OPTIMIZATION OF STRUCTURE OF NEURAL MODELS USING DISTRIBUTED COMPUTING ENVIRONMENT

Adam TOMANEK, Piotr PRZYSTAŁKA, Marek ADAMCZYK

Silesian University of Technology, Department of Fundamentals of Machinery Design
Konarskiego Street 18a, 44-100 Gliwice, Poland, e-mail: pprzystalka@polsl.pl, madamczyk@polsl.pl

Summary

The main aim of this paper was to identify the optimal structures of considered neural models using the distributed computing environment. In this paper distributed optimizing of feed-forward neural network architectures for given problems is presented. The computing environment is composed of a few important packages and modules and has been created by the authors in order to aid developing some soft computing methods [4], where a lot of calculations are needed. At the beginning the authors decided to adapt a simple systematic-search algorithm that searches through every possible combination of network structures. Since this class of algorithms requires large amount of computation the distributed computing system was employed.

Keywords: distributed computing, ad-hoc computing clusters, artificial neural networks, heuristic modeling, optimizing neural network architecture.

OPTYMALIZACJA STRUKTURY MODELI NEURONOWYCH Z ZASTOSOWANIEM ROZPROSZONEGO ŚRODOWISKA OBLICZENIOWEGO

Streszczenie

Głównym celem przeprowadzonych badań było zidentyfikowania optymalnej struktury rozpatrywanych modeli neuronowych z zastosowaniem środowiska do obliczeń rozproszonych. W artykule zaprezentowano zastosowanie systemu do rozproszonej optymalizacji struktury sztucznej sieci neuronowej typu perceptron wielowarstwowy dla zadanego problemu. Prezentowane środowisko obliczeniowe jest złożone z kilku pakietów oraz modułów i zostało utworzone przez autorów w celu wspomagania rozwoju metodologii modelowania heurystycznego [4], gdzie niezbędnych jest wiele obliczeń. W początkowym stadium rozwoju oprogramowania autorzy zastosowali prosty algorytm przeszukiwania systematycznego każdej możliwej kombinacji struktury sieci. Ponieważ tego typu algorytmy z reguły wymagają dużych mocy obliczeniowych, postanowiono wykorzystać system omawiany w niniejszym artykule.

Słowa kluczowe: rozproszone obliczenia, klastery typu ad-hoc, sztuczne sieci neuronowe, modelowanie heurystyczne, optymalizacja struktury sieci neuronowej.

INTRODUCTION

Heuristic modeling of objects and processes [4] is a very difficult task. Most of the physical processes realized by complex objects are stochastic and dynamic in nature. Sometimes it is impossible to use analytical methods [4] because modeling of such systems (i.e using physical processes) is a complicated task involving plenty of time and effort. It is necessary to do a lot of calculations. This purpose generates the need of having a high-powered CPU unit that can make a lot of calculations in a short time. In case like this there are at least two alternative ways to solve this problem. One is to use a supercomputer, while the other is to use a group of computers working together simultaneously [3]. A collection of PCs (nodes) is called a cluster.

Recently, clusters of personal computers (PCs for short) are high-powered technical computing platforms offering a low-priced alternative to traditional supercomputers [1, 2]. PCs clusters besides the low cost possess much more advantages such as: can be built in a modular fashion (for persons' requirements), the number of nodes, the node types and the network may be chosen, etc. In many articles it is possible to find classification of computer clusters by functionality: High Performance Computing [3], High Availability Computing [9], Load Balancing Cluster [5, 9].

This paper deals with distributed optimizing of feed-forward neural network architectures for heuristic modeling of a dynamic system and an industrial plant using the described computing environment.

This environment is easy-to-use, free-of-charge for non-commercial research purposes and is an ideal platform for all social scientists and even for graduate students. It is designed to be run on a usual PC-compatible computer without installing anything on the computer hard drive. The environment may be used in many applications but in this time it is applied for a well-known problem such as optimizing neural networks topology. There are many paper and books deals with this task [5, 7]. These methods usually need to do a lot of calculations. In other words many variants of network structures have to be tried. Therefore, there is a need to use a distributed system.

