Parallelization of the Simplex Method Based on the OpenMP Technology

Lesia Mochurad[0000-0002-4957-1512], Nataliya Boyko[0000-0002-6962-9363],

Nataliia Petryshyn[0000-0003-4642-1778], Myroslava Potokij, Mariia Yatskiv

Lviv Polytechnic National University, Lviv79013, Ukraine

lesia.i.mochurad@lpnu.ua, nataliya.i.boyko@lpnu.ua,
natalia.y.petryshyn@lpnu.ua,
myroslava.potokii.kn.2016@lpnu.ua, mikamause99@gmail.com

Abstract. In this work we propose a parallelization of the simplex method based on the technology of parallel programming OpenMP, which is used for solving linear programming problems. The advantages of this approach are to improve acceleration and efficiency. This, in turn, is important when processing large data. The obtained results of the parallel algorithm were compared with the conventional simplex method. Application of this method is important considering modern trends of multi-core architecture of processors. Also in this work was used such feature as multithreading. The proposed parallelization algorithm is easily scaled to a different number of processor cores. Without loss of generality, the example of solving the problem of distribution of cars between the freight fronts of the railway station have been carried out a number of numerical experiments. Data analysis showed that with increasing the number of threads and cores achieve the maximum performance of a program that implements a parallelization of the simplex method. These findings are depicted as bar charts.

Keywords: OpenMP parallel computing technology, multithreading, finite difference, linear programming, simplex method, parallelization, multicore.

1 Introduction

Linear programming (LP) was developed because of economy problems, finding a way to find the best decisions while using limited resources. The development and complication of economic processes and computing stimulates extensive use of mathematical methods in management, contributes to the growth of the role of linear programming as one of the important topics of applied mathematics [1-3].

According to American experts, about 75% of the total number of practical optimization problems, relate to the objectives of LP. About a quarter of the machine time spent in recent years on carrying out scientific researches was allocated to the decision of tasks of LP and their numerical modifications.

The simplex method is generic and it can be used to solve a wide range of tasks:

- maximum pairing (graph theory));
- maximum thread;
- transportation problem;
- a zero-sum game (a zero sum)

Improving the efficiency of algorithms for solving linear programming problems has important practical significance. Industrial production resources in the economy, the tension forces in the hostilities – it is a complex of numerous interrelated processes. In the management of these totally different phenomena one can distinguish essential features of similarity. When formulating optimization problems it turns out that problems with different content can be represented by the same type of mathematical models [4].

The linear programming model includes three main points:

- a set of nonnegative variables characterizing the investigated process or phenomenon;
- relations that establish the relationship between variables (restrictions) and the requirements of the reflecting task;
- criterion of optimality (goal function).

In the problems of linear programming constraints are a system of linear inequalities and equations, expressing the condition of the material balance (ie, that the consumption of any kind of raw material does not exceed the available stock of this raw material, etc.). The goal function also has a linear look.

The aim of the work is to parallelize algorithm of the simplex method to achieve maximum acceleration and efficiency in the processing of large volumes of input data.

Despite the fact that the simplex method is a very efficient algorithm that showed good results in solving linear programming problems, it is an algorithm with exponential complexity [4]. The reason for this is the combinatorial nature of the algorithm that sequentially traverses the vertices of the polyhedron of admissible solutions in the search for an optimal one.

Parallelization of the simplex method is a very relevant and popular topic. Available studies prove the effectiveness of a parallel algorithm, which allows you to increase the accuracy of the result. However, the implication of the simplex-method with modern trends in the development of computer technology becomes even more important, namely: a promising way in building computer systems based on multicore processors.

2 Review of the Literature

Nowadays, most tasks of linear programming are solved using algorithms which are based on the simplex method. Other known algorithms, including those with polynomial complexity, yield to the effectiveness of the simplex method in solving specific applications [5, 6].

The problem of linear programming is considered in the following canonical representation:

$$\max_{\bar{x}}\left(W = \sum_{i=1}^{n} b_i x_i \right), \bar{x} = \{x_i\}, i = \overline{1, n} \tag{1}$$

with restrictions

$$\begin{cases} x_i \le 0, i = \overline{1, n}, \\ \varphi_j(\bar{x}) = \sum_{i=1}^{n} a_{ji} x_i \le c_j, c_j \ge 0, i = \overline{1, n}, \; j = \overline{1, M} \le n \end{cases} \tag{2}$$

This formulation is considered to be classic, and this way any linear programming problem can be compiled [7]. The canonical representation is convenient because in this case the values b_i match with the values of the Lagrange multipliers, which are calculated for the active constraints $x_i \le 0, i = \overline{1, n}$

The simplex method has one important advantage – when you increase the dimension of the problem, the computational costs are small, because at each step it calculates only one value of the objective function.

The task of LP is usually written in abbreviated form and is represented as follows:

The task of linear programming is a wide range of control tasks to the functioning of economic systems.

