DOI https://doi.org/10.15407/usim.2019.03.038
UDC 519.163 + 681.5.015

V.S. STEPASHKO, Doctor Eng., Professor,
International Research and Training Center for Information Technologies and Systems,
of the NAS of Ukraine and of the MES of Ukraine, Acad. Glushkov ave., 40, Kyiv, 03187, Ukraine,
stepashko@irtc.org.ua

S.M. YEFIMENKO, PhD Eng., Senior Researcher,
International Research and Training Center for Information Technologies and Systems,
of the NAS of Ukraine and of the MES of Ukraine, Acad. Glushkov ave., 40, Kyiv, 03187, Ukraine,
syefim@ukr.net

A.V. PAVLOV, PhD Eng., Researcher,
International Research and Training Center for Information Technologies and Systems,
of the NAS of Ukraine and of the MES of Ukraine, Acad. Glushkov ave., 40, Kyiv, 03187, Ukraine,
andriypavlove@gmail.com

RECURRENT-AND-PARALLEL GMDH ALGORITHMS FOR HIGH-PERFORMANCE COMPUTING

The paper presents the conception, theoretical grounds and mathematical tools for designing high-performance searching and iterative GMDH algorithms on the basis of recurrent-and-parallel computing for modelling and prediction of complex processes. Its effectiveness is experimentally tested. Intelligent information technology for inductive modeling of complex processes on the basis of recurrent-and-parallel computing is constructed.

Keywords*: inductive modelling, recurrent-and-parallel computations, GMDH, COMBI, GRIA, vector autoregression.*

Inductive modelling problem

Inductive modeling is a process of mathematical model generation for an object, process or a system based on the empirical dataset. The modeling aims for forecasting, classification, extrapolation, interpolation and so on.

Generally, the model generation problem can be stated as follows. It is necessary to find the optimal model $\hat{y}_f^* = f^*(X, \hat{\Theta}_f)$ in a given set of models Ψ by minimization of a given criterion CR as the solution of the discrete optimization task:

$$f^* = \arg\min_{f \in \Psi} CR(y, f(X, \hat{\Theta}_f)), \qquad (1)$$

where parameters estimation vector $\hat{\Theta}_f$, $\dim \hat{\Theta}_f = s_f \times 1$, for each function $f(X, \Theta_f) \in \Psi$ is the solution of the following continuous optimization task:

$$\hat{\Theta}_f = \arg\min_{\Theta_f \in \Re^{s_f}} QR(y, f(X, \Theta_f)), \qquad (2)$$

where $QR(\cdot)$ is a criterion that estimates the quality of the solution for the parameter estimation problem; and $CR(\cdot)$ is a criterion that estimates the quality of the solution of the optimal model selection problem (1).

Group Method of Data Handling (GMDH) is one of the most effective methods for resolving

problems (1, 2). GMDH-based model generation process is based on the following principles: 1) successive complication of model structures; 2) "external supplement" [1]; and 3) non-final decisions [2].

The successive model structure complication principle is based on a self-organization concept of finding the optimal (adequate) model complexity for the modeling object in terms of minimization of the criterion *CR*. The model structure generation algorithm performs a set of consequent stages or iterations ($r = 1, 2, ...$). Each iteration generates a set of models (solutions) having more complex structures than the models at the previous iteration. The iterations are performed while criterion *CR* value is decreasing ($CR_r < CR_{r-1}$) or the difference $CR_r - CR_{r-1}$ approaches a given ε — the accuracy of the problem solution. It is assumed that if $CR_r > CR_{r-1}$ then the structure of the model at *r* iteration is overfitted and the iteration process should be stopped.

The *external supplement* principle is based on the Godel's incompleteness theorem [3]. According to that principle, the criteria should be utilized in (1) that are based on involving new information and have a minimum in the model complexity increasing process.

The principle of non-final decisions states that not the best model but several best models (solutions) are selected and passed to the next iteration of the algorithm. This allows increasing the probability to find the global minimum of the criterion *CR*.

The problem (1–2) in terms of GMDH may be specified as follows.

Let x_i, $i = \overline{1,m}$ are input variables or factors that are observed. First of all, we specify the set Ψ (model structure class), $\Psi = \{ f_k(X, \Theta_{f_k}) \}_{k=\overline{1,\infty}}$, which frequently is described by the following polynomial:

$$f(x_1, ..., x_m, \Theta) = \theta_0 + \sum_{i=1}^{m} \theta_i x_i + \sum_{i=1}^{m} \sum_{j=i}^{m} \theta_{ij} + \sum_{i=1}^{m} \sum_{j=i}^{m} \sum_{k=i+j}^{m} \theta_{ijk} x_i x_j x_k + \dots . \quad (3)$$

Assume that the polynomial (3) is limited by the maximal power *p*, then the number of terms in (3) is $l = C_{m+p}^{p}$. Let us denote variables after the coefficients θ_i, θ_{ij}, θ_{ijk}, ..., as z_j, and the corresponding vectors as z_j, $j = \overline{1,l}$, $Z = (z_1 \vdots z_2 \vdots ... \vdots z_l)$.

Let us give a definition of model structure in the form (3). Structural vector is an *l*-dimension binary vector $d_k^T = (d_{k1}, ..., d_{kl})$, $k = \overline{1,2^l}$ that is composed of s_k unities where $d_{kj} = 1$ in the vector manifests presence the corresponding variables z_j in the model structure (3), $s_k \le l$. The number s_k is called the complexity of the model (3). Consequently, the number of all structures of the model (3) and the power of the set Ψ is 2^l.

The vector of unknown parameters Θ_{d_k}, $\dim(\Theta_{d_k}) = s_k$, we define as a vector composed of non-zero components of the vector $\mathrm{diag}(d_k)\,\Theta$ where $\mathrm{diag}(v)$ is a diagonal matrix with elements of the vector *v*. Then a matrix Z_{d_k} is defined as a matrix composed of the vector-columns z_j of the matrix Z for which the corresponding elements of the vector d_k^T are equal to 1. Then the function $f_k(Z_{d_k}, \Theta_{d_k}) = Z_{d_k} \Theta_{d_k}$ with unknown parameters Θ_{d_k} is called a *model structure*.

