

Methods for finding shortest paths on graphs in organizational and economic systems and their implementation

V M Ramzaev¹, I N Khaimovich^{1,2} and I V Martynov²

¹Samara University of Public Administration «International Market Institute», G S Aksakova Street, 21, Samara, Russia, 443030

²Samara National Research University, Moskovskoe Shosse 34A, Samara, Russia, 443086

e-mail: kovalek68@mail.ru

Abstract. The article implements the functions in Postgre SQL DBMS, finding the shortest paths on graphs, using the wave algorithm method, the Dijkstra's method and the Floyd method. The authors determined models of dependencies of the running time of implementations of the shortest-path search algorithms on graphs on the number of graph vertices experimentally. A comparison of the data obtained as a result of the study was carried out to find the best applications of implementations of the shortest path search algorithms in the Postgre SQL DBMS.

1. Introduction

Modern organizational and economic systems are increasingly using technologies for collecting, analyzing and processing data [1,2], such systems in industrial enterprises include PDM - class systems. In these systems, data is arranged in undirected graphs with different weights. The main task of making decisions in PDM systems is the search for the optimal variant using graph methods [3, 4], this type of search is also actively used for analyzing social networks [5-8]. Currently, issues of the effectiveness of open source software and licenses, which practically do not impose restrictions on its use, are becoming increasingly topical. An example of such software is PostgreSQL DBMS, which by its capabilities comes close to the most common commercial Oracle database. Therefore, the study and analysis of the effectiveness of PostgreSQL DBMS is an important task [9].

In this paper, we study the theoretical foundations of constructing the shortest path search algorithms and the capabilities of the PL / PGSQL programming language built into the PostgreSQL DBMS. To implement the algorithms, databases and software have been developed for finding the shortest paths. The temporal characteristics of the algorithms on the test graphs are investigated and compared with each other.

2. Methods for finding shortest paths on graphs and analyzing the effectiveness of their implementation

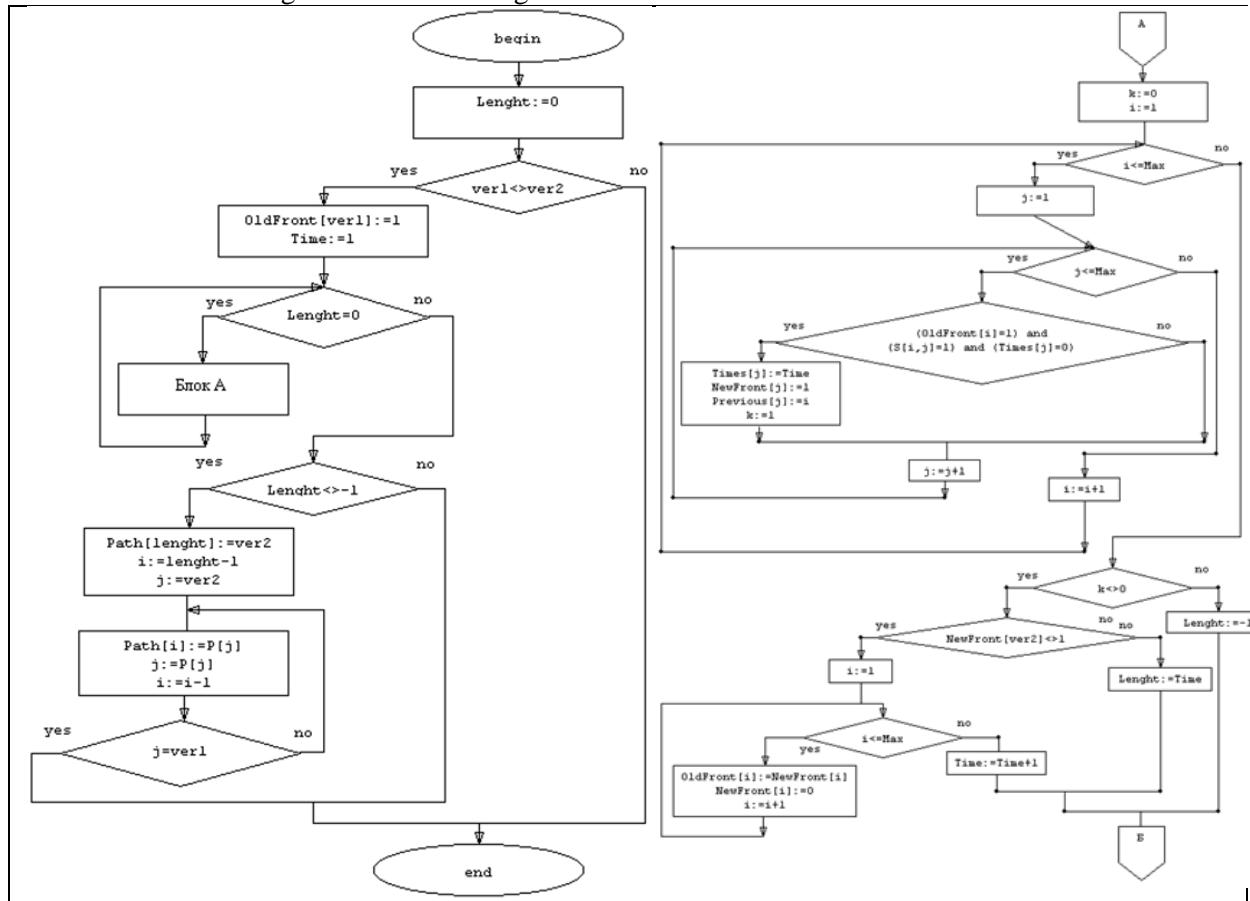
Let's consider the wave algorithm, Dijkstra's algorithm, and Floyd's algorithm for finding the shortest paths on graphs. Wave algorithm (wave-tracing algorithm, Lee algorithm) is an algorithm for finding the path, the algorithm for finding the shortest path on a planar graph. It belongs to algorithms based on wide search methods. The wave algorithm in the context of finding a path through a maze was proposed by E.F. Moore. Lee independently discovered the same algorithm when formalizing the routing algorithms for printed circuit boards in 1961.

