computing@tanet.edu.te.ua
www.tanet.edu.te.ua/computing

ISSN 1727-6209
International Scientific
Journal of Computing

EFFICIENCY ESTIMATION OF PARALLEL ALGORITHM OF ENHANCED HISTORICAL DATA INTEGRATION ON COMPUTATIONAL GRID

V. Turchenko [1-2], C. Triki [3], L. Grandinetti [1] and A. Sachenko [2]

[1] Center of Excellence of High Performance Computing, University of Calabria,
Via P. Bucci 22B, 87036, Rende (CS), ITALY
[2] Research Institute of Intelligent Computer Systems, Ternopil State Economic University,
Peremoga Square 3, 46004, Ternopil, UKRAINE
[3] Department of Mathematics, University of Lecce, 73100, Lecce, ITALY

Abstract: *The main feature of neural network using for accuracy improvement of physical quantities (for example, temperature, humidity, pressure etc.) measurement by data acquisition systems is insufficient volume of input data for predicting neural network training at an initial exploitation period of sensors. The authors have proposed the technique of data volume increasing for predicting neural network training using integration of historical data method. In this paper we have proposed enhanced integration historical data method with its simulation results on mathematical models of sensor drift using single-layer and multi-layer perceptrons. We also considered a parallelization technique of enhanced integration historical data method in order to decrease its working time. A modified coarse-grain parallel algorithm with dynamic mapping on processors of parallel computing system using neural network training time as mapping criterion is considered. Fulfilled experiments have showed that modified parallel algorithm is more efficient than basic parallel algorithm with dynamic mapping, which does not use any mapping criterion.*

Keywords: *sensor drift, integration historical data, neural networks, coarse-grain parallel algorithm, dynamic mapping, computational grids.*

1. INTRODUCTION

The authors have shown in [1-2], that the error of modern sensor data acquisition systems is much less than sensor's error in many cases. The accuracy improvement of physical quantity measurement is provided by (i) sensor calibration using special calibrator or (ii) sensor's periodic testing by reference sensor directly on exploitation place [3]. The frequency of calibration/testing procedure is called as inter-testing interval. However operations implementing calibration/testing procedures are rather laborious. Sensor drift prediction during inter-testing interval can reduce the laboriousness tremendously [1]. However, well known prediction methods, for example 5-degree polynomial, curvilinear alignment and cubic splines do not provide satisfactory results [2, 4]. Using artificial intelligence methods, in particularly, neural networks are more effective in this case [5-7].

Prediction using neural networks is used very widely and in the same time improvement of prediction accuracy traditionally is reached by improvement of neural network structure, using different neurons' activation functions, training algorithms, etc [8]. However mentioned approaches often do not provide satisfactory results and therefore it is necessary to use methods of special forming of neural network training set. Two such methods, additional approximating neural network and integration of historical data using set of Integrating Historical Data Neural Networks (IHDNNs), have been proposed and experimentally investigated in [2, 7, 9-10]. These methods allow considerably decreasing number of sensor calibration/testing by artificial increasing of the training set of predicting neural network. Experimental results of these methods showed [2, 4] that they allow increasing an accuracy of sensor drift prediction in 3-5 times at simultaneous increasing of inter-testing interval in 6-12 times. The enhanced method of integration historical data and its simulation modelling on mathematical models of sensor drift in comparison with the basis method of data integration are considered below.

The works [7, 9] show, that the method of integration historical data could require considerable computational recourses and time for its execution. In the previous works Turchenko has developed coarse-grain parallelization algorithms for set of Integrating Historical Data Neural Networks with static [11] and dynamic [12] mapping onto processors of parallel computer. The goal of this

paper is to estimate an accuracy of enhanced integration historical data method in comparison with the basic method and to investigate an efficiency of its parallelization on the parallel computer systems. The parallelization allows reducing total execution time of the method in order to be used in real-time intelligent data acquisition systems.

This paper is organised as follows. The basic integration historical data method is described in Section II, the difference of the enhanced integration historical data method is outlined in Section III, the architecture of multi-layer perceptron used as IHDNN model is considered in Section IV, the experimental researches of enhanced algorithm are compared with the basic method in Section V, an approach to enhanced integration historical data method parallelization is considered in Section VI, parallelization experiments are presented in Section VII, summary of the results concludes this paper in Section VIII.

2. BASIC INTEGRATION HISTORICAL DATA METHOD

It is proposed to use three groups of data of sensor drift in [7]: real, historical and hypothetical data. The real data are not available at the beginning of sensor exploitation. Using historical data, obtained as result of calibration or testing of the same type sensors in the similar operation conditions in the same data acquisition channel, can compensate this disadvantage.

The historical data should be integrated in order to account individual properties of drift of each sensor, those data are used as historical [13]. It is proposed to use a set of Integrating Historical Data Neural Networks for such integration. Let us consider the historical data of sensor drift as curves $x1...xn$ (Fig. 1), which are equal to values $xai, xbi, ..., xfi$, $i = \overline{1,n}$ into calibration points $a, b, ..., f$, where n is the number of available historical sensor drift curves. The first calibration of the new sensor allows correcting initial sensor error at 0 moment of time. The second calibration of the new sensor allows receiving the first real value xak of sensor drift in calibration point a. The goal of the IHDNN is to predict value xbk on the basis of xak and xai, $i = \overline{1,n}$, to predict next value xck on the basis of xbk and xbi, $i = \overline{1,n}$ etc. It is necessary to form training and prediction sets of IHDNN for fulfilling this task in special way [13].

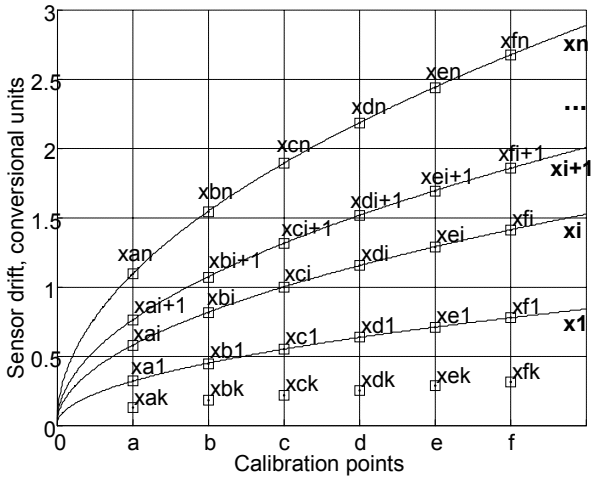

Fig. 1 - Historical data about sensor drift.