Let *D* is a set that holds all possible structural vectors d_k, $k = \overline{1,2^l}$, and some unimodal criterion $QR(y, f(Z_{d_k}, \Theta_{d_k}))$ is given. Then the problem (2) becomes as follows:

$$\widehat{\Theta}_{d_k} = \arg\min_{\Theta_{d_k} \in \Re^{s_k}} QR(y, f(Z_{d_k}, \Theta_{d_k})) , \ \forall d_k \in D \quad (4)$$

and the problem (1) takes the form:

$$f^* = \arg\min_{d_k \in D} CR(y, f(Z_{d_k}, \widehat{\Theta}_{d_k})) . \quad (5)$$

In forecasting, classification and extrapolation problems being solved using GMDH in the class of polynomial functions (3) being linear in parameters, the *residual sum of squares* $RSS_A = || y_A - Z_{A,d_k} \widehat{\Theta}_{A,d_k} ||^2$ is used as the *QR* criterion, and the *regularity criterion* $AR_B \overset{\Delta}{=} AR_{B/A} = || y_B - Z_{B,d_k} \widehat{\Theta}_{A,d_k} ||^2$ is used as the *CR* criterion.

Indices *A* and *B* indicate *training* and *testing* datasets, $A \cap B = \varnothing$, $A \cup B = W$. *W* is the initial dataset of observations or a set of vector-columns of the matrix $(Z \vdots y)$.

All the variety of GMDH algorithms can be classified into *searching* and *iterative* algorithms [4] based on specifics of the model structure generation process, Fig. 1.

The problem (1) is similar to a discrete programming problem and can be resolved using exhaus-

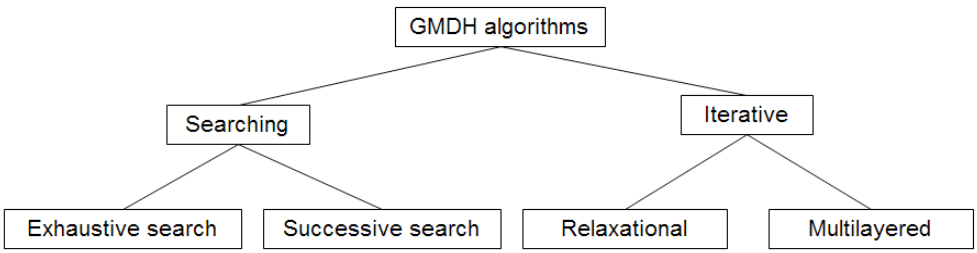

Fig. 1. GMDH algorithms classification

tive or successive search algorithms [4]. The basic searching GMDH algorithm is the combinatorial algorithm COMBI [5–8]. In order to speed up the algorithm, paper [6] proposes to utilize a recurrent procedure to estimate the parameters (bordering method [9]) and a special method for generation of model structures. This allows decreasing by a factor of ten the computational complexity of the parameters estimation procedure comparing to the Gaussian method [10].

The aim of successive search algorithms is to find the optimal solution obtained via exhaustive search algorithms. An example of such successive search algorithm is the algorithm MULTI [4] based on the principle of non-final decisions.

A number of iterative GMDH algorithms are well-known however the fastest among them is as for now the GRIA (Generalize Relaxational Iterative Algorithm) [10]. The computational complexity to build a model from iteration to iteration is linear in GRIA due to recurrent procedures for calculating the model parameters and *QR* and *CR* criteria [11].

The attention in this paper is paid to the development of parallelization methods for COMBI and GRIA and evaluation of their performance.

Combinatorial COMBI GMDH algorithm with recurrent computations

The combinatorial algorithm is designed to search for a better regression containing the most informative (effective) subset of input variables (regressors). It contains the following main blocks (which correspond to the main stages of the modelling process) [4]:

1. Data transformation according to the selected class of model structures linear in parameters.

2. Formation of models of different complexity (generation of all possible structures and parameters estimation by the least-squares method (LSM)).

3. Calculation of the value of external quality criteria and selection of the best models.

4. Estimation of the predictive quality of the best models on the third, examination (validation) sample (if it is given).

Since the same classes of structures and criteria can be used in any structural identification algorithm, their difference is determined by the type of structures generator. Therefore, we consider the principles of organizing calculations in a combinatorial generator, where the main computing costs are concentrated.

The basic operations performed in the block of models generation are: generating the next model structure (system of conditional equations); formation of a corresponding normal system of equations; solution of the received system (estimation the model coefficients).

In the case of a linear object with m inputs, the models of the form

$$\hat{y}_v = x_v\hat{\theta}_v,\ v = 1,\ldots, 2^m - 1, \qquad (6)$$

are compared in the process of the exhaustive search. Decimal number corresponds to binary number d_v in (6). Unit elements of d_v indicate inclusion regressors with corresponding numbers in the model, whereas zero elements signify exclusion. The parameter estimation problem is the most time-consuming in the combinatorial algorithm. Actually the problem is in computational speedup of linear equations systems solving. The bordering method is the traditional and effective recurrent method. The idea of this algorithm consists in step-by-step improving estimations of parameters using

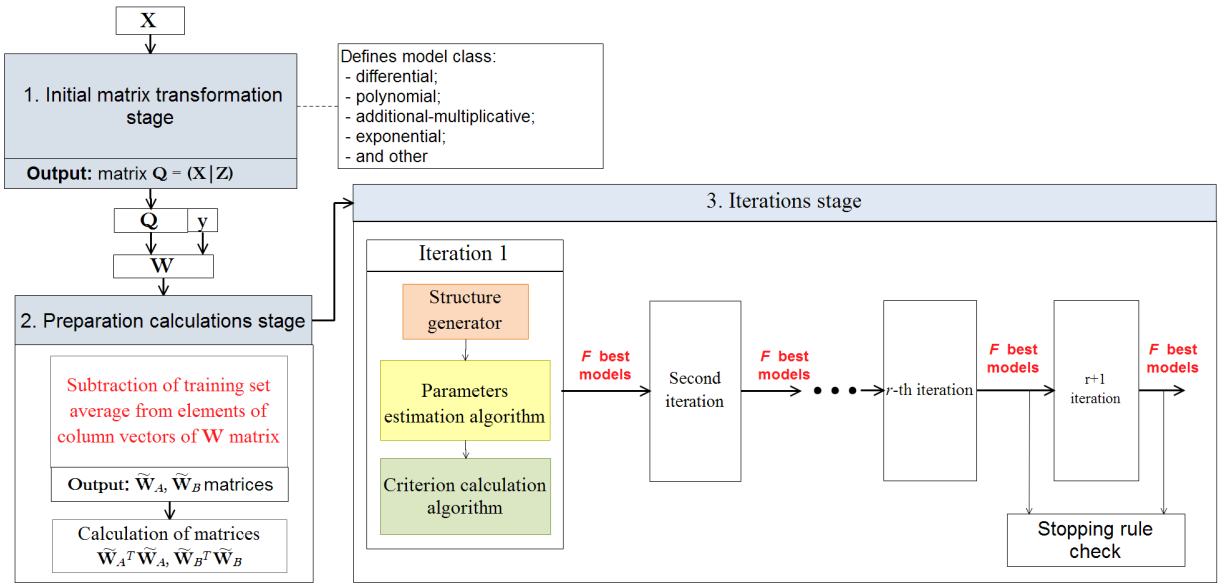

Fig. 2. Model generation process in GRIA

recurrent calculation of inverse matrix elements in an estimation procedure.

