Hierarchical Domain Representation in the AlgoWiki Encyclopedia: From Problems to Implementations*

Alexander Antonov ✉, Alexey Frolov, Igor Konshin, and Vladimir Voevodin

Lomonosov Moscow State University
Institute of Numerical Mathematics of the Russian Academy of Sciences
Moscow, Russia
asa@parallel.ru, frolov@inm.ras.ru, igor.konshin@gmail.com,
voevodin@parallel.ru
http://algowiki-project.org

Abstract. Algorithm description is the basic unit in the AlgoWiki Open Encyclopedia of Algorithmic Features. However, computational algorithms are not objectives in and of themselves: they are needed to address problems encountered in various fields of science and industry. On the other hand, there are many practical problems that can be tackled using various methods. This warrants the introduction of another basic term that fits between the concepts of a problem and an algorithm. Also, any algorithm can have different implementations, whether related to a single computing platform or to different platforms. The "problem–method–algorithm–implementation" chain is the basis for describing any subject area in AlgoWiki. This paper describes the permitted freedom in describing such chains, which arises when studying the approaches to address various practical problems.

Keywords: AlgoWiki, problem, method, algorithm, implementation, parallelism resource, parallel computing, supercomputers.

1 Introduction

The issues of efficiency and parallelism support throughout the entire supercomputer software stack are central to all global supercomputing forums today [1–3]. Judging by current trends, the degree of computing system parallelism will grow by an order of magnitude every several years [4]. This illustrates the relevance and importance of addressing this issue throughout the entire range of computing devices, from mobile platforms to exascale supercomputers. Scientists are

* The results described in sections 2–5 were obtained at Lomonosov Moscow State University with the financial support of the Russian Science Foundation (agreement № 14-11-00190). The research was carried out using the equipment of the shared research facilities of HPC computing resources at Lomonosov Moscow State University.

currently examining architectures that would potentially form the foundation for new generations of computers. Light and/or heavy computing cores, accelerators, SIMD and data-flow processing concepts can be used. Vector processing seems to be getting new life today. This is true for the new-generation Intel Xeon and Intel Xeon Phi processors. Scalable Vector Extensions (SVE) to ARM processor instruction set [5] were announced in August 2016. These extensions were developed together with Fujitsu, which is planning to use it in a new generation of CPU for creating the Post-K supercomputer [6]. Intel has started working on integrating classical multi-core CPUs and ARM processors with FPGA segments on the same chip [7, 8].

The Sunway TaihuLight supercomputer [9], the current leader on the TOP500 list, consists of more than 10 million cores, an unprecedented degree of parallelism. At the same time, its architecture has a few features that dictate a need to follow a certain style for writing efficient programs [10], namely to maintain a very high ratio of arithmetic operations execution speed to data transmission speed from RAM (flops/bytes)... These facts clearly show how important it is to know the properties and features of parallel algorithm structures today and in the future. Obviously, the full utilization of future computer capacity would require redesigning and rewriting source code; the important thing, though, is that the basic properties of the algorithms will remain the same.

2 About the AlgoWiki project

The AlgoWiki Open Encyclopedia of Parallel Algorithmic Features [11] project has been running at Lomonosov Moscow State University since 2014. During this time, the project has drawn attention from the computing community [12, 15]. All computational algorithms are described in AlgoWiki using the same universal structure, with special emphasis on properties related to parallelism. The description consists of two primary parts. The first describes the machine-independent properties of an algorithm. The second part describes the properties of its specific implementations [13]. As the number of algorithms described in the AlgoWiki Encyclopedia rapidly grew, the issue of classification and distribution by thematic categories was raised shortly after the project was launched.

A logical extension to the AlgoWiki project was the specific algorithm analysis [14] with expert-quality reviews of various approaches to addressing individual applied problems. Each problem can usually be addressed with several different algorithmic approaches or methods. Each method has its own features, and those algorithms that fit well a specific class of computers may not always be suitable for another class. The AlgoWiki project is acquiring new dimensions that help researchers move from analyzing individual algorithms to analyzing various algorithmic approaches to tackling problems.

