DOI: 10.17323/2587-814X.2025.2.77.88

Parallel implementation of the simplex method in matrix form using the PyTorch library for economics and management problems

Yuriy S. Ezrokh [a]
E-mail: ezrokh@corp.nstu.ru

Alexey V. Snytnikov [a,b]
E-mail: aleksej.snytnikov@klgtu.ru

Elena Yu. Skorobogatykh [b]
E-mail: elena.skorobogatykh@klgtu.ru

[a] Novosibirsk State Technical University, Novosibirsk, Russia
[b] Kaliningrad State Technical University, Kaliningrad, Russia

Abstract

The simplex method is widely used in economic planning and forecasting tasks. However, this method is used in real economic activity to find solutions to large-scale tasks, the speed of which is not a critical factor. This significantly limits the applied value of the simplex method in the economic sphere, since currently there is a certain tendency to move to more detailed economic models, which makes it urgent to accelerate calculations based on the simplex method. In these conditions, GPU (Graphical Processor

* The article is published with the support of the HSE University Partnership Programme

Unit) computing accelerators become the most important means of accelerating calculations. The authors propose the implementation of the simplex method in matrix form for computing on GPUs using the PyTorch library. This allows you to switch to using the computing power of graphics processors in a simple and reliable way. A linear programming problem with 900 constraints is solved on a graphics accelerator 6–9 times faster than the solution on a conventional processor. This paper identifies groups of applied economic problems for which the proposed algorithms and methods can be relevant.

Keywords: modified simplex method, acceleration of calculations, graphics processors, linear programming problems in economics

Citation: Ezrokh Y.S., Snytnikov A.V., Skorobogatykh E.Yu. (2025) Parallel implementation of the simplex method in matrix form using the PyTorch library for economics and management problems. *Business Informatics*, vol. 19, no. 2, pp. 77–88. DOI: 10.17323/2587-814X.2025.2.77.88

Introduction

The simplex method [1] is widely used in economic planning and forecasting tasks [2, 3]. In a broad sense, the task of linear programming [4] is that it is necessary to maximize or minimize some linear functional on a multidimensional space under given linear constraints [5–7].

The role of linear programming, in particular the simplex method in economic analysis and planning, is described in [8]. It says that linear programming was considered as a tool for implementing neoclassical economic principles at a time when the very concept of a market economy was under attack from several directions. Linear programming has become widely used in national economic planning [9–11], especially for developing countries, and for studying individual industries, especially energy. The article [12] describes the experience of applying linear programming to the US oil refining industry. The use of linear programming combined with a more careful selection of information sources provides greater predictive power compared to the use of forecasts based on time series. The article [13] also emphasizes the need for a complete understanding of the specifics of production processes, otherwise the methods of mathematical economics [14], in particular, linear programming, give only very approximate results. Works [15, 16] characterize the use of the simplex method as follows: this procedure can be interpreted as a search for market prices that balance the demand for factors of production with their supply.

At the same time, there are practically no works that would test the use of the simplex method to solve a large number of similar tasks with multiple conditions in a short time for individual users. In other words, the simplex method is used in real economic activity to find solutions to large-scale tasks, the speed of which is not a critical factor. This significantly limits the applied value of the simplex method in the economic sphere.

Thus, based on the fact that the simplex method is actively used and there is a definite tendency to switch to more detailed models, it can be concluded that there is a need to speed up calculations based on it. First of all, we are talking about calculations performed on household/office computers rather than on supercomputers [17–19]. In this case, the Graphical Processor Unit (GPU) becomes the most important means of speeding up calculations [20, 21].

GPUs are a set of a very large number (up to 10 000) of simplified processor cores over shared memory which can run several tens of thousands of parallel processes. This means that graphics accelerators are very well suited for solving linear algebra problems (such as matrix multiplication). The computing time on GPUs is usually considered in comparison with the computing time on the Central Processor Unit (CPU).