Efficient recurrent modifications of classic Gauss and Gramm-Schmidt algorithms were offered in [20].

Computational complexity of parameter estimation is proportional to the square of the model complexity for any recurrent algorithm and to the cube for non-recurrent one.

Generalized Relaxational Iterative GMDH Algorithm with recurrent computations

The model generation process in GRIA is shown in the figure 2 [10, 11].

As one can see from the figure above, the process consists of the three stages:

1. Stage of a transformation of the initial matrix *X*. The model class is generated here.
2. Preparation stage. The matrix of normal equations is generated here.
3. Iterations stage. Models are generated here.

In this algorithm, *F* selected models are passed from iteration to iteration. The algorithm includes two ways to find the solution of the problem (5):

- an exhaustive search in the given limited set of models at any iteration. It is performed based on a model structure generator with exhaustive search;
- successive model search being performed based on a model structure generator with a successive search; this generator reasonably truncates the exhaustive search in the case of a nonlinear model generation.

Vector autoregressive models for prediction of multidimensional interrelated processes

Vector autoregressive (VAR) model generalizes the autoregression model to multidimensional case [21]. It is built by the stationary time series. It is the system of equations in which every variable (component of multidimensional time series) is linear combination of all variables in the previous time points lags. The order of such model is determined by the order of the lags.

In the general case for *m* time series and *k* lags, the model will be the system of *m* equations and it matrix form will be of the form:

$$X(t) = \sum_{j=1}^{k} \Theta_j X(t-j), \qquad (7)$$

where Θ_j, $j = \overline{1,k}$ – matrices of model (7) parameters of the size $m \times m$.

The COMBI GMDH algorithm may be used for VAR modelling by exhaustive search of all possible variants and finding the best model for every time series containing the most informative subset of input arguments.

The general models structure in the form of the system of *m* difference equations is determined as a result of the sequence of such operations:

1. Data array of $m - k$ arguments is composed under the number of interrelated processes *m* and lags *k*.

2. Maximal complexity for restricted search is defined [22]. COMBI algorithm with sequentially complicated structures of models on the basis of recurrent-and-parallel computing is used for modelling. For every time series, the best *F* (by the value of the regularity criterion [23]) models are selected. Overall $F \cdot m$ models are passed to the next step.

3. The sorting-out of $G = F^m$ variants of model systems is carried out. The best system model (by the value of the systemic integral criterion of vector models quality) is selected. The value of the criterion is calculated on the given part of initial data set in the prediction mode of the process for the given steps number n_s:

$$B = \sum_{i=1}^{n_s} \sum_{j=1}^{m} (x_{ij} - x_{ij}^*)^2, \qquad (8)$$

where x_{ij}^* — result of step-by-step integration of the system of *m* equations.

The exponential dependence of number of system models *G* on the number of interrelated processes *m* results in a huge amount of variants enumeration and requires improvement of modelling means. Therefore, paralleling of computations and recurrent algorithms of parameters estimation are reasonable here.

Theoretical grounds for paralleling of computing in algorithms with recurrent parameters estimation

The problem of paralleling computations

The problem of parallelization of GMDH algorithms is not a new one. Many foreign and domestic scientists have been solvin git. Combinatorial algorithms were parallelized in the first place [12, 13] as they are the most time-consuming and suffer from "curse of dimensionality". Single-processor computational systems allow solving combinatorial search problem of 20 variables in 10 sec. while generating $2^{20} = 1048576$ models. Parallelization allows increasing the number of generated models proportionally to the number of processor cores, i.e., build $2^{26} = 67108864$ models in the same time using 64 cores.

The major problem in parallelization of combinatorial algorithms is approaching uniform load of all available cores, as the time to generate a model substantially depends on the number of variables (terms) included to the model. Paper [12] suggests using inverse structural vectors, in other words, a core computes a model whose structural vector is (1101110) and, in addition, computes its inverse version (0010001). This technique allows obtaining the cores load at 96% regardless of their number.

Paper [13] offers using a structural vector generation method with a consecutive complication. This allowed approaching almost maximal load of the cores on the level of 99,8%.

Methods and principles of parallelization of GMDH algorithms were considered in [14, 15]. Paper [14] proposes a parallelized version of GAME which is a GMDH-type neural network with different types of neurons and interlayer links based on symmetric multiprocessor (SMP) architecture that allows several cores to access to a single shared memory. Since each neuron is calculated independently and the time to create a thread is considerably less than the time to calculate a neuron, the algorithm computes each neuron in a single thread. However, the performance of such parallelization is turned out to be fairly low: if we get 1,7 speed up for 2 cores, then the algorithm performs only 3,5 times faster than its sequential version in the case of 8 cores.

Paper [15] considers three different type of speeding up:

1) parallelization tasks using threads,

2) vector parallel processing (vectorization) using an extended set of processor commands (*SSE* — Streaming *SIMD* Extensions),

3) using the 64-bit operating system.

The author suggests performing low-level computations (simple loops) using vectorization and more complex procedures (initialization functions, data manipulations, reading/writing data) in parallel in separated threads. The effectiveness of the proposed principles is turned out to be fairly high and has almost linear speed up function: 2 cores — 2x speed up, 8 cores — 7,4x speed up.

Libraries for parallelization

The most well-known libraries are: Message Passing Interface (MPI), Open Multi-Processing (OpenMP) and Threading Building Blocks (TBB).

MPI is an API developed for message transferring thatallow exchange messages between processes which perform one task. Primarily the interface designed for parallelizing of tasks between CPUs but not threads, therefore creating, deleting, synchronizing and message transfer between threads shoulder a programmer. That is the reason the interface is a quite low-level and complex, and isn't easy to master in a short term.

OpenMP is an open standard for program parallelization describing a set of compiler directives, library procedures and environment variables which purposed for creating multithread applications based on shared memory multiprocessor systems. The standard is implemented in the majority of well-known compilers (GCC, Microsoft Visual Studio, Intel, IBM, Oracle) and allows writing parallel applications easily just by parallelizing local code segments adding minimum changes to the original code.

TBB is a C++ template library for parallelism undertaking thread management that allow (as OpenMP) directly specify the section of code which should be parallelized. A programmer should think in terms of tasks, not threads when applying the library.