In step 1 of the algorithm, the function obtains the number of the initial vertex and calculates labels for all vertices of the graph, the resulting values are written to the nodes table.

In step 2 of the algorithm, the function obtains the number of the final vertex and calculates the shortest route to the initial vertex; the function outputs the result as an array with the vertex numbers.

To implement the search for the shortest path by the wave algorithm method on a given graph, the data are presented in the form of a table that contains the following fields: vertex number of the graph, list of adjacent vertices, path mark (No. of wave front of the wave algorithm), previous vertex number.

Here is a block diagram of the wave algorithm.



To analyze the operation time of the wave algorithm, calculations were performed on test graphs generated by the PostgreSQL DBMS using the function written by the authors. Each graph is a two-dimensional rectangular table of nodes with the size of N rows and M columns. Each node of the grid (except for the nodes of the last column and the last row) is connected by an edge with three nodes lying to the right, below and diagonally to the right-down from the current node. The lengths of all edges are 1.

To clarify, we present the data in the form of a graph (Figure 1).

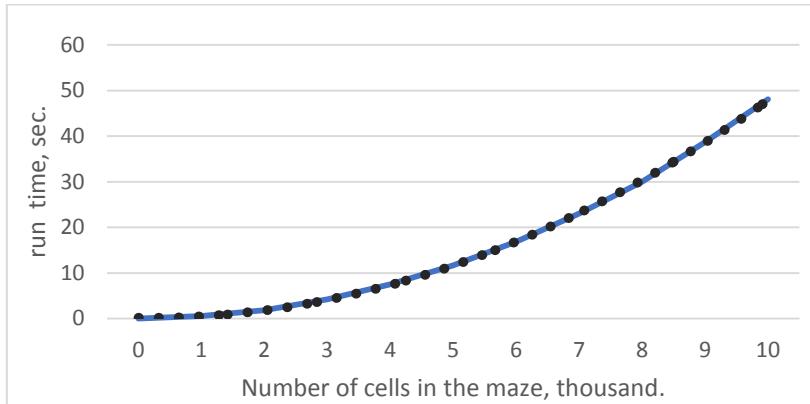


Figure 1. The dependence of the wave algorithm time on the number of vertices.

From the graph it is clear that the dependence is non-linear. We approximate, by a second-order polynomial, the obtained data, to find the function of the dependence of the algorithm time on the number of cells in the maze.

Here is the obtained function:

$$y = 0.47x^2 - 0.19x + 0.19.$$

The R-squared value (determination coefficient) of the current model is 0.9472, which corresponds to a model of good quality. The next shortest path search algorithm is Dijkstra's. This algorithm was proposed by the Dutch scientist Dijkstra in 1959. The algorithm is described in many sources [9].

Dijkstra's algorithm is applicable to an oriented weighted graph with non-negative weights [10].

In step 1 of the algorithm, the initial node S is assigned the status of a permanent label with zero value $d(S)$. This vertex is called basic and denoted by V.