1. Overview of parallel implementations of the simplex method

The article [22] presents the parallelization of the simplex method in a tabular version for systems with shared memory in order to solve large-scale linear programming problems with a dense matrix. The authors describe the general scheme of the method and explain the strategies adopted to parallelize each step of the standard simplex algorithm. The acceleration and parallel efficiency are analyzed in comparison with the standard simplex method when using a system with shared memory and 64 computing cores. Experiments were performed for several different tasks, up to 8 192 variables and constraints. The maximum acceleration achieved is about 19.

Work [23] considers parallelization of the simplex method based on OpenMP technology. This approach is used to improve acceleration and efficiency. The results of the parallel algorithm are compared with the usual simplex method. This work also used a feature of modern processors such as multithreading. The proposed parallelization algorithm scales easily to a different number of processor cores. In the general case, a number of computational experiments have been conducted using the example of solving the problem of distributing wagons between freight terminals of a railway station. An acceleration of 7 times is reported for 8 OpenMP streams with a data dimension of more than 102.

The article [24] discusses previous attempts to parallelize the simplex method in the context of improving the performance of sequential implementations of the simplex method and the nature of practical linear programming (LP) problems. For the main task of solving common large sparse LP problems, the author of the article did not find parallelization of the simplex method, which provides a significant improvement in performance compared to a good sequential implementation. However, some success has been achieved in the development of parallel solvers for dense LP problems or those or with special structural properties. As a result of the review, this paper determines the directions of future work aimed at developing parallel implementations of the simplex method with practical value. These directions are related to the use of parallel factorization methods and parallel sparse matrix inversion methods.

It was noted in [25] that studies devoted to the implementation of mathematical programming methods on the GPU are still in their infancy. One option is to modify existing algorithms in such a way as to achieve significant performance gains by executing on the GPU. The article [25] solves exactly this problem by presenting an effective implementation for the simplex method adapted for the GPU. The article describes how to perform the steps of the adapted simplex method in order to take full advantage of the GPU's capabilities. The experiments performed demonstrate significant acceleration compared to the sequential implementation, which highlights the enormous potential of the GPU.

The above review shows, firstly, the relevance of the work on the parallel implementation of the simplex method, and secondly, that this work should be mainly focused on computing systems with shared memory, in particular, on the GPU, which confirms the correctness of the problem statement for this article.

2. Description of the simplex method in matrix form

Due to the need to conduct a detailed analysis of the algorithm's performance, as well as to emphasize the difference from the more common simplex method

in the form of tables, we describe the simplex method in matrix form (or revised simplex method), following the book [26].

Let's consider the linear programming problem in the following form: it is required to minimize $\boldsymbol{cx}$ under the condition $Ax = \boldsymbol{b}$, $x \geq 0$, where A is a matrix of size $m \times n$, having rank m. The algorithm for solving the problem is shown in *Fig. 1.*

Next, we provide a step-by-step description of the algorithm.

2.1. Initialization step

Choose an initial basic feasible solution with basis $\boldsymbol{B}$.

2.2. The main part of the algorithm

1. Solve a system of linear algebraic equations $\boldsymbol{B}x_B = \boldsymbol{b}$.

2. Solve a system of linear algebraic equations $\boldsymbol{wB} = \boldsymbol{c}_b$ with a single solution $\boldsymbol{w} = \boldsymbol{c}_b\boldsymbol{B}^{-1}$. The vector **w** is commonly called a simplex multiplier due to the fact that its components are used as multipliers for the rows of the matrix A when converting it to a canonical form. Next, you need to calculate $z_j - c_j = \boldsymbol{wa}_j - c_j$ for all nonbasic variables. Let $z_j - c_j = \max\limits_{j \in J}\{z_j - c_j\}$, where J is the current set of indexes associated with nonbasic variables. If $z_k - c_k \leq 0$, then the current solution is the optimal solution. Otherwise, step 3 is performed with x_k.

3. Solve the system $\boldsymbol{By_k} = \boldsymbol{a}_k$. If $y_k \leq 0$, then the calculation stops, and it is concluded that the optimal solution is unlimited and lies on a straight line

$$\left\{\begin{bmatrix}\overline{b}\\ 0\end{bmatrix} + x_k \begin{bmatrix}-y_k\\ \boldsymbol{e}_k\end{bmatrix} : x_k \geq 0\right\},$$

where $\boldsymbol{e}_k$ is a vector of length $(n - m)$, consisting of zeros, except for the component with the number k, which is 1.

