The Problem of Software Implementation Iterative Algorithms of Inductive Modeling

Oleksandra Bulgakova [1,2], Viacheslav Zosimov [1,2]

[1] *Department of Applied information systems, Taras Shevchenko National University of Kyiv, Kyiv, Ukraine*
[2] *Department of computer science V.O.Sukhomlynsky National University of Mykolaiv*

Abstract

The article deals the software implementation of modeling algorithms, using the example of the generalized iterative algorithm of group method of data handling. Software implementation problems are described, which may cause the risk of data loss during the simulation stages. In order to increase the productivity of the generalized iterative algorithm and speed up the process of building models, the use of grids calculation and a method of speeding up the operation of the algorithm during combinatorial optimization are proposed.

Keywords

Modeling, grids calculation, generalized iterative algorithm, group method of data handling, software implementation, risk of information loss

1. Introduction

Currently, an increasing amount of research is being done in the field of distributed global computing. Middleware, libraries, and tools are being developed that allow you to jointly use geographically distributed but unified resources as a single, powerful platform for executing parallel and distributed applications. This approach to computing is known by several names such as metacomputing, scalable or global computing, etc.

Therefore, it is advisable to use all these technologies in the modeling process, namely in the process of software implementation of modeling algorithms.

2. General Characteristic of the Inductive Modeling Problem

The task of modeling is the construction of mathematical models for the quantitative description of the relationship between target indicators or dependent variables of modeled processes and input or independent variables.

The task of identifying the state of any object, process or system involves building a model based on the results of observations. To solve it, it is necessary to determine the model structure and estimate its parameters.

Problems of modeling complex systems can be solved with the help of either deductive (logical-mathematical) or inductive methods. Deductive and simulation methods have advantages in the case of simple modeling problems, if the theory of the modeled object is known, and therefore is possible to build a model based on physically based principles, applying knowledge about the processes in the object [1-4]. But these methods are not able to give a satisfactory result for complex systems. In this

CITRisk'2022: 3rd International Workshop on Computational & Information Technologies for Risk-Informed Systems, January 12, 2023, Neubiberg, Germany
EMAIL: sashabulgakova2@gmail.com (O.Bulgakova); zosimovvv@gmail.com (V.Zosimov);
ORCID: 0000-0002-6587-8573 (O.Bulgakova); 0000-0003-0824-4168 (O.Bulgakova)

case, gaining knowledge from the data, that is, finding a model based on experimental measurements, has advantages. Often the priory knowledge about such objects is absent.

One of the most well-known modeling methods, the application of which does not arise the problems described above, is the group method of data handling (GMDH), which find knowledge about the object from a sample of data [5-8].

In this computer modeling approach, instead of the traditional deductive path "from the general regularities of the object's functioning to a specific mathematical model", an inductive approach is used "from specific observational data to a general model": the researcher presents a sample, take a hypothesis about a possible class of models and sets the criterion for choosing the best model in this class. Next, the computer works, where becomes possible to minimize subjective factors and obtain a model as an objective result.

This approach of self-organization is based on sorting out gradually more complicated models and choosing the best solution according to the minimum external criterion. Such criteria are based on dividing the sample into parts, while parameter estimation and quality checking of models are performed on different subsamples. This makes it possible to do without a priori assumptions, because the division of the sample allows to automatically (implicitly) take into various types of a priori uncertainty [8].

