns. The boundary size of neural network size was equal to 70 neurons. The best found neural network had 68% generalization ability on the test set.

4. Conclusions

Our system has very good results for a little search space. With enlargement of the target neural network size the algorithm efficiency in terms of quality and speed obtained results gradually decrease. The big influence on the obtained results has a big number of parameters, for example parameters describing the structure of neural network (the number of neurons), the parameters of learning (an initial range of weights, the learning coefficient, momentum), suitable parameters of genetic algorithm (probabilities of mutation and crossover, the fitness function construction). The parameters characterizing parallel algorithm activity (such as population size, the neighbourhood radius, the schema of selection, the communication between cells are very important. Taking into account the number of parameters it is very difficult to choose their optimal value. The next reason of not very high working capacity of genetic algorithm one can suspect in too little size of tested population, so in the future this problem will be investigated.

References

Boers J.W., Kuiper H. (1992),"*Biological metaphors and design of modular artificial neural networks*" Master Thesis unpublished Departament of Computer Science Leiden University. ftp://ftp.wi.LeidenUniv.nl/pub/CS/MScTheses

Bornholdt S., Graudenz D. (1992), *General Asymmetric Neural Networks and Structure Design by Genetic Algorithms*, Neural Networks, Vol.5, pp. 327-334.

Cantu-Paz E. "*A survey of Parallel Genetic Algorithms*" IlliGAL Report No. 97003 , May 1997, Illinois Genettc Algorithms Laboratory.

Dasgupta D., McGregor D., R. (1992), *Designing Application-Specific Neural Networks using the Structured Genetic Algorithms*, International Workshop on Combinations of Genetic Algorithms and Neural networks, COGAN-92, IEEE Computer Society Press.

Heath G.E., MIT Lincoln Lab heath@ll.mit.edu, *Letter from Genetic Algorithms* discussion list, news comp.ai.genetic

Kwaśnicka H. (1997), *Evolutionary Approach to Artificial Neural Network Design – Good Way or Blind Alley*, MENDEL'97 III International Mendel Conference on Genetic, Optimization Problems, Fuzzy Logic, Neural Networks, Rough Sets, Technical University of Brno, Brno, Czech Republic, str. 342-347.

Medsker L. R.(1995), "*Hybrid Intelligent Systems*", Kluwer Akademic Publishers, Boston-Dordrecht-London

Michel O., Biondi J. "*From chromosome to neural network*" unpublished , Laboratory I3S – CNRS-UNSA.

Ossowski S. (1994), *Sieci Neuronowe* (*Neural Networks*), Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa.

Pit L. J. "Parallel Genetic Algorithms" Master Thesis unpublished Departament of Computer Science Leiden University. ftp://ftp.wi.LeidenUniv.nl/pub/CS/MScTheses

Schafer J.D., Whitley D., Eshelman L.J. (1992), *Combinations of Genetic Algorithms and Neural Networks: A Survey of the State* International Workshop on Combinations of Genetic Algorithms and neural Networks, Baltimore, Maryland.