If $y_k \geq 0$, then step 4 is performed.

4. Let x_k be included in the basis. Then r is the index of the blocking variable x_{B_r}, such that the basis remains unchanged as a result of the following check for the minimum ratio

$$\frac{\overline{b}_r}{y_{r,k}} = \min_{1 \leq i \leq m}\left\{\frac{\overline{b}_r}{y_{r,k} : y_{i,k} > 0}\right\}.$$

Next, you need to update the basis $\boldsymbol{B}$, where a_k replaces a_{B_r}, update the set of indexes J and repeat step 1.

2.3. The performance of the simplex method implementation in matrix form

The operating time of the simplex method in matrix form was measured on several model tasks on a computer with an Intel Core i7 950 processor, 3.07 GHz (*Table 1*). The model tasks were generated using an online linear programming task generator [27].

3. Software implementation of the simplex method in matrix form

Modern Python programming language tools, namely the Numpy library, facilitate extremely convenient implementation of computational algorithms based on operations with matrices. As a result, the simplex method was implemented in accordance with the procedure described in section 2.

One of the objectives of this work is to show that it is possible to use the power of high-performance computing to solve applied problems using very easy-to-use tools. In this regard, the PyTorch library, designed to work with neural networks, was used to implement the simplex method on the GPU. The PyTorch library has two important advantages from the point of view of