Thus, the values that should be used to form the training set of the IHDNN in order to predict each value $xbk, xck, ..., xfk$ can be described by following expressions:

$$xbk \Leftarrow \{xak, xai\}, i = \overline{1,n}, \quad (2.1)$$

$$xck \Leftarrow \{xbk, xbi\}, i = \overline{1,n}, \quad (2.2)$$

$$xdk \Leftarrow \{xck, xci\}, i = \overline{1,n}, \quad (2.3)$$

$$xek \Leftarrow \{xdk, xdi\}, i = \overline{1,n}, \quad (2.4)$$

$$xfk \Leftarrow \{xek, xei\}, i = \overline{1,n}. \quad (2.5)$$

The general algorithm of IHDNNs training and prediction sets forming [13] can be described by the following steps (see Fig. 1):

1. To choose the input data described by (2.1).
2. To form IHDNN training set in order to predict xbk value by the following:
   2.1. To choose one curve of sensor drift xi which will describe real data about sensor drift, all other curves xj, $j = \overline{1,i-1}$, $j = \overline{i+1,n}$ will describe historical data;
   2.2. To calculate absolute deviation of xai values from all other xaj values according to $\Delta_{ij} = |xai - xaj|$, where $i = \overline{1,n}$, $j = \overline{1,i-1}$, $j = \overline{i+1,n}$;
   2.3. To sort all absolute deviations Δ_{ij}, calculated in the previous step 2.2, in decreasing order; to calculate maximum $\Delta_{ij}^{\max} = \max \Delta_{ij}$ and minimum $\Delta_{ij}^{\min} = \min \Delta_{ij}$ values of the absolute deviations;
   2.4. To generate i training vector as the set of values xbi, xai, xaj, where xaj, $j = \overline{1,i-1}$, $j = \overline{i+1,n}$ values must be putted into training vector according to sorted (in decreasing order) values of absolute

deviations xaj from value xai according to the expression (2.6);

2.5. To repeat the steps 2.1-2.4 for all $i = \overline{1,n}$.

3. To form IHDNN prediction set in order to predict xbk value by the expression (2.7).
4. To choose the input data described by the next expression from the expressions (2.2)-(2.5).
5. To execute steps 2-4 above in order to form the training and prediction sets for all IHDNNs, which will predict the next values of sensor drift xck, xdk, xek, xfk.

$$\left\{xaj\middle|\Delta_{ij} = \Delta_{ij}^{\max}, xaj\middle|\Delta_{ij} = \Delta_{ij}^{\max} - 1, ..., xaj\middle|\Delta_{ij} = \Delta_{ij}^{\min}, xai\right\} \Rightarrow xbi \quad (2.6)$$

$$\left\{xaj\middle|\Delta_{ij} = \Delta_{ij}^{\max}, xaj\middle|\Delta_{ij} = \Delta_{ij}^{\max} - 1, ..., xaj\middle|\Delta_{ij} = \Delta_{ij}^{\min}, xak\right\} \Rightarrow xbk \quad (2.7)$$

It is necessary to note, that the length of prediction vector (2.7) should be equal to the length of the training vector (2.6) for appropriate functioning of the integration historical data method. The data set (2.1) for forming the training and prediction sets (2.6)-(2.7) can be considered as "window" of historical data. Execution of step 5 of the algorithm above requires (i) shifting this window on one position to the right and choosing the input data according to (2.2) for xck value prediction, (ii) shifting this window on two positions to the right and choosing the input data according to (2.3) for xdk value prediction, (iii) shifting this window on three positions to the right and choosing the input data according to (2.4) for xek value prediction, (iii) shifting this window on four positions to the right and choosing the input data according to (2.5) for xfk value prediction. These shifts are noted in relation to the current position of the "window" described by (2.1). Thus, according to the basic algorithm of integration historical data, the training and prediction sets of each IHDNN is forming separately and therefore each IHDNN implementation does not relate to any other IHDNN implementation by input data.

3. ENHANCED INTEGRATION HISTORICAL DATA METHOD

The basic method takes into account the values of sensor drift, which are placed in the "window" only and it is a disadvantage of this method. For example, these values are xai and xbi, $i = \overline{1,n}$ for the calibration point b. At the same time the character of each available sensor drift curve is not accounting in relation to the investigated sensor. The main idea of enhanced integration historical data method is the necessity to take into account all the values of sensor drift located on all available historical sensor drift curves [10]. Thus, the goal of IHDNN is to predict value xbk on the basis of xak and historical values xai, $i = \overline{1,n}$, to predict next value xck on the basis of $\{xbk, xak\}$ and historical values $\{xbi, xai\}$, $i = \overline{1,n}$, etc (see Fig. 1). The values that should be used to form the training set of the IHDNN according to enhanced method can be described by expressions (3.1)-(3.5).

The general algorithm of IHDNNs training and prediction sets forming within enhanced method can be described by the following steps (see Fig.1):

1. To choose the input data described by (3.1).
2. To form the training and prediction sets of IHDNN which will predict value xbk analogously to steps 2 and 3 of the basic method from Section II.
3. To choose the input data described by (3.2).
4. To form IHDNN training set in order to predict xck value by the following:

4.1. To choose one curve of sensor drift xi which will describe real data about sensor drift, all other curves xj, $j = \overline{1,i-1}$, $j = \overline{i+1,n}$ will describe historical data;

4.2. To calculate absolute deviation of xbi values from all other xbj values according to $\Delta_{ij} = |xbi - xbj|$, where $i = \overline{1,n}$, $j = \overline{1,i-1}$, $j = \overline{i+1,n}$;

$$xbk \Leftarrow \{xak, xai\}, i = \overline{1,n} \quad (3.1)$$

$$xck \Leftarrow \{\{xbk, xak\}, \{xbi, xai\}\}, i = \overline{1,n} \quad (3.2)$$

$$xdk \Leftarrow \{\{xck, xbk, xak\}, \{xci, xbi, xai\}\}, i = \overline{1,n} \quad (3.3)$$

$$xek \Leftarrow \{\{xdk, xck, xbk, xak\}, \{xdi, xci, xbi, xai\}\}, i = \overline{1,n} \quad (3.4)$$

$$xfk \Leftarrow \{\{xek, xdk, xck, xbk, xak\}, \{xei, xdi, xci, xbi, xai\}\}, i = \overline{1,n} \quad (3.5)$$

4.3. To sort all absolute deviations Δ_{ij}, calculated in the previous step 4.2, in decreasing order; to calculate maximum $\Delta_{ij}^{\max} = \max \Delta_{ij}$ and minimum $\Delta_{ij}^{\min} = \min \Delta_{ij}$ values of the absolute deviations;

4.4. To generate i training vector placing the values $\{xbj, xaj\}$ in decreasing order according to the values of absolute deviations Δ_{ij} from value xbi by expression (3.6);