In step 2, the vertices directly reachable from the base vertex V are assigned temporary label values equal to $d(V) + w(V \rightarrow U)$, where $w(V \rightarrow U)$ is the edge weight from the vertex V to the vertex U.

In step 3, among the vertices marked with temporal values, the vertex U is selected with the minimum mark and assigned the status of a permanent mark. The value of the constant label will be equal to the length of the shortest path from the initial vertex S to the vertex U.

In step 4, the vertex U is taken as the base vertex V and the transition to step 2 takes place.

The algorithm ends when all reachable vertices are marked with permanent labels.

To implement the algorithm in the PostgreSQL DBMS, it is necessary to create an Edge table in which the number of the initial vertex for each edge is stored, as well as the number of the final vertex, and the weight of the edge. To restore the path, add the Node table in which everything will be stored.

To implement the algorithm, we use the PL / pgSQL [10] programming language built into the DBMS, in which besides executing SQL queries, we can use standard high-level control instructions for languages.

To analyze the operation time of the Dijkstra algorithm, calculations were made on test graphs. The lengths of all the edges were chosen randomly in the range from 1 to 1,000,000.

For clarity, we present the data in the form of a graph (Figure 2).

The graph shows clearly that the dependence is non-linear. We approximate, by a second-order polynomial, obtained from the data, to find the function of the dependence of the running time of the algorithm on the number of cells in the maze.

Here is the obtained function:

$$y = 1.64x^2 - 0.06x + 0.05.$$

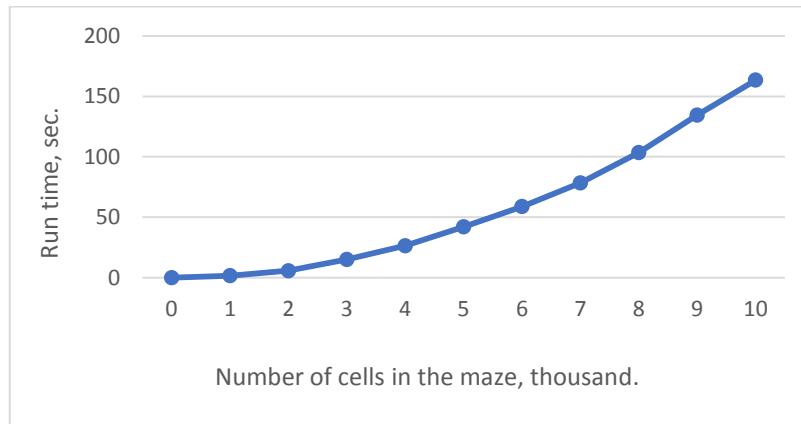


Figure 2. The dependence of the time of the Dijkstra's algorithm on the number of vertices.

The R-squared value (determination coefficient) of the current model is 0.9141, which corresponds to a model of good quality. The next algorithm for finding the shortest paths on graphs is the Floyd's algorithm. This algorithm is a dynamic algorithm for finding the shortest distances between all vertices of a weighted oriented graph. It was developed in 1962 by Robert Floyd and Stephen Warshal.

In step 1, the algorithm calculates and writes into the ADS table of the mD matrix which is the shortest path length matrix; mS is the auxiliary matrix to restore the path.

In step 2, the algorithm will calculate the optimal route from the vertex u to the vertex v of the graph.

To implement the algorithm in PostgreSQL DBMS, create an Edge table in which for each edge we will store the record number, and in fact the task number; the number of vertices of the graph; adjacency matrix; shortest path length matrix; auxiliary matrix to restore the path.

To implement the algorithm, we use the PL / pgSQL [10] programming language built into the DBMS, in which besides executing SQL queries, we can use standard high-level control instructions for languages.

To analyze the Floyd's algorithm time, calculations were made on test graphs. The lengths of all the edges were chosen randomly in the range from 1 to 1,000,000.

To clarify, we present the data in the form of a graph (Figure 3).

From the graph it is clear that the dependence is non-linear. We approximate, by a second-order polynomial, obtained from the data, to find the function of the dependence of the running time of the algorithm on the number of cells in the maze.

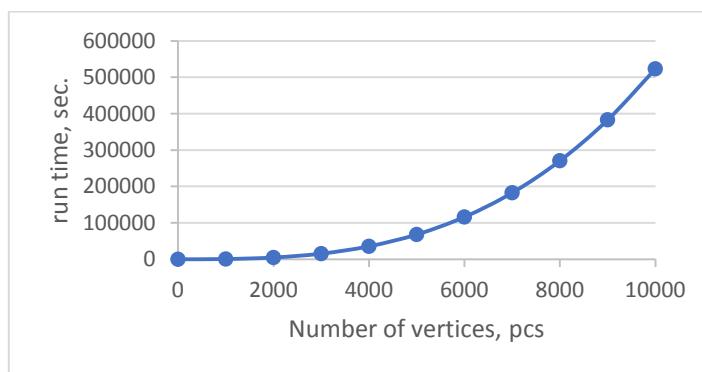


Figure 3. The dependence of the Floyd algorithm time on the number of vertices.

