Parallelization of the Process of Calculating the Optimal Route for a Strike Aircraft Flight

Lesia Mochurad [1][0000-0002-4957-1512], Nataliya Boyko[1][0000-0002-6962-9363], Vasyl Sheketa[2][0000-0002-1318-4895]

[1] Department of Artificial Intelligent Systems, Lviv Polytechnic National University, 12 S. Bandery str., Lviv, 79000, Ukraine
[2] Ivano-Frankivsk National Technical University of Oil and Gas

lesia.i.mochurad@lpnu.ua, nataliya.i.boyko@lpnu.ua, vasylsheketa@gmail.com

Abstract. In this article an approach is proposed for parallelizing the method of automation to determine the rational and shortest flight path of shock aviation to the set goal in a space containing a limited flight zone. In the research process, the graph model was used and the problem of finding the shortest path for shock air transport was considered. In order to secure the flight, by constructing the shortest route, bypassing the lines of fire, and to estimate the fuel consumption, the Dijkstra's algorithm is parallelized using the OpenMP parallel programming technology. For optimal use of the multi-core platform, one has been selected for a single core stream. A number of numerical experiments on a computer with two- and four-nuclear architecture were conducted. Estimates of the acceleration and efficiency of the proposed parallel algorithm, which can be significantly improved with the use of computers with more cores. As a result of the research, we managed to find not only the shortest, but also the least costly and safe way, which is relevant for our country, because an anti-terrorist operation takes place on the territory of Ukraine, where flights are conducted daily on the fire zone to assess the situation in which Ukrainian army is located.

Keywords: Dijkstra's Algorithm, OpenMP Technology, Acceleration, Efficiency.

1 Introduction

In the conditions of the rapid development of armaments and military equipment, and the constant automation of operational and tactical calculations, the question is raised about the optimization, efficiency and validity of the definition of the flight route of shock aviation [1, 2]. The calculation consists of the reliability of orientation and output to the main and secondary objective with a more favorable direction at the specified time, the need to ensure the lowest fuel consumption, the need for a covert flight over the enemy territory, taking into account the terrain and weather conditions, and the number of programmable points of the flight, all these calculations and conditions when choosing a route affect the time, optimality and fidelity of the choice of the airship flight route, for the successful completion of combat missions. In the process of these calculations, optimization of the route identification process is required, which will improve the time characteristics. One way to reduce the execution time is to parallelize the algorithm or its parts to further implement it in parallel architecture [3]. Convert an existing algorithm to a parallel can be done using experience, intuition, or mathematical apparatus of equivalent transformations.

The development of automation in the military sphere directly affects the increase of the combat capability of the troops, which entails an increase in the percentage of successful implementation of the tasks assigned to the Armed Forces of Ukraine in particular. Considering the tasks that are solved by automation at the control points can reduce the time to make a decision and reduce errors in the calculation process, which will significantly affect the outcome of the task.

The object of study is a study of the method of solving the problem of determining the impact route of a strike aircraft.

The subject of study is OpenMP's parallel programming technology for finding the optimal route for an airspace.

The purpose of the work is the application of the Dijkstra's method with modern trends in the development of computer technology. Building multiprocessor systems is the first and most important direction in multi-threading, but for this path development requires a lot of money. The leading manufacturers of processors, speak about a prospective way of developing computers, which consists in the construction of computer systems based on multi-core processors. The use of multi-core systems is the best opportunity to increase the power of processors [4]. So, for today, multi-core architecture is rapidly developing. Therefore, the main purpose of this work is to develop a program for the implementation and application of the algorithm, which allows the basic calculations to be conducted in parallel with the use of modern architecture of multi-core processors and analyze the advantages in calculating the speed of computing.

2 Related Works

In the problem of the shortest paths [5, 6], a weighted oriented graphis given $G = (V, E, \omega)$ where V − the set of vertices of the graph G; E − the set of its edges; $\omega : E \rightarrow R$ − weight function of the edges of the graph G.

The Dijkstra's algorithm unlike Floyd's algorithm [7], by which one can find the shortest route between any two vertices of a graph, is intended to find the route of the shortest length from the given vertex of the graph to all others.

Let t it be the supporting peak; V_1 − set of vertices of the graph for which the shortest path from the reference vertex t is calculated; n − number of vertices of the graph; v − current top of the graph.

