Studying the Structure of Parallel Algorithms as a Key Element of High-Performance Computing Education

Vladimir Voevodin, Alexander Antonov(✉), and Nina Popova

Lomonosov Moscow State University, Moscow, Russia
{voevodin,asa}@parallel.ru, popova@cs.msu.su

Abstract. Since the computing world has become fully parallel, every software developer today should be familiar with the notion of "parallel algorithm structure." If in recent years, students have studied a basic introduction to algorithms; today, parallel algorithm structure must become a vital part of computer science education. In this work we present two years of experience teaching a "Supercomputer Modeling and Technologies" course, and running practical assignments at the Computational Mathematics and Cybernetics faculty of Lomonosov Moscow State University, aimed at teaching students a methodology for analyzing parallel algorithm properties.

Keywords: Structure of parallel algorithms
High-performance computing education · Parallel programming
Educational curricula · Computer science curricula
Undergraduate students

1 Introduction

Today, computing technologies are used in all areas of science, industry and economics, which imposes strict requirements on higher education systems training computer science specialists in all countries. One recent example is India's "National Supercomputing Mission" [1], during which the government set a 7-year target for training 20,000 specialists in the area of parallel and distributed computer technologies. The demand for actively developing education in the areas of computational sciences, high-performance computing, and mathematical modeling using supercomputers is evidenced throughout the entire global educational community [2–5].

The results described in Sects. 4, 5 were obtained in Lomonosov Moscow State University with the financial support of the Russian Science Foundation (Agreement № 14–11–00190). The research is carried out using the equipment of the shared research facilities of HPC computing resources at Lomonosov Moscow State University supported by the project RFMEFI62117X0011.

G. Mencagli et al. (Eds.): Euro-Par 2018 Workshops, LNCS 11339, pp. 199–210, 2019.
https://doi.org/10.1007/978-3-030-10549-5_16

A number of major national and international projects can be noted that offer recommendations for developing training materials in these areas [6–10]. Interesting results have been discussed at international seminars dedicated to the issue: EduHPC, EduPAR, Euro-EduPAR [11]. Books have been published that are entirely dedicated to the best materials and pedagogical practices in the area of PDC [12]. These activities are fueling growth in the number of educational courses and programs, with the mathematical modeling industry engaging specialists from various applied areas. This is further promoted by the emergence of new areas where high-performance computing is in high demand. The most recent examples where industries received a development boost thanks to HPC technologies are deep learning, artificial intelligence, and big data analytics.

This work consists of 5 sections. In Sects. 2 and 3 we give a brief overview of supercomputer education at the Computational Mathematics and Cybernetics faculty of Lomonosov Moscow State University, and describe how the "Supercomputer Modeling and Technologies" course is organized. Sections 4 and 5 describe two versions of practical assignments that are part of the course, which allow for two different perspectives on studying the structure of algorithms. From our point of view, focusing on the study of parallel algorithm structure in the form presented is a new approach, representing the key content of this work. Section 6 contains recommendations and conclusions based on two years of experience from teaching this course and conducting practical assignments in the form presented.

This approach to studying the structure of parallel algorithms is in line with existing proposals on the content of educational curricula, e.g., Computer Science Curricula [13], NSF/IEEE-TCPP Curriculum Initiative on Parallel and Distributed Computing [14], and can be used at many universities.

2 Supercomputer Education at MSU

Lomonosov Moscow State University provides a good basis for supercomputer education. MSU's supercomputer center is currently the most powerful in Russia. It is centered around the Lomonosov-2 (4.9 Petaflops) and Lomonosov (1.7 Petaflops) supercomputers and IBM Blue Gene/P (28 Teraflops). The basics of parallel computations are taught at several faculties within MSU: Computational Mathematics and Cybernetics, Mechanics and Mathematics, Physics, Chemistry, Bioengineering and Bioinformatics, and a few others.

CMC faculty is a leading educational center in Russia offering specialist training that combines applied mathematics, computational technologies and information science. It has about 2000 full-time students, with about 200 PhD candidates. Training at the faculty is provided as part of an integrated Master's degree program: for their first four years, students study within a Bachelor's degree program, followed by two years in one of twenty-two available Master's degree programs. This dual-level system offers basic fundamental training for students during years 1–4, with deeper specialization as a part of the Master's degree programs. Studying supercomputer-related disciplines is mandatory for all students at the faculty.