GMDH algorithms are used to solve the modeling problem based on observational data, which consists in building models of different structures (1)

$$\hat{y}_f = f(X, \hat{\theta}_f) \tag{1}$$

in the polynomial functions class of the Kolmogorov-Gabor polynomial type and finding the optimal structure of the model (2)

$$f^* = \underset{f \in \Phi}{\operatorname{argmin}}\, CR(y, f(X, \hat{\theta}_f)), \tag{2}$$

$$\hat{\theta}_f = \arg \min_{\theta \in R^{s_f}} QR(y, X, \theta_f) \tag{3}$$

under the condition of the external selection criterion minimum (*CR*) in the role of which applied criteria of regularity (4) or unbiasedness (6) based on divided data sample.

$$AR_B(s) = AR_{B|A}(s) = \left\| y_B - \hat{y}_{B|A}(s) \right\|^2 = \left\| y_B - X_{Bs} \hat{\theta}_{As} \right\|^2, \tag{4}$$

$$CB = CB_{W|A,B} = \left\| \hat{y}_W(A) - \hat{y}_W(B) \right\|^2 = \left\| X_W \hat{\theta}_A - X_W \hat{\theta}_B \right\|^2. \tag{5}$$

$$W = [X \vdots y] = \begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} X_A \vdots y_A \\ X_B \vdots y_B \end{bmatrix}, \; n = n_A + n_B$$

where $X[n \times m]$ - input variables, $y[n \times 1]$- output variable, s_f is the complexity of the model *f*, $QR(\cdot) \neq CR(\cdot)$.

3. Generalized Iterative Algorithm GIA GMDH

During a long period of time, many researchers both from Ukraine and abroad (USA, Japan, Czech Republic etc.) elaborate GMDH-type tools on the basis of the original multilayered iterative algorithm MIA GMDH [9-13].

Generalized Iterative Algorithm GIA GMDH was described in detail in [14] and can be used to build both linear and nonlinear models. The generalized iterative algorithm combines the following neuron modifications, based on neurons formed from, Figure 1:

- intermediate arguments (multilayer iterative algorithm MIA);
- both intermediate and initial arguments (relaxation iteration algorithm RIA);
- intermediate arguments as well as intermediate and initial ones (combined iterative algorithm CIA);

- in all three versions, combinatorial optimization of the complexity of partial descriptions is used being linear, bilinear or quadratic.

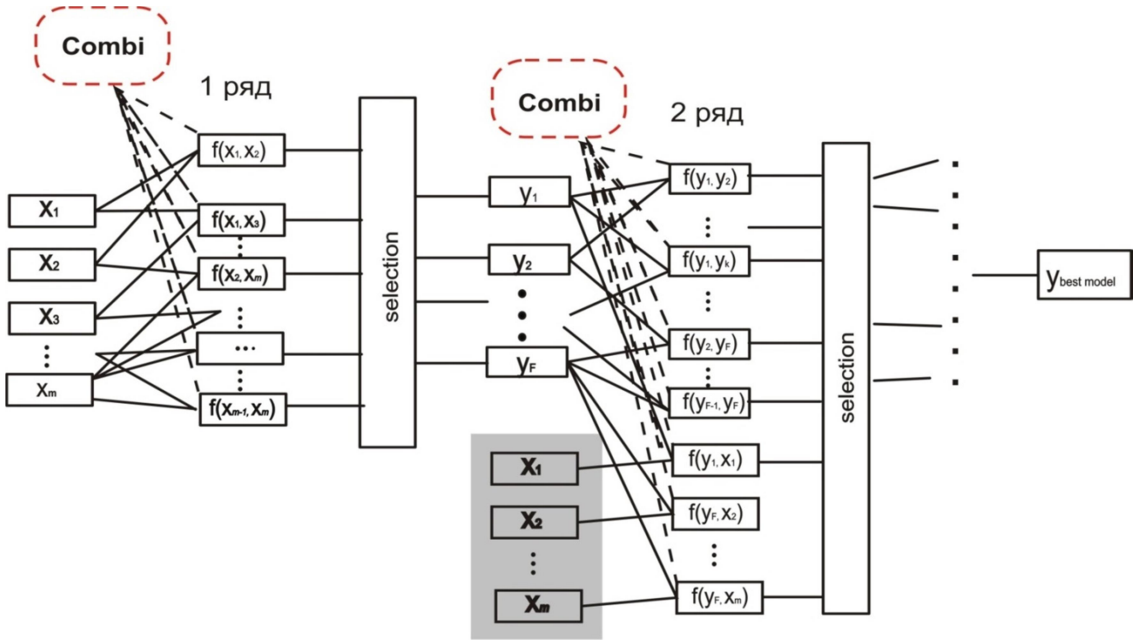

Figure 1: The schema of the generalized iterative algorithm

The proposed variant of hybridization of structures makes it possible to significantly improve the architecture of the classical MIA GMDH algorithm and thereby provide the following complex of new effective properties:

- not to lose informative arguments that can be lost out at the previous layers of modeling;
- sift out non-informative arguments that may be included at previous layers;
- prevent overfitting the built model due to combinatorial optimization of the partial models complexity.

GIA GMDH can be defined by a vector of the following three elements: Dialogue Mode, DM; Iterative-Combinatorial, IC; Multilayer-Relaxation, MR; they produce a set of iterative and iterative-combinatorial algorithms, Figure 2.

Due to that, various iterative GMDH algorithms can be defined as a special case of the generalized iterative algorithm GIA(DM, IC, MR). For example, DM may get three values: 1 – standard automatic mode, 2 – scheduled automatic mode, 3 – interactive mode; IC does two values: 1 – iterative, 2 – iterative-combinatorial algorithms; MR does three values: 1 – classical iterative, 2 – relaxational, 3 – combined algorithms [14]. In this case, all the iterative-combinatorial algorithms belong to the networks with active neurons.

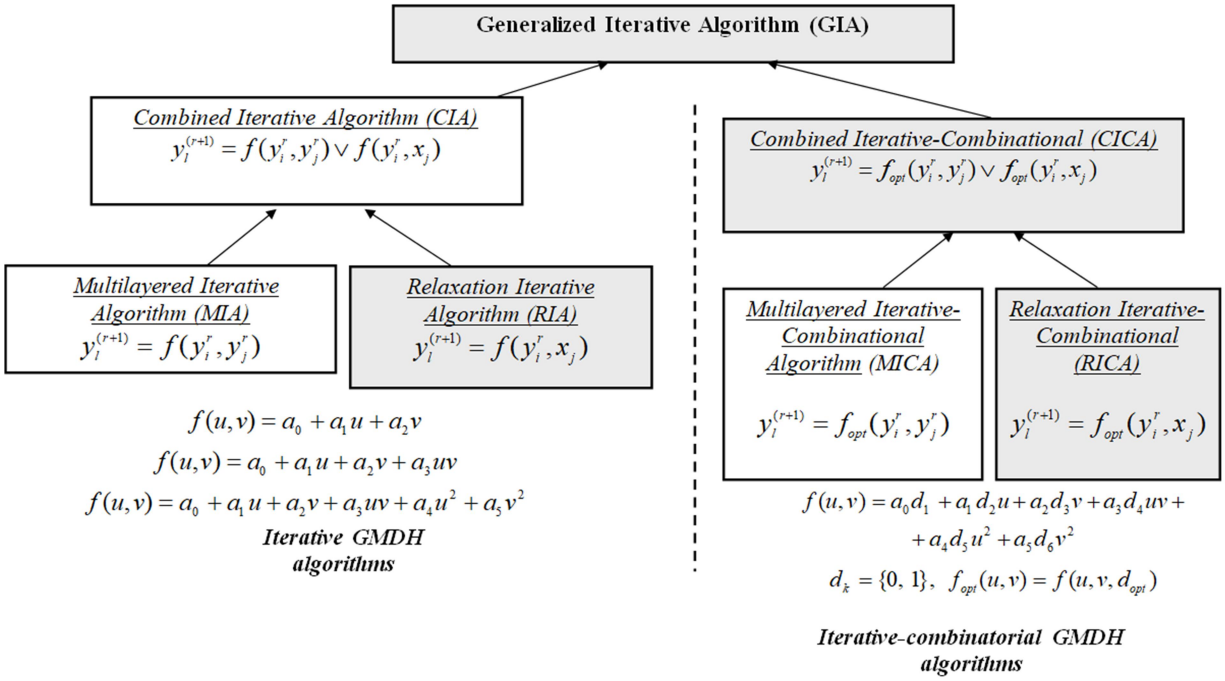

Figure 2: The general block diagram of the approach to the construction of the architecture of the generalized iterative algorithm, where *r* is the layer number (iteration), *opt* is the best model found by combinatorial optimization of a partial model, and *d* is the structure vector.

4. Features of Software Implementation

In the software complex, the modeling process can be implemented in three modes - two automatic and one interactive [15-16]:

- automatic – when the process of self-organization models is performed automatically, and this automatic mode is implemented in two variants:
  – standard – when the partial description and the freedom of choice are set to be the same for all selection rows without exception;
  – planned – when the process of self-organization models is performed automatically according to a given plan, that is, when the partial description and freedom of choice are set differently for different series.
- interactive - when you can directly intervene in the process of self-organization models, using the following possibilities:
  – include or exclude modifications on any layer;
  – change the complexity of the partial description;
  – change the number of models that will move to the next layer;
  – use different criteria for choosing the best models.

In addition, the process of self-organization can be stopped at an arbitrary stage of calculations, and then at any moment of time the calculations can be continued, while all intermediate calculations will be saved.

The main code of the program is written in procedural style. It has several modules, the main interaction with it is carried out through web access. However, you can work with the program locally by launching several processes at the same time without additional time.