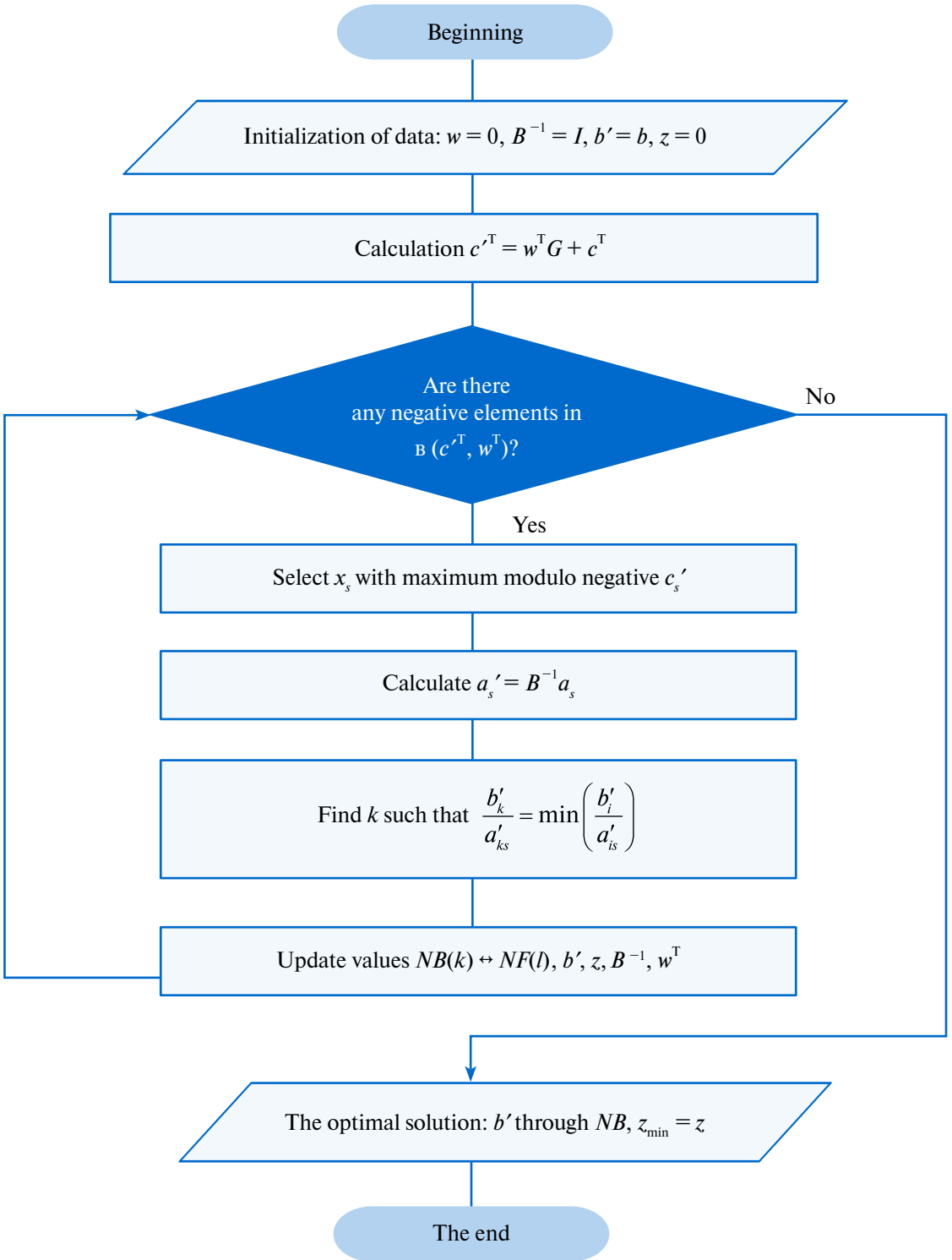

Fig. 1. Algorithm of the revised simplex method.

Table 1.

The operating time of the simplex method in matrix form

Number of restrictions	Matrix size *A*	Time, milliseconds
10	10 × 20	9.3
10	10 × 910	27.1
100	100 × 600	36.7
900	900 × 1800	222.3

the problem being solved: a large set of tools for matrix calculations and the ability to transfer calculations to the GPU without changing the program, just adding one instruction that makes sense to "transfer the matrix to the GPU." All further calculations with the specified matrix will be performed on the GPU. As can be seen from *Table 2*, when transferring the calculation to the GPU using the PyTorch library, the calculation time is noticeably reduced.

Without giving code listings, however, we can say that when using alternative implementations of the simplex method, the main of which is CUDA technology, it is necessary to perform several preparatory operations before calculating, for example, a matrix product, namely, to move the matrices from computer memory to GPU memory and convert them to a format that is optimal in terms of computing on the GPU. Then, in fact, the implementation of the matrix product is performed based on asynchronous calculations using the maximum possible number of parallel processes.

Thus, it can be seen that porting computational algorithms to the GPU using the PyTorch library is much easier than using specialized programming tools such as CUDA, OpenACC, cuBLAS, etc.

The performance of the simplex method implemented on the GPU was measured on the Nvidia TITAN X graphics accelerator, as well as on Nvidia Volta (computing cluster of the NSC NGU). The measurement results are presented in *Table 2*.

Table 2.

Performance of the simplex method implemented on the GPU

Number of restrictions	Matrix size *A*	Time on Titan, ms	Time on Volta, ms	Time on CPU, ms
10	10 × 20	1.9	1.3	9.3
10	10 × 910	4.4	4.4	27.1
100	100 × 600	5.2	5.2	36.7
900	900 × 1800	81.9	23.7	222.3

Shorter billing time for Nvidia Volta is achieved due to the so-called tensor cores, which additionally speed up operations such as matrix multiplication.

The program's performance was also analyzed on the Nvidia A100 Ampere graphics accelerator installed on the KSTU's Euler computing complex. *Figure 2* shows the dependence of the operating time of the matrix conversion procedure, as the mathematically most time-consuming operation, on the parameter N, which, from the point of view of the simplex method, represents the number of variables. Here it is important to pay attention to the fact that matrix inversion is implemented in CUDA using the PyTorch library, and the graph shown does not mean that a linear dependence of the operating time of matrix inversion on the number of variables has been obtained. To clarify this point, consider the number of CUDA threads. Due to the fact that the matrix reversal algorithm is not implemented manually on CUDA, the number of threads is selected automatically. However, the profiler makes it possible to see how many threads are used. The result is shown in *Fig. 3*.