4.5. To repeat steps 4.1-4.4 for all $i = \overline{1,n}$.

5. To form IHDNN prediction set in order to predict xck value by the following expression (3.7).
6. To choose the input data described by the next expression from the expressions (3.3)-(3.5).
7. To execute steps 4-6 above in order to form the training and prediction sets for all IHDNNs, which will predict the next values of sensor drift xdk, xek, xfk. The training and prediction sets should be formed by (3.8)-(3.9) for the xdk value, by (3.10)-(3.11) for the xek value and by (3.12)-(3.13) for the xfk value.

$$\left\{\{xaj, xbj\}\middle|\Delta_{ij} = \Delta_{ij}^{\max}, \{xaj, xbj\}\middle|\Delta_{ij} = \Delta_{ij}^{\max} - 1, ..., \{xaj, xbj\}\middle|\Delta_{ij} = \Delta_{ij}^{\min}, \{xai, xbi\}\right\} \Rightarrow xci \quad (3.6)$$

$$\left\{\{xaj, xbj\}\middle|\Delta_{ij} = \Delta_{ij}^{\max}, \{xaj, xbj\}\middle|\Delta_{ij} = \Delta_{ij}^{\max} - 1, ..., \{xaj, xbj\}\middle|\Delta_{ij} = \Delta_{ij}^{\min}, \{xak, xbk\}\right\} \Rightarrow xck \quad (3.7)$$

$$\left\{\{xaj, xbj, xcj\}\middle|\Delta_{ij} = \Delta_{ij}^{\max}, \{xaj, xbj, xcj\}\middle|\Delta_{ij} = \Delta_{ij}^{\max} - 1, ..., \{xaj, xbj, xcj\}\middle|\Delta_{ij} = \Delta_{ij}^{\min}, \{xai, xbi, xci\}\right\} \Rightarrow xdi \quad (3.8)$$

$$\left\{\{xaj, xbj, xcj\}\middle|\Delta_{ij} = \Delta_{ij}^{\max}, \{xaj, xbj, xcj\}\middle|\Delta_{ij} = \Delta_{ij}^{\max} - 1, ..., \{xaj, xbj, xcj\}\middle|\Delta_{ij} = \Delta_{ij}^{\min}, \{xak, xbk, xck\}\right\} \Rightarrow xdk \quad (3.9)$$

$$\left\{\{xaj, xbj, xcj, xdj\}\middle|\Delta_{ij} = \Delta_{ij}^{\max}, ..., \{xaj, xbj, xcj, xdj\}\middle|\Delta_{ij} = \Delta_{ij}^{\min}, \{xai, xbi, xci, xdi\}\right\} \Rightarrow xei \quad (3.10)$$

$$\left\{\{xaj, xbj, xcj, xdj\}\middle|\Delta_{ij} = \Delta_{ij}^{\max}, ..., \{xaj, xbj, xcj, xdj\}\middle|\Delta_{ij} = \Delta_{ij}^{\min}, \{xak, xbk, xck, xdk\}\right\} \Rightarrow xek \quad (3.11)$$

$$\left\{\{xaj, xbj, xcj, xdj, xej\}\middle|\Delta_{ij} = \Delta_{ij}^{\max}, ..., \{xaj, xbj, xcj, xdj, xej\}\middle|\Delta_{ij} = \Delta_{ij}^{\min}, \{xai, xbi, xci, xdi, xei\}\right\} \Rightarrow xfi \quad (3.12)$$

$$\left\{\{xaj, xbj, xcj, xdj, xej\}\middle|\Delta_{ij} = \Delta_{ij}^{\max}, ..., \{xaj, xbj, xcj, xdj, xej\}\middle|\Delta_{ij} = \Delta_{ij}^{\min}, \{xak, xbk, xck, xdk, xek\}\right\} \Rightarrow xfk \quad (3.13)$$

As in the case of basic method, the length of training and prediction sets for any value $xbk, xck, ..., xfk$ should be the same. The training set for each predicted sensor drift value is forming independently, therefore each IHDNN implementation does not relate to any other IHDNN implementation by input data. The models of single-layer perceptron described in details in [4, 7-9] and multi-layer perceptrons described below are used as IHDNNs.

4. MATHEMATICAL MODEL OF IHDNNS

IHDNNs should provide a non-linear transfer function because the drift of the majority of modern sensors is non-linear. Therefore the multi-layer perceptron should be used as a model for the IHDNN with a nonlinear activation function, such us the logistic function. This kind of neural networks has the advantage of being simple and to provide nice generalized properties [8].

The output value of three-layer perceptron (Fig. 2) can be formulated as:

$$y = F_3\left(\sum_{i=1}^{N} w_{i3} h_i - T\right), \quad (4.1)$$

where N is the number of neurons in the hidden layer, w_{i3} is the weight of the synapse from neuron i in the hidden layer to the output neuron, h_i is the output of neuron i, T is the threshold of the output neuron and F_3 is the activation function of the output neuron.

The output value of neuron j in the hidden layer is given by:

$$h_j = F_2\left(\sum_{i=1}^{M} w_{ij} x_i - T_j\right), \quad (4.2)$$

where w_{ij} are the weights from the input neurons to neuron j in the hidden layer, x_i are the input values

and T_j is the threshold of neuron j. The logistic activation function is used for the neurons of the hidden layer and the linear activation function, having a coefficient k, is used for the output neuron.

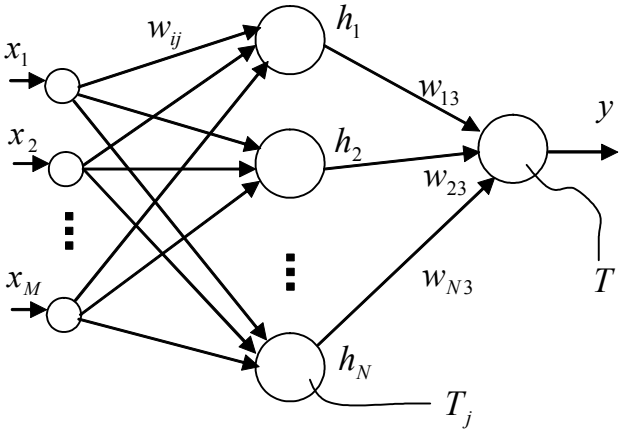

Fig. 2. The structure of IHDNN

The back propagation error algorithm [4] is used for the training algorithm. It is based on the gradient descent method and provides an iterative procedure for the weights and thresholds updating for each training vector p of the training sample:

$$\Delta w_{ij}(t) = -\alpha \frac{\partial E^p(t)}{\partial w_{ij}(t)}, \quad \Delta T_j(t) = -\alpha \frac{\partial E^p(t)}{\partial T_j(t)}, \quad (4.3)$$

where α is the learning rate, $\frac{\partial E^p(t)}{\partial w_{ij}(t)}$ and $\frac{\partial E^p(t)}{\partial T_j(t)}$ are the gradients of the error function on each iteration t for the training vector p with $p \in \{1,...,P\}$, where P is the size of the training set.