Linear programming problem closely related to the tasks of game theory, so to solve them it is possible to apply numeric methods of game theory $Z = \max(\min) \sum_{j=0}^{n} c_j x_j$ [8].

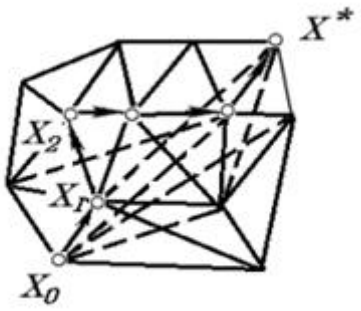


Fig. 1. Graph model of the simplex method..

The geometric meaning of simplex method consists of consecutive transition edges from one vertex of a polyhedron solution (basic plan) to another in the direction of the

vertex *X**, in which the objective function reaches the highest (lowest) value (see Fig. 1).

3 Statement of the Problem

General task of linear programming (GLP), presented in any form of record, called a task in which you must determine the optimum (maximum or minimum) of the objective function:

$$F = c_1 x_1 + c_2 x_2 + ... + c_n x_n \tag{3}$$

under the following restrictions:

$$\sum_{j=1}^{n} a_{ij} x_j \leq \{\geq\} b_i ; (i = \overline{1, s}),$$

$$\sum_{j=1}^{n} a_{ij} x_j = b_i ; (i = \overline{s+1, m}), \tag{4}$$

$$x_j \geq 0,\ j = \overline{1, n}$$

here a_{ij}, b_i, c_j – some of the coefficients.

We introduce the notation:

$$\Delta_j := \sum_{i=1}^{m} C_{b_i} a_{ij} - c_j, (j = \overline{1, n}) \tag{5}$$

where C_b - the vector of objective function coefficients at basic variables.

Matrix *A*, that consists of elements a_{ij}, the dimension of $m \times n$ is called the matrix of the problem, the column vector is called the vector of free members (a vector of constraints in the problem), a row vector c_k – the coefficients of the objective function at variable x_k, a column vector $X = \begin{bmatrix} x_1 \\ x_2 \\ ... \\ x_n \end{bmatrix}$ – the vector of unknowns (variables).

In vector form, the problem (3)-(4) has the following form:

$$F = Cx$$

$$p_1x_1 + p_2x_2 + p_3x_3 + ... + p_mx_m + p_{m+1}x_{m+1} + p_nx_n = p_0$$

$$p_1 = \begin{bmatrix} a_{11} \\ a_{21} \\ a_{31} \\ ... \\ a_{m1} \end{bmatrix}; p_2 = \begin{bmatrix} a_{12} \\ a_{22} \\ a_{32} \\ ... \\ a_{m2} \end{bmatrix}; p_3 = \begin{bmatrix} a_{13} \\ a_{23} \\ a_{33} \\ ... \\ a_{m3} \end{bmatrix} \ldots;$$

$$p_m = \begin{bmatrix} a_{1m} \\ a_{2m} \\ a_{3m} \\ ... \\ a_{mm} \end{bmatrix}; p_{m+1} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ ... \\ 0 \end{bmatrix}; p_n = \begin{bmatrix} 0 \\ 0 \\ 0 \\ ... \\ 1 \end{bmatrix} \ldots$$

When you build simplexe table uses the value p_0 – the right part of the problem constraints.

Without loss of generality, consider the problem of distribution of cars on the freight fronts of the railway station, which is usually solved by methods of linear programming in multicriterial statement.

4 Methods of Solving

Let freight station arrives a certain number of cars that must be unloaded during the day on three cargo fronts. At various technical equipment the cost of railway on unloading is different. Let the different fronts they are respectively 9, 10 and 16.s.u/weights. (standard unit)

Find the maximum number of wagons can be unloaded by a cargo station during the day:

$$F = 9x_1 + 10x_2 + 16x_3 \rightarrow \max$$

There are also the following constraints for the problem:

$$\begin{cases} 18x_1 + 6x_2 + 5x_3 \le 360, \\ 15x_1 + 4x_2 + 3x_3 \le 192, \\ 12x_1 + 8x_2 + 3x_3 \le 180. \end{cases}$$

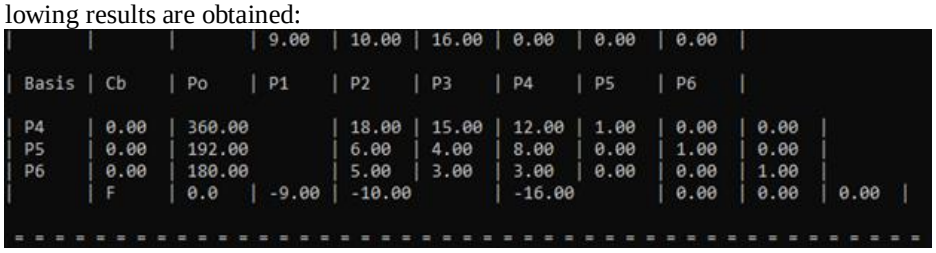
Demonstrate a few of the implementation steps of the simplex method on this problem.