3 The Supercomputer Modeling and Technologies Discipline

The “Supercomputer Modeling and Technologies” course is the only general lecture course for all 22 Master’s degree programs at CMC faculty. It is taught to second-year Master’s degree students with a total of about 240 students taking the course. All students are expected to have basic knowledge of mathematical modeling, parallel computing systems and supercomputer architecture, and the basics of parallel computing.

The discipline totals 7 credits. The course consists of a lecture module, seminars and several practical assignments. Lectures are conducted for two academic hours a week from September to November. Two hours of seminars are also conducted each week. The seminars are used to discuss problem definitions and implementation details, to offer consultations on assignments, and for students to present reports on the assignments they have completed.

The course lasts for one semester, and two full cycles have been completed to date in the fall semesters of 2016 and 2017. Teachers from the various departments at the CMC faculty are invited to take part in delivering lectures, along with representatives from leading IT companies.

Students are offered three practical assignments as part of the course, of which two are mandatory. The first assignment requires studying and describing the structure and properties of parallel algorithms. The second assignment involves implementing a parallel algorithm to solve a three-dimensional hyperbolic equation using MPI and OpenMP. The third assignment is given by the lecturers in specific subject areas, and students can choose which lecturer’s assignment they would like to perform. While the second and third assignments are classical assignments that directed to parallel implementations; the first one requires additional clarifications and is central to this work: Sects. 4, 5 and 6 of this article describe two options for the first assignment that are aimed at studying the properties of parallel algorithms.

The first assignment is indeed unusual and non-trivial, so students were allowed to work in pairs. Any parallel computers could be chosen as the target computing platform. By default, all students were provided access to MSU’s supercomputers: Lomonosov [15] and IBM Blue Gene/P. Some students were granted access to the Lomonosov-2 supercomputer [16], clusters with Intel Xeon Phi (KNL) and/or NVIDIA P100 processors, clusters with the new “Angara” interconnect and several others. This enabled comparison of results across different processors (multicore/manycore Intel, NVIDIA GPU, IBM PowerPC), and various communication networks (InfiniBand, proprietary) with different network topologies (fat tree, 3-dimensional torus, flattened butterfly).

The results verification form is an important part of the assignment. The goal was not just to grade the work, but to make sure that students completed their assignments at a good level of quality. In fact, the idea was to teach students to find a proper approach to this kind of assignment. Instead of immediately grading the work, the tutors formulated their comments; the students would incorporate the feedback and send the assignment results back for further review.

This interaction was repeated as necessary, usually limited only by the deadline for grading course results at the end of the term. This is very different from the traditional assignment grading process. The purpose is not so much about making sure the course materials are absorbed correctly. Rather what is most important is to teach students an effective approach to analyzing algorithm properties in a proper and high-quality manner. This requires tutors to have a much higher level of qualification and to dedicate more time to the assessment, but the final results are comparable in quality with the best teaching practices.

4 Version of the Practical Assignment: Description of Parallel Algorithm Properties

Assignment: *Describe the structure and properties of the chosen algorithm.*

While this wording sounds extremely simple, it masks a number of small but important nuances. It is also important to note that in order to successfully complete this assignment, students need to use knowledge previously obtained over various disciplines at the faculty.

4.1 Methodological Comments on the Assignment

What does it mean to describe an algorithm's structure and properties? This is not a simple question, as there is no universally recognized standard specifying which properties of an algorithm are important and exactly how they must be described. The students were offered the algorithm description structure used by the AlgoWiki Open Encyclopedia of Algorithm Properties [17,18]. This description structure was developed as a universal one that can be applied to any algorithm, giving particular emphasis to the properties related to parallelism.

