Intelligent adaptive algorithms for solving the algebraic eigenvalue problem on modern supercomputers*

Oleksandr Khimich [1,†], Oleksii Chystiakov [1,*,†]

[1] V.M. Glushkov Institute of Cybernetics, NAS of Ukraine, Akademika Glushkova Avenue, 40, 03187, Kyiv, Ukraine,

Abstract

Mathematical modeling of complex objects often reduces to solving of an algebraic eigenvalue problem (AEP) of extremely large sparse matrices of various structures. The most important parameters of parallel algorithms are: computational time, used computer resources, and the obtained results reliability. Modern supercomputers architecture becomes more complicated, their performance increases due to increment the number of processors and various coprocessors [1]. Computational efficiency issues often arise when using supercomputers for mathematical modeling. There are significant differences between maximum and operational productivity of modern computers due to losses in the communication component [2, 3] and other factors. There are also problems with the computer results reliability, related to approximate data, features of machine arithmetic and rounding. Therefore, it is necessary to provide computer research of mathematical properties of the problems and analysis of the obtained solution results [4].

The article presents intelligent adaptive algorithms for solving AEP for large sparse matrices of various structures on modern supercomputers. Based on the results of a computer study of the portrait and structure of the matrix and the mathematical properties of the problem, using artificial intelligence (neural networks and knowledge bases), the algorithms automatically determine an effective parallel computing model and topology, required number of computing elements, and the bit depth of computations to ensure the results reliability. The general scheme and adaptive algorithms implementation steps are described. Theoretical studies and numerical experiments confirmed high efficiency and adaptability of the developed algorithms on various computers architectures, providing stable results in modeling of various processes.

Keywords

Mathematical modeling, intelligent adaptive algorithms, linear algebra algorithms, parallel algorithms, algebraic eigenvalue problem

1. Introduction

Increased computing capabilities (high performance and significant amounts of memory) make it possible to solve new scientific and technical problems and organize numerous experiments, which significantly reduce the cost and time of developing modern technology. However, the wide variety of existing algorithmic software created for modern parallel computers of different architectures in different programming languages and operating under different operating and hardware platforms, and the large amount of documentation on the use of new computing tools - all this requires significant intellectual effort and time from users.

In addition, nowadays there is a significant increase in the requirements for the completeness of the studied discrete models of objects from different subject areas. Thus, the size of the problems that have to be solved on computers is significantly increasing. To ensure the reliability of the studied properties of such modeling objects, it is necessary to develop new numerical methods and parallel algorithms for solving problems using new programming technologies involving arbitrary bitness, artificial intelligence tools, etc. In order to effectively use such algorithms on different computer architectures, it is necessary to provide for parallelism that scales at different stages of the computational process.

*International Workshop on Computational Intelligence, co-located with the IV International Scientific Symposium "Intelligent Solutions" (IntSol-2025), May 01-05, 2025, Kyiv-Uzhhorod, Ukraine

1,* Corresponding author.

khimich505@gmail.com (O. Khimich); alexej.chystyakov@gmail.com (O. Chystiakov)

0000-0002-8103-4223 (O. Khimich), 0000-0001-6456-2094 (O. Chystiakov)

Mathematical models of many engineering problems, for example, in mechanics, construction, aircraft construction, etc., are described by systems of differential equations or difference equations, the solution of which consists in determining the eigenvalues and eigenvectors of matrices of various structures [4].

New intelligent algorithms are proposed for solving the algebraic eigenvalue problem with sparse matrices of various structures with automatic tuning functions for an efficient computing environment, which will ensure obtaining reliable solution results with effective use of computer resources and time. At the same time, intellectual support for users is increased [5–8].

2. General characteristics of the intelligence of new parallel algorithms for solving eigenvalue problems

The algebraic eigenvalue problem (AEP) consists in finding such numbers λ for which there exist non-zero solutions to systems of linear algebraic equations [4]:

$$Ax = \lambda Bx,\ A, B \in M^{n \times n},\ x \in C^n,\ \lambda \in C, \tag{1}$$

where $M^{n \times n}$, is the set of square matrices of order *n*. The numbers λ are the eigenvalues of problem (2.1), and the vectors x are the eigenvectors of this problem. Problem (2.1) is called the generalized eigenvalue problem.