The problem can be represented in the form of a Table. 1.

Table 1. The input data

Number of cargo front	Expenses of each of the front			Capacity of the fronts
I	18	6	5	360
II	15	4	3	192
III	12	8	3	180
Fee	9	10	16	

The result of the first iteration of the simplex method on the given example, the following results are obtained:

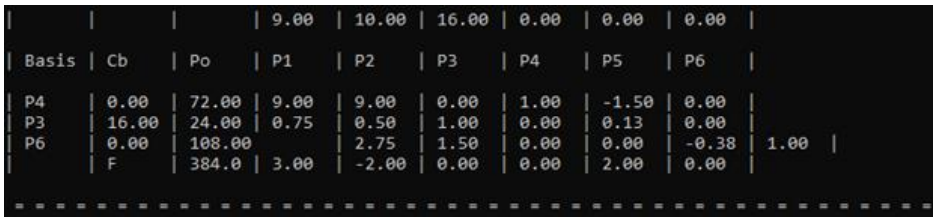


			9.00	10.00	16.00	0.00	0.00	0.00	
Basis	Cb	Po	P1	P2	P3	P4	P5	P6	
P4	0.00	360.00		18.00	15.00	12.00	1.00	0.00	0.00
P5	0.00	192.00		6.00	4.00	8.00	0.00	1.00	0.00
P6	0.00	180.00		5.00	3.00	3.00	0.00	0.00	1.00
	F	0.0	-9.00	-10.00		-16.00		0.00	0.00	0.00

Fig. 2. The results of the first phase of the simplex method

In Fig. 2 the value of the function equal to 0.0, however, there are negative values Δ, in particular, $\Delta_1 = -9.00$, $\Delta_2 = -10.00$, $\Delta_3 = -16.00$. You need to go to the next support plan with a list of vector values $p_1, p_2, p_3, p_4, p_5, p_6$.

The results of the execution of the next iteration is shown in Fig. 3:

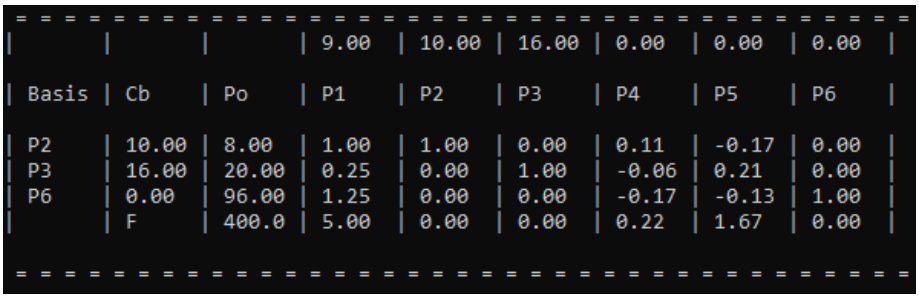


			9.00	10.00	16.00	0.00	0.00	0.00	
Basis	Cb	Po	P1	P2	P3	P4	P5	P6	
P4	0.00	72.00	9.00	9.00	0.00	1.00	-1.50	0.00	
P3	16.00	24.00	0.75	0.50	1.00	0.00	0.13	0.00	
P6	0.00	108.00		2.75	1.50	0.00	0.00	-0.38	1.00
	F	384.0	3.00	-2.00	0.00	0.00	2.00	0.00	

Fig. 3. The results of the second iteration of the simplex method

As can be seen from Fig. 3, the optimal value of the function increased $(F = 384{,}00)$, however, the target value has not been reached, because $\Delta_2 = -2.00$.

After the last iteration (see Fig. 4) the simplex method the optimal solution (see Fig. 5):



			9.00	10.00	16.00	0.00	0.00	0.00
Basis	Cb	Po	P1	P2	P3	P4	P5	P6
P2	10.00	8.00	1.00	1.00	0.00	0.11	-0.17	0.00
P3	16.00	20.00	0.25	0.00	1.00	-0.06	0.21	0.00
P6	0.00	96.00	1.25	0.00	0.00	-0.17	-0.13	1.00
	F	400.0	5.00	0.00	0.00	0.22	1.67	0.00

Fig. 4. The results of the third iteration of the simplex method

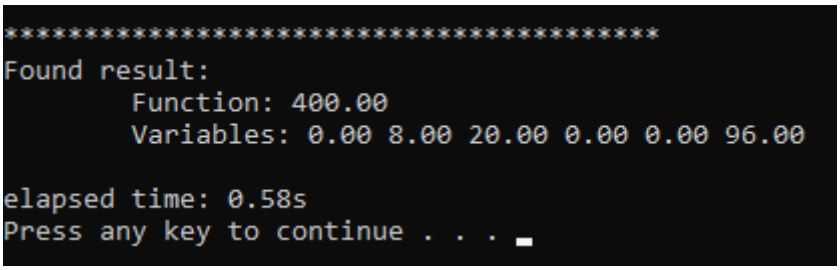


Fig. 5. The solution of the problem

The optimal value function $F = 400{,}00$. The values of the unknown $x_1 = 0{,}00$, $x_2 = 8{,}00$, $x_3 = 20{,}00$, $x_4 = 0{,}00$, $x_5 = 0{,}00$ and $x_6 = 96{,}00$. Since all values $\Delta \geq 0$, the problem is solved.