Some sections of the AlgoWiki description were left out of this assignment due to their complexity (for example, sections describing the data locality or dynamic characteristics of an algorithm's implementation). Ultimately, the following structure was recommended for students to use in their descriptions of the algorithm's properties:

1. General description of the algorithm.
2. Mathematical description of the algorithm.
3. Computational kernel of the algorithm.
4. Macro structure of the algorithm.
5. Implementation scheme of the serial algorithm.
6. Serial complexity of the algorithm.
7. Information graph.
8. Parallelism resource of the algorithm.
9. Input and output data of the algorithm.
10. Properties of the algorithm.
11. Scalability of the algorithm and its implementation.
12. Existing implementations of the algorithm.
13. References.

Notably, a number of examples are available in AlgoWiki for each item, which helped students to complete the assignment. The first ten items in the description require studying the algorithm's theoretical properties, while items 11 and 12 are oriented towards studying the properties of its specific implementations. The main focus of the assignment was not on the actual algorithm description (this part could simply be taken from textbooks), but on studying its properties — primarily the algorithm's information structure and parallelism resource. These properties are rarely described in the literature, so this part of the assignment required conducting independent research.

The central task in describing the algorithm properties would be to build and analyze an information graph (Item 7 of the above structure) [19,20]. Figure 1 shows the example information graph of the Cooley-Tukey algorithm with input and output data.

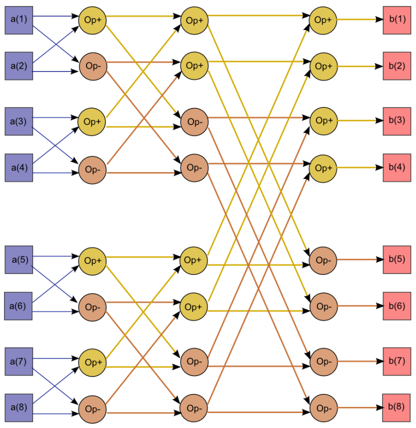

Fig. 1. The information graph of the Cooley-Tukey algorithm for n = 8. Op+ denotes the addition of two complex numbers, while Op- denotes the subtraction of two complex numbers followed by multiplying the result by another complex number (a twiddle factor). The edges correspond to the transmission of data between the vertices.

An information graph is vital for studying algorithm properties, as it contains all the necessary information about its parallel structure. The skills for working with an information graph are also very important in practice, as they help to evaluate an algorithm's parallel complexity and an application's parallelism resource, to understand the algorithm's bottlenecks and to find different options for parallel implementation. This is why special attention was paid to information graph analysis — both when formulating the student assignments and when checking the completed work.

To prepare a specific description, the students were asked to choose one of 30 preselected algorithms, specifically: Jacobi's method for the singular value decomposition, Gram-Schmidt orthogonalization process, fast discrete Fourier transform, and others. These algorithms are all certainly different in complexity. However, the essence of the assignment was not about developing algorithms, nor even about implementing them, so the complexity of the algorithm itself didn't affect the complexity of the assignment so much.

4.2 Organization and Results of the First Practical Assignment

This version of the assignment was completed by 246 Master's degree students in 2016, during their second year of education. As a result, each of the 30 proposed algorithms was described by 4–5 groups of 1–2 students each. The students could use any literature or online sources in their algorithm descriptions, as long as they were appropriately cited. Moreover, when the assignment was distributed, each algorithm was accompanied by references to well-known sources that explain the algorithm. This addressed two issues at once: the students would get a reliable source of information, and both the student and the tutor would be guaranteed to have an unequivocal understanding of which specific algorithm was to be described.

Due to the volume of the work produced, the resulting descriptions were verified in two independent stages. The first stage involved a purely formal verification of the descriptions for compliance with the requirements. This included checking for the presence of all relevant description sections, the clarity of the formulas, the information content of any drawings used, the inclusion of all parameters and conditions under which the algorithm properties were studied, references to sources, etc. The content of the algorithm descriptions was not checked at this stage, to reduce the requirements for inspector qualifications and the time needed to perform the verification. The second stage of the verification required a review of content. The algorithm description was checked for accuracy, the proper definition and description of its properties, proper formulas and the accuracy of the results. These checks required much more time and substantially higher tutor qualifications.