If B is an identity matrix of order n (i.e. $B \equiv I_n$), then the problem

$$Ax = \lambda x \tag{2}$$

is called the standard eigenvalue problem. In this case, the numbers λ and the vectors x are called the eigenvalues and vectors of the matrix A, respectively. There can be the following problems of the Eigenvalue Problem [8]: a complete eigenvalue problem – find all eigenvalues and all eigenvectors; a partial eigenvalue problem – find one or more eigenvalues and their corresponding vectors or only eigenvalues (all or some).

Intelligent parallel algorithms for solving AEP with sparse matrices of arbitrary structure have been developed: the alternating triangular method for calculating the minimum eigenvalue in problem (1); the generalized conjugate gradient method for calculating the minimum eigenvalue in problem (2), the subspace iteration method for calculating several minimum eigenvalues and their corresponding eigenvectors of problem (1). Let's consider their general functional characteristics.

2.1. Computer identification and classification of sparse matrix of unknown structures

Nowadays, the concept of a “large” problem is changing radically, involving the use of numerical methods and algorithms focused on hundreds or thousands of processors of parallel supercomputers of various architectures. Modern problems of mathematical modeling of physical and mechanical processes, which are reduced to solving the AEP, most often have sparse matrices of various structures and very large orders. There is a need to determine the matrix structure in the computer and, if necessary, apply effective methods of reducing them to a regular form [9–11]. Since when processing a sparse matrix, operations are performed only with non-zero elements, using a certain rearrangement method, they can be placed in such a way as to reduce the matrix filling and the number of interprocessor exchanges, as well as to ensure balancing of processes during calculations.

The proposed intelligent algorithms use neural networks to determine the portrait of a sparse matrix [12]. If the matrix has an arbitrary structure, it is automatically reduced to a regular form. In [8], methods of processing sparse matrices of various structures for their effective use in mathematical modeling problems on computers are considered in detail.

To arrange sparse matrices of arbitrary structure in proposed intelligent algorithms, the parallel section method is used, as a result of which the sparse matrix has a bordered block-diagonal form.

Such a matrix structure is effective because each individual block will be placed entirely in one parallel process on the computer.

When solving applied problems, AEP with exact initial data of the form (1) or (2) rarely arise. The most typical formulation of these problems is to specify the corresponding errors in the initial data:

$$\|A-\overline{A}\|=\|\Delta A\|\le\varepsilon_A\|A\|,\|B-\overline{B}\|=\|\Delta B\|\le\varepsilon_B\|B\| \quad (3)$$

In this case, the structure of the matrices (matrix) of the original problem 1 and the perturbed problem (1), (3), or (2), (3) does not change. That is, if the original matrix is symmetric, then the perturbed one remains symmetric, if the original one is strip, then the perturbed one is strip.

Therefore, it is necessary to consider not a problem with exact initial data of the form (1) or (2), but a problem with approximate data of the form (1), (3) or (2), (3) and to estimate the perturbation of the solution depending on the perturbation of the initial data (3), because the proximity of the elements A and $\overline{A}$ (as well as B and $\overline{B}$ for the generalized problem) does not always ensure the proximity of the eigenvalues.

In the considered intelligent algorithms, to study the mathematical properties of problems of the form (1), (3) or (2), (3), the algorithms are adapted to data flows using neural networks [8]. In this case, the data stream should be understood as sets of input data for the problem and a knowledge base from a given subject area, on the basis of which the mathematical properties of matrices entered into the computer (positive definiteness, degeneracy, etc.) are automatically studied, as well as the parallelization model and computer bit capacity are determined to ensure the reliability of the solution results with the effective use of computing resources. This takes into account factors of the external environment required for the algorithm. This includes the hardware environment: required CPU RAM; required number of GPUs; application programs running simultaneously with the algorithm, etc.

2.2. Automatic tuning of an efficient computing environment to the mathematical properties and scope of the problem

The presence of large-scale practical problems that are subject to mathematical modeling and a wide variety of powerful computing equipment of various architectures pose the following tasks to developers of algorithmic and software support: to create adaptive parallel algorithms and programs with functions for their automatic adjustment to the mathematical properties of the problem and an effective (variable) computer environment (multi-level parallelism, variable topology of interprocessor connections, multi-bit arithmetic, mixed bitness, etc.), which will ensure the reliability of computer results for solving problems with approximate data with effective use of computing resources. [13].