The Sum-Squared Error (SSE), for training iteration t, is calculated as:

$$E^p(t) = \frac{1}{2}\left(y^p(t) - d^p(t)\right)^2, \quad (4.4)$$

where for the training vector p, $y^p(t)$ is the output value on iteration t and $d^p(t)$ is the target output value.

During training, the total error is calculated as:

$$E(t) = \sum_{p=1}^{P} E^p(t). \quad (4.5)$$

The steepest descent method for calculating the learning rate [8] is used for removing the classical disadvantages of the back propagation error algorithm. Thus, the adaptive learning rate for the logistic and linear activation functions are given, respectively, by:

$$\alpha(t) = \frac{4}{\left(1+(x_i^{p}(t))^2\right)} \times \frac{\sum_{j=1}^{N}(\gamma_j^p(t))^2 h_j^p(t)(1-h_j^p(t))}{\left(\sum_{i=1}^{N}(\gamma_j^p(t))^2 (h_j^p(t))^2 (1-h_j^p(t))^2\right)},$$

$$\alpha(t) = \frac{1}{\sum_{i=1}^{N}(h_i^p(t))^2 + 1} \quad (4.6)$$

where, for the training vector p and iteration t, $\gamma_j^p(t)$ is the error of neuron j and $h_i^p(t)$ is the input signal of the linear neuron.

The error of neuron i with logistic activation function can be determined by the relation:

$$\gamma_i^p(t) = \sum_{j=1}^{N} \gamma_3^p(t) w_{i3}(t) h_j^p(t)(1-h_j^p(t)), \quad (4.7)$$

where $\gamma_3^p(t) = y^p(t) - d^p(t)$ is the error of the output neuron, $w_{i3}(t)$ is the weight of the synapses between the neurons of the hidden layer and the output neuron.

A slight modification of the back propagation error algorithm, called multiple propagation error, has been implemented in order to stabilize the training process [4]. This approach consists in modifying the weights of only one layer of the neural network during a single training iteration. This algorithm includes thus the following steps:

1. Set the desired value of SSE to $E_{\min}$;
2. Initialize the weights and the thresholds of the neurons by values in the range (0-0.5);
3. Set a counter for the number of neural network layers, $LAYERS$;
4. If $LAYERS = 2$ then calculate the output value $y^p(t)$ using expression (4.2) for the training vector p and perform the steps 5 and 6;
5. Calculate the error of the output neuron: $\gamma_3^p(t) = y^p(t) - d^p(t)$;
6. Update the weights and the thresholds of the output neuron by (4.3) using the adaptive learning rate given by (4.6);
7. Decrease the number of current layer $LAYERS$ by one unit;
8. If $LAYERS = 1$ then calculate the error $\gamma_j^p(t)$ of the neurons of the hidden layer by (4.7);
9. Update the weights and the thresholds of the neurons of the hidden layer by (4.3) using the adaptive learning rate (4.6) for the logistic activation function;
10. Calculate the SSE for the training iteration t using (4.4);
11. Repeat the steps from 3 to 10 for all the other vectors in the training set;
12. Calculate the total SSE, $E(t)$ of the neural network using (4.5);

13. If $E(t)$ is still greater than the desired error $E_{\min}$ then go to step 3, otherwise stop the training process.

5. EXPERIMENTAL RESEARCHES OF BASIC AND ENHANCED METHODS

As discussed in Sections II and III, it is necessary to have f copies of the IHDNN to predict f drift values of the new sensor k (Fig. 1). In [4], Turchenko has shown that 6 values on the drift curve of the new sensor k are enough to provide a prediction in the future moments of time. Therefore we set f =6. However, for more complex intelligent data acquisition systems and next generation sensors it may worth investigating innovative solutions with multiple data acquisition channels. Scientific investigations with larger number of sensor drift curves help to discover the limitation conditions of the enhanced data integration method, its potential abilities, how to form training vectors, and to determine the optimal number of hidden neurons, etc. In our case the historical data integration was conducted by using 10 sensor drift curves [4], therefore n is assumed to be 10.

The use of real data of the sensor calibration is not expedient for IHDNNs investigation because real data do not fully describe the behavior of a sensor drift. Thus, mathematical models of sensor drift are usually developed for experimental researches [14]. The results of industrial sensors calibrations in real environment are the basis of these mathematical models. The real data about the drift are supplemented by additional components that model non-stationarity and the non-uniformity of the drift, systematic and random errors of standard sensors, methodical errors, noises and other errors. As a result the model “with saturation” (Fig. 3) corresponds to the drift of the thermo-resistor 30K5A1 at a working temperature of 150°C has been used for the experiments.

The total number of IHDNNs used in this study of basic and enhanced integration historical data methods is equal $N = n \times f = 50$. The expressions (2.6)-(2.7) are used to form the training and prediction sets for all predicting values according to the basic method.

The expressions (2.6)-(2.7) for the *xbk* value, (3.6)-(3.7) for the *xck* value, (3.8)-(3.9) for the *xdk* value, (3.10)-(3.11) for the *xek* value and (3.12)-(3.13) for the *xfk* value are used to form the training and prediction sets according to the enhanced method of integration historical data. We have considered several scenarios in training of each IHDNN when sum-squared error (SSE) has been set to SSE=10^{-3}, 10^{-4}, 10^{-5} and 10^{-7}.

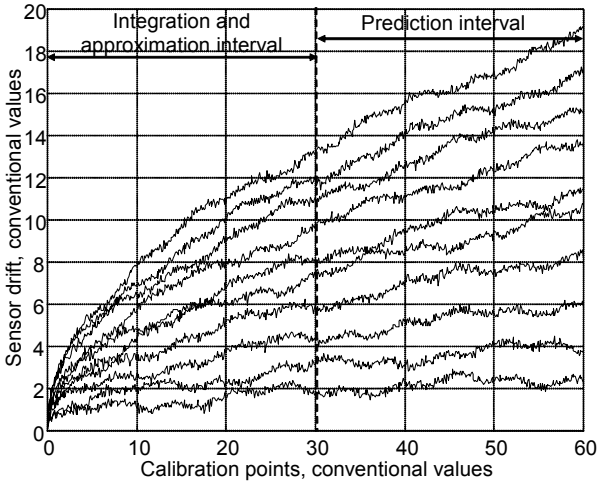

Fig. 3. Mathematical model of sensor drift "with saturation".