According to the obtained results, construct a table of time measurements of execution of a sequential algorithm with reference to the respective computer architecture.

Table 2. Temporal measurements of the sequential execution of the algorithm on a Quad-core

Number of cores	Run time, sec
1	3.713
2	3.381
4	2.572
8	0.998

Based on the analysis of the results of sequential algorithm execution (see Table. 2) we saw the need for parallelization of the simplex method when processing large volumes of input data.

5 Parallelization of the Simplex Method

To achieve the maximum performance was performed to parallelize the critical sections of the program. It is possible to reduce the computational time of the algorithm

by performing multiple tasks simultaneously, and has also led to a decrease in the number of iterations, making the algorithm more effective for use in problems with a large amount of data [9, 13]. In particular, it was distributed one of the most important critical areas of the program – calculation of parameters with the arguments in the objective function. For this we used technology OpenMP is set of compiler directives, library procedures and environment variables intended for programming multi-threaded applications on multiprocessor systems with shared memory in C, C++ [10].

Advantages of OpenMP [11,12]:

- Thanks to the idea of "incremental paralleling", OpenMP best suits the developers seeking to quickly parallel their calculating programs with large parallel loops. The developer does not create a new parallel program but just consecutively adds text of a program OpenMP directives.
- At the same time, OpenMP is a flexible mechanism that gives the developer great control over the behavior of the parallel program.
- It is assumed that the OpenMP program on a uniprocessor platform can be used as a sequential program, i.e. there is no need to support serial and parallel versions. The OpenMP directives are simply ignored by a sequential compiler, and OpenMP procedure call can be inlined stub (stubs), the text of which is given in the specifications [14].
- One of the advantages of OpenMP developers find support for the so-called "orphan" (separated) directives, that is directives of synchronizing and distributing work that may not enter directly into the lexical context of the parallel region [15].

In this work the parallelization of the critical sections of the algorithm, in particular:

- The scalar product of the vector C_b and P_i to expedite plan check for optimality:

```
#pragma omp parallel for
   for (int i = 0; i < numberOfVariables; i++) {
     double sum = 0;
     for (int j = 0; j < numberOfRestrictions; j++) {
     sum += functionCoefs[currentBasis[j]] * restrictionCoefs[j*numberOfVariables + i];
     }
     currentDeltas[i] = sum - functionCoefs[i];
     if (currentDeltas[i] < 0) {
       negativeDeltaPresent = true;
     }
   }
  }
#pragma omp parallel for
     for (int i = 0; i < numberOfRestrictions; i++) {
     currentFunctionValue += functionCoefs[currentBasis[i]] * restrictionVals[i];
          }
```

- The process of determining Δ, which in the absolute value takes the maximum value:

```
#pragma omp parallel for
     for (int i = 0; i < numberOfVariables; i++) {
```

```
        if (i != maxVal_num && abs(currentDeltas[i]) == abs(currentDeltas[maxVal_num])) {
          equalDeltas.push_back(i);
        }
      }
      #pragma omp parallel for
        for (int i = 0; i < numberOfVariables; i++) {
          if (abs(currentDeltas[i]) > abs(currentDeltas[max])) {
            max = i;
          }
        }
      #pragma omp parallel for
          for (int i = 0; i < (int)equalDeltas.size(); i++) {
            if (functionCoefs[equalDeltas[i]] > functionCoefs[equalDeltas[validMax]]) {
              validMax = equalDeltas[i];
            }
          }
```

– The construction of the next simplex table and determining all of its coefficients according to the new basis:

```
      #pragma omp parallel for
        for (int i = 1; i < numberOfRestrictions; i++) {
          if (restrictionCoefs[i*numberOfVariables + absMaxDelta] > 0 &&
            allDivisions[i] < allDivisions[minDivVal]) {
            minDivVal = i;
          }
        }
      #pragma omp parallel for
        for (int i = 0; i < numberOfRestrictions; i++) {
          if (i == minDivVal) {
            currentBasis[i] = absMaxDelta;
            break;
          }
        }
```

The form was created to process the input data (see Fig. 6.). This form accepts input parameters of three cargo fronts and performs data processing after clicking the "Calculate" button.