The paper is composed as follows. In Section 1 a brief description of this environment is given. Next, in Section 2 some performance tests are presented. At the end of the paper, there are two examples presented. They show modeling of a dynamic system and modeling of an electric furnace by means of the described environment.

1. ENVIRONMENT DESCRIPTION

1.1. The main idea

The basic schema of the distributed computing environment for heuristic modeling of objects and processes is given in Figure 1. It is multilayer application architecture. There are two major layers: the low-level layer (realizing migration of processes or threads) and the high-level one (e.g. user interface, web server, database, neural network simulator engine, etc. for solving given problems).

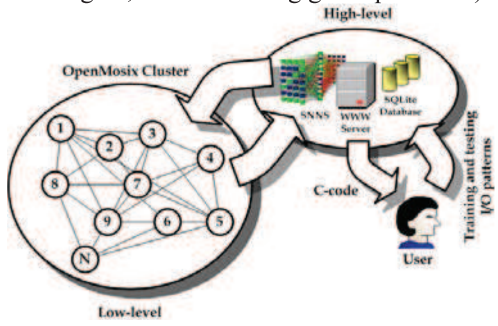


Fig.1. Architecture of the distributed computing environment for a neural modeling

1.2 Main components The low-level layer

The main part of the computing environment includes the openMosix system for parallel computing. openMosix is a Linux kernel extension for single-system image clustering [2, 5]. This kernel extension turns a network of ordinary computers into a supercomputer for Linux applications. The openMosix technology is composed of two major parts [5]: a Preemptive Process Migration (PPM) mechanism and a set of algorithms for adaptive resource sharing (both are implemented at the kernel level and they are completely transparent for user level). The PPM is able to migrate any process, at any time, to any available node by using information gathered from one of the resource sharing algorithms. openMosix has no central control (no master/slave relationship between nodes), each node operates as an autonomous system and it allows dynamic configuration. There are two main resource sharing algorithms of openMosix [5]: the load-balancing and the memory ushering.

Using a Linux system and OpenMosix technology as a low-level layer unlocked an easier way to distribute processes and threads seamlessly across any machine on the network. Instead of needing a lengthy and complex installation process, the operating system with environment runs directly from a CD.

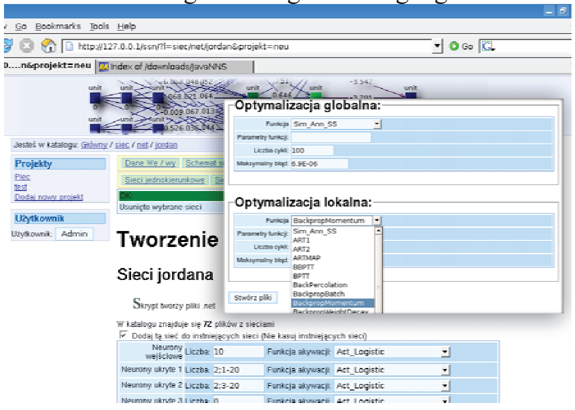
The high-level layer

There are also included tools based on the GNU Compiler Collection (GCC) and programs (Apache, SQlite, etc.) for programming in C, C++, Fortran, and Java etc. For modeling task the environment gives Stuttgart Neural Network Simulator (SNNS for short, a software simulator for neural networks on Unix/Linux workstations) [6].

Additionally, one is able to access files on USB memory devices, floppy, Zip or hard drives connected to the computer, as well as networked resources on Windows and Macintosh networks and on the Internet.