The scheme of the sequential Dijkstra's algorithm can be represented in the form [8, 9]:

- Procedure Dijkstra_Single$\left(V, E, \omega, t\right)$.
- $V_1 := \{t\}$.
- For all vertices $v \in \left(V \setminus V_1\right)$.
- If ($\left(p, v\right)$ exist), then $s[v] := \omega\left[p, v\right]$.
- If ($\left(p, v\right)$ does not exist), then $s[v] := \infty$.
- Until $\left(V_1 \neq V\right)$ perform.
- Find a vertex u such that $s[u] := \min\left\{s\left[v\right] \middle| v \in (V \setminus V_1)\right\}$.
- $V_1 := V_1 \setminus \{u\}$.
- For all vertices $v \in \left(V \setminus V_1\right)$.
- $s[v] := \min\left\{s\left[v\right], s\left[u\right] + \omega[u, v]\right\}$.

The above algorithm uses a set V_1, containing vertices of the graph for which the shortest path from the original vertex t is already found. It also uses a priority queue $Q = V_1 / V$ and an attribute $s\left[1...n\right]$ (where n − number of vertices of the graph G), in which for each vertex $v \in V$ the current value of the shortest path is stored.

At each such iteration of the algorithm, a new vertex u is added to the set V_1, such that $s[u] := \min\left\{s\left[v\right] \middle| v \in (V \setminus V_1)\right\}$. After adding the vertex, all array values $s[v]$ are updated if the sum of the distance to the vertex u and the edge of the “fringe” is less than the current distance to the vertex. The algorithm finishes its work when $V_1 = V$.

The execution time of the Dijkstra's algorithm depends on the implementation of the priority queue [10]. In the case where the priority queue is maintained due to the fact that all the vertices are numbered from 1 to $|V|$. The attribute $s[v]$ is simply placed as an array element with an index v. At the same time, each operation of adding a vertex to the set V_1, and the transition to the next vertex takes time $O(1)$, and each operation of the minimum search ($s[u] := \min\left\{s\left[v\right] \middle| v \in (V \setminus V_1)\right\}$) $O(V)$, since it searches the entire array. As a result, the time of the algorithm is generally equal $O(V^2)$. Note that in the case of an unsaturated graph G, the priority queue is set to be a binary tree. Then the search operation value $s[u] := \min\left\{s\left[v\right] \middle| v \in (V \setminus V_1)\right\}$ takes $O(\lg V)$, and such operations − $|V|$ that is, the total complexity of the algorithm - $O((V + E)\lg V)$. And this value is equivalent $O(E \lg V)$, provided that all vertices of the graph G are achievable from the reference vertex t.

3 Setting the Task

It is necessary to create a program for the parallelization of the automation algorithm to determine the rational and shortest flight path of the strike aircraft to the target set in the space containing the restricted areas for the flight. It is known that the flight route is selected taking into account the actions of the armed forces of the opposing sides, the established corridor of the flight of the front line, the terrain. The route chosen for flight to the target must provide reliable and accurate air traffic in the conditions of counteraction by the enemy, as well as the reach of the target for aviation in different conditions. In working out this task, we will take into account that certain areas of the flight are prohibited, since this fire line and the projectile can hit the plane. That is, the itinerary should go bypassing the forbidden zones. The zones in which the flight is forbidden, we ask statically in the program realization. And then the program builds the route, without taking into account these vertices of the graph. If suddenly the flight will pass through a closed territory, the flight coordinator and the executor will be subject to penalties. This will look like this. For example, there are three closed zones. And the program is building a way to bypass them. The shortest paths can be several. We will choose the one weighing less, to save fuel costs. In this case, the shortest path whose weight is the smallest is indicated by a solid line. Otherstwo, marked with a dotted line, are also variants of the shortest path, but they are bigger in them. Since large-scale missions are spent on strike aircraft, we decided that with this algorithm you can determine the estimated amount of fuel that is needed for the flight. The fuel consumption of the aircraft varies greatly depending on the length of the route, the chosen speed, load, wind speed, flight altitude. For an average strike aircraft with good weather conditions, about 8,000 kg of fuel for 900 km/h is spent per hour of flight at an altitude of 7-8 km. The fuel consumption can be calculated because of the weight of the path. For example, 1 weight is ~ 10 km. And thus, about 1 kg of fuel is spent on 1 weight of the way.