The technical evaluation was successfully supported by useful features of AlgoWiki, based on MediaWiki technology. The students prepared algorithm descriptions in their personal spaces and interacted with tutors using a built-in collaboration mechanism. This facilitated communication student — tutor, in addition to tracking every stage of the assignment, including any changes made in the descriptions, tutor comments, date of response to tutor feedback, etc.

The final grades of the 146 groups comprising the 246 students were distributed as follows: 59 works received a 5 (Excellent) grade; 36 were graded at 4 (Good), 48 received a 3 (Satisfactory) grade, and three works were evaluated as 2 (Unsatisfactory). Thus, the average grade for the assignment was 4.03, which indicates a generally high level of description quality given the complexity and novelty of the assignment.

Remarkably, some of the student works were completed with such a high quality level that they were included in the AlgoWiki Encyclopedia. In some cases, the students went beyond the assignment formulation, conducting additional studies of other issues related to parallelism. Moreover, some of the students became so engaged in studying the selected algorithms that they continued enhancing their results even after the semester ended.

5 Version of the Practical Assignment: Studying Algorithm Scalability

Assignment: *Studying the scalability of algorithms and their implementations on various computing platforms when changing the size of the problem and the number of processors available.*

5.1 Methodological Comments on the Assignment

When performing the second version of the assignment as part of the "Supercomputer Modeling and Technologies" course, students needed to perform a series of computational experiments, collect the relevant data, interpret it correctly, then draw a conclusion on the algorithm's level of scalability. Additionally, they needed to determine from the data obtained, which combination of problem size and the quantity of processors maximized performance.

Graph algorithms were chosen as the subject of study in 2017. Five key problems were considered: Single Source Shortest Path, Breadth-First Search, Page Rank, Minimum Spanning Tree, and Strongly Connected Components.

The students could choose one of several available algorithms for each problem. For example, the options for the "Single Source Shortest Path" problem were the Bellman-Ford, Dijkstra's and Delta-Stepping algorithms. Since the objective of this assignment was not to study parallel programming technologies, up to 5 different ready-made implementations were offered for each algorithm, which were to be used in computational experiments on the chosen computer platform. As a result, each student chose a unique combination within which scalability [21] was to be examined:

Problem → Algorithm → Implementation → Computing platform.

Computer performance for these algorithms is frequently measured in TEPS (Traversed Edges Per Second), indicating the number of graph edges the computer can pass (process) in one second using a given implementation of a given algorithm. This parameter was used in our assignment to assess performance.

Special attention was paid to processing large graphs, which are common in practical applications: social networks, road maps, chemical compounds and many other real-life objects are described using graphs with millions and billions of vertices and edges. At the same time, as the graph size increases, its implementation performance can drop substantially, as data no longer fit in cache memory at different levels; hence the interest in carefully measuring the dependence between the size of the problem, number of processors and performance.

Another important issue is that dynamic characteristics of graph algorithms can change significantly with changes in the structure and properties of the graphs being processed. For this reason, each student needed to study scalability for two types of graphs: RMAT and SSCA2. These are synthetic graphs reflecting different properties of real-life graphs: RMAT graphs are suitable for modeling the structure of social networks [22], while SSCA2 graphs are good for describing a set of interconnected communities [23]. To obtain input RMAT/SSCA2 graphs of an arbitrary size, students were provided ready-made parallel generators.

As a result, the assignment for each student was formulated as follows: for each of the two graph types: RMAT and SSCA2, within the chosen combination "Problem → Algorithm → Implementation → Computing platform," it was required to:

– build a chart showing the dependence (MTEPS) on the number of processors (or threads) used and the graph size;
– find the combination of processor number and graph size that maximizes performance.

Since the assignment was focused on analyzing large graphs, the maximum performance point was to be calculated only for those problem sizes where the graph did not fit entirely within cache memory. Figure 2(a) shows the dependence experimentally determined by one of the students for the Breadth First Search algorithm without considering this requirement, where maximum performance is achieved on a small graph of 2^{12} vertices.