User interface

In order to simplify usage of the distributed system for both the administrator and end-user the website was designed using PHP language.


http://127.0.0.1/ssn/?l=siec/net/jordan&projekt=neu

Jesteś w katalogu: Główny / siec / net / jordan

Projekty: Piec, test, Dodaj nowy projekt

Użytkownik: Admin

OK: Usunięto wybrane sieci

Tworzenie

Sieci jordana

Skrypt tworzy pliki .net

Optymalizacja globalna:

Funkcja	Sim_Ann_SS
Parametry funkcji:	
Liczba cykli:	100
Maksymalny błąd:	6.9E-06

Optymalizacja lokalna:

Funkcja	BackpropMomentum
Parametry funkcji:	
Liczba cykli:	
Maksymalny błąd:	

Sim_Ann_SS, ART1, ART2, ARTMAP, BBPTT, BPTT, BackPercolation, BackpropBatch, BackpropMomentum, BackpropWeightDecay

Stwórz pliki

W katalogu znajduje się 72 plików z sieciami

☑ Dodaj tą sieć do instniejących sieci (Nie kasuj instniejących sieci)

Neurony wejściowe	Liczba: 10	Funkcja akywacji: Act_Logistic
Neurony ukryte 1	Liczba: 2;1-20	Funkcja akywacji: Act_Logistic
Neurony ukryte 2	Liczba: 2;3-20	Funkcja akywacji: Act_Logistic
Neurony ukryte 3	Liczba: 0	Funkcja akywacji: Act_Logistic

Fig.2. Easy to access user interface (in Polish)

Figure 2 shows an example of the sub-site, where the user has to set some parameters for global and local optimization algorithms of neural network topology. The user is also able to modify nearly all the options available in the SNNS. Thus, it makes possible to prepare several different network topologies and many different types of learning procedures automatically. The above-described environment also gives a simple interface for analyzing errors. The user is able to analyze quality of the model by looking at figures generated on a webpage and he/she can use some formal quality measures as well. Three various quality measures were defined: Thiel's coefficient *(U)*, mean absolute percentage error (*MAPE*), and confirming with Obuchowicz coefficient *(J)* (detailed description of quality measures was omitted in the paper).

The last advantage of this interface is that after the optimizing stage the user may get a complete C-code with an embedded optimal neural structure and put it into a dedicated system.

2. PERFORMANCE AND SPEED TESTS

There are at last two ways to create programs that can be distributed among different machines in a network (employing the cluster). One way is to use simple BASH scripts which run several independent processes or the second is to write C/C++ programs using POSIX threads API [7].

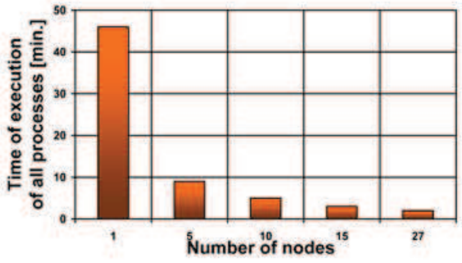


Fig.3. Performance test of the cluster

In particular, the authors have carried out a series of performance tests (all the trials were carried out on 27 computers with Pentium 4 dual channel processor and 2 GB RAM connected with each other by 1-Gb Ethernet network infrastructure). These tests depended on running sixty processes in the same time (each process needed about one minute to be executed). In Figure 3 average results are shown.

The time of execution of all processes for 27 nodes, on the average, is over twenty times faster than for a single PC.

3. EXAMPLES OF HEURISTIC MODELING

In the following part of the paper two examples of applications will be demonstrated. The first one shows results of using the computing environment and commercial Matlab environment (without distributed engine and also without optimization of network topology) for modeling a dynamic system. The second one presents results of modeling an industrial furnace with using the distributed system only.

3.1. Modeling dynamic system

For the given input-output pairs *{u(k),y(k)}* generated by the non-linear system defined by:

$$y(k)=\frac{y(k-1)\cdot y(k-2)\cdot u(k)\cdot u(k-2)\cdot[u(k)-1]+u(k-1)}{1+y(k-2)^2+u(k)^2}, \quad (1)$$

identify a topology of the network and its parameters. In the training stage the input-target patterns were represented as follows:

$$\begin{gathered}\{(\mathbf{P}(k),\mathbf{T}(k))\mid k=1,\dots,N\}\\ \mathbf{P}(k)=[u(k)\quad u(k-1)\quad \dots\quad u(k-5)], \qquad (2)\\ \mathbf{T}(k)=[y(k-1)\quad y(k-2)\quad \dots\quad y(k-5)]\end{gathered}$$

where: *u(k)* was initialized with random values (1000 samples with a uniform distribution in the range <-2; 1.2>). In the testing stage three different input signals were used: polyharmonic signal, hard-limit signal and pseudo-random binary sequence (in Tables 1 and 2 results only for the first signal are shown).