The advantages of TBB over OpenMP are:

- any data type support;
- ready to use class templates, allowing memory access from several threads simultaneously;
- automatic detection and creation the optimal number of working threads in runtime.

The decision about using the TBB library for parallelization the GRIA was made, taking into account the advantages of the TBB mentioned above.

Parallelization of recurrent computations in iterative GMDH algorithms

Paper [16] offers a parallelized version of GRIA with recurrent computations based on SMP architecture [17] considered in [14].

The model generation process in GRIA consists of three stages:

1) transformation of the initial matrix,
2) calculation of the normal equations matrix,
3) iterations stage where models are built.

When building models, it is necessary to calculate standard error measures at different parts of the dataset. If the number of models and observations is big enough, this post-process stage could be time-consuming. Taking into account that the first stage does not take long, we should parallelize the last two stages of the algorithm and the post-process stage where error measures are calculated.

The preparation stage calculates matrices $\mathbf{X}_A^T\mathbf{X}_A$, $\mathbf{X}_B^T\mathbf{X}_B$, $\mathbf{X}_A^T\mathbf{y}_A$, $\mathbf{X}_B^T\mathbf{y}_B$ by implementing a double loop. To approach the uniform load of cores, it is offered [16] to create tasks in the following way. Each task should execute the body of the internal loop for two values of the variable i of the external loop that are equidistant from $n/2$ (for example, $i = 0$ and $i = n - 1$; $i = 1$ and $i = n - 2$;...), where n is the initial dataset length. Then, having k cores (n is even), each of them executes:

$$N = \begin{cases} N_1 = n/(2k)\ tasks,\ if\ N_1 - int \\ first\ \lfloor n/(2k) \rfloor\ cores\ execute\ \lceil n/(2k) \rceil + 1, \\ \quad the\ rest\ -\lceil n/(2k) \rceil \end{cases}$$

where $\lfloor\ \rfloor$ denotes a remainder, $\lceil\ \rceil$ denotes integer result of the division.

Model generation process in GRIA is the iterative one [10]. Each model can be calculated independently of others at any iteration. The algorithm passes a given number F of best models from iteration to iteration. It is calculated m new models at the current iteration based on every model that was passed from the previous iteration. Thus, the total number of tasks (models that must be built) at every iteration of the algorithm is Fm. Articles [18, 26]

suggest the following scheme for the tasks parallelization (k is the number of cores):

1. $Fm < k$. Fm cores perform one task, the rest ($k - Fm$) are free.

2. $Fm = k$. All the cores perform one task.

3. $Fm > k$. The first $\lfloor Fm / k \rfloor$ cores perform $\lceil Fm / k \rceil + 1$ tasks and the rest $\lceil Fm / k \rceil$ tasks.

It is also offered to have the threads number equal to the number of cores in order to avoid delays due to switching between the threads.

Parallelization of recurrent computations in COMBI GMDH algorithm

The scheme of the combinatorial algorithm parallelization with the *standard binary generator* of structural vectors and the recurrent parameters estimation using the modified Gauss algorithm for solving linear equation systems developed in [24]. In this scheme, the change of states of binary structural vector with elements 0 or 1 is organized on the basis of the binary counter.

Table 1 shows the approximate modeling time using this scheme. Already for more than 50 arguments, an exhaustive search (in acceptable modeling time) becomes impossible even for cluster system containing one hundred processors.

The scheme with sequential binary counter uses such sequence of binary numbers generation when all combinations with one unit in structural vector appears first of all, then with two units, and so on to complete model comprising all arguments.

Table 1. **Approximate time of the exhaustive search**

Arguments	Models	Time	
		1 proc.	100 proc.
20	1 048 575	1 s	0,01 s
21	2 097 151	2 s	0,02 s
...			
40	1,1E+12	~ 12 days	~ 3 hours
...			
50	1,1E+15	~ 34 days	~ 124 hours

Table 2. **Approximate time of the restricted search**

Arguments	Complexity	Models	Time, hours	
			1 proc.	100 proc.
50	15	3,7E+12	984	~ 10
100	9	2,1E+12	558	~ 6

This scheme allows to partially solve the problem of exhaustive search when arguments number exceeds capability of the algorithm with a standard binary generator. In this case it is advisable to execute an exhaustive search not among all possible models but only for models of the restricted complexity.

Table 2 shows approximate modeling time of COMBI with successive complication of model structures of complexity no more than 15 out of the total 50 arguments, when all models with 50-elements binary structural vectors containing from 1 to 15 units are built [22].

Paralleling the recurrent computations in VAR modeling

To build models of vector autoregression, the combinatorial COMBI GMDH algorithm (with sequential complication of model structures), recurrent parameters estimation (with the use of the modified Gauss algorithm) and computations paralleling is used. Linear models are compared for each of the m interrelated processes with $g = m \cdot k$ arguments (inputs).

The scheme of algorithm with sequential complication of structures for building VAR models is chosen because it allows partially solving the problem of exhaustive search in the case when it becomes impossible even with parallelization. In this case, it is advisable to perform exhaustive search not among all possible models, but only models of limited complexity.

Another reason for applying the restriction on complexity may be the insufficient number of points in the training part of the sample n_A when the search is performed among the complexity models no more than n_A. Since the modeling time is actually determined by the time of the search of variants for the systems of models, the efficiency of parallelization of this stage of the algorithm will be even more important than the efficiency of paralleling the stage of constructing the F best models for each of the m outputs (time series).

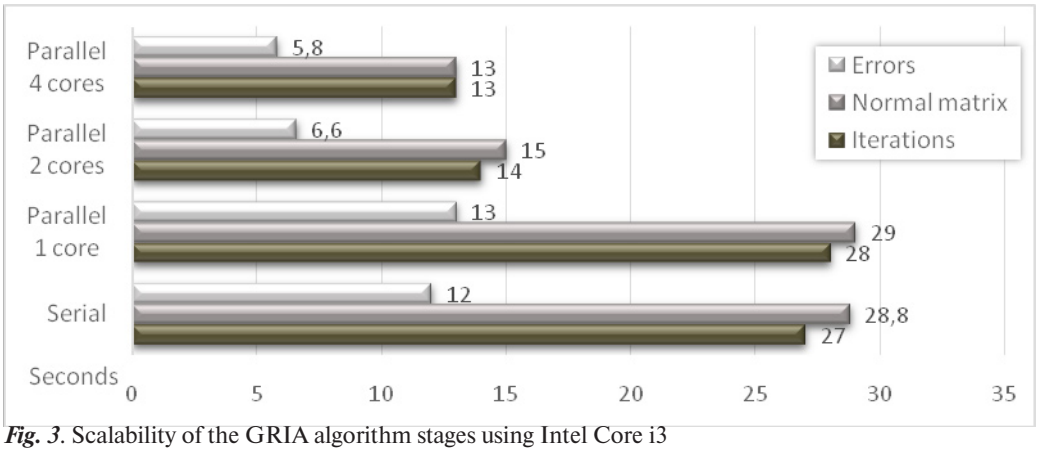

Fig. 3. Scalability of the GRIA algorithm stages using Intel Core i3

The investigation of the paralleling effectiveness

The effectiveness of the constructed modeling tools was calculated according to the following criteria:

- the acceleration of modeling time;
- the percentage of the computers load.