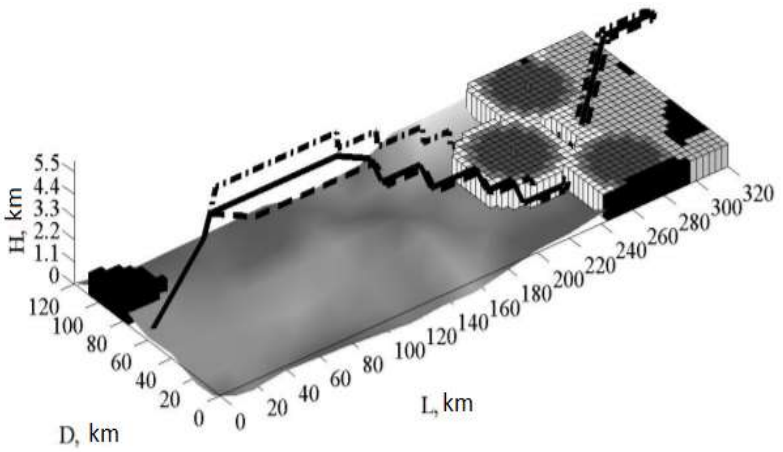


Fig. 1. Three-dimensional view of the flight through the restricted zones

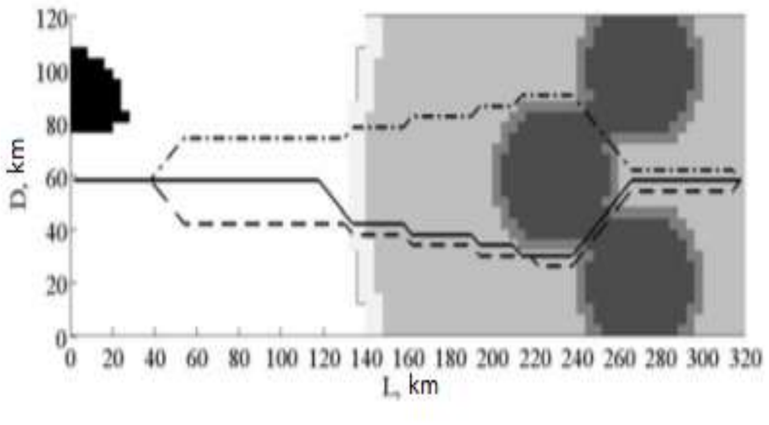


Fig.2. Horizontal flight profile

It should be noted that the task posed in the work is relevant to our country, because an anti-terrorist operation is conducted on the territory of Ukraine, where flights over the zone of fire are conducted daily to assess the situation in which the Ukrainian army is located.

4 Materials and Methods

When solving a given problem, we can use the Dijkstra's algorithm, which allows us to find the shortest paths from some vertex of the graph to all its other vertices, since a weighted oriented graph, which is a graph model of our problem, does not have arcs of negative weight. In order to accelerate the application of this algorithm, the work takes into account trends in the development of multi-core architecture of modern computers and uses the technology of parallel programming OpenMP [11-13]. Using the Dijkstra's algorithm, we assume that each vertex of a directed graph is a certain coordinate in space. That is, on space we impose the coordinate grid and each point in this grid is the vertex of the graph. In this program, the user introduces the start and end point of the flight. After receiving data from the user, the program displays the most optimal way. Hazardous and forbidden zones in the program are static, the distance to these points is assigned 0, thus we exclude the peaks through which can pass the flight path. By obtaining a minimum distance to the end point, we can calculate the amount of fuel to be used. The fuel consumption for the flight is calculated according to the following formula:

$$K = D \times k,$$

where K − the amount of fuel consumed during the flight, D − the weight of the path, k − the amount of fuel, which is spent at a distance of 10 km ~ 1 weight of the path.

In order to achieve the maximum productivity of the above algorithm in the work, the parallelization of critical areas of the program was performed, the parallel execution of which allowed reducing the execution time of the algorithm by executing several tasks simultaneously, and also led to a reduction in the number of iterations, which makes the algorithm more effective for use in tasks with a large amount of data. For this purpose, OpenMP was used − a set of compiler directives, library procedures and environment variables that are intended to run multithreaded applications on multiprocessor systems with shared memory in C, C ++ [14, 15].

Advantages of OpenMP:

1. Due to the idea of incremental parallelization, OpenMP is ideally suited to developers who want to quickly parallelize their computing programs with large parallel cycles. The developer does not create a new parallel program, but simply adds sequentially to the text of the program OpenMP-directive.
2. At the same time, OpenMP is a flexible mechanism that gives developers great control over the behavior of a parallel program.
3. It is assumed that the OpenMP program on a single-processor platform can be used as a sequential program, that is, there is no need to maintain a consistent and parallel version. OpenMP directives are simply ignored by a sequential compiler, and stubs can be used to call OpenMP procedures, the text of which is given in the specifications.
4. One of the advantages of OpenMP, developers consider support for so-called "orphan" (detached) directives, that is, directives for synchronization and distribution of work may not be included directly in the lexical context of the paramaterial area.

In the program realization of the set task in work used on-off directives.