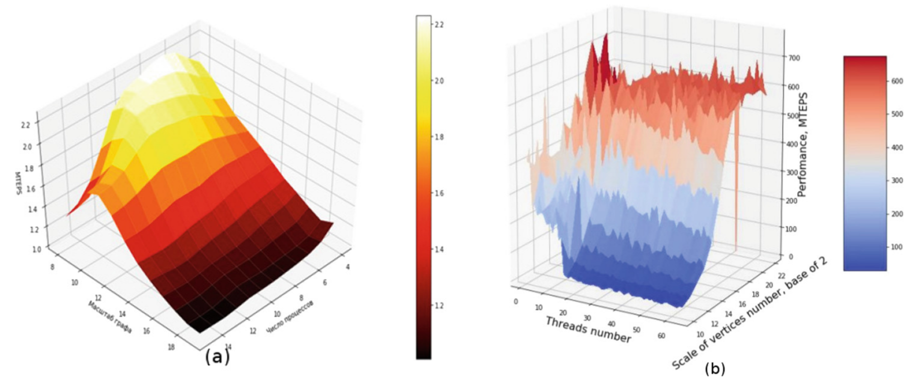

Fig. 2. Particular aspects of the assignment: (a) impact of cache memory on maximum performance value, (b) value fluctuations in the absence of multiple runs

One has to note the substantial computing resources needed to perform the assignment properly. The assignment required the programs to be run multiple times: the performance values were to be assessed for different graph sizes and different processor numbers, for each of the two graph types (RMAT and SSCA2). Moreover, performance values on nearly every computer would change from one run to another, so ideally several experiments needed to be conducted and the maximum value chosen, otherwise the resulting chart would contain obvious artifacts, like the example shown in Fig. 2(b).

To motivate students to conduct a more thorough analysis of the scalability figures obtained, it is useful to show the results obtained by other students using other algorithms, other implementations and other computers (an example is shown in Table 1). When comparing their maximum performance figures to other results, students begin to ask the question: “Why am I doing worse?” Finding the answer would require analyzing the entire chain “Problem → Algorithm → Implementation → Computing platform,” which helps students realize the need for a comprehensive approach for studying scalability.

Table 1. Comparison of maximum performance for different algorithms solving the “Single Source Shortest Path” problem using different implementations on different platforms.

Algorithm	Implementation	Computing Platform	MTEPS	GraphType	GraphSize
Bellman-Ford	RCC for GPU	Lomonosov	1309.0	SSCA2	2^{20}
Bellman-Ford	Ligra	Lomonosov-2	1035.0	RMAT	2^{21}
Delta Stepping	PBGL MPI	Cluster/“Angara”	809.5	SSCA2	2^{21}
Delta Stepping	GAP	Lomonosov-2	616.0	RMAT	2^{21}
Bellman-Ford	RCC for CPU	Lomonosov	435.0	SSCA2	2^{21}
Bellman-Ford	RCC for CPU	Lomonosov-2	426.0	RMAT	2^{21}
Bellman-Ford	Graph500 MPI	Lomonosov	350.0	RMAT	2^{20}
Dijkstra’s	PBGL MPI	IBM BlueGene/P	8.9	SSCA2	2^{20}
Dijkstra’s	PBGL MPI	Lomonosov	5.3	SSCA2	2^{21}

5.2 Organization and Results of the First Practical Assignment

This version of the first practical assignment was performed in 2017 by 143 groups of Master’s degree students and the grades were distributed as follows: 121 works received a 5 (Excellent) grade; 15 were graded at 4 (Good), 5 received a 3 (Satisfactory) grade, and two works were evaluated as 2 (Unsatisfactory). The average grade for scalability description was 4.78. This is much higher than the average for the 2016 assignment (4.03), which is not surprising: the scalability study assignment was simpler and more familiar than the task of studying and describing algorithm properties. In addition, when students described algorithm properties in 2016, they had to present, among other things, their considerations for algorithm’s scalability, as this was required in the description structure (Item 11). At the same time, the assignment form chosen in 2017, turned out successfully in a different way. By combining a simple assignment statement with the need to interpret the obtained data, we achieve our goal: students begin to think not just about the scalability analysis methodology and the notions of weak and strong scalability, but also to learn the techniques for studying parallel program scalability in practice. Moreover, when analyzing scalability data, students recognize the need for the joint (and specifically joint) study of the various algorithm properties, implementations and computing platforms.