Principles of Internet access organization. The program is divided into several modules, most of which support the cli-command line interface. However, the main means of interaction with the program is web access. For its implementation, the Perl module main.cgi is used on the server side and the javascript module index.htm on the client side. We will describe them in more detail below, and for now we will consider the general scheme of web applications that belong to the client-server category.

The software complex features. The server part of the functional application is made in the Perl language, as well as the main code. All interactions with the web server are collected in the main.cgi module. It implements the initial analysis and verification of parameters, authenticates the user and determines the data available, selects the appropriate module for processing the request, converts the received data to the appropriate form, ensures correct processing of fatal errors in the called module. Communication with the web server follows the cgi-protocol, text information for the client part is transmitted in the form of JSON [17].

The client part is written in javascript using the extJS object-oriented framework (framework) and the AJAX concept, and it is an index.htm page. Requests to the server are sent without restarting, the received responses change its appearance. In case of server unavailability or its failure, error messages are displayed to the user. To reduce the number of requests to the server, forms are checked and some elements are disabled until authentication is completed, and cookies are used to save information about this. Security checks are performed on the server side.

5. The risk of program stopping due to high load on the server

During the operation of the generalized iterative algorithm, thousands and tens of thousands of various partial models are formed, and the structure of each of them is optimized. Since the optimization of the structure of partial models takes place with the help of a combinatorial algorithm, i.e. a complete search of all possible combinations of a partial description (COMBI), it requires considerable time. To find opportunities to speed up operations, you should analyze the entire procedure of sorting partial model.

The main stages of operation of the COMBI algorithm [18-19], which are the basic process: transformation of the initial data according to the selected system of reference (basic) functions, in which the model is sought; generation (sorting) in this basis of a complete set of partial models, which gradually become more complicated; estimation using LMS of parameters of each generated model; calculation of the values of the external selection criterion and sequential selection of the best individual models according to this criterion. Note that the process of selecting model structures is organized using the orderly formation of so-called structural vectors d with dimensions of $1\times m$, containing binary elements, and element 1 indicates the presence of a corresponding argument in the model structure, and 0 indicates its absence. For example, if m=3, the vector 101 means that the partial model contains the arguments x_1 and x_3, but does not contain x_2.