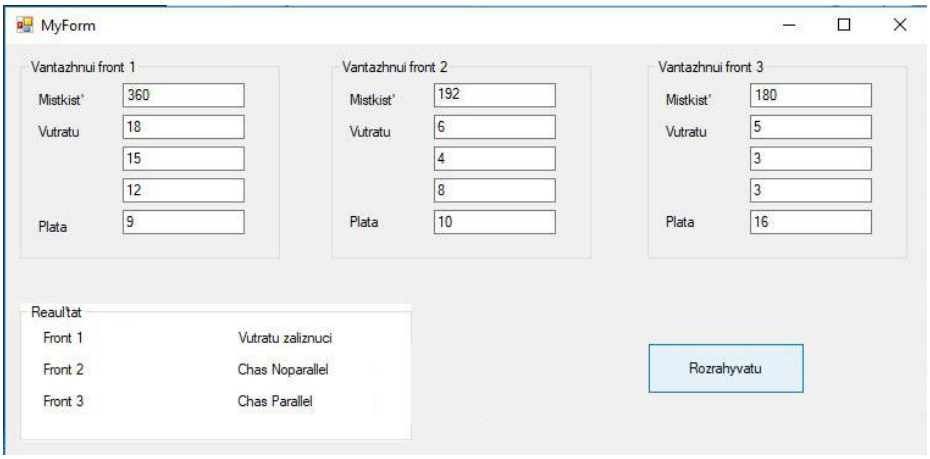


Fig. 6. Data input/output form

Form was created to simplify the user experience with the program, and was made using Windows Forms technology. This is a smart client technology for the .NET Framework platform, a collection of managed libraries that simplify the execution of common tasks, such as reading and writing in a file system. While using the Visual Studio development environment intelligent Windows Forms applications can be created that display information, request input from users and allow to interact with remote computers in the network.

In Windows Forms form is a visual surface on which information to the user is displayed. Usually the Windows Forms application is build by dragging control elements onto a form and writing code to respond to user actions such as mouse clicks or keystrokes. A control element is a separate element of the user interface that can be used to display or enter data.

When a user performs any action with a form or one of its control elements, an event is created. The application responds to these events using code and processes events when they occur. If an existing control does not meet the needs, in Windows Forms, you can create custom controls using the User Control class.

6 Numerical Experiments

After carrying out a numerical experiments, it has been shown that with a fairly small input data, the parallel implementation of the simplex method algorithm stops to work efficiently. It was found that when entering data less than 10^2, the results of parallel processing of data are not significantly different from the results obtained with the sequential execution of the program (see Fig. 7).

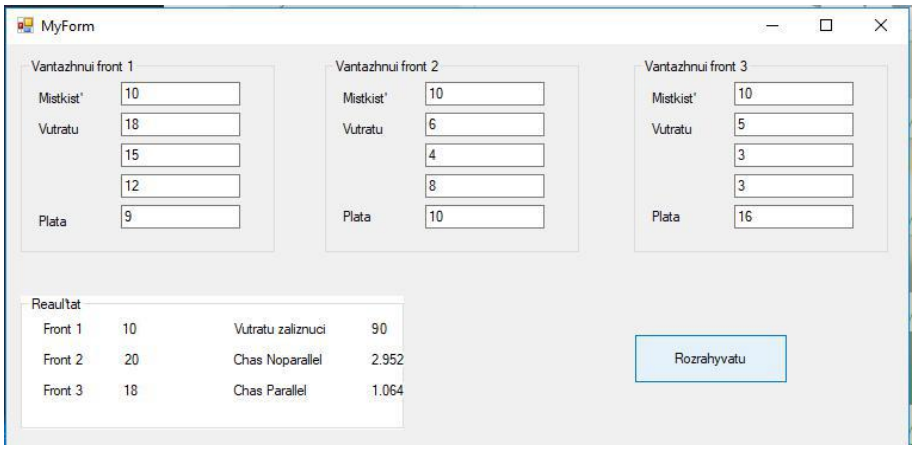


Fig. 7. The result of a program with rather small input data

Further, on Fig. 8 shows the result of parallelizing the algorithm of the simplex method to solve the problem of distributing of cars between the freight fronts of the railway station when processing large amounts of data.