Determining the optimal number of computing devices. This is a non-trivial task, since the time it takes to solve the problem depends on many factors: the amount of RAM on the nodes of the parallel computer, the performance of the processors and the communication environment, as well as the software implementation of the problem-solving algorithm.

According to Amdahl's law, the time to solve a problem consists of two components: the time to execute sequential operations and the time to execute parallel operations, taking into account the number of processor elements. However, when solving problems on parallel computers, the time for performing additional operations necessary to exchange information between computing devices is added to the time for actually solving the applied problem, i.e., overhead. They directly depend on the hardware characteristics of the computing system: the data transfer rate in the communication environment, the performance of processor elements and memory capacity, as well as the number of transmitted messages to solve the problem.

Increasing the number of computing devices, on the one hand, allows you to reduce the time for solving a problem by simultaneously processing several blocks of information, but, on the other hand, the increasing number of transmitted messages on a parallel computer has the opposite effect.

As is known, speedup S_p and efficiency E_p coefficients are used to assess the quality of a parallel algorithm [3]:

$$S_p = T_1 / T_p, E_p = S_p / p,$$

where p is the number of processes used for calculations, T_1 is the time to solve the problem by one process, T_p is the time to solve the same problem by p processes.

For hybrid computers, if we denote: T_1 – the time to solve a problem on the architecture using one CPU and one GPU, T_p – the time to solve the same problem using pCPU and pGPU , then the time to solve the problem can be calculated by the formulas:

$$T_1 = O t_g, T_p = O t_g + M_1 t_{opg} + M_2 t_{opp} + Q_1 t_{cpg} + Q_2 t_{cpp}.$$

Here we have the following notations: O – the number of algorithm execution operations; t_g – the average execution time of one arithmetic operation on the GPU; t_{opg} – the time of information exchange between the CPU and the GPU; t_{opp} – the time of exchange of one machine word between two CPU processes; t_{cpg}– the time to establish a connection between the CPU and the GPU; t_{cpp} – the time to establish a connection between two processes on the CPU; M_i and Q_i – respectively, the number of exchanges and synchronizations between the CPU and the GPU on the i-th iteration ($i = 1,2,\ldots$).

Based on such formulas, for each of the considered intelligent algorithms, the acceleration and efficiency coefficients for various parallel computing models were theoretically proven and tested when solving experimental and practical problems on parallel computers of various architectures.

Building an effective parallelization model. When creating algorithms for solving problems on modern supercomputers, it is necessary to take into account the multi-level parallelism model, the availability of computer memory of various types and volumes, the number and types of processor elements, and the peculiarities of the connections between them [13].

In intelligent algorithms, it is assumed that two main levels of parallelism are distinguished: the upper one – macro-operations are performed in parallel (subtasks - logically independent parts of algorithms) and the lower one - parallelization of the execution of each of the macro-operations) [14].

The first level of the parallel computing model (top level, MIMD model) is process level parallelism (PLP), in which processes execute subtasks in parallel, using both distributed and shared CPU memory, synchronizing computations and data exchanges. For parallelizing processes on distributed memory, the most effective system is MPI [15], and on shared memory, OpenMP [16].

The second level (lower level, SIMD model) is thread level parallelism (TLP) – parallelization of macro-operations using multiple threads and shared memory. In this case, each of the top-level macro-operations is parallelized between a number of threads on CPU cores with shared memory using the OpenMP system, as well as on coprocessors, for example, GPUs using NVIDIA CUDA technology [17].

The use of distributed memory on a parallel computer creates certain problems with data exchanges between CPU processes, as well as between CPU and GPU, which can significantly exceed arithmetic operations in terms of duration. Intelligent algorithms use a hypercubic topology of interprocessor connections, a block-cyclic method of matrix parallelization between CPU processes, and also provide for the execution of data array transfers with simultaneous execution of arithmetic operations. Such implementation of parallelization on a hybrid computer minimizes the total execution time of the task.

Using arbitrary bit depth calculations. Solving many practical problems with approximate data on modern supercomputers requires increasing the accuracy of calculation results. The problems of enormous volume that arise in the modeling of physical and mechanical processes are particularly critical to the accuracy of computer results. To ensure sufficient accuracy, some practical problems

require a two-fold increase in bit depth, others a four-fold increase, and there are also problems that require hundreds of bits of calculation.