It is quite obvious that in this algorithm, the greatest amount of time is spent on evaluating model parameters, which requires the following rather time-consuming operations: formation of matrices of systems of conditional equations for each variant of the structure of each partial model; calculation of the corresponding matrices of systems of normal equations based on these matrices; solving these systems of linear algebraic equations to obtain numerical estimates of the parameters of each such model.

In addition, some time is required to calculate the criterion value of each of the model options, after which the best option is selected based on the minimum of these values, that is, a partial model of optimal complexity. This is actually a description of the active neuron of the GMDH polynomial neural network.

But in [71] it was established that in the combinatorial algorithm of MGUA, the formation of matrices of conditional equations for each partial description with subsequent calculation of the corresponding normal systems is completely impractical, as it leads to multiple calculations of the same values. For example, for models with structures 010, 011 and 110, the same value is calculated three times $x_2^T x_2 = \sum_{i=1}^{n} x_{i2}^2$. Therefore, when applying combinatorial optimization, it is enough to calculate the matrices of the complete system of normal equations $X^T X\theta = X^T y$, which have the form

$$X^T X = \begin{bmatrix} x_1^T x_1 & x_1^T x_2 & \dots & x_1^T x_m \\ x_2^T x_1 & x_2^T x_2 & \dots & x_2^T x_m \\ \dots & \dots & \dots & \dots \\ x_m^T x_1 & x_m^T x_2 & \dots & x_m^T x_m \end{bmatrix} \quad X^T y = \begin{bmatrix} x_1^T y \\ x_2^T y \\ \dots \\ x_m^T y \end{bmatrix} \tag{6}$$

and containing elements of all possible partial normal systems. To obtain any partial normal system, it is enough to take the elements of the matrix $X^T X$, which are located at the intersection of those rows and columns indicated by the units of the structural vector *d*, and the corresponding elements of the vector $X^T y$.

Thus, in order to obtain estimates of the coefficients of all possible variants of the models of each partial description, it is enough to construct the full matrix once and "extract" the necessary partial normal systems from it. Demonstrate the implementation of these procedures on an example *m*=3, *n*=5:

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ \cdots & \cdots & \cdots \\ x_{51} & x_{52} & x_{53} \end{bmatrix}; \quad y = \begin{bmatrix} y_1 \\ \cdots \\ y_5 \end{bmatrix}.$$

:

1. Calculate the normal matrices for the complete model $y = a_1 x_1 + a_2 x_2 + a_3 x_3$:

$$X^T X = \begin{bmatrix} x_1^T x_1 & x_1^T x_2 & x_1^T x_3 \\ x_2^T x_1 & x_2^T x_2 & x_2^T x_3 \\ x_3^T x_1 & x_3^T x_2 & x_3^T x_3 \end{bmatrix}, \quad x_j^T x_j = \sum_{i=1}^{n} x_{ij}^2, \; x_j^T x_k = \sum_{i=1}^{n} x_{ij} x_{ik},$$

$$X^T y = \begin{bmatrix} x_1^T y \\ x_2^T y \\ x_3^T y \end{bmatrix}, \quad x_j^T y = \sum_{i=1}^{n} x_{ij} y_i .$$

2. From the elements of the matrix $X^T X$ and the corresponding elements of the vector $X^T y$, extract partial systems of normal equations depending on the type of structural vector *d* (Table 1):

Table 1

View of partial models depending on the structural vector for *m*=3

#	structural vector	partial models
1	111	$y = a_1 x_1 + a_2 x_2 + a_3 x_3$
2	011	$y = a_2 x_2 + a_3 x_3$
3	101	$y = a_1 x_1 + a_3 x_3$
4	110	$y = a_1 x_1 + a_2 x_2$
5	100	$y = a_1 x_1$