3 The basic concepts are: problem, method, algorithm and implementation

Algorithm description the basic unit in the AlgoWiki Open Encyclopedia of Algorithmic Features. The notion of an algorithm has been in existence for a long time, with the actual word originating from the name of al-Khwarizmi, an Arab scientist who lived between the 8th and 9th centuries [17]. Many definitions of the term "algorithm" exist, probably the most famous one being attributed to Donald Knuth [18]: an algorithm is "a finite set of rules that gives a sequence of operations for solving a specific type of problem," and it "has five important features:

- Finiteness...
- Definiteness...
- Input...
- Output...
- Effectiveness."

A common universal structure was proposed for describing in a standard form these and other important properties of various computational algorithms in the AlgoWiki project. According to D. Knuth, "[a]lgorithms are the threads that tie together most of the subfields of computer science." In the AlgoWiki Encyclopedia, algorithm descriptions also act as the central link in the chain connecting applied scientific problems with the results of supercomputer experiments.

At its initial stage of development, AlgoWiki was a simple list of algorithm descriptions. As the project expanded, the algorithms were divided into thematic subcategories, eventually developing into a form of algorithm classification. However, computational algorithms are not objectives in and of themselves: they are needed to solve *problems* faced in various areas of science and industry. For this reason, descriptions of the practical problems to be addressed were added to the AlgoWiki Encyclopedia. The problems can be described at various levels: from a specific practical problem being addressed (for example, "a general atmospheric circulation model") to mathematical formulations ("solving elliptical equations").

Many practical problems can be addressed using various *methods*: this warrants the introduction of another basic term that fits between the concepts of a problem and an algorithm. A problem can potentially be addressed via different methods (for example, elliptical equations can be solved using the direct method, Fourier transforms or iterative methods [19–21]). Each method has its advantages and may be preferred over others under certain conditions. These conditions can be dictated by the target software and hardware environment (e.g., iterative methods are preferred when solving elliptical equations on parallel computers with distributed memory). Some methods can also use other methods in sequence, which requires AlgoWiki to have several method levels.

Also, any algorithm can have different *implementations*, whether related to a single computing platform or to different platforms. For example, the following

parallel implementations are known for three-dimensional Fast Fourier Transform: FFTW (using MPI+OpenMP technologies) [22], MKL FFT (MPI) [23], AccFFT (MPI, CUDA) [24], etc. Within the algorithm description, AlgoWiki allows the properties of each specific implementation of the algorithm to be described, with visualization of the results obtained on various software and hardware platforms.

Thus, a *"problem–method–algorithm–implementation" chain* is being built as part of the AlgoWiki Open Encyclopedia of Algorithmic Features, which forms the basis for describing any subject area and follows the concept of linked representation of various algorithmic approaches to addressing one and the same problem. Effectively, the AlgoWiki project is acquiring new dimensions that help researchers move from analyzing individual algorithms to analyzing various algorithmic methods for addressing problems. Whether evaluating a definite integral, finding characteristic vectors, or searching for the minimum spanning tree of a graph, multiple algorithms can be offered to address the problem at hand, each having its own properties that can be key to efficiently implementing it on a specific computing system.

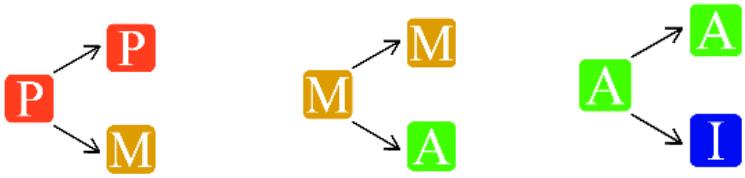
The classification of algorithms in the AlgoWiki project by their compatibility with specific supercomputer infrastructure will become the basis for comparing various algorithms to each other, which is needed for switching from analyzing individual algorithms to analyzing algorithmic methods for addressing problems. This markup makes it possible to compare the compliance of algorithms to the properties of a specific computer architecture, understand the advantages of each specific approach compared to others, compare the theoretical potential of various algorithmic approaches to the same problem, and draw a variety of other conclusions.

4 Interrelationships between basic concepts

4.1 From problem to method

The Problem (P) level is the most general of the basic concepts considered in AlgoWiki. Algorithms addressing problems from various areas of study can be described in the AlgoWiki Encyclopedia. From various areas in mathematics and more specific problems in computational physics, quantum chemistry and biomathematics, to practical issues in construction, design and all areas of science where computer simulation can be applied. More complex problems may require addressing additional problems (P–P link), for some of which solutions are already available (see Fig. 1a).