Performance evaluation of the parallelized GRIA algorithm

Evaluation of the performance of the parallelized GRIA was carried out using two different processors [16]:

- 4-core processor Intel Core i3 M 350 2.27 GHz (2 physical and 2 logical cores);
- 8-core processor Intel Core i7 4700 HQ 2.4 GHz (4 physical and 4 logical cores).

The initial matrix held 4000 observations, 10 true variables and 990 false variables. The dataset was generated using a pseudo-random generator Mersenne twister [19]. The matrix values were generated using the uniform distribution law in the interval [0; 1]. The initial dataset was divided into an examination (500 records), learning (3000 records), and validation (500 records) sets. The values for the algorithm's parameters were the following: choice of freedom $F = 400$, the iteration number $R = 50$.The algorithm should to discovere the following model:

$$y = 6{,}29447 + 9{,}37736x_1 + 8{,}26752x_2 - -8{,}04919x_{3+} + 8{,}11584x_4 - 7{,}46026x_5 - 7{,}29046x_6 + +6{,}70017x_7 - 5{,}57932x_8 - 3{,}83666x_9 + 2{,}64719x_{10}.$$

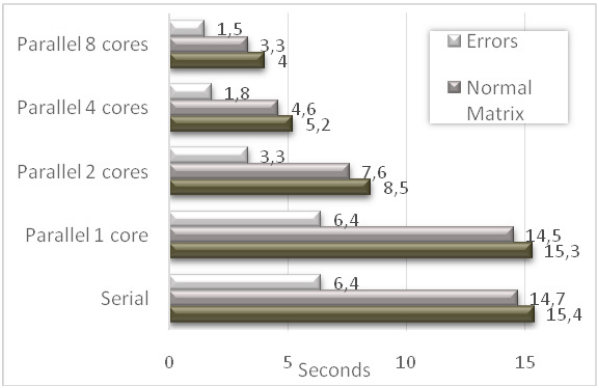

Fig. 4. Scalability of the algorithm stages using Intel Core i7

The algorithm has built the following model:

$$\hat{y} = 6{,}29447 + 9{,}37736x_1 + 8{,}26752x_2 - -8{,}04919x_3 + 8{,}11584x_4 - 7{,}46026x_5 - 7{,}29046x_6 + +6{,}70017x_7 - 5{,}57932x_8 - 3{,}83666x_9 + 2{,}64719x_{10} + 3{,}5\ 10^{-8} \cdot x_{145} + 1.4\ 10^{-9} \cdot x_{258}.$$

The figures 3 and 4 show the scalability of the three mentioned above stages of the algorithm that were parallelized. The legend description of the figures: Errors — the post-process stage (errors measure are calculated), Normal matrix — the preparation stage, Iterations — the iterations stage.

As one can see from the figures, the parallelization allows obtaining the almost equal time of the serial and the parallel versions using one core, hence usage of physical cores allow speeding up every stage in 2x using 2 cores of Core i3 and in 1,86x using 2 cores of Core i7. Scalability of the stages

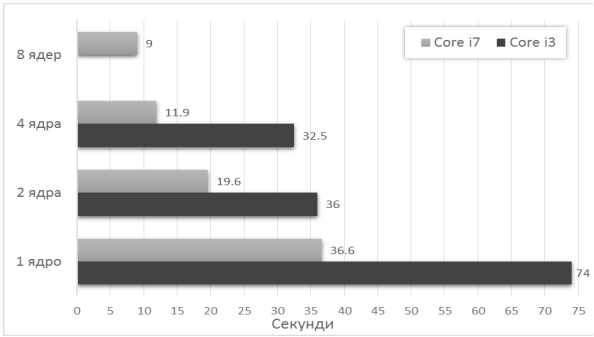

Fig. 5. Performance of parallelized GRIA

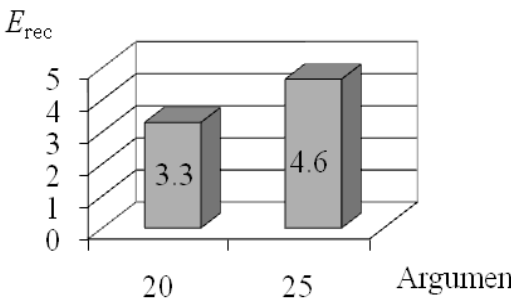

Fig. 6. Comparative effectiveness

differ when using 4 cores of Core i7: the iterations stage has been speeding up in 2,9x, the normal matrix calculation stage — in 3,15x and the errors calculation stage — in 3,55x.

As it may be seen, we have a low scalability when involving logical cores into computations: usage of 4 processor cores Core i3 gives us only 2,3x speed up and usage of 8 cores Core i7 — 3.8x at the iteration stage, 4,39x at the normal matrix calculation stage, and 4,26x at the error measures calculation stage.

Let's look at the scalability of parallelized GRIA (see figure 5). The performance of the developed application is represented in table 3.

As one can see from the table, the parallelized version allowed obtaining 2,28x speed up at Core i3 processor and 4x speed up using Core i7 and approaching uniform load of processor cores that one can see from the last two columns of the table.

Performance evaluation of the parallelized COMBI algorithm

The experiment was carried out at the SCIT IC complex of the National Academy of Sciences of Ukraine [25] with the use of CPU IntelXeonE5-2600 2.6 GHz.

The test task was formed as follows: the matrix X of size 70×50 (70 records for 50 arguments) for the system of conditional equations was generated. The vector y was formed in the form of a linear combination of only five arguments:

$$y = x_{10} + x_{20} + x_{30} + x_{40} + x_{50} \qquad (9)$$

The time of exhaustive search with a standard binary generator based on recurrent computations for 50 arguments would last about 34 years. However, this task can be solved by using an algorithm with sequential complication of structures limited to models of complexity 7. In this case 6 nodes with 24 cores of SCIT-4 computing cluster were used for modeling.