One of the easiest ways to minimize errors associated with rounding and loss of precision when performing computer calculations is to further increase the bit depth. There are various ways to improve the accuracy of computer results, such as using symbolic calculations. This means that, for example, when calculating a polynomial, real numbers are not represented in the traditional floating-point format, but as rational fractions. Among the application packages that provide arbitrary bit depth of calculations using a character processor, one can note Matlab [18].

The considered intelligent algorithms use a special GMP (GNU Multiple-Precision Library) program library to implement high-precision calculations [19]. Today, the main disadvantage of using existing methods of increased bit rate is a significant increase in calculation time. To solve such problems, it is necessary to use exaflop computers.

Fine-Tiling Algorithms. Despite the wide variety of powerful parallel computers of MIMD architecture with various types of coprocessors, powerful multi-core computers have been actively developed and used in recent years. For such computers, American scientists under the leadership of Prof. J. Dongarra proposed small-tile algorithms for solving SLAE by direct methods based on ¿^{T}, LU and QR – expansion of dense non-degenerate matrices, in which operations on matrices of levels BLAS, BLAS2, BLAS3 (matrix-matrix operations) are performed on small square blocks of data ("tiles"). on different (mixed) bitness – single and double [20].

Intelligent algorithms for solving AEP provide for their use on multi-core computers. Small-tile algorithms have been implemented with calculations performed on single and double bitness. This makes it possible to effectively solve small-scale problems on multi-core computers [7].

In fig. 1 shows the scheme of application of intelligent adaptive algorithms in mathematical modeling of physico-mechanical processes, which boil down to the solution of the partial generalized AEP of sparse positive-definite matrices.

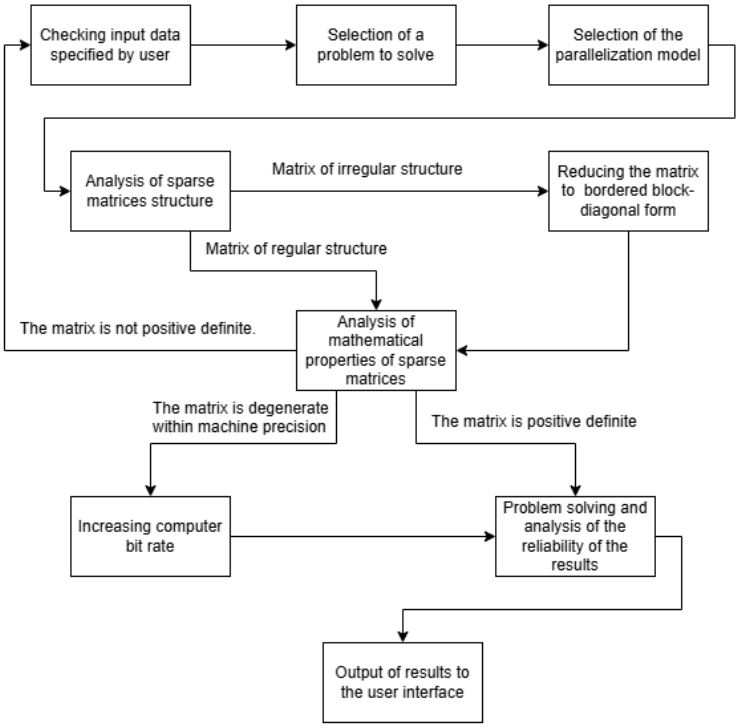

Figure 1: Scheme of the application of intelligent algorithms in mathematical modeling of physical and mechanical processes on parallel computers.

The algorithm starts with entering, checking and evaluating the complexity of the input data specified by the user (Step 1: Checking input data specified by user). Then the task to be solved is selected (Step 2: Selection of a problem to solve). In our case, the AEP is solved. In the next step (Step 3: Selection of the parallelization model), the mechanism of parallelization of calculations is determined. Depending on the size and characteristics of the input data, various parallel computing models and their combinations can be selected, such as: MPI, OpenMP, CUDA, MPI + OpenMP, MPI + CUDA, etc.