6	010	$y = a_2 x_2$
7	001	$y = a_3 x_3$

3. For the corresponding partial system of normal equations, find coefficient estimates
   - for 100 of the complete normal system matrices, take the following known values: $x_1^T x_1$ and $x_1^T y$ and find the coefficient *a* from the equation:
   $$x_1^T y = a x_1^T x_1$$
   - for 010 of the complete normal system matrices, take the following known values: $x_2^T x_2$ and $x_2^T y$ and find the coefficient *a* from the equation:
   $$x_2^T y = a x_2^T x_2$$
   - for 110 of the complete normal system matrices, take the following known values: $x_1^T x_1$, $x_1^T x_2$, $x_2^T x_1$, $x_2^T x_2$ and $x_1^T y$, $x_2^T y$ and find the coefficients from the equation:
   $$\begin{bmatrix} x_1^T y \\ x_2^T y \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \cdot \begin{bmatrix} x_1^T x_1 & x_1^T x_2 \\ x_2^T x_1 & x_2^T x_2 \end{bmatrix};$$
   - for 101 of the complete normal system matrices, take the following known values: $x_1^T x_1$, $x_1^T x_2$, $x_3^T x_1$, $x_3^T x_3$ and $x_1^T y$, $x_3^T y$ and find the coefficients from the equation:
   $$\begin{bmatrix} x_1^T y \\ x_3^T y \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \cdot \begin{bmatrix} x_1^T x_1 & x_1^T x_3 \\ x_3^T x_1 & x_3^T x_3 \end{bmatrix} \text{ and etc.}$$

Thus, in order to obtain estimates of the coefficients of all possible variants of the models of each partial description, it is enough to build the full matrix once and select the necessary parts of the normal system from it.

The described suggestions allow you to perform all the necessary operations without explicitly writing the complete system of conditional equations into the computer's RAM. At the same time, the calculation time is significantly reduced. This approach is used for radical acceleration of combinatorial optimization of partial descriptions.

6. Using GRID systems to parallelize the computing process

In the research, two main options for parallelization can be distinguished:

1. By tasks - when threads perform different tasks;
2. By data - when threads perform one task, but each with its own part of the common data.

Consider each of these options separately.

Parallelization by tasks. Consideration of this issue is easier to start with the main tasks that the program should perform:

1. Work with the interface (formation of widgets, reaction to change.
2. Resizing and moving windows, response to button presses, changing tabs, opening and selecting in drop-down lists, etc.).
3. Working with the repository (creating, deleting, changing, obtaining a list of various project files, obtaining and decoding information from calculations and results files).
4. Construction of graphs.
5. Calculations directly.

It is obvious that all these tasks differ significantly in execution time. Calculations are the most time-consuming task, which can take several orders of magnitude longer than other actions. When the “fast” tasks are performed sequentially, they will be forced to wait for the completion of the “slow”

tasks. As a result, when starting the calculation, the user will lose the opportunity to see other results, compare their graphs, and the interface will “hang” in general. The problem is exacerbated in a multi-user environment: if the wait due to one's own actions can still be endured, then constant freezes due to the actions of other users are unacceptable. To solve this problem, it is necessary to ensure the possibility of performing all possible actions in different streams.

In this case, it is most convenient to use the generally accepted client-server model.

The interface is handled by the client part, which sends requests to the server to receive data. The server accepts requests and starts a separate thread for each of them, which ceases to exist after receiving the results and returning the response to the client.

If there is a lack of hardware resources, can run the server and the client on the same computer, including if there is only one processor core. The operating system is engaged in switching tasks on its own - although there will be no gain in execution time, but from the point of view of the user's work with the interface, multitasking will be preserved. When expanding the hardware base, the server can be run on a separate computer, then if it has several processor cores, work will be accelerated, because several calculations will be able to be performed simultaneously, each on its own core.

At this stage, you should take into account the number of cores and, for maximum efficiency, do not run more calculations at the same time than there are cores in the system, and it is better to leave even one core for the execution of short-term requests and for the operation of the operating system itself. If it is possible to allocate several computers to the server at once, then this is done according to the backend-frontend scheme (Figure 3).