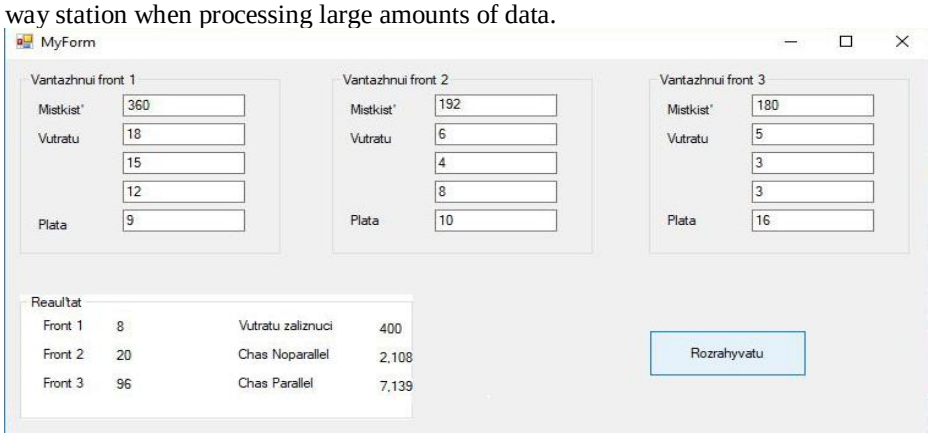


Fig. 8. The result of the program

Table 3 shows the time schedules for the parallel execution of the program. As you can see, the parallelization of the simplex method several times speeds up the program while processing large data.

Table 3. Time measurements of parallel algorithm execution on a four core CPU

Number core	Execution time, s
1	3.623
2	3.004
4	1.547
8	0.580

In this case, the values of the acceleration and efficiency of the parallel algorithm on the quad core processor are shown in Table 4. Fig. 9 shows a graph of execution time (in seconds) depending on the number of threads on quad core processor.

Table 4. Acceleration and efficiency ratios

Number of flow	Acceleration factor	Efficiency ratios
1	1.0249268	1.0249268
2	1.1254993	0.56274966
4	1.6625727	0.415643175
8	1.72068968	0.21508620

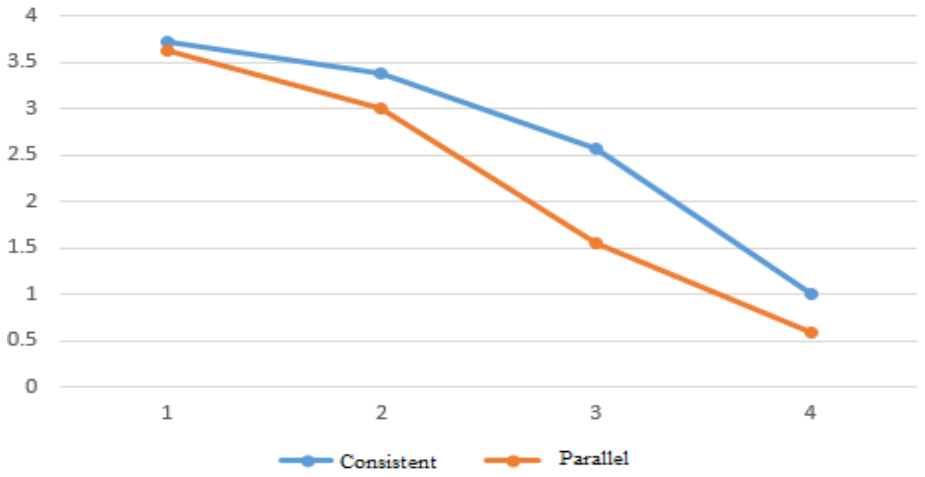


Fig. 9. Graph of execution time depending on the number of threads

Fig. 10-11 shows comparisons` graphs of the acceleration and efficiency of parallel execution for two and four - core processor.