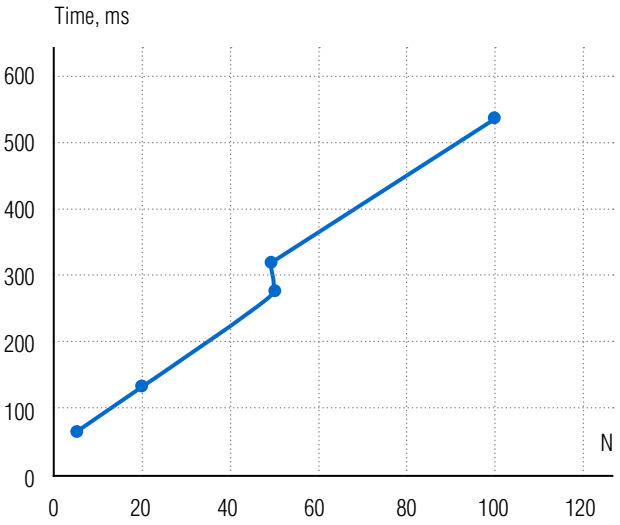

Fig. 2. Dependence of the matrix reversal time on the graphics accelerator Nvidia A100 from the number of simplex method variables.

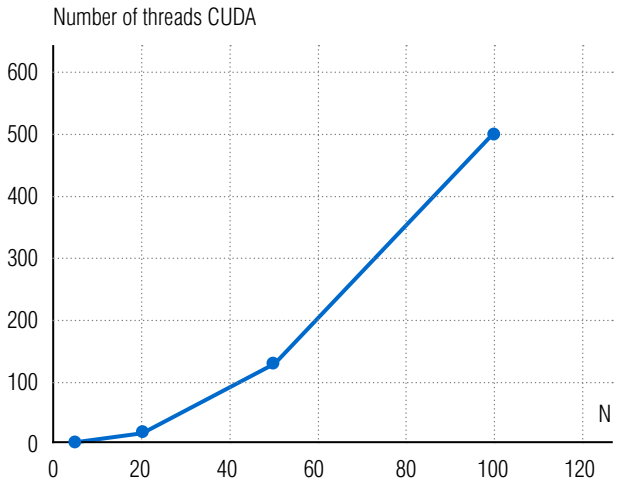

Fig. 3. The number of CUDA threads used when performing matrix inversion in the implementation of the simplex method on the Nvidia A100 graphics accelerator.

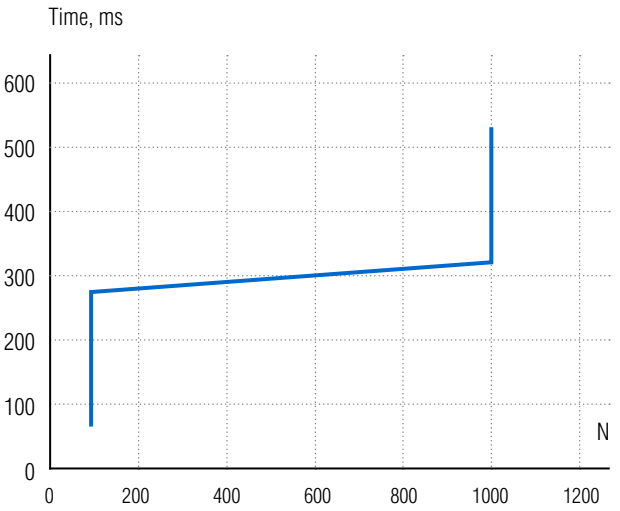

Fig. 4. Dependence of the matrix reversal time on the graphics accelerator Nvidia A100, depending on the number of limitations of the simplex method.

In addition, the dependence of the matrix reversal time on the number of constraints of the simplex method M looks interesting (*Fig. 4*).

4. Directions of applied economic application of the software implementation of the simplex method in matrix form

The algorithms and approaches described above can be used to solve three main groups of applied economic problems, including in the field of marketing, management (in the framework of customer experience management, as well as the implementation of assortment and product policies), plus finance.