Next (Step 4: Analysis of sparse matrices structure) the structure of the input matrices is checked and if the matrices have an irregular structure, they are reduced to a bordered block-diagonal form. When performing the next step (Step 5: Analysis of mathematical properties of sparse matrices), if it is determined that the matrix is positive definite, then the computational process begins (Step 6: Problem solving and analysis of the reliability of the results). Otherwise, if the matrix is not positive definite, the user will be returned a message about the input data error. If the matrix is singular within the machine precision, then the bit depth of the computations is increased. Upon completion of the computations, the results together with the report of the computational process are transferred to the user interface (Step 7: Output of results to the user interface).

3. Experimental study of the effectiveness of the intelligent algorithm of the subspace iteration method for solving the AEP

The study of the new intelligent algorithm was conducted on the SKIT supercomputer complex of the V.M. Glushkov Institute of Cybernetics [21]. Fig. 1 – 5 presents the time characteristics of solving the APVZ when using sparse matrices of various structures and volumes from the Florida collection [22]: G2_circuit – matrix order 150 102, number of non-zero elements 726 674; Bone010 – matrix order 986 703, number of non-zero elements 47 851 783; Emila_923 – matrix order 923 136, number of non-zero elements 47 851 783. bmwcra_1 – matrix order 148 770, number of non-zero elements –10 641 602.

Figure 2 shows the acceleration of the intelligent algorithm when finding the few minimum eigenvalues and their corresponding eigenvectors of sparse matrices of different structures using different numbers of computing devices. The experimental results show that the developed adaptive algorithm provides good scalability, i.e. the task execution time decreases proportionally with the increase in the number of computing devices.

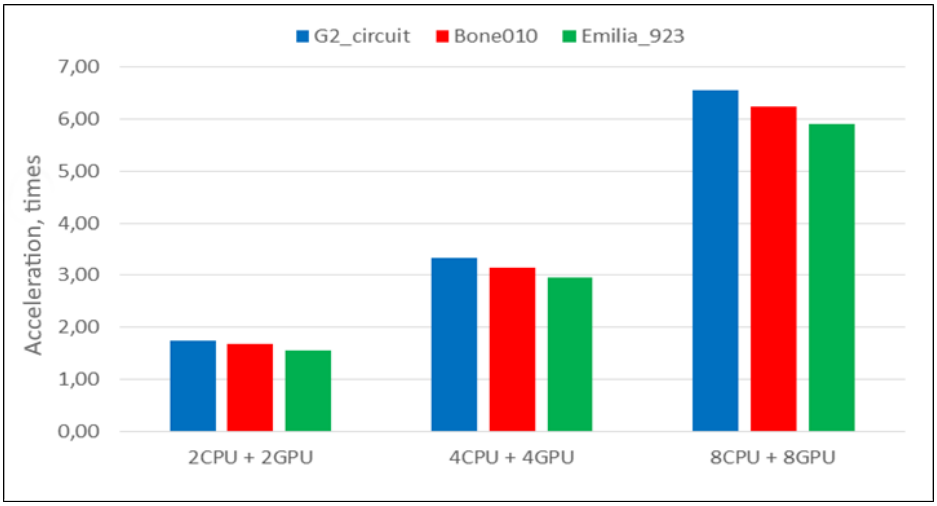

Figure 2: Acceleration of the subspace iteration algorithm for bordered block-diagonal matrices.

An experimental study of the algorithm's efficiency on computational models of different levels of hybrid computers was also conducted (Fig. 2 – 4). The results of the research are also given in [6–8].

The graphs (Figure 3) demonstrate the acceleration of the adaptive algorithm of the subspace iteration method on the MIMD architecture (MPI parallelization) using a different number of MPI processes.