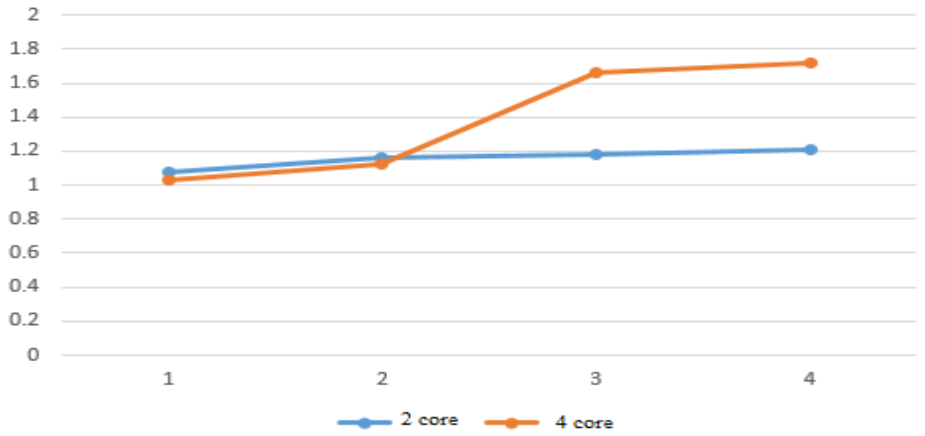


Fig. 10. Comparison of acceleration indexes

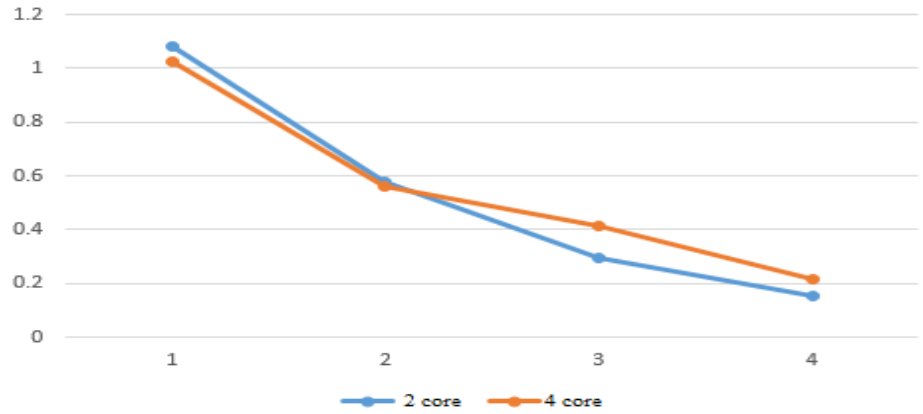


Fig. 11. Comparison of performance

7 Analysis of the Results

The effectiveness of the simplex method depends on the following factors:

1. Numbers of iterations;
2. Machine time.

As a result of numerical experiments, the obtained results showed that the machine time is proportional to m^2. The number of constraints has a greater impact on computing efficiency than on the number of variables, therefore, it was concluded that when forming a linear programming problem we should aim to reduce the number of constraints.

The software product developed in the work calculates the solution without losing data and presents the results as integers. The program interface is easy to use to enter task data. However, the single-core processor can not be achieved even with the parallel execution of the program. Also, it was investigated that with a sharp contrast input data for freight fronts there is an incorrect paralleling and failures in the implementation of the algorithm.

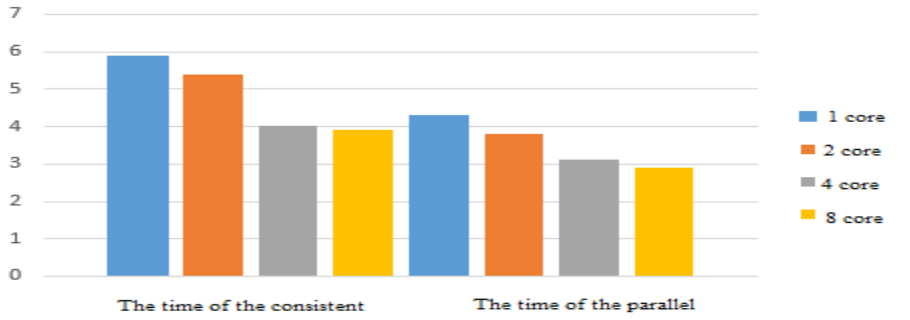


Fig. 12. Performance results on a single core processor

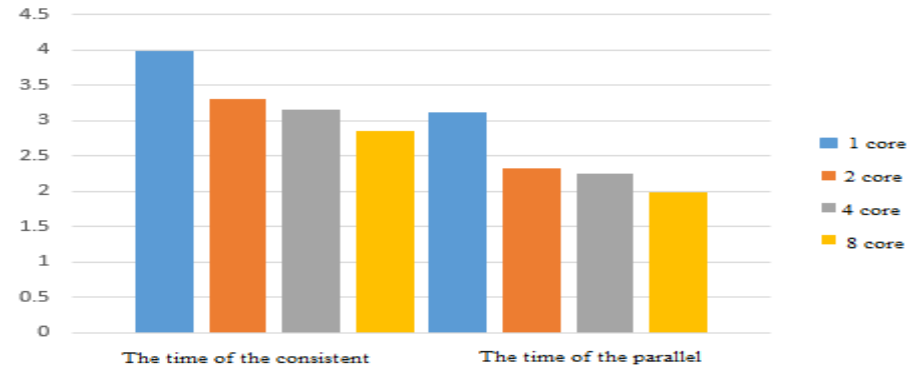


Fig. 13. Performance results on a dual-core processor

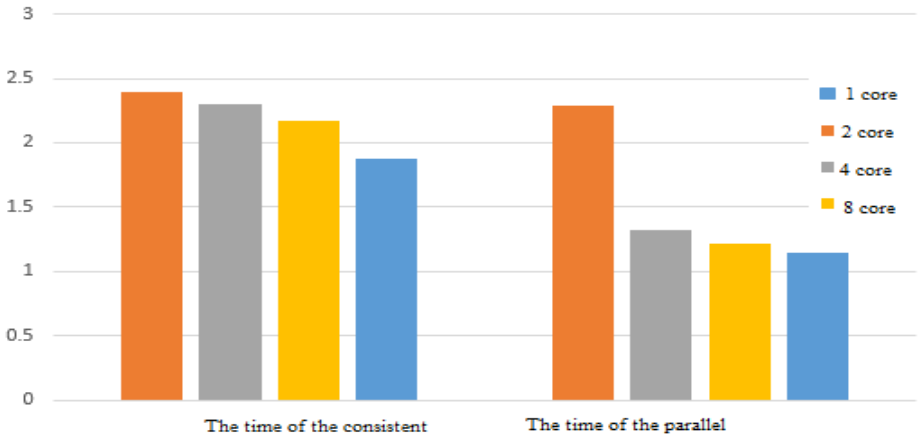


Fig. 14. Performance results on an eight-core processor

Data analysis (see Fig. 12-14) shows that with the increasing the number of threads and cores the maximum productivity of the program is reached. For a small number of threads on a single core processor, the algorithm is performed relatively long with other processors, and there are possible minor differences in starting with the same large enough input parameters.