The first group is commercial tasks in the field of marketing promotion of goods/services that require an instant solution in the form of an online response in order to obtain a positive super-effect of spontaneous purchases. An example of this is the application of grocery hypermarkets, which generates the final consumer basket of the current order in the presale mode. Given the increasing level of detail of information about the customer (his personal data, as well as the purchase history), the number of parameters acting as limitations within the simplex method is becoming large enough to justify its use. In such a situation, the client is not ready to wait for the time required to calculate the CPU (taking into account the queue of clients), as well as to transfer information. Given the speed of modern life, the client is focused on obtaining exceptionally instant results achieved with the use of graphics accelerators, which is accompanied by an appropriate explanation and justification.

The second group of economic tasks is the intellectualization of the work of high-performance chatbots using fuzzy logic, that is, generating non-standard answers to pre-programmed questions. This will significantly increase the ability to perform the function of a personal assistant, including: (a) expand the planning horizon of the schedule, which will be able to take into account the limitations of family members, work schedules, climatic seasonality, etc.; (b) prepare sound recommendations in the field of a healthy lifestyle, including the composition and formulation of combined products that take into account gerontological, anthropometric, personal data, taste preferences, etc. It is obvious that the increasing scale of the use of assistant bots will allow for more targeted interaction with consumers, as well as the use of recommendation system algorithms for sellers (tasks at the interface with those described above in the first group). For example, based on the characteristics of food products (from the simplest composition of proteins, fats and carbohydrates, to the characteristics of the complementarity of individual products or the impossibility of their joint consumption), the problem of forming the most optimal portfolio of orders in hypermarkets delivering their products to customers can be solved. In such a situation, the linear constraint matrix may significantly exceed the limits defined in the article, which emphasizes the importance of practical application. At the same time, it is worth emphasizing that the customer flow of large e-commerce entities tends to grow sharply, a trend that may continue at least in the medium term. At the same time, even a decrease in the growth rate of Internet business in 3–5 years will most likely lead not to a recession, but to the release of relevant economic indicators to a plateau, while maintaining a significant amount of remote interaction between companies and customers in the long term.

The third group combines financial and economic tasks, where the speed of decision-making is one of the key factors for successful algorithmic trading on the exchange and over-the-counter markets. It is worth noting that this group of tasks is currently, on the one hand, one of the largest in terms of the number of market participants, since the vast majority of traders are focused on using various mathematical models when organizing their activities. On the other hand, the competition of available algorithmization methods and the high "cost of error" of trading, which is often carried out at the expense of customers or using margin trading schemes, make it difficult to test hypotheses, which inevitably occurs as part of the application of new machine learning methods.

It is worth emphasizing that the above groups do not exhaust the tasks where the simplex method can be effectively applied in matrix form on the GPU. These include traditional economic issues in the field of optimization, taking into account limited resources, as well as other, for example, creative tasks (Basadur Simplex).

Conclusion

The research we conducted has shown that transferring the implementation of the simplex method in matrix form to the GPU using the PyTorch library makes solving the problem much easier than using specialized programming tools such as CUDA, OpenACC, cuBLAS, etc. The question arises as to how good the results of this simpler method are. For the test calculations performed, when transferring the calculation to the GPU using the PyTorch library, the calculation time is reduced by 3–5 times, depending on the size of the task.

On the other hand, it is important to be able to speed up calculations using the simplex method for relatively small tasks and for ordinary workstations used in real economic calculations, rather than for superlarge problem statements that can only be solved on a supercomputer, which corresponds to real practice, since even calculations carried out within the framework of economics do not provide examples of very large tasks for the simplex method. This paper presents three main groups of applied economic problems that can be effectively solved using the tools described in the article.

It should also be noted that CUDA technology allows us in almost all cases to achieve the highest efficiency of implementation on the GPU compared to PyTorch. However, CUDA is so complex that its use to solve some real problem is usually a separate big question. This work uses a less effective but much simpler tool. ■

References

1. Kanorovich L.V. (1939) *Mathematical methods of organization and planning of production*. Leningrad: Leningrad State University Press (in Russian).

2. Makarov V.L., Bakhtizin A.R., Sushko E.D., Sushko G.B. (2022) Creating a supercomputer simulation of society with active agents of different types and its testing. *Herald of the Russian Academy of Sciences*, vol. 92, no. 5, pp. 458–466 (in Russian).