In a case of basic integration historical data method single-layer perceptron has 9 inputs. Multi-layer perceptron has 9 input neurons, 7 neurons in the hidden layer with logistic activation function and one output neuron. The number of input neurons can be justified by the fact that in the case of 10 sensor drift curves one of them should play the role of an investigated curve and the other curves are used to form the training vector of the IHDNN according to (2.6). The number of the hidden neurons should be less than the number of the input data (corresponding in our case to 7 neurons) in order to ensure good generalization properties of the IHDNN and to avoid transforming it to an associative memory [8]. The output neuron provides the predicted value of the sensor drift for future calibration moments b,c,d,e,f which is the only result expected from each IHDNN.

In a case of enhanced integration historical data method we have used the same number of neurons of hidden and output layers of multi-layer perceptron. The difference in the neural network structure is relating to number of input neurons of single-layer and multi-layer perceptrons used for prediction the sensor drift values in different calibrations b,c,d,e,f. In particular, IHDNN had 9 inputs for the *xbk* value prediction, 18 inputs for the *xck* value prediction, 27 inputs for the *xdk* value prediction, 36 inputs for the *xek* value prediction and 45 inputs for the *xfk* value prediction.

The experimental results, comparing an accuracy of the basic and enhanced methods of integration historical data, have showed in Fig. 4. The averaged relative error of data integration within basic method is equal to 14-57% at using single-layer perceptron and 12-20% at using multi-layer perceptron. The averaged relative error of data integration within enhanced method is equal to 5-9%. Taking this result we have got together the best results provided

as single-layer as well as multi-layer perceptrons. Thus, enhanced method of integration historical data improves accuracy in two times in comparison with the basic method.

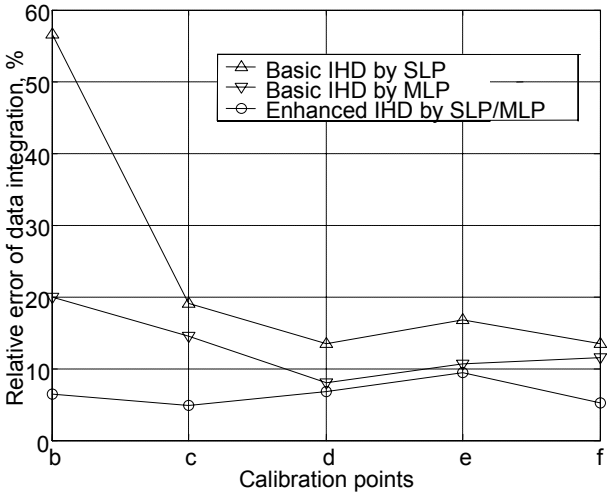

Fig. 4. Accuracy comparison of basic and enhanced integration historical data (IHD) method.

However, experiments have shown that enhanced integration historical data method requires considerable computational and time recourses for its execution. The distribution of IHDNN (multilayer perceptron) training times among all calibration points at mentioned scenarios of SSE=10^{-3}, 10^{-4}, 10^{-5} and 10^{-7} is presented in Fig. 5. In particular, training time is equal to 36 seconds approximately at IHDNN training till SSE=10^{-7} for all calibration points xbk, xck, xdk, xek, xfk at usage of personal computer with Intel Celeron 1,8 GHz processor, 256 Mb RAM and Windows® operation system. Taking into account additional time needed for approximation of these values and prediction of the sensor drift in the future moments of time, the time of enhanced integration historical data method execution considerably decreasing an efficiency of its application in the real-time intelligent data acquisition systems. In a case of scientific research using 10 curves of sensor drift increases the computational time in 10 times at least. Therefore let us consider parallelization approaches of the enhanced integration historical data algorithm in the Section V below.

6. PARALLELIZATION OF ENHANCED INTEGRATION HISTORICAL DATA METHOD

Several approaches to parallelize neural networks have been proposed in the literature: according to the architecture of the network [15], taking advantage of the matrix learning rule calculations [16], or parallelizing the presentation of examples [17]. In [18], three nested levels of parallelism in neural algorithms have been considered: connection parallelism (parallel execution on sets of weights), node parallelism (parallel execution of operations on sets of neurons), and example (modular) parallelism (parallel execution of examples on replicated networks).

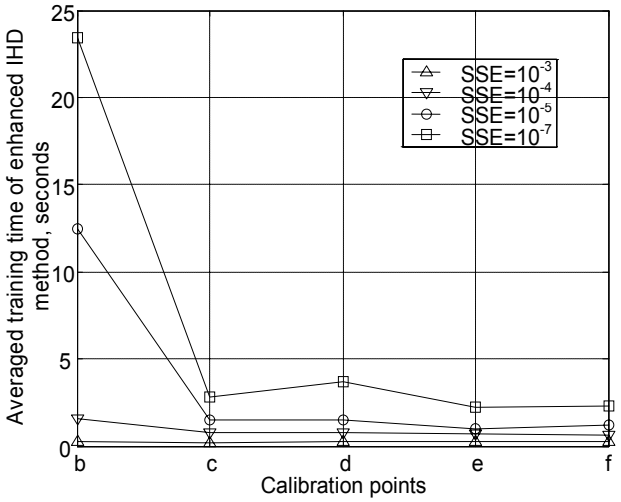

Fig. 5. Distribution of the training times of enhanced IHD method among calibration points.

The first two levels are a fine-grain parallelism and the third level is a coarse-grain parallelism. Fine-grain parallel algorithms require a lot of low-level communications, for example to combine the results of the parallel calculation of weights of each neuron. Their use is, thus, more effective on processors arrays or network of transputers [19-20]. Vice versa coarse-grain algorithms are useful when big independent computation tasks should be processed and communications are rarely required. The use of high-performance computers with powerful of the parallel processors is recommended for the implementation of such parallel algorithms [15].

Turchenko in [11-12] has proposed to use coarse-grain approach to parallelize the basic method of integration historical data and developed two coarse-grain parallel algorithms with static and dynamic mapping of IHDNNs onto processors of parallel computer. The experimental results, achieved on parallel computers SGI Origin 300 and NEC TX-7, showed that parallel algorithm with dynamic mapping has better efficiency on 13% in comparison with parallel algorithm with static mapping for mathematical model of sensor drift "with saturation". Therefore we will use coarse-grain parallel algorithm with dynamic mapping for parallelization of the set of Integrating Historical Data Neural Networks according to enhanced integration historical data method.