For this task two experiments were done. Firstly, the authors used simple scripts written using Matlab language. There were implemented three kinds of a neural network: a simple feed-forward neural network (*ffbp*), feed-forward neural network with delay lines (*ffbptd*) and locally recurrent one $iir^{(2,1)}arx^{(1,1)}$. Their topologies and training methods will not be discussed in this paper. The exemplary results are presented in Table 1.

Table 1. Results obtained for static and locally recurrent neural networks (using Matlab program without distributed engine)

J [-]	U [-]	MAPE [%]	Neural network
8,35E-05	**1,19E-01**	2,28E-01	*ffbptd*
9,12E-05	**2,11E-01**	2,56E-01	$iir^{(2,1)}arx^{(1,1)}$
12E-05	**2,28E-01**	2,76E-01	*ffbp*

Table 2. Results obtained for feed-forward neural networks with optimizing their structures (using distributed computing system)

	J [-]	U [-]	MAPE [%]	Neural network
1	5,28E-05	**1,52E-01**	1,87E-01	11.8.21.1
2	5,47E-05	**1,54E-01**	1,97E-01	11.21.20.1
3	5,54E-05	**1,55E-01**	1,99E-01	11.16.16.1
4	5,66E-05	**1,57E-01**	2,04E-01	11.12.12.1
.	.	.	.	.

Next, the authors employed the distributed computing system in order to find optimal feed-forward neural network architecture. Before starting optimization of a structure learning algorithms and their parameters were selected (simulated annealing for global optimization and backpropagation algorithm with momentum technique for local optimization). The maximum network complexity was set up as a network with 11 neurons with hyperbolic activation function in the input layer, 30 neurons with hyperbolic function in the first hidden layer, 30 neurons with hyperbolic function in the second hidden layer and one unit with linear transfer function in the output layer. About eight hours were necessary to find an optimal network topology with using the distributed system (with 27 nodes). The results received during the testing stage are included in Table 2.

3.2. Modeling electric furnace

This example describes the application of the discussed distributed system to data gathered from the real process. In order to create a submodel of the process which is described in [4] a non-linear discrete difference equation is proposed (Eq. 2, inputs and output were selected based on information from staff maintaining this object and technical documentation dealing with this process):

$$\hat{y}_{Cu}(k+\Delta k) = \hat{f}\left[\mathbf{I}(k), \mathbf{L}(k), \hat{\mathbf{I}}(k), \hat{\mathbf{L}}(k), p(k), y_{Cu}(k)\right], \quad (3)$$

where: $\mathbf{I}(k)$ is a vector of three currents, $\mathbf{L}(k)$ is a vector that determines positions of three electrodes; $p(k)$ is a position of a claw of the transformer; $y_{Cu}(k)$ is a current state parameter describing the copper concentration in slag; Δk is the time horizon of prediction (for this example it equals 15, 20 and 60 min.); $\hat{f}$ represents a non-linear input-output relation of the feed-forward neural network; $\hat{\mathbf{I}}(k), \hat{\mathbf{L}}(k)$ are suitably preprocessing currents and electrodes positions.