Model (9) was received by such computing system in less than 2 seconds when 118145035 models

Table 3. **The performance of the developed application**

Processor	Number of cores	Performance		
		Speed up	Physical cores load, %	Logical cores load, %
Logical cores load, %	1	1x	100	Not used
	2	2x	90	
	4	2,28x	90	99
Core i7	1	1x	100	Not used
	2	1,86x	100	
	4	3x	99	
	8	4x	99	99

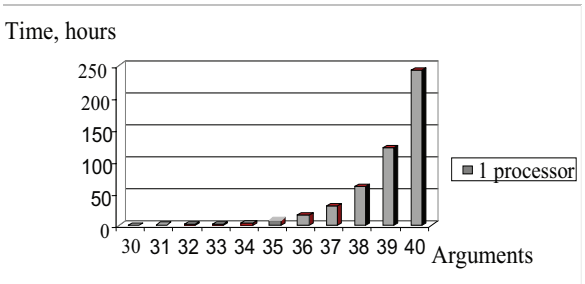

Fig. 7. Run-time estimation

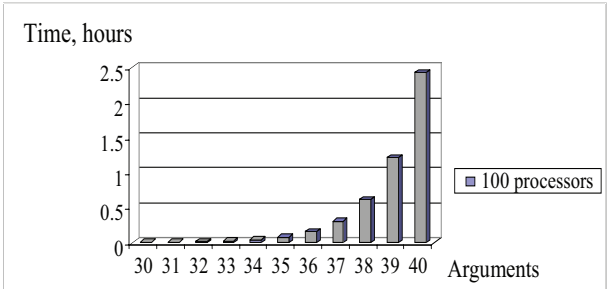

Fig. 8. Run-time estimation

were built (structures were generated, parameters were estimated, and the quality criteria were calculated) among which the best one according to the regularity criterion was chosen.

The purpose of the next test experiment was to compute the comparative effectiveness E_{rec} of the parallel algorithm with recurrent parameters estimation versus the parallel algorithm with non-recurrent computations. The effectiveness was determined as the ratio of the time of execution of the corresponding programs (for the number of arguments, equal to 20 and 25).

The result of the experiment, presented in figure 6, shows an increase in the value of E_{rec} with an increase in the number of arguments.

Figures 7 and 8 give estimates of the run-time of the combinatorial algorithm with recurrent computations on single and 100 processors. With the increase in the arguments number, the efficiency of recurrent-and-parallel computations increases, and for 40 arguments the gain reaches by a factor of a hundred.

Construction of intelligent information technology for inductive modeling of complex processes on the basis of recurrent-and-parallel computations

Taking into account mentioned features for GMDH algorithms as well as schemes of parallel computations, it is possible to propose intelligent information technology of inductive modeling on the basis of recurrent-and-parallel computations.

The technology in automatic mode takes into account number of arguments, number of available

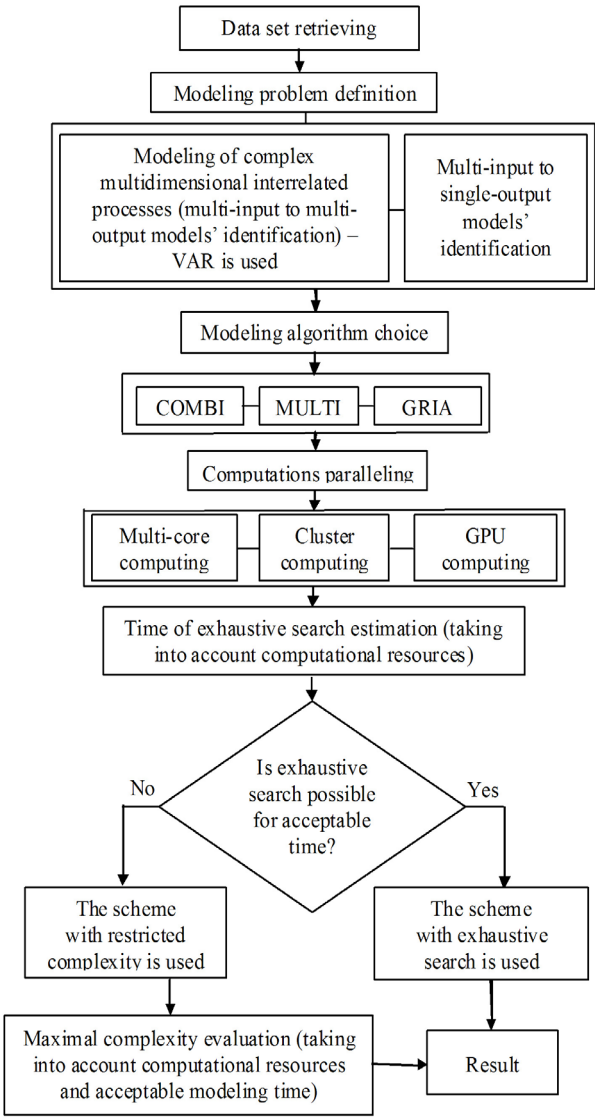

Fig. 9. Flow chart of the intelligent information technology

computational resources and running time restriction. Figure 9 represents a block-diagram of information technology suggested in [27].

Conclusion

The principle of high-performance parallel computations in the problems of inductive modelling on the basis of searching and iterative GMDH algorithms with recurrent parameters estimation is developed.

A project of the intelligent information technology for inductive modeling of complex processes on the basis of recurrent-and-parallel computations is presented.