Parallel algorithm with dynamic mapping [12] is developed by using a "centralized" planning approach with only one processor *Master* having the

role of task planner and each of the other processors (called *Slaves*) will train the IHDNNs assigned by the *Master*. The *Master* starts with assigning the first M IHDNNs to the M available processors and then continues the execution of the mapping procedure consisting in assigning dynamically the tasks to the *Slaves* as soon as they become idle. Once a *Slave* has finished its task, it asks the *Master* for a new one till no tasks are left. We note here that, besides the role of planner, the *Master* does not fulfill any further calculation. The communications between the *Master* and the *Slaves* are ensured by using the standard MPI sending/receiving functions *MPI_Send()* and *MPI_Recv()*. More details about this algorithm can be founded in [12].

It was shown in [12] that reducing of the efficiency of this parallel algorithm is caused by a non-uniform training time of the IHDNNs. As a possible solution it was proposed to estimate a priori the training time of each IHDNN module. In this case it will be possible to use the information produced by such a pre-processing procedure in order to improve the load balance among the different *Slaves*. The analysis of the IHDNNs training time distribution according to enhanced integration historical data (Fig. 5) shown, that the IHDNNs, which predict sensor drift values in b calibration, have biggest training time at SSE value increasing. For example, the average training time is equal to 15 and 25 seconds at $SSE=10^{-5}$ and $SSE=10^{-7}$ respectively. We propose to use this training time as mapping criteria and modify therefore basic parallel IHDNN parallelization algorithm with dynamic mapping. Thus, the most "long" tasks, i.e. IHDNNs predicting sensor drift in calibration b for all curves, should be mapped first. All other IHDNNs can be mapped after since all of them have practically equal training time (see Fig. 5). The modification of the parallel algorithm consists in calculation of the IHDNN reference number taking into account the training time of this IHDNN module.

7. PARALLELIZATION EXPERIMENTAL RESULTS

The experimental results have been collected by using the computational grid with Globus middleware [21-22]. The computational grid consists in 4 dual-processor personal computers Compaq ML350T01 with Pentium III 933 MHZ processors, 128 Mb PC133 MHz RAM, integrated L2 cash 256 Kb, system bus clock rate 133 MHz, 9.1 Gb SCSI HDD, Fast Ethernet 100 Mbit/s network connection to 24-port 3Com Switch 100Mbit/s. Operation system of each computer is RedHat Linux 9 with Globus toolkit v.3.2.1.

Since the architecture of the computational grid is scalable by 2^n processors we trained the 50 IHDNNs on 2, 4 and 8 processors in order to avoid any eventual overhead due to the architectural characteristics of the system. The sequential and parallel routines have been developed by using C as programming language, standard MPI v.1.2 [23] and grid-enabled MPICH-G2 v.1.2.6 [24] as message passing libraries and MPE v.1.9.2. as performance visualization package.

The execution time of 50 IHDNNs parallelization according to enhanced integration historical data method is reported in Table 1 by using the basic dynamic mapping algorithm and in Table 2 by using the modified dynamic mapping algorithm. Speedup and efficiency of the basic mapping algorithm are depicted in Figs. 6-7. Speedup and efficiency of the modified mapping algorithm are depicted in Figs. 8-9. An efficiency comparison of both algorithms is reported in Table 3. As it is seen, the modified mapping algorithm, which uses IHDNN training time as mapping criterion, shows 5% better efficiency in average in comparison with the basic mapping algorithm and 13-16% better efficiency within two scenarios of IHDNNs training till $SSE=10^{-5}$ and $SSE=10^{-7}$, which had low efficiency in the basic mapping case.

Table 1. Training Time (Seconds) of 50 IHDNNs by Basic Dynamic Mapping Algorithm

CPU(s)	1	2	4	8
$SSE=10^{-3}$	12.39	6.20	3.10	1.66
$SSE=10^{-4}$	43.49	21.75	10.99	5.69
$SSE=10^{-5}$	177.09	88.83	68.28	57.46
$SSE=10^{-7}$	339.48	170.64	100.35	74.31

Table 2. Training Time (Seconds) of 50 IHDNNs by Modified Dynamic Mapping Algorithm

CPU(s)	1	2	4	8
$SSE=10^{-3}$	12.39	6.26	3.20	1.62
$SSE=10^{-4}$	43.49	22.20	10.99	5.62
$SSE=10^{-5}$	177.09	89.48	54.77	52.48
$SSE=10^{-7}$	339.48	172.15	86.48	69.91

Table 3. Efficiency Comparison Between Basic and Modified Dynamic Mapping Algorithms

Efficiency	2	4	8
$SSE=10^{-3}$	0.00	-0.02	0.03
$SSE=10^{-4}$	-0.01	0.00	0.02
$SSE=10^{-5}$	0.00	0.16	0.03
$SSE=10^{-7}$	0.00	0.13	0.04
Average:	0.05		

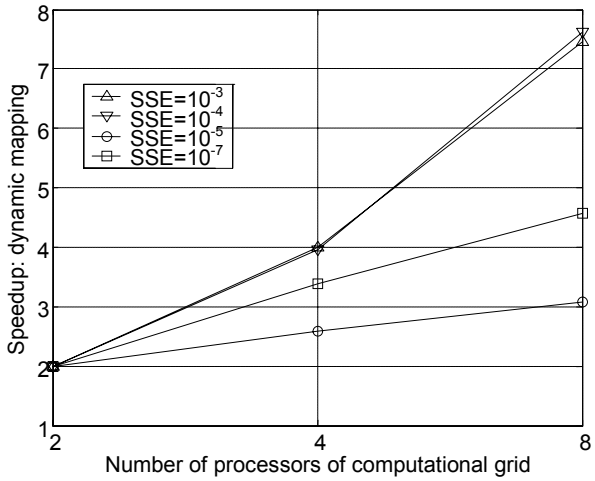

Fig. 6. Speedup vs number of processors using basic dynamic mapping algorithm.

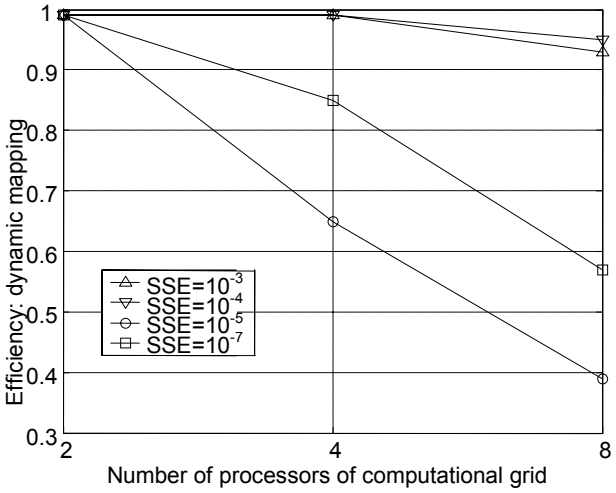

Fig. 7. Efficiency vs number of processors using basic dynamic mapping algorithm.