To select parallel fragments of the program:

#pragmaompparallel

If a loop operator has met in a parallel region, then, according to the general rule, it will be executed by all streams of the current group, that is, each stream will execute all the iterations of the given loop. For the distribution of iteration cycles between different threads, you can use the directive for:

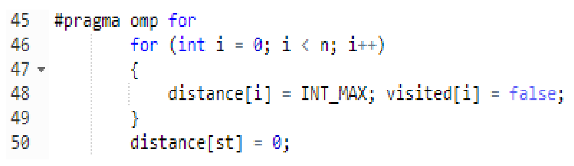
#pragmaompfor

The directive used to determine the structural block of the program, which will run exclusively in the main stream (parallel to the thread with a zero number) from the whole set of parallel streams:

#pragmaomp master

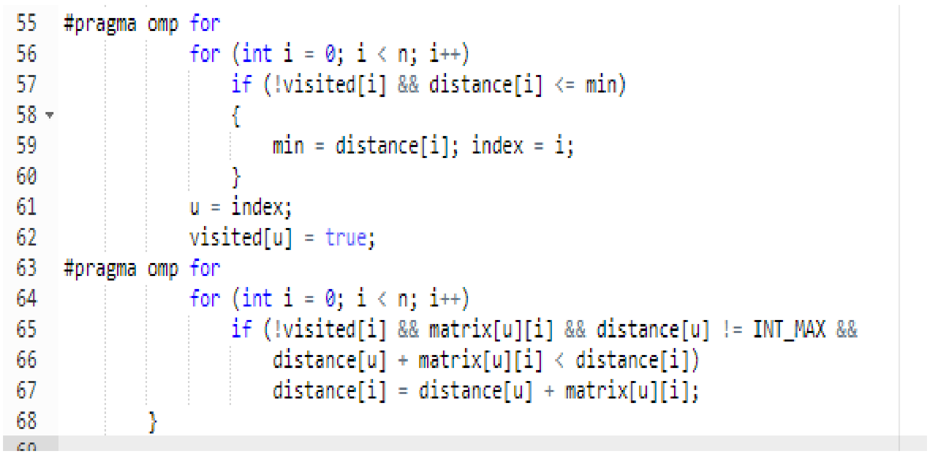
Implicit synchronization does not imply this directive.

For example, a parallel code that is responsible for filling an array of distances between the vertices of the initial values (during the program execution, the values will change) and the array of visited vertices (each element is assigned a False value):



```
#pragma omp for
        for (int i = 0; i < n; i++)
        {
            distance[i] = INT_MAX; visited[i] = false;
        }
        distance[st] = 0;
```

A parallel code that finds the minimum distances from a given vertex in each vertex of a directed graph:



```
#pragma omp for
          for (int i = 0; i < n; i++)
              if (!visited[i] && distance[i] <= min)
              {
                  min = distance[i]; index = i;
              }
          u = index;
          visited[u] = true;
#pragma omp for
          for (int i = 0; i < n; i++)
              if (!visited[i] && matrix[u][i] && distance[u] != INT_MAX &&
                  distance[u] + matrix[u][i] < distance[i])
                  distance[i] = distance[u] + matrix[u][i];
      }
```

The application of a parallel algorithm to the problem posed in the work prints the finding of the optimal path, allowing the user to timely regress in critical situations.The main feature of the parallel programming model is the higher performance of the programs.

However, it should be noted that parallel computations using OpenMP technology are used in areas related to large-scale calculations, therefore, for small amounts of use, they are inappropriate [16].

5 Results

Having analyzed the task and available programs for implementation, we decided to use the C ++ software and OpenMP technology, implemented in the Visual Studio 2017 software environment.
The program was tested on a two- and four-core processor. At the same time, the parallel algorithm gives a significant gain over time compared with the successive, but with a small dimension of the matrix, the sequential algorithm is more efficient.
With the dimension of a matrix smaller than 1000 × 1000, the parallel algorithm not only shows the same results but worse than in the sequential one.
Analyzing the data presented in Table 1 and Table 2, we can conclude that the implementation time of the program improves with the increase in the dimension of the input data, which for our task corresponds to the increase in the scale of the territory on which we are looking for the shortest flight of shock air.
On Fig. 3 and Fig. 4 show schedules of the program execution time from the matrix dimension (value n) for a certain number of threads on dual-core and quad-core processors.
And on Fig. 5 and Fig. 6 show graphs of the dependence of the program execution time on the number of threads on a two- and four-core processor at $n = 9000$.