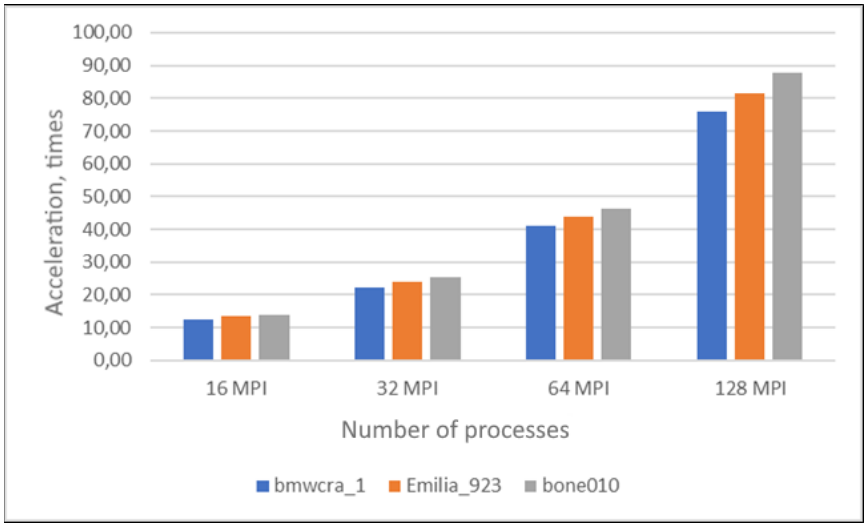

Figure 3: Acceleration of an intelligent algorithm on a first-level parallelism architecture.

The graphs (Figure 4) show the acceleration of the intelligent algorithm on a multi-core MIMD architecture using different numbers of processes and threads on the cores (MPI+OpenMP parallelization).

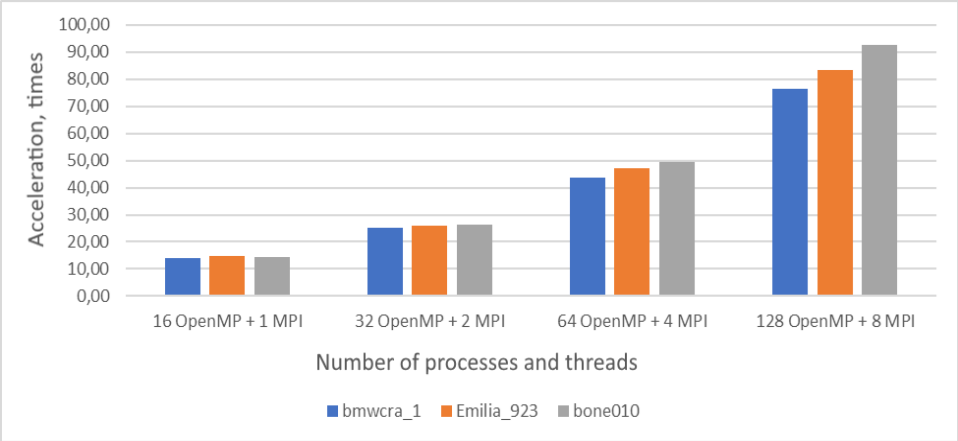

Figure 4: Acceleration of the subspace iteration method algorithm on the first and second level parallelism architecture.

As can be seen from Figures 2 – 4, the use of multi-level parallelism allows for more efficient use of computing resources and provides greater acceleration of the algorithm with the same computing resources. For example, the acceleration obtained for the Bone010 matrix – matrix order 986,703, number of non-zero elements 47,851,783, increases by an average of 4% – 9% with a multi-level scheme.

The graphs (Figure 5) show the acceleration of the algorithm on a multi-core architecture of the first and second levels of parallelism of a MIMD computer when using graphics accelerators.

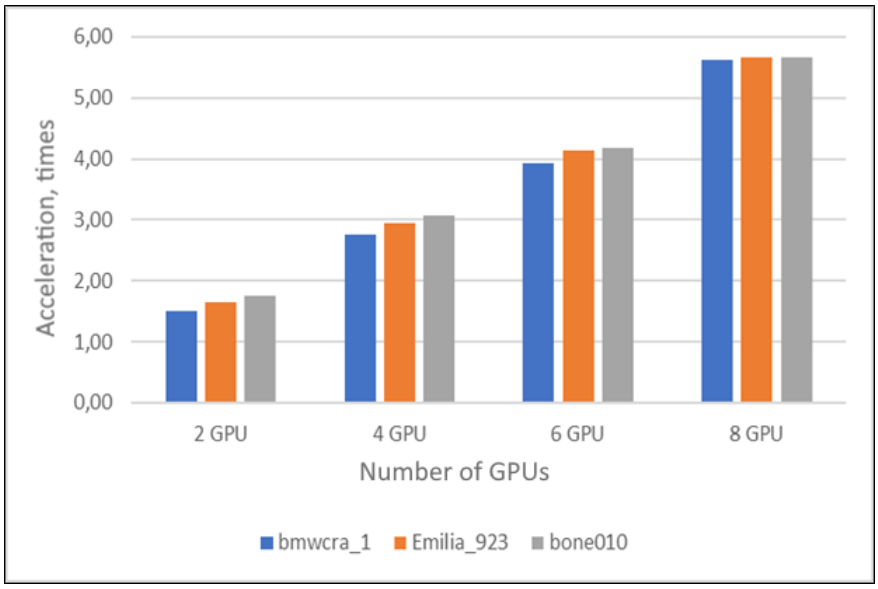

Figure 5: Acceleration of an intelligent algorithm on the architecture of first and second level parallelism and using GPU.