6 Lessons Learnt from the 2-Year Experience

Let's look at some important issues that one must keep in mind when using similar assignments in the future. Some of them we chose to point out to students at the very beginning, when distributing the assignments; others were faced during the course of work, causing difficulties for students or tutors.

When describing algorithm properties, it is important to realize that not just floating point arithmetic matters, but also read-write memory operations which determine the execution time for many algorithms. In particular, it is necessary to describe the computational core and sequential complexity of algorithms.

When defining the information structure of an algorithm, it is important to define a level of details for operations. Otherwise, the students could produce a linear graph of 3–4 vertices reflecting the sequential stages of the algorithm: while this isn't necessarily wrong, it clearly isn't very informative either.

Some algorithms are based on other, simpler algorithms. In these cases, it was advisable to use "macro-operations" that corresponded to simpler algorithms, as they could be more traditional, expressive and clear for describing and understanding the structure of the original algorithms.

The information structure of the algorithms could be expressed in different ways, with no set standards. However, the students were best off using a system of axes related to the loop nesting structure: in that case, the information graph reflects the computation structure used in the program and is more intuitive.

Describing an algorithm's potential parallelism is challenging for the students. This is not a habitual notion, and students don't always immediately learn to look at the algorithm structure in general. Methodological materials need to be developed that contain sample descriptions of potential parallelism, clearly showing what kind of results is expected from the students.

When studying scalability, it is important to draw the students' attention to explain all of the feature points on the performance charts: peaks, inflection points, asymptote starting points, etc. Detailed analysis is not simple, but if peculiarities consistently repeat between runs, then there must be an explanation.

For self-written programs independently, the students needed to clarify the testing technology for the program implementing the given algorithm. This is important, as the results presented will otherwise not be trustworthy.

7 Conclusion

Overall, we consider the two-year track record in teaching the "Supercomputer Modeling and Technologies" course along with the practical assignments to be a highly positive experience. A spacious inter-disciplinary approach to problems under study, supported by specific practical assignments on actual supercomputers, creates a solid foundation for using the knowledge gained in further professional activities. Indeed, obtaining good results required serious efforts from both students and professors. But it was worth it! Students must use knowledge and skills from previous courses, which is a good way to bring them closer to completing their Master's degree studies.

Practical assignments can easily be adapted to the specific environment of another faculty by using a different set of algorithms, placing particular emphasis on studying different algorithm properties, analyzing their own implementations, studying existing source codes, focusing on the scalability of a specific computing platform, and many other aspects.

The courses implemented in 2016 and 2017 are considered to be a pilot program. Given the positive results achieved, we are planning to modify the topics covered during the lecture part of the course, including areas such as supercomputer climate modeling, high-performance image processing methods, deep learning, and big data analytics.

Acknowledgments. We are sincerely grateful to our colleagues form the Faculty of Computational Mathematics and Cybernetics and the Research Computing Center who helped us to deliver the lectures and organize the practical assignments—completing the educational program in this form without their help would simply have been impossible.

References

1. National Supercomputing Mission. https://www.nsmindia.in/. Accessed 4 May 2018
2. Secretary of Energy Advisory Board. Report of the Task Force on Next Generation High Performance Computing, U.S. Department of Energy. August 18, 2014. https://www.energy.gov/sites/prod/files/2014/10/f18/SEAB%20HPC%20Task%20Force%20Final%20Report.pdf. Accessed 4 May 2018
3. Ezell, S.J., Atkinson, R.D.: The Vital Importance of High-Performance Computing to U.S. Competitiveness. http://www2.itif.org/2016-high-performance-computing.pdf. Accessed 4 May 2018
4. SIAM Working Group on CSE Education. SIAM: Graduate Education for Computational Science and Engineering (2014). http://www.siam.org/students/resources/report.php. Accessed 4 May 2018
5. Chapman, B., et al.: DOE: assessment of workforce development needs in office of science research disciplines. http://science.energy.gov/~/media/ascr/ascac/pdf/charges/ASCAC_Workforce_Letter_Report.pdf. Accessed 4 May 2018
6. Dongarra, J., et al.: Applied Mathematics Research for Exascale Computing, US - DOE Report, March 2014. https://science.energy.gov/~/media/ascr/pdf/research/am/docs/EMWGreport.pdf. Accessed 4 May 2018
7. Future Directions in CSE Education and Research. Report from a Workshop Sponsored by the Society for Industrial and Applied Mathematics (SIAM) and the European Exascale Software Initiative (EESI-2). http://wiki.siam.org/siag-cse/images/siag-cse/f/ff/CSE-report-draft-Mar2015.pdf. Accessed 4 May 2018
8. Voevodin, V., Gergel, V., Popova, N.: Challenges of a systematic approach to parallel computing and supercomputing education. In: Hunold, S., et al. (eds.) Euro-Par 2015. LNCS, vol. 9523, pp. 90–101. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-27308-2_8
9. Supercomputing Education in Russia. Final report on the national project “Supercomputing Education”, Supercomputing Consortium of the Russian Universities (2012). http://hpc.msu.ru/files/HPC-Education-in-Russia.pdf. Accessed 4 May 2018