Tabl. 1. The program execution time on the dual-core processor, minutes

Dimensionalityn	Number of streams		
	1	2	4
1000	0,02107	0,01979	0,01988
3000	0,13148	0,08966	0,07848
5000	0,42794	0,24516	0,2348
7000	0,91067	0,45916	0,42431
9000	1,47441	0,78206	0,69478

Tabl. 2. Time to execute the program on a quad-core processor, minutes

Dimensionality n	Number of streams		
	1	2	4
1000	0,007	0,00593	0,00616
3000	0,063	0,0455	0,02989
5000	0,222	0,10078	0,08373
7000	0,469	0,2232	0,18923
9000	0,752	0,47864	0,31487

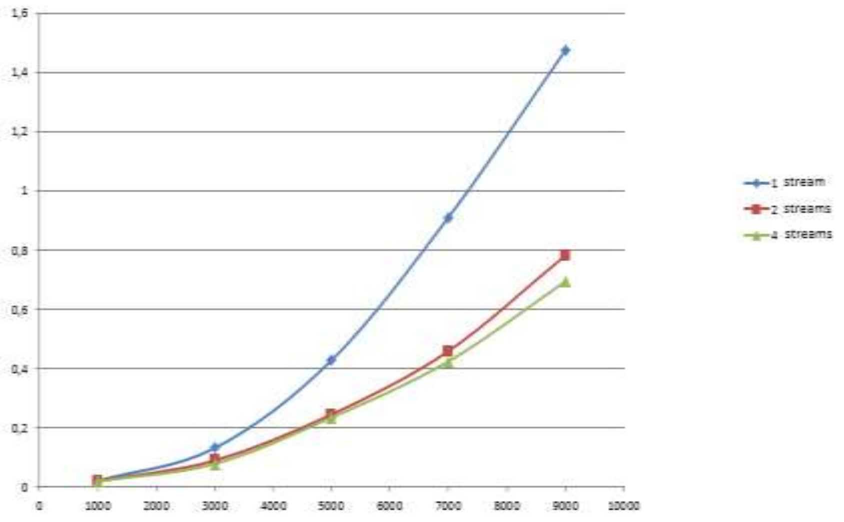


Fig.3. The graph of the dependence of the program execution time on the matrix dimension (value n) with a certain number of threads on a dual-core processor

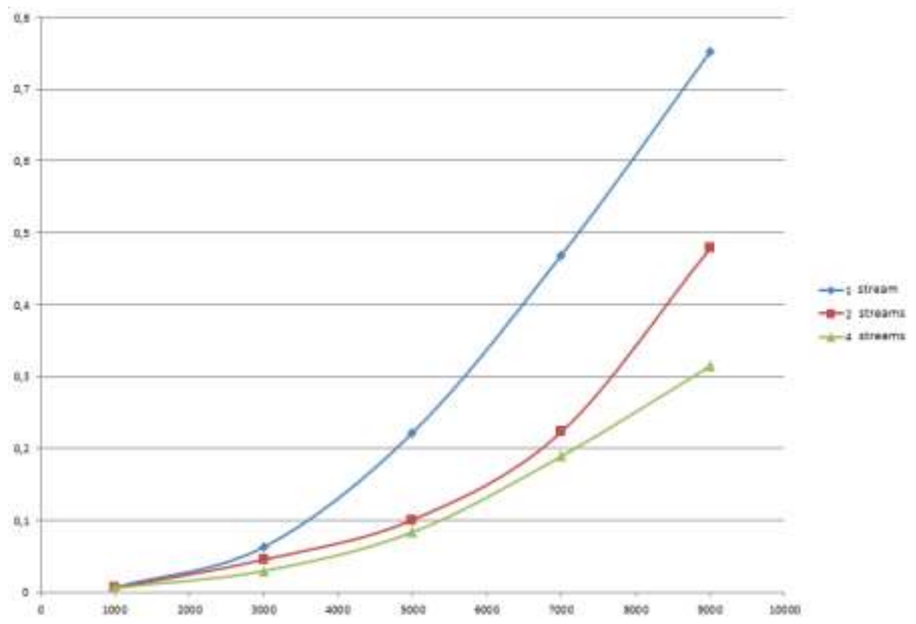


Fig. 4. Schedule of the dependence of the program execution time on the dimension of the matrix (value n) with a certain number of threads on the quad-core processor