The graphs in Figure 6 demonstrate the dependence of the acceleration of the adaptive algorithm on the block size in the matrix distribution on a multi-core MIMD architecture using different numbers of processes and cores of a hybrid computer, using one MPI process and 16 OpenMP threads.

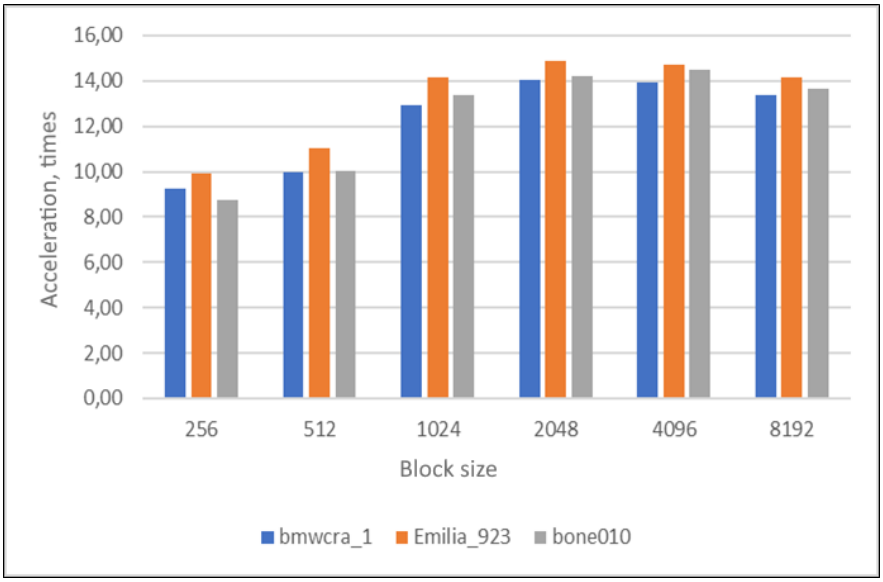

Figure 6: Dependence of algorithm acceleration for block-diagonal matrices on the size of the matrix block on the architecture of parallelism of the first and second levels.

As theoretical and experimental studies have shown, for the effective implementation of intelligent algorithms for solving AEP of sparse matrices of different structures and volumes, it is necessary to coordinate the scope of the problem and the effective parallelism model. The proposed algorithms automatically study the input AEP data and determine the effective parallelism model. This significantly speeds up problem solving.

Conclusions

The need to develop more refined mathematical models of objects in various subject areas, taking into account as many factors as possible, ensuring the reliability of computer solutions, automating the processing of large volumes of data require new conceptual solutions and approaches to the creation of algorithms and programs for solving the problems of computational mathematics, in particular, for solving the algebraic problem of eigenvalues of sparse matrices, on modern supercomputers of various architectures. The proposed intelligent adaptive algorithms meet the growing requirements for accurate and efficient modeling of complex tasks and processes on parallel computers of various architectures. Such algorithms, realizing in the computer an automatic study of the type of sparse matrices and mathematical properties of the problem with approximately given data, simplify the use of multilevel parallelism, increase the efficiency of processing sparse matrices of various structures. Theoretical and experimental studies have shown that the proposed algorithms not only increase the efficiency of computing due to scalability and optimization of the used resources, but also increase the accuracy of the results obtained by adjusting the necessary bit rate of computing. This makes algorithms in demand for solving engineering and scientific problems, reducing the time of application development and increasing the quality of numerous simulations.

In the future, the proposed intellectualization approaches can be adapted for parallel algorithms for solving other problems of linear algebra. Also, further improvements of intellectual algorithms are possible using Regression Neural Networks for forecasting the necessary computing resources, as well as Reinforcement Learning for the redistribution of computations during a complex and long computing process.