The way a problem is addressed can be called a Method (M); the resulting link is denoted as P–M (Fig. 1a). The description of a method usually contains a mathematical description of the way it addresses the problem, an explanation of its precision level, some considerations on the solution accuracy, the way to arrive at a solution and a justification for this being the only solution. Let us assume that a method is a mathematically justified sequence of actions by which one can follow to arrive at a solution to the original problem.



a) Possible connections from P level

b) Possible connections from M level

c) Possible connections from A level

Fig. 1. Possible connections between nodes and links in the "problem–method–algorithm–implementation" chain. We denote with "P" a problem, with "M" a method, with "A" an algorithm, and with "I" an implementation

As an example of the sub-chain being considered, we can describe a solution to a specific problem (P level), for instance, the analysis of the flow around an aircraft. The solution to gas dynamic equations in a complex-shaped area requires solving the subproblem (P) of construction of a computational grid in the given region. The latter problem is usually reduced to solving a set of linear elliptical equations (P), which requires performing the decomposition (P) of a coefficient matrix into triangular factors (P). Matrix decomposition can be done using the Cholesky factorization method (M level). Thus, the sequence considered above can be presented as follows: P–P–P–P–P–M. It should be noted that the last three nodes in this sub-chain are already described in AlgoWiki.

4.2 From method to algorithm

The method level (M) stands for a known solution to a certain (most commonly mathematical) problem. Methods usually operate with mathematical objects of a general nature (matrices, vectors, graphs, arrays, etc.). Very complex methods can include solving certain secondary subproblems using other known methods (M–M links; Fig. 1b). These contain a description of the general method for tackling a problem, frequently providing an opportunity to detail the method further.

The method is detailed by recording a fixed sequence of computations that result in the concept of an algorithm (A) and, consequently, an M–A link (Fig. 1b). One method can lead to a variety of algorithms with different computational properties. An algorithm implies a precise indication of the entire set of operations needed to implement the method and their sequence of operations. Even though a Gaussian elimination method for a linear system is not sensitive to the order of calculation of the partial sums in each line of the matrix, this order needs to be defined in the algorithm. At the same time, it should be noted that two algorithms related to the same Gaussian method but with a different order

of calculation of partial sums (ascending or descending order), besides difference in accuracy due to round-off errors, would have completely different properties: one is strictly serial, while the other has a good parallel implementation.

As an example, we can quote a sub-chain fully described in AlgoWiki. If matrix factoring is performed using Gaussian elimination by finding a triangular factorization (M), and its option LU-decomposition [25] without using transposition (M), with the compact Gaussian elimination method (M), or more specifically, the compact Gaussian elimination for tridiagonal matrices (M), then, for example, serial algorithm (A) for the compact factoring scheme described above can be considered. The chain above can be represented as follows: M–M–M–M–A.

4.3 From algorithm to implementation

The algorithm level (A) is the most branched structure in AlgoWiki, owing to the use of various clarifications of a specific method, or to the application of certain tricks to solve the problem faster. Some algorithms can engage many different operations, each being an independent algorithm (A–A link; Fig. 1c). For example, an algorithm (A) for solving a system of linear equations by the conjugate gradients method [21], in which every iteration uses auxiliary algorithms, is also described in AlgoWiki Encyclopedia: multiplication of a densely populated matrix by a vector (A), scalar multiplication of vectors (A), finding the vector norm (A), and vector operations (A) such as AXPY. This can result in a long chain of algorithms or trees of algorithms.

Once an algorithm is selected for solving a problem, a data structure should be developed. After the data structure is fixed, the most suitable programming language can be chosen and work can start on writing a specific implementation (I) of the chosen algorithm as a computer program (A–I link; Fig. 1c). It should be noted that the preliminary design and data structure development for complex algorithms may take a substantial amount of time, comparable to the time it takes to write the actual computer program.

As a result, each algorithm can be implemented using different programming languages. Many developers publish their implementations with detailed descriptions. This makes it possible to find either the necessary program or the information on how to write the most efficient program. Most pages in the AlgoWiki Encyclopedia, together with algorithm descriptions, contain reviews of existing implementations and direct links to open source codes.