3. Kolev M., Georgiadou S. (2022) On simulation and modeling in economics. *Asian-European Journal of Mathematics*, vol. 15, no. 10, article 2250239. https://doi.org/10.1142/s1793557122502394

4. Davydov I., Kochetov Yu., Tolstykh D., et al. (2023) Hybrid variable neighborhood search for automated warehouse scheduling. *Optimization Letters*, vol. 17, no. 9, pp. 2185–2199. https://doi.org/10.1007/s11590-022-01921-6

5. Chistyakova T.B., Shashikhina O.E. (2022) Intelligent software complex for modeling the process of planning multi-assortment industrial productions. *Applied Informatics*, vol. 17, no. 5(101), pp. 41–50 (in Russian). https://doi.org/10.37791/2687-0649-2022-17-5-41-50

6. Beklaryan L.A., Beklaryan G.L., Akopov A.S., Khachatryan N.K. (2024) Dynamic and agent-based models of intelligent transport systems. *Economics and Mathematical Methods*, vol. 60, no. 2, pp. 105–122 (in Russian). https://doi.org/10.31857/S0424738824020091

7. Mochurad L., Boyko N., Sheketa V. (2020) Parallelization of the method of simulated annealing when solving multicriteria optimization problems. Proceedings of the *CEUR Workshop, Lviv, May 21, 2020*, pp. 12–24.

8. Arrow K.J. (2008) George Dantzig in the development of economic analysis. *Discrete Optimization*, vol. 5, no. 2, pp. 159–167. https://doi.org/10.1016/j.disopt.2006.11.007

9. Ratushnyi A., Kochetov Y. (2021) A column generation based heuristic for a temporal bin packing problem. Proceedings of the *20th International Conference Mathematical Optimization Theory and Operations Research (MOTOR 2021), Irkutsk, Russia, July 5–10, 2021* (eds. P. Pardalos, M. Khachay, A. Kazakov). Lecture Notes in Computer Science, vol. 12755, pp. 96–110. Springer, Cham. https://doi.org/10.1007/978-3-030-77876-7_7

10. Kochetov Yu.A., Shamray N.B. (2021) Optimization of placement and relocation of emergency medical teams. *Discrete Analysis and Operation Research*, vol. 28, no. 2(148), pp. 5–34 (in Russian). https://doi.org/10.33048/daio.2021.28.702

11. Kovesnikov V.A., Mehtiev A.Ya. (2020) Study of the accumulative-sorting method for solving parametric optimization problems. *Control Issues*, no. 2, pp. 28–35 (in Russian). https://doi.org/10.25728/pu.2020.2.3

12. Manne A.S. (1958) A linear programming model of the U.S. petroleum refining industry. *Econometrica*, no. 26, pp. 67–196.

13. Chenery H.B. (1949) Engineering production functions. *The Quarterly Journal of Economics*, no. 63, pp. 507–531.

14. Makarov V.L., Bakhtizin A.R., Beklaryan G.L., Akopov A.S. (2021) Digital plant: methods of discrete-event modeling and optimization of production characteristics. *Business Informatics*, vol. 15, no. 2, pp. 7–20. https://doi.org/10.17323/2587-814X.2021.2.7.20

15. Scarf H.E. (1990) Mathematical programming and economic theory. *Operations Research*, vol. 38, no. 3, pp. 377–385.

16. Apalkova T.G., Kosorukov O.A., Mishchenko A.V., Tsourkov V.I. (2024) Mathematical models for managing the production and financial activities of an enterprise. *Herald of the Russian Academy of Sciences. Theory and Systems of Control*, no. 2, pp. 107–129. https://doi.org/10.31857/S0002338824020109

17. Gergel V., Grishagin V., Liniov A., Shumikhin S. (2021) Parallel computations in integrated environment of engineering modeling and global optimization. Proceedings of the *16th International Conference (PaCT 2021), Kaliningrad, Russia, September 13–18, 2021* (ed. V. Malyshkin), Lecture Notes in Computer Science, vol. 12942, pp. 413–419. Springer, Cham. https://doi.org/10.1007/978-3-030-86359-3_31