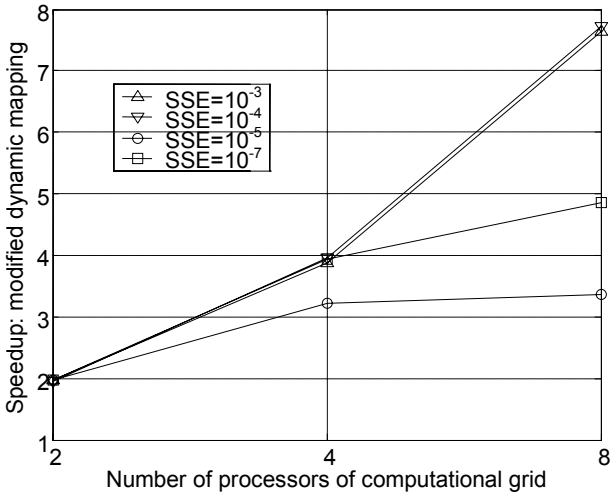

Fig. 8. Speedup vs number of processors using modified dynamic mapping algorithm.

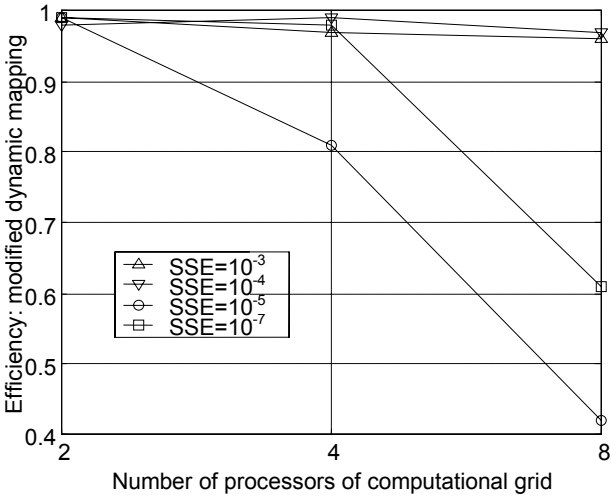

Fig. 9. Efficiency vs number of processors using modified dynamic mapping algorithm.

8. CONCLUSIONS

The enhanced integration historical data method using the set of Integrating Historical Data Neural Networks is described in this paper. The main idea of enhanced method is to account all available data on all historical curves of the sensor drift. Experiments showed that enhanced method has better accuracy in two times in comparison with basic integration historical data method. However the enhanced method requires considerable time and computational resources for its execution. The parallelization of the enhanced method has been done using coarse-grain approach with dynamic mapping of IHDNNs onto processors of parallel computer system. We have used a modified mapping strategy which uses the training time of the IHDNN as mapping criterion. Parallelization experiments done on computation grid under Globus middleware have showed that the modified mapping algorithm is more efficient on 5-16% in comparison with the basic mapping algorithm, which does not use any mapping criterion.

9. ACKNOWLEDGEMENTS

This work is fulfilled within INTAS Postdoctoral Fellowship for Young Scientists of the corresponding author Dr. Volodymyr O. Turchenko, grant reference number INTAS YSF 03-55-2493 “Development of Parallel Neural Networks Training Algorithms on Advanced High Performance Systems”. This support is gratefully acknowledged.

10. REFERENCES

[1]. A. Sachenko, V. Kochan, V. Turchenko, "Intelligent Distributed Sensor Network,” *Proceedings of 15th IEEE Instrumentation and Measurement Technology Conference IMTC/98*, St. Paul, USA, 1998, pp. 60-66.

[2]. A. Sachenko, V. Kochan, V. Turchenko, "Instrumentation for Data Gathering," *IEEE I&M Magazine*, vol. 6, no. 3, September 2003, pp. 34-40.

[3]. J. Brignell, "Digital compensation of sensors," *Scientific Instruments*, vol. 20, no 9, 1987, pp. 1097-1102.

[4]. V. Turchenko, "Neural network-based methods and means for improving the effectiveness of distributed sensor data acquisition and processing networks," *Ph.D. Thesis*, National University "Lvivska Politechnika," Lviv, p. 188, 2001 (in Ukrainian).

[5]. A. Alippi, A. Ferrero, V. Piuri, "Artificial Intelligence for Instruments & Applications," *IEEE I&M Magazine*, June 98, pp. 9-17.

[6]. P. Daponte, D. Grimaldi, "Artificial Neural Networks in Measurements," *Measurement*, vol. 23, 1998, pp. 93-115.

[7]. A. Sachenko, V. Kochan, V. Turchenko, V. Golovko, J. Savitsky, A. Dunets, T. Laopoulos, "Sensor Errors Prediction Using Neural Networks," *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks IJCNN'2000*, Como (Italy), vol. IV, 2000, pp. 441-446.

[8]. V. Golovko, A. Galushkin, *"Neural Networks: training, models and applications,"* Moscow, Radiotechnika, p. 256, 2001 (in Russian).

[9]. A. Sachenko, V. Kochan, V. Turchenko, "Sensor Drift Prediction Using Neural Networks," *Proceedings of the International Workshop on Virtual and Intelligent Measurement Systems VIMS'2000*, Annapolis, USA, 2000, pp. 88-92.

[10]. V. Turchenko, V. Kochan and A. Sachenko, "Advanced Method of Historical Data Integration Using Neural Networks," *Sensor and Systems*, Moscow, vol. 7 (38), 2002, pp. 35-38 (in Russian).

[11]. V. Turchenko, "Static Mapping of Integrating Historical Data Neural Networks on Parallel Computer," *Proceedings of the 16th IASTED International Conference on Parallel and Distributed Computing and Systems*, 2004, MIT, Cambridge, MA, USA, pp. 884-889.

[12]. V. Turchenko, "Parallel Algorithm of Dynamic Mapping of Integrating Historical Data Neural Networks," *Information Technologies and Systems*, vol. 7, no. 1, 2004, pp. 45-52.

[13]. Patent #50380 Ukraine, IPC 7 G06F15/18, "*Method of the training set formation for neural network predicting drift of data acquisition device,"* A.Sachenko (UA), V.Kochan (UA), V.Turchenko (UA), V.Golovko (BY), J.Savitsky (BY), T.Laopoulos (GR), filled 04 Jan 2000, issued 15 Nov 2002, p. 14.

[14]. A. Sachenko, V. Kochan, R. Kochan, V. Turchenko, K. Tsahouridis, Th. Laopoulos, "Error Compensation in an Intelligent Sensing Instrumentation System," *18th IEEE Instrumentation and Measurement Technology Conference IMTC/2001*, Budapest, Hungary, May 21-23, 2001, pp. 869-874.