10. Voevodin, Vl.V., Gergel, V.P.: Supercomputing education: the third pillar of HPC. In: Computational Methods and Software Development: New Computational Technologies, vol. 11, no. 2, pp. 117–122. Moscow State University Press, Moscow (2010)
11. 3rd European Workshop on Parallel and Distributed Computing Education for Undergraduate Students (Euro-EDUPAR). http://www.cs.man.ac.uk/~rizos/euroedupar/. Accessed 4 May 2018
12. Prasad, S.K., Gupta, A., Rosenberg, A.L., Sussman, A., Weems Jr., C.C. (eds.): Topics in Parallel and Distributed Computing: Introducing Concurrency in Undergraduate Courses. Morgan Kaufmann, San Francisco (2015)
13. Computer Science Curricula 2013. https://www.acm.org/binaries/content/assets/education/cs2013_web_final.pdf. Accessed 4 May 2018
14. NSF/IEEE-TCPP Curriculum Initiative on Parallel and Distributed Computing. http://www.cs.gsu.edu/~tcpp/curriculum. Accessed 4 May 2018
15. Sadovnichy, V., Tikhonravov, A., Voevodin, Vl., and Opanasenko, V.: "Lomonosov": supercomputing at Moscow State University. In: Contemporary High Performance Computing: From Petascale Toward Exascale, pp. 283–307. Chapman & Hall/CRC Computational Science), CRC Press, Boca Raton (2013)
16. MSU Supercomputers: "Lomonosov-2". http://hpc.msu.ru/?q=node/159. Accessed 4 May 2018
17. Open Encyclopedia of Parallel Algorithmic Features. http://algowiki-project.org/en. Accessed 4 May 2018
18. Antonov, A., Voevodin, V., Dongarra, J.: Algowiki: an open encyclopedia of parallel algorithmic features. J. Supercomput. Front. Innov. **2**(1), 4–18 (2015)
19. Voevodin, V.: Mathematical Foundations of Parallel Computing. Series in Computer Science, vol. 33. World Scientific Publishing Co. (1992)
20. Voevodin, V., Voevodin, Vl.: Parallel Computing. BHV-Petersburg, St. Petersburg (2002)
21. Antonov, A., Teplov, A.: Generalized approach to scalability analysis of parallel applications. In: Carretero, J., et al. (eds.) ICA3PP 2016. LNCS, vol. 10049, pp. 291–304. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-49956-7_23
22. Chakrabarti, D., Zhan, Y., Faloutsos, C.: R-MAT: a recursive model for graph mining. In: Proceedings of 4th International Conference on Data Mining, Brighton, UK, pp. 442–446 (2004). https://doi.org/10.1137/1.9781611972740.43
23. Bader, D.A., Madduri, K.: Design and implementation of the HPCS graph analysis benchmark on symmetric multiprocessors. In: Bader, D.A., Parashar, M., Sridhar, V., Prasanna, V.K. (eds.) HiPC 2005. LNCS, vol. 3769, pp. 465–476. Springer, Heidelberg (2005). https://doi.org/10.1007/11602569_48