4.4 From implementation to computations

When running the computations using the written program or a specific existing implementation, one should pay attention to the correct operation of the program; the most experienced developers design tests to debug the program. An important indicator of the quality of a program is its computational performance. This issue receives much attention in AlgoWiki Encyclopedia articles.

The computation locality and data usage locality are analyzed, criteria for assessing parallel efficiency are developed, and the results of the parallel computation are presented with a scalability analysis.

As a conclusion to the review of the various forms of the "problem–method–algorithm–implementation" chain, it should be noted that the least experienced developers make their first attempt to consider features of the computer architecture only during the very last stage, when analyzing the results of their first runs. However, the highest efficiency and scalability of the program can be achieved at the topmost levels, when choosing a method for solving the problem. One of the main goals for creating the AlgoWiki Open Encyclopedia of Algorithmic Features is to specifically give the user an opportunity well in advance to visualize the entire chain in full detail, from problem to implementation, and to choose the methods and algorithms that will produce the most efficient solutions.

4.5 Interconnection of sub-chains

In fact, different sub-chains may intersect or have many common nodes and links since each method can usually be applied to the solution of various problems. Short chains can be composed of longer ones. For example, all the sub-chains considered in Subsections 4.1–4.3 could be combined into one. Thus, the subtask of the solution of a system of linear elliptic equations (P) considered in Subsection 4.1 can be solved by using the algorithm (A) for the solution of linear systems by the conjugate gradients method (see Subsection 4.3), which will use as a preconditioner an incomplete triangular factorization, referred to in Subsection 4.2. Such long chains can also arise when solving other practical problems. Another example of a complete chain description is considered in the next section.

5 An example description of the "problem-method-algorithm-implementation" chain in the AlgoWiki Encyclopedia

Let us take a look at one of the full "problem-method-algorithm-implementation" chains already described in the AlgoWiki Encyclopedia — the chain that goes from the "Matrix factorization" group of problems to Householder's (reflection) method of QR factorization [26–29] of a square matrix, floating point variant (see Fig. 2).

Starting with the program (Matrix factorization), an AlgoWiki reader would see the "P" icon, indicating the problem level. In this case, however, it is not a single problem but a group of problems: Matrix factorization as a problem requires decomposing a matrix into a series of special matrices (unitary, triangular, etc.), depending on what is used in a higher-level problem (solving a system of linear equations, eigenvalue problems, etc.). Next, the reader sees that the detailed description of the problem raises another level with the same "P" sign, and again, these are not individual problems but groups of problems:

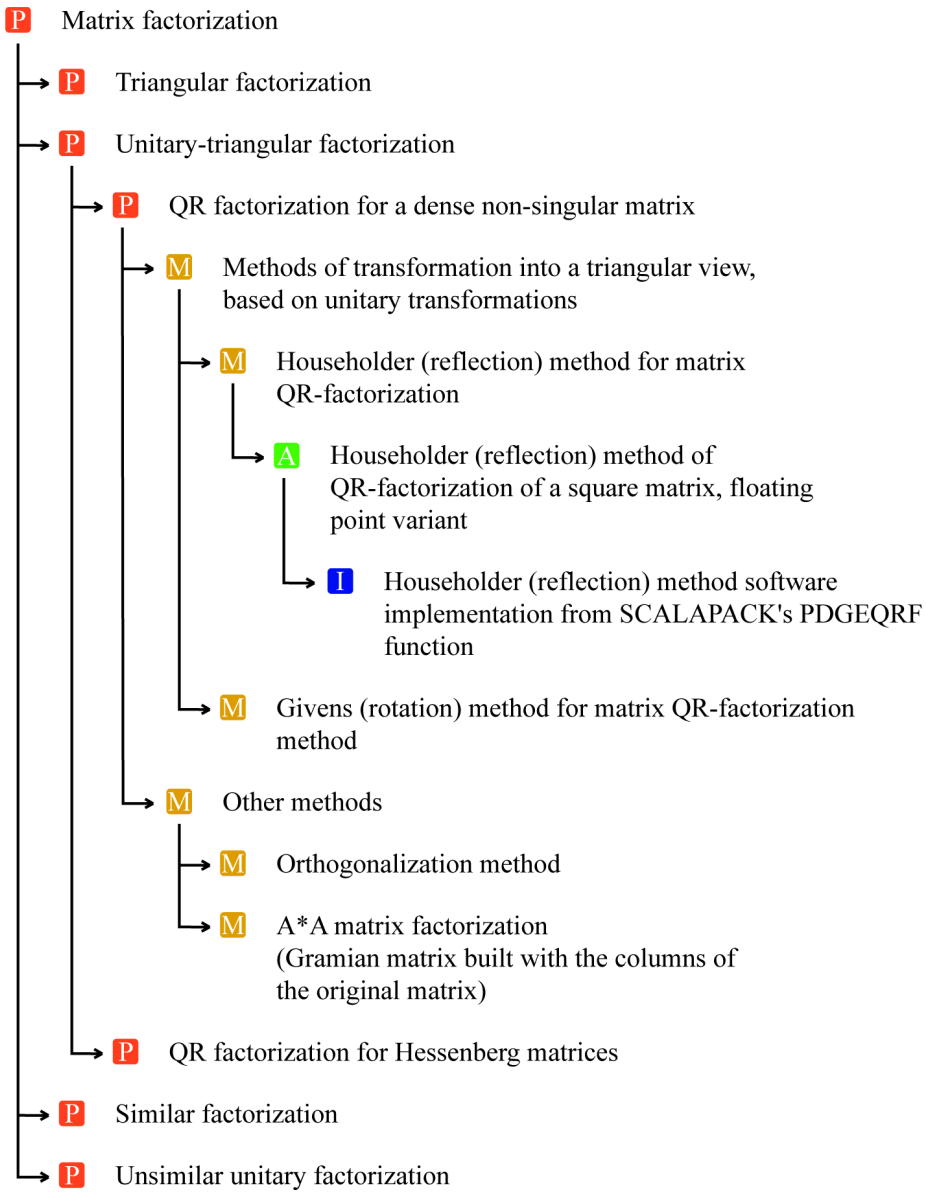


Fig. 2. Example of the full "problem–method–algorithm–implementation" chain

- "Triangular factorization": different versions of Gaussian decomposition, with or without transpositions, using the original matrix structures, up to compact schemes for tridiagonal matrices; a place is also prepared for precomputed factorization of known matrices.
- "Unitary-triangular factorization": the intermediary point we seek in our chain (we will look at it in more detail later).
- "Similar factorization": reduction of the matrix by means of a two-sided similarity transform to a Hessenberg form or tridiagonal symmetric matrix form, and spectral factorization.
- "Unsimilar unitary factorization": reduction of the matrix by means of an unbounded similarity transform to a two-diagonal form, and singular factorization.

Before proceeding with the chosen factorization group, note that the pages describing variations at the lower "algorithmic" level (e.g., Gaussian method with the selection of a leading element) are quite brief, but we can say in advance that no detailed description will be available for at least two of them since they are not used in practice and are only mentioned as a tribute to the history of the algorithm development.

In the chosen group of problems, "Unitary-triangular factorization," after moving a level lower, the reader finally gets to the individual problem level. There are two of them in the classification so far: QR factorization for a dense non-singular matrix and for Hessenberg matrices. The latter only has a short description at the method level, as the actual algorithm is not used in practice (factorization is not explicitly used in the QR algorithm but as iterations with implicit shifts). The LQ decomposition is also mentioned in the description at this level as a variation of QR factorization for a matrix transposed from the original matrix.

A more detailed description of the problem "QR factorization for dense non-singular matrix" is the last step in our chain, with the "P" icon. This is why, following the rules for describing the single-problem level, a brief list of solution methods is given along with a brief comparison of the above methods.

Digressing from the chosen chain towards the Householder method, one can note that different chains do not always end at the algorithm level. For example, the last method (using Gramian matrix factorization) does not have such descriptions. The reason is that, owing to a narrow application area (in addition to requiring a non-singular matrix, the A*A matrix conditioning is the square of the original matrix conditioning, so the error margin for this method is much greater), this algorithm has not been implemented by anyone.