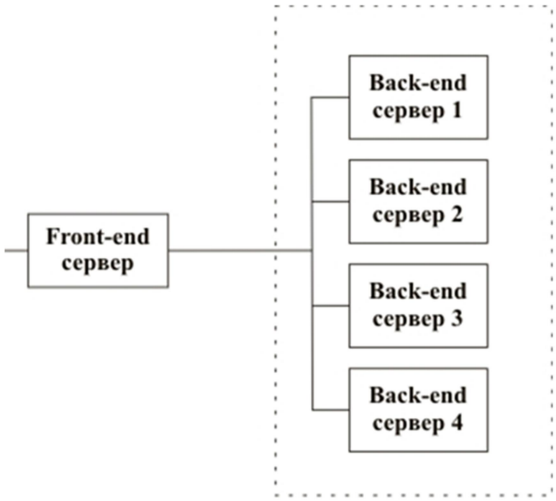

Figure 3: The “backend-frontend” scheme

Most of the computers make up the backend part and do the main work - each of them runs a standard web server and application. A smaller part of the machines is the frontend part. A web server running in reverse-proxy mode is launched on it. Frontend receives user requests, distributes them between working web servers, receives and caches responses from them, and returns results to users.

To ensure transparency at the file level, network storage is used, which is connected to all backend servers. This scheme allows to practically linearly increasing the power of the computing cluster, and also ensures continuous operation in the event of failure of individual backend servers. It is important to note that no changes to the application code are required to achieve scaling.

Task parallelization has many advantages, but there is one serious disadvantage. Despite the possibility of significantly speeding up the execution of several calculations at the same time, there is a complete absence of the possibility of speeding up any individual calculation. The separate calculation will be able to use only one core. To solve this problem, it is necessary to apply a completely different variant of parallel computing - parallelization by data.

Parallelization by data.

Three sections are always available in the developed software complex:

1. the initial section, where data is prepared and divided into groups;

2. section of parallel execution, where computational flows work independently of each other;
3. the final section, where the results of individual flows are collected together, analyzed, then either the completion of the task or the return to the first section follows.

At the same time, it should be taken into account that the creation of computational flows and the exchange of data with them also require additional costs of processor time: if these costs are comparable to the operating time of the computational flow, then instead of acceleration, a significant slowdown will result. If the number of computing threads exceeds the number of cores, then again we will get a slowdown instead of an acceleration.

The most rational is parallelization after compiling combinations of variables, but not on separate equations, but on their groups. The size and number of groups is chosen according to the number of available cores. From a technical point of view, the problem is much more complicated than in the case of task parallelization.

There is no unequivocally winning choice: gaining an advantage in some conditions, we lose it in others. As a result, the implementation implemented in the program in no way claims to be recognized as the best, it is simply one of the possible ones, and the compromise between efficiency and universality is shifted towards universality.

7. Conclusions

The paper presents the problems of software implementation of the iterative method of the inductive algorithm, which can lead to the risk of data loss at the modeling stages. To improve the performance of the generalized iterative algorithm GMDH and speed up the process of building models, two approaches have been proposed:

- in the process of combinatorial optimization, matrices of a complete normal system are calculated instead of matrices of conditional and normal equations for each partial description;
- the use of parallel computing and the organization of network multi-access to the program.

Parallelization by tasks allows you to organize multi-access to the software package, allocating a separate processor core for the calculation of each task. But such an implementation does not have the ability to speed up a separate calculation. Therefore, another type of parallelization is implemented in the work - according to data, which can significantly speed up the execution of each individual task.

References

[1] P. Lynn, J. Hall, V. Brown, G. Nicolaas, Extended field efforts to reduce the risk of non-response bias: do they pay off?, ISER working Paper Series, 2011-24, 2011, p. 31.

[2] P. Marsden, J. Wright, Handbook of survey research. Emerald Group Publishing, 2nd edition, 2010, 886 p.