Table 3. Results received for four-layered feed-forward neural networks with optimizing their structures (using distributed computing system)

	J [-]	U [-]	MAPE [%]	Neural network
Δk=15				
1	4,11E-03	**9,54E-01**	2,43E+00	15.7.11.1
2	4,17E-03	**9,64E-01**	2,63E+00	15.4.12.1
3	4,18E-03	**9,64E-01**	2,72E+00	15.7.4.1
.	.	.	.	.
Δk=20				
1	7,27E-03	**9,58E-01**	3,14E+00	15.11.6.1
2	7,40E-03	**9,65E-01**	3,22E+00	15.9.6.1
3	7,44E-03	**9,67E-01**	3,20E+00	15.8.7.1
.	.	.	.	.
Δk=60				
1	4,52E-02	**8,36E-01**	8,75E+00	15.8.5.1
2	4,65E-02	**8,49E-01**	8,49E+00	15.5.5.1
3	4,67E-02	**8,51E-01**	8,60E+00	15.6.4.1
.	.	.	.	.

In this example the authors have proceeded similarly as in the first case. Nevertheless, here four-layered network (15 neurons in the input layer with hyperbolic transfer function, two hidden layers with 30 neurons in each with hyperbolic function as well and one output neuron with linear activation function) was set up as the maximum network complexity. Some results of optimization are given in Table 3. They are not as good as in the first task but they are still acceptable. Note that the modeled plant is very complex, therefore simple feed-forward networks are not able to represent their completely dynamic behavior (authors are going to employ Jordan and Elman networks to do so in the future).

4. SUMMARY

With respect to the experiments conducted here it can be stated that generally, a distributed computing is very fascinating domain of research. The authors have presented a complete computing environment that might be applied for modeling tasks (using soft modeling methods). It is a tool free of charge in case of research purposes.

Note that the authors did not want to show that their system is better than commercial products but they would like to present some advantages of freeware systems and their applications.

5. FUTURE WORK

For the reason that systematic-search algorithms are not very effective the authors will try to use the ENZO application that offers genetic algorithm for optimizing network topology (it is an extension of the SNNS). In case when migration of this application would not be possible to realize, they would create their own BASH scripts. There will also be a trial with applying other types of architectures such as: Jordan networks, Elman and extended hierarchical Elman networks.

ACKNOWLEDGMENTS

Paper presents a selected part of the research supervised by Prof. W. Moczulski. This research has been partially supported by the Ministry of Education and Scientific Research under grant No. 4 T07B 018 27.

REFERENCES

[1] Barak, A.; Shiloh, A.; Amar, L.: *An organizational grid of federated MOSIX clusters.* Computer Systems and Software Engineering, 1996., Proceedings of the Seventh Israeli Conference on 12-13 June 1996 Page(s):38 - 45 Digital Object Identifier 10.1109/ICCSSE. 1996.554847

[2] Lottiaux, R.; Gallard, P.; Vallee, G.; Morin, C.; Boissinot, B.: *OpenMosix, OpenSSI and Kerrighed: a comparative study*. Cluster Computing and the Grid, 2005. CCGrid 2005. IEEE International Symposium on Volume 2, 9-12 May 2005 Page(s):1016 - 1023 Vol. 2 Digital Object Identifier 10.1109/CCGRID. 2005.1558672.

[3] Buyya R. *High Performance Cluster Computing: Architectures and Systems*. Prentice Hall PTR, NJ, USA, 1999.

[4] Moczulski W.: *Methodology of Heuristic Modelling of Dynamic Objects and Processes for Diagnostics and Control.:* Recent Developments in Artificial Intelligence Methods. AI-METH 2005, p. 123 – 126.

[5] Korbicz J., Koscielny J.M., Kowalczuk Z., Cholewa W. (Eds.): *Fault diagnosis. Models, Artificial Intelligence, Applications.*: Springer-Verlag Berlin Heidelberg 2004 New York. ISBN 3-540-40767-7.

[6] http://openmosix.sourceforge.net/, The openMosix Project.

[7] http://www-ra.informatik.uni-tuebingen.de /SNNS/, Stuttgart Neural Network Simulator home page.

[8] http://yolinux.com/TUTORIALS/LinuxTutorial PosixThreads.html#BASICS, YoLinux Tutorial: POSIX thread (pthread) libraries.

[9] http://lcic.org, Linux Clustering Information Center.