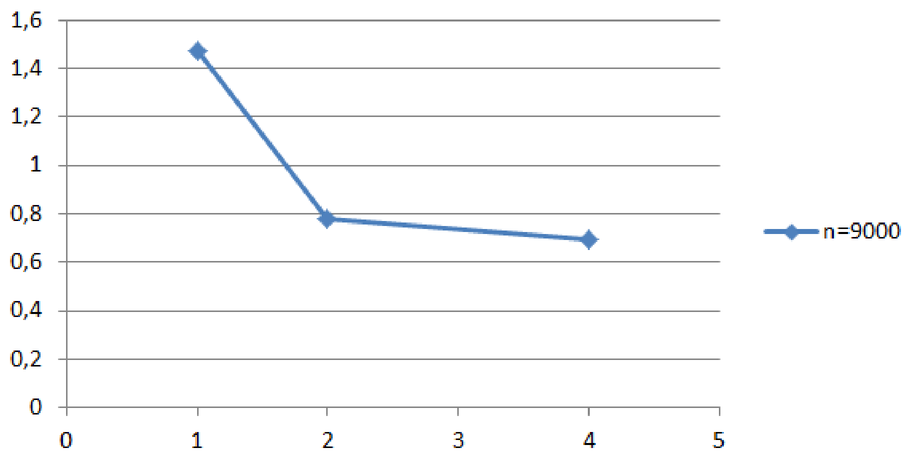


Fig. 5. The graph of the time dependence of the program implementation on the number of threads on the 2-core processor at n =9000

Having conducted a number of numerical experiments with a different number of threads, we confirmed that for optimal use of the multi-core platform it is necessary to take into account that the number of threads is equal to the number of processor cores [4]. If the number of threads is greater than this does not affect the speed of the calculations. The program's execution time on a quad-core processor yields as many rewards as dual-core, since the given matrix sizes corresponding to grid splits are not large enough.

It is known [17] that when performing parallel calculations for the analysis of the obtained results, the acceleration and efficiency indicators are important. The speedup obtained by using a parallel algorithm for p processors is determined by the value $S_p(n) = T_1(n)/T_p(n)$ in comparison with the successive version of the calculation, and n is used to parameterize the computing complexity a solvable problem and can be understood, for example, as the number of input data of a task. The efficiency of using a parallel processor algorithm in solving a problem is determined by the ratio of $E_p(n) = T_1(n)/(p \cdot T_p(n)) = S_p(n)/p$.

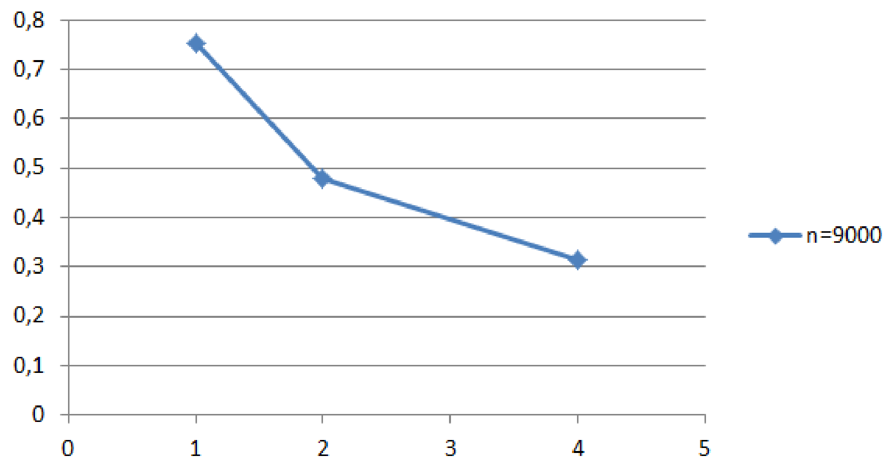


Fig. 6. The graph of the dependence of the program execution time on the number of threads on the 4-core processor at n =9000

The efficiency value determines the part of the algorithm execution time, which processors are really designed to solve the problem. From the above relationships, it can be shown that in the best case: $S_p(n) = p$ and $E_p(n) = 1$.

Tabl. 3. Parameters of acceleration and efficiency of parallel algorithm

n	Parallel Algorithm Indicators			
	2-core processor		4-core processor	
	parallel speedup	parallel efficiency	parallel speedup	parallel efficiency
1000	1,064679131	0,532339565	1,136363636	0,284090909
3000	1,466428731	0,733214365	2,107728337	0,526932084

n	Parallel Algorithm Indicators			
	2-core processor		4-core processor	
	parallel speedup	parallel efficiency	parallel speedup	parallel efficiency
5000	1,745553924	0,872776962	2,651379434	0,662844858
7000	1,983339141	0,991669571	2,47846536	0,61961634
9000	1,885290131	0,942645066	2,38828723	0,597071807

The calculations were made for the data obtained when launching the program on a 2-core and 4-core processor.

The number of threads was selected for optimal use of the multi-core platform, that is one stream per core. Tabl. 3 shows the values of acceleration and efficiency indicators.