[3] E. Singer, Nonresponse bias in household surveys, Public Opinion Quarterly, 70 (5), 2006, pp. 637-645

[4] H. Madala, A. Ivakhnenko, Inductive Learning Algorithms for Complex Systems Modeling. CRC Press, New York, 1994.

[5] Ivakhnenko A.G. The group method of data handling in prediction problems, Soviet Automatic Control c/c of Avtomatika, 9, 6, 1976, pp. 21-30

[6] Ivakhnenko A.G., Inductive sorting-out GMDH algorithms with polynomial complexity for active neurons of neural networks, Proceedings of the International Joint Conference on Neural Networks, Piscataway, New Jersey, USA, IEEE, 1999

[7] Mahfoud S. W. Niching methods for genetic algorithms, Technical Report 95001, Illinois Genetic Algorithms Laboratory (IlliGaL), University of Ilinios at Urbana-Champaign, May 1995.

[8] V. Stepashko, Inductive modeling from historical perspective, Proc. Int. Conf. on Computer Science and Information Technologies CSIT-2017, Lviv: LNPU, 1, 2017, pp. 537-542

[9] S. Yefimenko, Advances in GMDH-based Predictive Analytics Tools for Business Intelligence Systems, Proc. Int. Conf. on Advances in Computer Information Technologies ACIT-2018, Ceske Budejovice, 2018, pp. 254-257
[10] T.Aksyonova, V.Volkovich, I.Tetko, Robust Polynomial Neural Network in Quantative-Structure Activity Relationship Studies, Systems analysis modeling simulation, 43(10), 2003, pp. 1331-1341
[11] A.Ivakhnenko, D.Wunsh, G.Ivakhnenko, Inductive sorting-out GMDH algorithms with polynomial complexity for active neurons of neural networks, Proc. of the Int. Joint Conf. on Neural Networks, IEEE, New Jersey: Piscataway, 1999, pp. 1169-1173
[12] T. Kondo, GMDH neural network algorithm using the heuristic self-organization method and its application to the pattern identification, Proceedings of the 37th SICE Annual Conference SICE'98 IEEE, 1998, pp. 1143-1148
[13] P. Kordík, Náplava, M. Šnorek, P. Genyk-Berezovskij, The Modified GMDH Method Applied to Model Complex Systems, Proceedings of the International Conference on Inductive Modelling ICIM'2002, Lviv: LNPU, 2002, pp. 134-138
[14] V. Stepashko, O. Bulgakova, V. Zosimov, Construction and research of the generalized iterative GMDH algorithm with active neuronsб Advances in Intelligent Systems and Computing, 689, 2018, pp. 492–510
[15] V. Zosimov, O. Bulgakova, Web Data Displaying Approach Based on User's Semantic Profile Templates, International Scientific and Technical Conference on Computer Sciences and Information Technologies, 1, 2020, pp. 428-431
[16] V. Zosimov, O. Bulgakova, Calculation the Measure of Expert Opinions Consistency Based on Social Profile Using Inductive Algorithms, Advances in Intelligent Systems and Computing, AISC Book Series, 1020, 2020, pp. 622-636
[17] O. Bulgakova, V. Mashkov, V. Zosimov, P. Popravkin, Risk of Information Loss Using JWT Token, CEUR Workshop Proceedingsthis link is disabled, 3101, 2021, pp. 292-299
[18] V. Stepashko, R. Voloschuk, S. Yefimenko, A Technique for Integral Evaluation and Forecast of the Performance of a Complex Economic System, 2020 10th International Conference on Advanced Computer Information Technologies, ACIT 2020 - Proceedings, 2020, pp. 704-707
[19] V. Stepashko, S. Yefimenko, Improvement of a sorting-out GMDH algorithm using recurrent estimation of model parameters, Advances in Intelligent Systems and Computing, 1080, 2020, pp. 523-534