Choosing the Householder method next, the user arrives at the method level ("M" icon on screen), specifically at the "Householder (reflection) method for matrix QR-factorization". Naturally, at this level the user is presented with just a basic mathematical concept for the method in question, as the actual algorithm is described in more detail at the algorithm-level page ("Householder (reflection) method of QR-factorization of a square matrix, floating point variant"). Other versions of the algorithm have not been described yet, but they are mentioned in

the method description: “In addition to the classical point version, the method has many other implementations such as the block version.”

Following the link to the algorithm, the user sees the “A” icon and a detailed description of all features for the chosen algorithm. Among other things, the general and mathematical descriptions of the algorithm contain not just general words about the Householder transformation, but also specific formulas for each step in the computation and for reducing the transformations to a sequence of scalar multiplications and weighted vector sums. These basic operations are parts of the computational core of the algorithm and are described in the respective section. A description of the algorithm macro structure shows why the main parts of scalar multiplications in the same step can be performed independently.

Looking at the implementation chart of the serial algorithm, the reader will see a description of the mathematical essence of the stages within one step. Serial complexity is presented as a formula for the number of floating point multiplications and additions/subtractions.

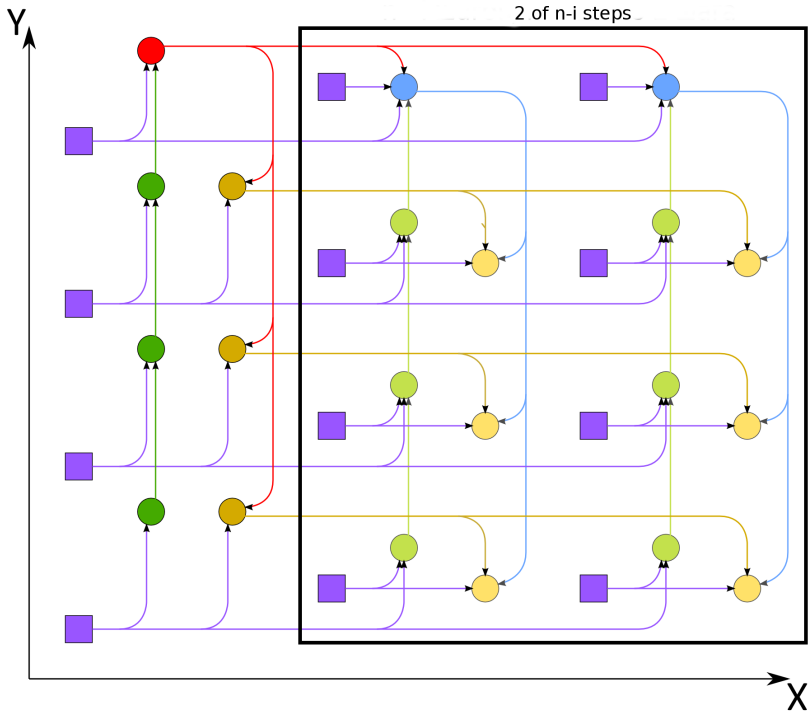


Fig. 3. Algorithm step graph (zeroing column i). Squares represent input data for this step (taken from the input data or from previous step), circles show operations. The outlined group of operations is repeated independently $n - i$ times

The information graph (see Fig. 3) is given for a single step as no parallelism is observed between individual steps. The parallelism resource is calculated assuming serial execution of scalar multiplication, but there are also assessments for other methods, namely the serial-parallel and pairing methods. Input and output data for the algorithm and its other properties are described further.

As one can clearly see, the correlation between serial and parallel complexities is *linear*, which provides a good incentive for paralleling the algorithm execution. However, the fastest level-parallel form of the graph has a *square-law* width, which indicates an imbalance between device loads in an attempt to actually program it. That is why it is more reasonable to keep the number of devices (e.g., cluster nodes) linear to the matrix size, even in the presence of a good (fast) communication network, which doubles the critical path for the level parallel form.

In this case, the algorithm's computational efficiency, expressed as the ratio of the number of operations to the total amount of input and output data, is *linear*.

The algorithm is fully determined within the selected version.

The computational error in the Householder (reflection) method grows in a *linear* manner, as well as in the Givens (rotations) method.

The next item after the description of the actual algorithm is the description of its software implementation from SCALAPACK's PDGEQRF function. The "Implementation peculiarities of the serial algorithm" section presents a fragment of the program in Fortran implementing the given algorithm.