REFERENCES

1. Beer, S., 1964. Cybernetics and Management. John Wiley & Sons, Inc., 214 p.
2. Gabor, D., 1971. "Cybernetics and the future of industrial civilization". J. Cybern., 1971, 2 (1), pp. 1–4.
3. Nagel, E., Newman J. R., 1989. Gedel's Proof. Routledge, 118 p.
4. Ivakhnenko, A.G., Stepashko, V.S., 1985. Noise-immunity of modeling. Kiev: Naukova dumka, 216 p. (In Russian).
5. Stepashko, V.S., 1979. Optimization and Generalization of Model Sorting Schemes in Algorithms for the Group Method of Data Handling. Soviet Automatic Control. 12(4), pp. 28–33.
6. Stepashko, V.S., 1981. "A Combinatorial Algorithm of the Group Method of Data Handling with Optimal Model Scanning Scheme". Soviet Automatic Control, 14(3), pp. 24–28.
7. Stepashko, V.S., 1983. "Potential noise stability of modelling using the combinatorial GMDH algorithm without information regarding the noise". Soviet Automatic Control, 16(3), pp. 15–25.
8. Stepashko, V.S., 1983. "A Finite Selection Procedure for Pruning an Exhaustive Search of Models". Soviet Automatic Control, 16(4), pp. 88–93.
9. Faddeev, D.K., Faddeeva, V.N., 1963. Computational methods of linear algebra, 2nd ed. M.: Nauka, 656 p. (In Russian).
10. Pavlov, A.V., 2011. Generalized relaxation iterative algorithm of GMDH. Inductive modeling of complex systems. Coll. of science works, Issue 2. K.: IRTC, pp. 95-108. (In Russian).
11. Pavlov, A.V., 2013. "Generalized relaxational iterative GMDH algorithm of GMDH and its analysis". Proc. of the Int. Conf. on Inductive Modelling ICIM-2013, pp. 89–96.
12. Koshulko, O.A., Koshulko, A.I., 2007. "Adaptive parallel implementation of the combinatorial GMDH algorithm". Proc. of the International workshop on inductive modelling IWIM-2007. Prague: CTU, pp. 71–77.
13. Stepashko, V., Yefimenko, S., 2008. "Optimal Paralleling for Solving Combinatorial Modelling Problems". Proc. of the 2nd International Conference on Inductive Modeling ICIM-2008. Kyiv, pp. 172–175.
14. Kordik, P., Spirk, J., Simecek, I., 2008. "Parallel computing of GAME models". Proc. of the 2nd Int. conf. on inductive modeling ICIM-2008. Kyiv, pp. 160–163.
15. Lemke, F., 2008. "Parallel Self-Organizing Modeling". Proc. of the II Int. Conf. on Inductive Modelling ICIM-2008, 15–19 Sept. 2008, Kyiv, Ukraine. Kyiv: IRTC ITS NASU, pp. 176–183.
16. Pavlov, A., 2014. "Parallel Relaxational Iterative Algorithm of GMDH". Inductive modelling of complex systems. Coll. of science works, Issue 6. K.: IRTC, pp. 33–40. (In Russian).
17. Message Passing Interface, [online]. Available at: <http://en.wikipedia.org/wiki/Message_Passing_Interface> [Accessed 27 Dec. 2018].
18. Pavlov, A.V., 2013. "Principles of parallel computations of relaxational iterative GMDH algorithm". Inductive modeling of complex systems. of science works, Issue 5. K.: IRTC, pp. 220–225. (In Russian).
19. Mersenne twister, [online]. Available at: <http://en.wikipedia.org/wiki/Mersenne_twister> [Accessed 20 Dec. 2018].
20. Stepashko, V.S., Efimenko, S.M., 2005. "Sequential Estimation of the Parameters of Regression Models". Cybernetics and Systems Analysis, 41(4), pp.631–63.
21. Lutkepohl, H., 1993. "Introduction to multiple time series analysis". Springer-Verlag Berlin Heidelberg, 545 p.
22. Yefimenko, S., Stepashko, V., 2015. "Intelligent Recurrent-and-Parallel Computing for Solving Inductive Modeling Problems". 16th Int. Conf. on Computational Problems of Electrical Engineering (CPEE), Lviv, pp. 236–238.
23. Madala, H.R., Ivakhnenko, A.G., 1994. Inductive Learning Algorithms for Complex Systems Modeling. London, Tokyo: CRC Press Inc., 384 p.

24. Yefimenko, S., 2013. "Comparative Effectiveness of Parallel and Recurrent Calculations in Combinatorial Algorithms of Inductive Modelling". Proceedings of the 4th International Conference on Inductive Modelling ICIM'2013, Kyiv, pp. 231–234.
25. Supercomputer of IC, [online]. Available at: <http://icybcluster.org.ua/index.php?lang_id=3&menu_id=1> [Accessed 07 Dec. 2018].
26. Pavlov, A.V., Stepashko, V.S., 2011. "Recurrent algorithms for calculating coefficients and selection criteria in the GMDH relaxational algorithm". Cybernetics and computing engineering, 165, pp. 72–82. (In Russian).
27. Yefimenko, S., 2018. "Construction of Intelligent Information Technology for Inductive Modeling of Complex Processes on the Basis of Recurrent-and-Parallel Computations". Proc. of the XIII IEEE Int. Conf. CSIT-2018& International Workshop on Inductive Modeling, Sept. 11–14, 2018, Lviv, Ukraine. Lviv: Publisher "Vezha&Co", pp. 440–443.

Received 29.03.2019

В.С. Степашко, д-р техн. наук, професор, старший науковий співробітник,
Міжнародний науко-навчальний центр інформаційних технологій та систем
НАН та МОН України, просп. Академіка Глушкова, 40, Київ, 03187, Україна,
stepashko@irtc.org.ua

С.М. Ефіменко, канд.техн. наук, старший науковий співробітник,
Міжнародний науко-навчальний центр інформаційних технологій та систем
НАН та МОН України, просп. Академіка Глушкова, 40, Київ, 03187, Україна,
syefim@ukr.net

А.В. Павлов, канд. техн. наук, науковий співробітник,
Міжнародний науко-навчальний центр інформаційних технологій та систем
НАН та МОН України, просп. Академіка Глушкова, 40, Київ, 03187, Україна,
andriypavlove@gmail.com

РЕКУРЕНТНО-ПАРАЛЕЛЬНІ АЛГОРИТМИ МГУА ДЛЯ ВИСОКОПРОДУКТИВНИХ ОБЧИСЛЕНЬ

Вступ. Індуктивне моделювання є процесом побудови математичних моделей об'єктів, процесів та систем на основі статистичних даних. Метод групового урахування аргументів (МГУА) є одним з найбільш ефективних методів обчислювального інтелекту. Процес побудови моделей на основі МГУА базується на принципах послідовного ускладнення структур моделей, «зовнішнього доповнення» та неостаточних рішень. Все різноманіття алгоритмів МГУА, виходячи з особливостей процесу генерації структур моделей, можна розділити на перебірні та ітераційні алгоритми.

Мета цієї статті полягає у розробленні методів розпаралелювання обчислень у перебірному алгоритмі *COMBI* та узагальненому релаксаційному ітераційному алгоритмі *GRIA* і визначенні обчислювальної ефективності розпаралелювання.

Результати. У статті описано розроблені принципи розпаралелювання операцій у комбінаторному алгоритмі *COMBI* МГУА з рекурентним оцінюванням параметрів моделей. При розпаралелюванні використано схеми обчислень зі стандартним генератором двійкових чисел та з послідовним ускладненням структур моделей, згідно з якими кожен процесор автономно обчислює початковий двійковий структурний вектор та кількість моделей, які він будуватиме. Також гарантується неповторюваність структур у різних процесорах. Завдяки цьому значно підвищується ефективність розпаралелювання, оскільки немає втрат часу на міжпроцесорну взаємодію.