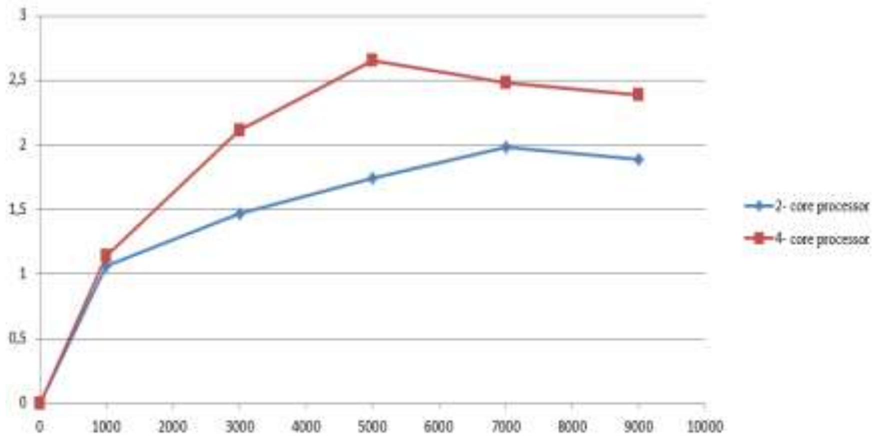


Fig. 7. The graph for comparison of acceleration indicators

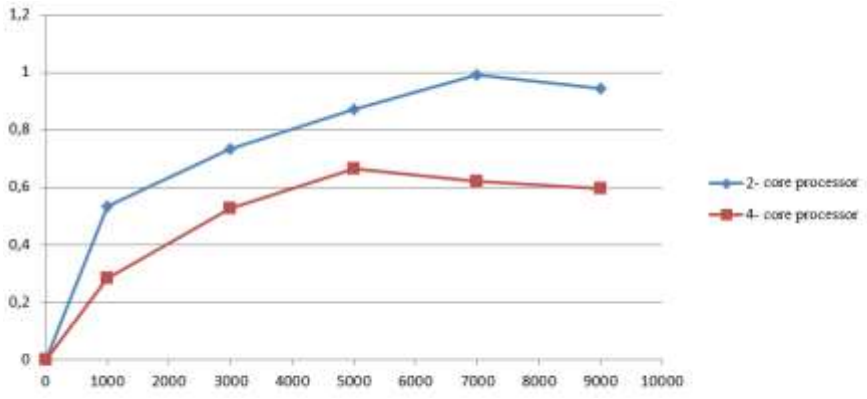


Fig. 8. The graph for comparison of performance indicators

On Fig. 7 and Fig. 8 graphs of comparison of acceleration and efficiency indicators for two- and four-core architecture, respectively. At the same time, it was possible to achieve the maximum value of the rate of acceleration and efficiency on the dual-core processor. Obviously, with the increase in input data, the same result could be achieved for the four nuclear archeology.

6 Conclusion

The main task of optimizing the algorithm is to improve the time characteristics with increasing the size of the output graph by creating parallel versions. Improved these characteristics through the use of parallel programming OpenMP technology. When solving the problem set in the work, not only the shortest path but also the least costly and safe using the Dijkstra's algorithm is found.

The scientific novelty of the obtained results is that in the work the software was developed that allows to implement the procedure of parallelization of the calculation of the optimal impact flight path of the strike aircraft using multi-core systems and such a property as multithreading based on OpenMP technology. The application of this technology has made it possible to significantly improve the time characteristics in calculating the optimum impact route for strike aircraft. At the same time, the coefficient of brute force was increased by approximately 2, 4 times with the use of 2- and 4-core processors, respectively. The practical significance of the results obtained is that the results obtained in the work, in turn, make it possible to increase the percentage of successful implementation of the tasks assigned to the Air Forces of the Armed Forces of Ukraine. The prospects for further research are that the alignment and efficiency indicators can be greatly improved by increasing the number of cores and varying variations in the number of streams, which is relevant in the development of computer systems based on multi-core processors.