The next session presents an analysis of the algorithm locality and its qualitative assessment.

Next, the user can see how the algorithm's dynamic characteristics change with different computation performance parameters for the same algorithm implementation.

Algorithm implementation can have relatively good performance, owing to the use of the SCALAPACK implementation of BLAS libraries, etc. Finally, at the end of the chain, the user can see some recommendations, particularly the advice to avoid using the classical version of the method and to use the block versions instead, for which numerous research works are available.

6 Conclusions

The paper considers the hierarchical approach used in the AlgoWiki Open Encyclopedia of Algorithmic Features to represent the structure of the subject area. A description is offered in the form of a "problem–method–algorithm–implementation" chain, which corresponds to numeric descriptions of the problems used in computational mathematics. In the paper, we show the interconnection between the basic concepts and analyze the descriptions of such chains obtained while working on additional content for the AlgoWiki Encyclopedia.

The proposed approach to the description of "problem–method–algorithm–implementation" chains is now being implemented in descriptions within the

AlgoWiki Open Encyclopedia of Algorithmic Features. This makes it possible to describe subject areas in which computational algorithms are used more clearly and according to a single scheme.

References

1. Patwary, Md.M.A., Satish, N.R., Sundaram, N., Park, J., Anderson, M.J., Vadlamudi, S.G., Das, D., Pudov, S.G., Pirogov, V.O., Dubey, P.: Parallel Efficient Sparse Matrix-Matrix Multiplication on Multicore Platforms. High Performance Computing. ISC High Performance 2015. Lecture Notes in Computer Science. Vol. 9137. Springer, Cham, pp. 48–57 (2015). DOI: 10.1007/978-3-319-20119-1_4
2. Solc, R., Kozhevnikov, A., Haidar, A., Tomov, S., Dongarra, J., Schulthess, T.C.: Efficient implementation of quantum materials simulations on distributed CPU-GPU systems. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC'15). ACM, New York, NY, USA, pp. 10:1–10:12 (2015). DOI: 10.1145/2807591.2807654
3. Alam, M., Khan, M., Vullikanti, A., Marathe, M.: An Efficient and Scalable Algorithmic Method for Generating Large-Scale Random Graphs. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE Press, Piscataway, NJ, USA, pp. 32:1–32:12 (2016). DOI: 10.1109/SC.2016.31
4. Dongarra, J., Beckman, P. , Moore, T., Aerts, P., Aloisio, G., Andre, J., Barkai, D., Berthou, J., Boku, T., Braunschweig, B. et al.: The international exascale software project roadmap. International Journal of High Performance Computing Applications, vol. 25, N 1, pp. 3–60 (2011). DOI: 10.1177/1094342010391989
5. Arm HPC tools for SVE. `https://developer.arm.com/products/software-development-tools/hpc/sve`
6. Post-K computer. `http://www.aics.riken.jp/en/postk/project`
7. Intel Eases Use of FPGA Acceleration: Combines Platforms, Software Stack and Ecosystem Solutions to Maximize Performance and Lower Data Center Costs. `https://newsroom.intel.com/news/intel-eases-use-fpga-acceleration-combines-platforms-software-stack-//ecosystem-solutions/`
8. Intel Enables 5G, NFV and Data Centers with High-Performance, High-Density ARM-based Intel Stratix 10 FPGA. `https://newsroom.intel.com/news/intel-enables-5g-nfv-data-centers-high-performance-high-density-arm-//based-intel-stratix-10-fpga/`
9. Fu, H., Liao, J., Yang, J., Wang, L., Song, Z., Huang, X., Yang, C., Xue, W., Liu, F., Qiao, F., Zhao, W., Yin, X., Hou, C., Zhang, C., Ge, W., Zhang, J., Wang, Y., Zhou, C., Yang, G.: The Sunway TaihuLight supercomputer: system and applications. Science China Information Sciences, vol. 59, issue 7 (2016). DOI: 10.1007/s11432-016-5588-7
10. Zhang, J., Zhou, C., Wang, Y., Ju, L., Du, Q., Chi, X., Xu, D., Chen, D., Liu, Y., Liu, Z.: Extreme-Scale Phase Field Simulations of Coarsening Dynamics on the Sunway TaihuLight Supercomputer. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC'16). IEEE Press, Piscataway, NJ, USA, pp. 4:1–4:12 (2016). DOI: 10.1109/SC.2016.3
11. Open Encyclopedia of Parallel Algorithmic Features. `https://algowiki-project.org/en`