18. Gergel V., Kozinov E. (2021) Parallel computations for solving multicriteria mixed-integer optimization problems. *Communications in Computer and Information Science*, vol. 1437, pp. 92–107. https://doi.org/10.1007/978-3-030-81691-9_7

19. Shichkina Y., Kupriyanov M., Awadh AM.M.H. (2020) Application of methods for optimizing parallel algorithms for solving problems of distributed computing systems. Proceedings of the *International Conference on Cyber-Physical Systems and Control (CPS&C'2019), St. Petersburg, Russia, June 10–12, 2019* (eds. D. Arseniev, L. Overmeyer, H. Kälviäinen, B. Katalinić), Lecture Notes in Networks and Systems, vol. 95, pp. 212–224. Springer, Cham. https://doi.org/10.1007/978-3-030-34983-7_21

20. Makarov V.L., Bakhtizin A.R., Beklaryan G.L., Akopov A.S., Strelkovskii N.V. (2022) Simulation of migration and demographic processes using FLAME GPU. *Business Informatics*, vol. 16, no. 1, pp. 7–21. https://doi.org/10.17323/2587-814X.2022.1.7.21

21. Beklaryan A.L., Akopov A.S., Beklaryan L.A. (2021) Implementation of the Deffuant model within the FLAME GPU framework. *Advances in Systems Science and Applications*, vol. 21, no. 4, pp. 87–99. https://doi.org/10.25728/assa.2021.21.4.1161

22. Coutinho D.A.M., Lins e Silva F.O., Aloise D., Xavier-de-Souza S. (2019) A scalable shared-memory parallel simplex for large-scale linear programming. *arXiv:1804.04737v2*. https://doi.org/10.48550/arXiv.1804.04737

23. Mochurad L. (2020) Parallelization of the Simplex method based on the OpenMP technology. Proceedings of the *4th International Conference on Computational Linguistics and Intelligent Systems (COLINS 2020), Lviv, Ukraine, April 23–24, 2020*, pp. 952–963.

24. Hall J.A.J. (2010) Towards a practical parallelization of the Simplex method. *Computational Management Science*, vol. 7, pp. 139–170. https://doi.org/10.1007/s10287-008-0080-5

25. Bieling J. (2010) An efficient GPU implementation of the revised simplex method. Proceedings of the *IEEE International Symposium on Parallel Distributed Processing, Workshops and PhD Forum (IPDPSW), Atlanta, GA, USA*, pp. 1–8. https://doi.org/10.1109/IPDPSW.2010.5470831

26. Bazaraa M.S., Jarvis J.J., Sherali H.D. (2010) *Linear Programming and Network Flows*. Hoboken, NJ: John Wiley & Sons, Inc.

27. Instituto Superior Técnico, Universidade de Lisboa (2025) *LP random problem generator*. Available at: https://web.tecnico.ulisboa.pt/~mcasquilho/compute/or/Fx-LP-generator.php (accessed 20 April 2025).

About the authors

Yuriy S. Ezrokh

Doctor of Sciences (Economics), Associate Professor;

Head of Department, Department of Economic Informatics, Novosibirsk State Technical University, 6, Karl Marx St., Novosibirsk 630073, Russia;

E-mail: ezrokh@corp.nstu.ru

ORCID: 0000-0002-8367-1840

Alexey V. Snytnikov

Doctor of Sciences (Technology);

Professor, Department of Applied Informatics, Kaliningrad State Technical University, 1, Sovetskiy Ave., Kaliningrad 236022, Russia;

E-mail: aleksej.snytnikov@klgtu.ru

ORCID: 0000-0003-4111-308X

Elena Yu. Skorobogatykh

Candidate of Sciences (Pedagogy), Associate Professor;

Associate Professor, Department of Applied Informatics, Kaliningrad State Technical University, 1, Sovetskiy Ave., Kaliningrad 236022, Russia;

E-mail: elena.skorobogatykh@klgtu.ru

ORCID: 0000-0001-6050-4831