Схема розпаралелювання з послідовним ускладненням структур моделей дозволяє частково розв'язувати задачу повного перебору у випадку, коли кількість аргументів для перебору перевищує можливості алгоритму зі стандартним двійковим генератором, і повний перебір доцільно виконувати не серед усіх можливих моделей, а лише моделей обмеженої складності.

Описано принцип розпаралелювання обчислень в комбінаторному алгоритмі *COMBI* МГУА з рекурентним оцінюванням параметрів моделей для побудови дискретних прогнозних моделей динаміки складних багатовимірних взаємозв'язаних процесів. Описано принципи та схеми розпаралелювання обчислень в узагаль

неному релаксаційному ітераційному алгоритмі МГУА *GRIA*, які дозволяють збільшити швидкість роботи алгоритму пропорційно кількості обчислювачів при максимальному (майже рівномірному) їх завантаженні.

Виконано дослідження ефективності різних схем розпаралелювання обчислень в алгоритмі *COMBI* та *GRIA* за допомогою обчислювальних експериментів на персональному комп'ютері та кластерному багатопроцесорному комплексі.

Як свідчать експерименти з тестування ефективності розпаралелювання алгоритму *GRIA*, швидкість роботи розроблених програмних засобів збільшується лінійно із додаванням нового обчислювального елемента (процесора чи ядра процесора).

Розроблені схеми дозволяють істотно підвищити ефективність алгоритмів МГУА шляхом виконання рекурентно-паралельних обчислень.

Враховуючи особливості алгоритмів МГУА та схеми паралельних обчислень, розроблено концепцію інтелектуальної інформаційної технології індуктивного моделювання на основі рекурентно-паралельних обчислень. Така технологія при побудові моделей в автоматичному режимі враховує кількість аргументів, кількість доступних обчислювальних ресурсів і встановлені користувачем обмеження на час моделювання.

Висновки. Розроблено технологію високопродуктивних паралельних обчислень в задачах індуктивного моделювання на основі перебірних та ітераційних алгоритмів МГУА з рекурентним оцінюванням параметрів моделей. Запропоновано концепцію інтелектуальної інформаційної технології індуктивного моделювання складних процесів на основі рекурентно-паралельних обчислень.

***Ключові слова*:** *індуктивне моделювання, рекурентно-паралельні обчислення, МГУА, COMBI, GRIA, векторна авторегресія.*

В.С. Степашко, д-р техн. наук, профессор, старший научный сотрудник,
Международный научно-учебный центр информационных технологий и систем
НАН и МОН Украины, просп. Академика Глушкова, 40, Киев, 03187, Украина,
stepashko@irtc.org.ua

С.Н. Ефименко, канд. техн. наук, старший научный сотрудник,
Международный научно-учебный центр информационных технологий и систем
НАН и МОН Украины, просп. Академика Глушкова, 40, Киев, 03187, Украина,
syefim@ukr.net

А.В. Павлов, канд. техн. наук, научный сотрудник,
Международный научно-учебный центр информационных технологий и систем
НАН и МОН Украины, просп. Академика Глушкова, 40, Киев, 03187, Украина,
andriypavlove@gmail.com

РЕКУРРЕНТНО-ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ МГУА ДЛЯ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛЕНИЙ

Введение. Индуктивное моделирование — это процесс построения математических моделей объектов, процессов и систем на основе статистических данных. Метод группового учета аргументов (МГУА) — один из наиболее эффективных методов вычислительного интеллекта. Процесс построения моделей на основе МГУА базируется на принципах последовательного усложнения структур моделей, «внешнего дополнения» и неокончательных решений. Все многообразие алгоритмов МГУА, исходя из особенностей процесса генерации структур моделей, можно разделить на переборные и итерационные алгоритмы.

Цель этой статьи состоит в разработке методов распараллеливания вычислений в переборном алгоритме *COMBI* и обобщенном релаксационном итерационном алгоритме *GRIA* и определении вычислительной эффективности распараллеливания.

Результаты. В статье описаны разработанные принципы распараллеливания операций в комбинаторном алгоритме *COMBI* МГУА с рекуррентным оцениванием параметров моделей. При распараллеливании использованы схемы вычислений со стандартным генератором двоичных чисел и последовательным усложнением структур моделей, согласно которым каждый процессор автономно вычисляет начальный двоичный структурный вектор и количество моделей, которые он будет строить. Также гарантируется неповторяемость структур в различных про-

цессорах. Благодаря этому значительно повышается эффективность распараллеливания, поскольку нет потерь времени на межпроцессорное взаимодействие.

Схема распараллеливания с последовательным усложнением структур моделей позволяет частично решать задачу полного перебора в случае, когда количество аргументов для перебора превышает возможности алгоритма со стандартным двоичным генератором, и полный перебор целесообразно выполнять не среди всех возможных моделей, а только среди моделей ограниченной сложности.

Описан принцип распараллеливания вычислений в комбинаторном алгоритме *COMBI* МГУА с рекуррентным оцениванием параметров моделей для построения дискретных прогнозных моделей динамики сложных многомерных взаимосвязанных процессов. Описаны принципы и схемы распараллеливания вычислений обобщенного релаксационного итерационного алгоритма МГУА *GRIA*, которые позволяют увеличить скорость работы алгоритма пропорционально количеству вычислителей при максимальной (почти равномерной) их загрузке.

Выполнены исследования эффективности различных схем распараллеливания вычислений в алгоритме *COMBI* и релаксационном итерационном алгоритме с помощью вычислительных экспериментов на персональном компьютере и кластерном многопроцессорном комплексе.

Как свидетельствуют результаты экспериментов по тестированию эффективности распараллеливания алгоритма *GRIA*, скорость работы разработанных программных средств увеличивается линейно с добавлением нового вычислительного элемента (процессора или ядра процессора).

Разработанные схемы позволяют существенно повысить эффективность алгоритмов МГУА путем выполнения рекуррентно-параллельных вычислений.

Учитывая особенности алгоритмов МГУА и схемы параллельных вычислений, разработана концепция интеллектуальной информационной технологии индуктивного моделирования на основе рекуррентно-параллельных вычислений. Такая технология при построении моделей в автоматическом режиме учитывает количество аргументов, количество доступных вычислительных ресурсов и установленные пользователем ограничения на время моделирования.

Выводы. Разработана технология высокопроизводительных параллельных вычислений в задачах индуктивного моделирования на основе переборных и итерационных алгоритмов МГУА с рекуррентным оцениванием параметров моделей. Предложена концепция интеллектуальной информационной технологии индуктивного моделирования сложных процессов на основе рекуррентно-параллельных вычислений.

***Ключевые слова*:** *индуктивное моделирование, рекуррентно-параллельные вычисления, МГУА, COMBI, GRIA, векторная авторегрессия.*