12. Voevodin, Vl., Antonov, A., Dongarra, J..: AlgoWiki: an Open Encyclopedia of Parallel Algorithmic Features. Supercomputing Frontiers and Innovations, Vol. 2, N 1, pp. 4–18 (2015). DOI: 10.14529/jsfi150101
13. Antonov, A., Voevodin, Vad., Voevodin, Vl., Teplov, A.: A Study of the Dynamic Characteristics of Software Implementation as an Essential Part for a Universal Description of Algorithm Properties. In: 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing Proceedings, 17th-19th February 2016, pp. 359–363 (2016). DOI: 10.1109/PDP.2016.24
14. Antonov, A., Frolov, A., Kobayashi, H., Konshin, I., Teplov, A., Voevodin, Vad., Voevodin, Vl.: Parallel Processing Model for Cholesky Decomposition Algorithm in AlgoWiki Project. Supercomputing Frontiers and Innovations, Vol. 3, N 3, pp. 61–70 (2016). DOI: 10.14529/jsfi160307
15. Voevodin, Vl., Antonov, A., Dongarra, J.: Why is it hard to describe properties of algorithms? Procedia Computer Science, Vol. 101, pp. 4–7 (2016). DOI: 10.1016/j.procs.2016.11.002
16. Antonov, A.S., Volkov, N.I.: An AlgoView Web-visualization System for the AlgoWiki Project. Communications in Computer and Information Science. Vol. 753, pp. 3–13 (2017). DOI: 10.1007/978-3-319-67035-5_1
17. Boyer, C.B.: The Arabic Hegemony. A History of Mathematics (Second ed.). John Wiley & Sons, Inc. (1991).
18. Knuth, D.: The Art of Computer Programming. Vol. 1. Fundamental Algorithms. Third Edition. Reading, Massachusetts: Addison-Wesley, 650 pp. (1997).
19. Gloukhov, V.: Parallel implementation of the INM atmospheric general circulation model on distributed memory multiprocessors. Lecture Notes in Computer Science. Vol. 2329, pp. 752–762 (2002) DOI: 10.1007/3-540-46043-8_76
20. Hess, R., Joppich, W.: A comparison of parallel multigrid and a fast Fourier transform algorithm for the solution of the Helmholtz equation in numerical weather prediction. Parallel Computing. Vol. 22, pp. 1503–1512 (1997).
21. Saad, Y.: Iterative Methods for Sparse Linear Systems. SIAM, Second edition, 520 p. (2003). DOI: 10.1137/1.9780898718003
22. FFTW Home Page. `http://www.fftw.org`
23. Computing Cluster FFT. `https://software.intel.com/node/521992`
24. AccFFT. A New Parallel FFT Library. `http://accfft.org`
25. Davis, T.A.: Direct Methods for Sparse Linear Systems. SIAM, 215 p. (2006). DOI: 10.1137/1.9780898718881
26. Golub, G., Van Loan, C.F.: Matrix Computations. Johns Hopkins University Press, Baltimore, MD, third ed. (1996).
27. Chu, E., George, A.: QR Factorization of a Dense Matrix on a Hypercube Multiprocessor. SIAM Journal on Scientific and Statistical Computing, 11, pp. 990–1028 (1990). DOI: 10.1137/0911057
28. Paige, C.: Some aspects of generalized QR factorization, in Reliable Numerical Computations. M. Cox and S. Hammarling, eds., Clarendon Press (1990).
29. Dongarra, J.J., D'Azevedo, E.F.: The design and implementation of the parallel out-of-core ScaLAPACK LU, QR, and Cholesky factorization routines. Department of Computer Science Technical Report CS-97-347, University of Tennessee, Knoxville, TN (1997).
30. Antonov, A., Teplov, A.: Generalized Approach to Scalability Analysis of Parallel Applications. Lecture Notes in Computer Science. Vol. 10049, pp. 291–304 (2016). DOI: 10.1007/978-3-319-49956-7_23