8 Conclusion

The linear programming problem and the method of its solution that are described in the work - is only a separate example of a huge number of linear programming tasks.

Linear programming is the most commonly used method of optimization. Linear programming is one of the main parts of that section of modern mathematics, which is called mathematical programming.

Simlex method is a typical example of iterative computations used in solving most of the optimization tasks. For computer realization of the simplex method a method of using artificial variables is developed, which allows to find the initial basic solution of the problem.

The main task of this work was to develop a program for solving a linear programming problems with a sufficiently large dimensionality and the ratio of the number of variables to the number of constraints. According to the results of numerical experiments, we can conclude high efficiency of the parallelized algorithm of the simplex method based on the OpenMP parallel programming technology. After parallelizing critical sections of the algorithm to speed up the program, it is noted that the acceleration factor increases with the increase in the number of cores, which means it directly depends on the architecture of the computer. This approach is aimed at supporting the latest developments of multi-core processors. However, the total amount of computations can be improved by some optimizations of the implemented algorithm, which are goals for further work.

References

1. Yakovlev, M.F., Gerasymova, T.O., Nesterenko, A.N.: Characteristic feature of the solving both of non-linear systems and systems of ordinary differential equations on parallel computer. In Proceedings of international symposium “Optimization problems of computations” (OPC - XXXV). Kyiv: V.M. Glushkov Institute of cybernetics of NAS of Ukraine, 2009. Kyiv: Vol. 2. P. 435-439. (2009).
2. Yakovlev, MV.F., Nesterenko, A.N., Brusnikin, V.N.: Problems of the efficient solving of non-linear systems on multi-processor MIMD-architecture computers. Mathematical machines and systems. (4). P. 12-17. (2014).
3. Khymych, A.N., Molchanov, Y.N., Popov, A.V. other: Parallel algorithms for solving problems of computational mathematics. Kiev: Scientific Opinion, 248 pp. (2008).
4. Autar, K.: NONLINEAR EQUATIONS - Newton-Raphson Method-More Examples, Civil Engineering. August 7, 4 pp. (2009)
5. Autar, K.: NONLINEAR EQUATIONS - Newton-Raphson Method-More Examples, Mechanical Engineering. August 7, 3 pp. (2009).

6. Kakhaner, D., Mouler, K., Nesh, S.: Numerical methods and software. «Myr» publishing house, 575 pp., (1998).
7. Mochurad, L.I.: Method of reduction model for calculation of electrostatic fields of electronic fields of electronic optics systems. Science journal Radioelektronika, informatics, management, 1(48), pp. 29-40 (2019). (In Ukrainian)
8. Voss, M.: OpenMP Share Memory Parallel programming. Toronto, Kanada (2003).
9. Chapman, B., Jost, G.: Using OpenMP: portable shared memory parallel programming (Scientific and Engineering Computation). Cambridge, Massachusetts: The MIT Press (2008).
10. Chandra, R., Menon, R., Dagum, L., Kohr, D., Maydan, D., McDonald, J.: Parallel Programming in OpenMP. Morgan Kaufinann Publishers (2000).
11. Grama, A., Gupta, A., Karypis, G., Kumar, V.: Introduction to Parallel Computing. Addison Wesley, ISBN- 0-201-64865-2, 856 p. (2003).
12. Mochurad, L., Boyko, N.: Solving Systems of Nonlinear Equations on Multi-core Processors. Advances in Intelligent Systems and Computing IV Selected Papers from the International Conference on Computer Science and Information Technologies, CSIT 2019, September 17–20, pp. 90-106, Lviv, Ukraine (2019)
13. Shakhovska, N., Boyko, N., Zasoba, Y., Benova, E.: Big data processing technologies in distributed information systems. Procedia Computer Science, 10th International conference on emerging ubiquitous systems and pervasive networks (EUSPN-2019), 9th International conference on current and future trends of information and communication technologies in healthcare (ICTH-2019), Vol. 160, 2019, pp. 561–566, Lviv, Ukraine (2019)
14. Boyko, N., Shakhovska, Kh., Mochurad, L., Campos, J.: Information System of Catering Selection by Using Clustering Analysis, Proceedings of the 1st International Workshop on Digital Content & Smart Multimedia (DCSMart 2019), pp. 94-106, Lviv, Ukraine (2019)