Declaration on Generative AI

During the preparation of this work, the authors used CahtGPT-4o for grammar and spelling check.

References

[1] TOP 500 List-November 2024. URL: https://top500.org/lists/top500/2024/11/

[2] J. Dongarra, P. Beckman and et al., The International Exascale Software Project Roadmap. International Journal of High Performance Computing Applications. Vol. 25, N 1, 2011, pp. 3–60

[3] I. V. Sergienko, O. M. Khimich, Mathematical modeling: From small to exaflops. Bulletin of the National Academy of Sciences of Ukraine. N. 8, 2019, pp. 37–50.

[4] A. N. Khimich, I. N. Molchanov, A. V. Popov, T. V. Chistyakova, M. F. Yakovlev, Parallel algorithms for solving problems in computational mathematics. Kiev: Naukova dumka, 2007.

[5] A.V. Chistyakov, On improving the efficiency of mathematical modeling of the problem of stability of construction. Artificial Intelligence. N 8. 25(3), 2020, pp. 27–36.

[6] O.M. Khimich, O.V. Popov, O.V. Chystyakov. V.A. Sidoruk, A Parallel Algorithm for Solving a Partial Eigenvalue Problem for Block-Diagonal Bordered Matrices. Cybernetics and Systems Analysis. V. 56, N 6, 2020, pp. 61–74. doi:10.1007/s10559-020-00311-z

[7] O.M. Khimich, A.V. Popov, O.V. Chystyakov, V.O. Kokhanovsky, Adaptive Algorithms for Solving Eigenvalue Problems in the Variable Computer Environment of Supercomputers. Cybernetics and Systems Analysis.Vol. 59, N 3, 2023, pp. 480–492. URL: https://link.springer.com/article/10.1007/s10559-023-00583-1

[8] O. Khimich, A. Popov, A. Chistyakov, Effective Use of Sparse Matrices in Problems of Mathemaical Modeling. CEUR Workshop Proceeding. Vol. 3501, 2022, pp. 212–221. URL: https://www.scopus.com/record/display.uri?eid=2-s2.0-85175705798&origin=resultslist

[9] S. Pissanetzky Sparse Matrix Technology. Academic Press, London, 1984.

[10] G. H. Golub and C. F. Van Loan. Matrix Computations, third edition. The Johns Hopkins University Press, Baltimore, MD, 1996.

[11] Saad, Y. Iterative Methods for Sparse Linear Systems. 2nd ed. Society for Industrial and Applied Mathematics, Philadelphia, 2003.
[12] S. Khaikin, Neural networks: a complete course, 2nd ed. M.: OOO I.D. Williams, 2008.
[13] O.V. Popov, O.V. Rudich, O.V. Chistyakov, Multilevel model of parallel computing for linear algebra problems. Problems of programming. No 2-3, 2018, pp. 83-92
[14] O. Popov, O. Chystiakov, On the efficiency of algorithms with multilevel parallelism. PHYSICAL AND MATHEMATICAL MODELING AND INFORMATION TECHNOLOGIES. No 33, 2021, pp. 133-137. doi:10.15407/fmmit2021.33.133
[15] Message Passing Interface Forum. URL: https://www.mpi-forum.org/
[16] OpenMP. Architecture Review Board. URL: http://www.openmp.org/
[17] CUDA Toolkit 11.6 Update 2 Downloads. NVIDIA Developed. URL: https://developer.nvidia.com/cuda-downloads
[18] MATLAB. URL: https://www.mathworks.com/help/matlab/
[19] E.A. Nikolaevskaja, A.N. Chimich, T.V. Chistyakova, Programming with Multiple Precision. Springer-Verlag. Studies in Computational Intelligence, Berlin, Vol. 397, 2012.
[20] A. Buttari , J. Langou, J. Kurzak, J. Dongarra, A Class of Parallel Tiled Linear Algebra Algorithms for Multicore Architectures. Parallel Computing. V. 35, issue 1, 2009, pp. 38–53. doi:10.1016/j.parco.2008.10.002 .
[21] SKIT supercomputer complex. URL: http://icybcluster.org.ua
[22] The Suite Sparse Matrix Collection. URL: https://sparse.tamu.edu/