References

1. Vorobyov, E.S., Pavlenko, M.A., Chertok, O.A., Gladyshev, M.G.:Rationale for the method of automation of the calculation of the optimal route for strike aircraft of the air force of the Armed Forces of Ukraine. Collection of abstracts of reports of the scientific-practical conference "Service and combat activity of the National Guard of Ukraine: state of affairs, problems and perpetuates", 6-9 (2018).
2. Vorobyov, E.S.: The use of cellular automata in the method of ranking variants of the flight route of strike aircraft for the destruction of ground targets. Collection of scientific works of Kharkiv National University of the Air Force, № 2, 39-47 (2018).
3. Pogoriliy, S.D., Boyko, Yu.V., Belous, R.V.: Formation and analysis of parallel schemes of Dijkstra's algorithm. Mathematical Machines and Systems, Vol. 4, 61–71 (2008).
4. Mochurad, Lesia,Boyko, Nataliya:Solving Systems of Nonlinear Equations on Multi-core Processors. DOI: 10.1007/978-3-030-33695-0_8, 17 p. (2020)

5. Abraham, I., Delling, D., Goldberg, A., Werneck, R.Labeling Algorithm for Shortest Paths on Road Networks. Symposium on Experimental Algorithms, 230-241 (2011).
6. Rolf H., Mohring, Heiko, Schilling, Birk, Schutz, Dorothea, Wagner, and Thomas,Willhalm: Partitioning Graphs to SpeedupDijkstra’s Algorithm. ACM Journal of Experimental Algorithmics, Vol. 11, Article No. 2.8, 1–29 (2006).
7. Revyakin, A. M., Evgrafova, N. V.: Development of a Minimum-Duration Route for a Truck Transporting Communication Devices. Economic and socio-humanitarian studies, № 4(16), 179-182(2017).
8. Crauser, A., Mehlhorn, K., Meyer, U., and Sanders, P.: A Parallelization of Dijkstra's Shortest Path Algorithm. MFCS'98- LNCS 1450, Lubos Prim et al. (Eds.), SpringerVerlag Berlin Heidelberg, 722-731 (1998).
9. Pogorilyy, S. D., Slynko, M. S., &Rustamov, Y. I.: Research and development of Jonhson’s algorithm parallel schemes in GPGPU technology. TWMS Journal of Pure and Applied Mathematics, 8(1), 12-21 (2017).
10. M., Chen, R.A., Chowdhury, V., Ramachandran, D.L. Roche, L. Tong: Priority Queues and Dijkstra’s Algorithm, Technical Report TR-07-54, Computer Science Department, University of Texas at Austin, 25 p. (2007)
11. Lesia,Mochurad, Khrystyna,Shakhovska, Sergio, Montenegro:Parallel Solving of Fredholm Integral Equations of the First Kind by Tikhonov Regularization Method Using OpenMP Technology. Advances in Intelligent Systems and Computing IV, 11 p. (2020)
12. Chapman, B., Jost,G., Ruud van der Pas: Using OpenMP: portable shared memory parallel programming (Scientific and Engineering Computation). Cambridge, Massachusetts: The MIT Press (2008).
13. Voss, M.: OpenMP Share Memory Parallel programming. Toronto, Kanada (2003).
14. Nikolskyi, Y.V., .Pasichnyk, V.V., Shcherbyna, Y.M.: Discrete Math. 368 pp. (2007).
15. Sedzhvyk, R.: Fundamental algorithms in C ++. Algorithms on graphs. Per. from English, 496 pp. (2002).
16. Mochurad, L., Solomiia, A.: Optimizing the Computational Modeling of Modern Electronic Optical Systems. Lecture Notes in Computational Intelligence and Decision Making. ISDMCI 2019. Advances in Intelligent Systems and Computing, vol 1020. Springer, Cham. 597–608 (2020).
17. Kvurt, L., Tsylylyk, L.: The use of the laws of Amdal and Gustafson in estimating the acceleration rate in multiprocessor systems. Measurement Engineering and Metrology, Lviv: Issue Nat. Lviv Polytechnic University, Issue 70, 55–57 (2009).
18. Schahovs’ka, N., Syerov, Yu.: Web-community ontological representation using intelligent dataspace analyzing agent. 10th International Conference - The Experience of Designing and Application of CAD Systems in Microelectronics. Polyana–Svaliava (Zakarpattya), 2009. pp. 479–480.
19. Fedushko S., Michal Gregus ml., Ustyianovych T. Medical card data imputation and patient psychological and behavioral profile construction. Procedia Computer Science. Volume 160, pp. 354-361 (2019). https://doi.org/10.1016/j.procs.2019.11.080
20. Shakhovska N., Fedushko S., Greguš ml. M., Melnykova N., Shvorob I., Syerov Yu. Big Data analysis in development of personalized medical system. Procedia Computer Science, Volume 160, 2019, pp. 229-234. https://doi.org/10.1016/j.procs.2019.09.461
21. Fedushko S., Ustyianovych T. Predicting Pupil’s Successfulness Factors Using Machine Learning Algorithms and Mathematical Modelling Methods. Advances in Computer Science for Engineering and Education II. ICCSEEA 2019. Advances in Intelligent Systems and Computing, vol 938. Springer. pp 625-636 (2020). DOI 10.1007/978-3-030-16621-2_58