[15]. S. Wang, "Reducing the communication cost in simulating layered neural networks on a hypercube machine," *Proceedings of Parallel Computing, Elsevier*, Amsterdam, 1989, pp. 375-380.

[16]. A. Petrowski, G. Dreyfus, C. Girault, "Performance analysis of a pipelined back-propagation parallel algorithm," *IEEE Transactions on Neural Networks*, vol. 4, 1993, pp. 970-981.

[17]. H. Paugam-Moisy, "Optimal speedup conditions for a parallel back-propagation algorithm," *Proceedings of CONPAR'92-VAPP V, Lecture Notes in Computer Science*, vol. 682, Springer-Verlag, 1992, pp. 719-724.

[18]. H. Hopp, L. Prechelt, "CuPit-2: A Portable parallel programming language for artificial neural networks," *Proceedings of the 15th IMACS World Congress Scientific Computation Modeling and Applied Mathematics*, vol. 6, Berlin, 1997, pp. 493-498.

[19]. Z. Hanzálek, "A parallel algorithm for gradient training of feed-forward neural networks," *Parallel Computing*, vol. 24, no. 5-6, 1998, pp. 823-839.

[20]. J.M.J. Murre, "Transputers and neural networks: An analysis of implementation constraints and performance," *IEEE Transactions on Neural Networks*, vol. 4, no. 2, 1993, pp. 284-292.

[21]. I. Foster, C. Kesselman, "Globus: a metacomputing infrastructure toolkit," *International Journal of Supercomputer Application*, vol. 11, no. 2, 1997, pp. 115-128.

[22]. Global Grid Forum webpage. http://www.gridforum.org/

[23]. J. Dongarra, D. Laforenza, S. Orlando (Eds), "Recent Advances in Parallel Virtual Machine and Message Passing Interface," *Lecture Notes in Computer Science*, Berlin: Springer-Verlag, 2003, vol. 2840, ISBN 3-540-20149-1.

[24]. N.T. Karonis, B. Toonen, I. Foster, "MPICH-G2: A Grid-enabled implementation of the Message Passing Interface," *Journal of Parallel and Distributed Computing*, vol. 63, no. 5, 2003, pp. 551-563.

Volodymyr Turchenko *was born in 1973, in Ternopil, Ukraine. He received his Engineer Diploma in systems engineering from Brest Polytechnic Institute, Brest, Belarus (1995) and his Ph.D. degree in computer engineering from National University "Lviv Polytechnics", Lviv, Ukraine (2001). Now he is Associate Professor of the Information Computing System and Control Department of the Faculty of Computer Information Technologies, Ternopil State Economic University (TSEU), Ternopil, Ukraine. He is a leader of Neural Network and Parallel Computing Research Team within the Research Institute of Intelligent Computer Systems of TSEU. He has published more than 60 scientific papers. His main research interests are neural networks, parallel computing and intelligent distributed sensor networks.*

Chefi Triki *has received his Engineer Degree in Electromechanical Engineering at "Ecole Nationale d'ingenieurs de Tunis", Tunisia (1993) and his Ph.D. in Engineering Systems and Computer Science from the University of Calabria, Italy (1998). Actually he is assistant professor of Operations Research at the University of Lecce, Italy. His major research interests lie in the field of optimization and simulation with application to logistics, energy and telecommunications. He has published and served as reviewer in a variety of international scientific journals.*

Lucio Grandinetti *is Full Professor at the Department of Electronics, Informatics and Systems of University of Calabria, Italy (since 1986) and scientific director of the Parallel Computing Laboratory (PARCOLAB) at the University of Calabria, Italy.*

He is director of the Center of Excellence on High Performance Computing established in the year 2000 by the Italian Ministry of University and Research at University of Calabria.

Lucio Grandinetti is co-director of the project SPACI (Southern Partnership for Advanced Computational Infrastructure) aimed to build a geographic GRID in Southern Italy (partners involved: University of Naples, University of Lecce, University of Calabria and Hewlett Packard) financed by the Italian Government.

He is member of the Partners Board of the main European Project on Grid, named EGEE

Prof. Grandinetti is co-director of NATO ARW on Software for Parallel Computation, Italy (1992) and NATO ARW on High Performance Computing, Italy (1996); member of several organising and scientific committees of international conferences on high-performance parallel computing (e.g. EUROPAR, HPCN Europe, PARCO) and member of the IEEE Technical Committee on Parallel Processing.

His areas of expertise are the design of numerical algorithms for parallel and distributed computer systems, modeling and simulation of large scale systems, numerical optimization methods for complex problems, software engineering aspects related to parallel processing, grid scheduling models.

He has been and currently is involved in research projects sponsored and financed by the National Research Council of Italy, by the European Commission, and by Italian Ministry of Research.

He has been evaluator and reviewer of European Commission Research Projects in the IT Programme ESPRIT, during 1993 and 1995. He has also been evaluator and reviewer of ESPRIT projects in the 4th framework programme. He has been evaluator and reviewer of research projects in the 5th Framework Programme (INFSO Programme) and is currently reviewer of a few projects in the 6th Framework Programme in the same field (Information Society).

Prof. Grandinetti is co-author of more than 60 papers in refereed journals, and co-editor of several books on numerical methods for non-linear optimization, computational engineering, parallel algorithms and software for vector and parallel computing, grid computing.

He is member of the editorial board of the following international journals:

- Parallel Computing (Elsevier)

- Optimization Methods and Software (Taylor and Francis)

- International Journal on Computing (Ukraine).

Lucio Grandinetti is co-Editor of the book series "Scientific and Engineering Computation" published by MIT Press, Boston (USA).

He has received several administrative and managing appointments at the University of Calabria, Italy (Department's Chairman, Member of Administration Council) and currently is Vice-Rector of the same University (since November 1st, 1999).

Anatoly Sachenko *is Professor and Head of the Department of Information Computing Systems and Control and Director of American-Ukrainian Program in Computer Science, Ternopil State Economic University.*

He earned his B.Eng. Degree in Electrical Engineering at L'viv Polytechnic Institute in 1968 and his PhD Degree in Electrical Engineering at L'viv Physics and Mechanics Institute in 1978 and his Doctor of Technical Sciences Degree in Electrical and Computer Engineering at Leningrad Electrotechnic Institute in 1988. Since 1991 he has been Honored Inventor of Ukraine, since 1993 he has been IEEE Senior Member.

His main Areas of Research Interest are Implementation of Artificial Neural Network, Distributed System and Network, Parallel Computing, Intelligent Controllers for Automated and Robotics Systems. He has published over 300 papers in areas above.