Here is the obtained function:

$$y=5 \cdot 10^{-7} \cdot x^3 + 0.0003x^2 - 0.0735x + 2.8405.$$

The R-squared value (determination coefficient) of the current model is 0.9462, which corresponds to a model of good quality. Let us analyze the time of the wave algorithm and the Dijkstra's algorithm on graphs with unit weights containing from one thousand to ten thousand vertices.

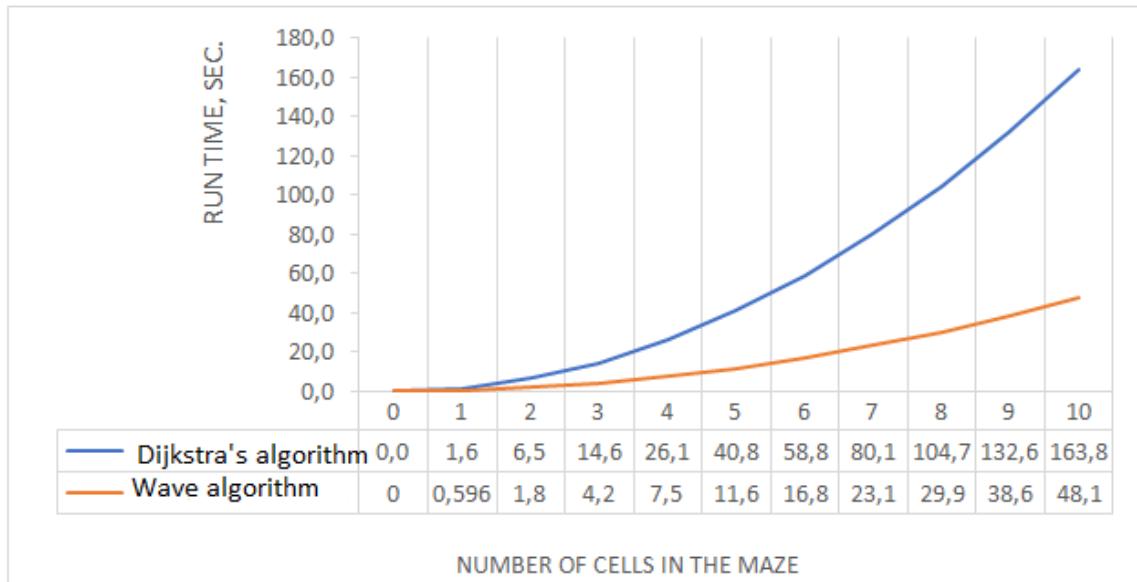


Figure 4. Comparison of the running time of the wave algorithm and the Dijkstra's algorithm.

Let us write the equation of the dependence of the operating time of the proposed implementation of the wave algorithm on the number of vertices in the graph:

$$y=0.47x^2-0.19x+0.19.$$

The R-squared value (determination coefficient) of the current model is 0.8534, which corresponds to a model of good quality. Let us write the equation of the dependence of the operating time of the proposed implementation of the Dijkstra's algorithm on the number of vertices in the graph:

$$y=1.64x^2-0.06x+0.05.$$

The R-squared value (determination coefficient) of the current model is 0.9472, which corresponds to a model of good quality. To calculate the time difference of the proposed implementations of these algorithms, we compare the coefficients at the highest powers in the obtained equations:

$$a = 1.64 / 0.47 = 3.48$$

It turns out that the proposed implementation of the wave algorithm copes with the task by 3.5 times faster than the proposed implementation of the Dijkstra's algorithm. From which it can be concluded that in not oriented graphs with unit weights it is best of all to use the wave algorithm.

Let us consider a comparison of the running time of the Floyd's and Dijkstra's algorithm on test graphs.

Due to the fact that Floyd's algorithm is based on finding the shortest distances between all the graph vertices, and Dijkstra's algorithm for finding the shortest distance from one of the graph vertices

to all others, we will use the following method to compare their operation time. We will compare the time of the Floyd's algorithm and the time of the Dijkstra's algorithm multiplied by n , where n is the number of vertices of the graph.

Let us present data in the form of a Figure 5.

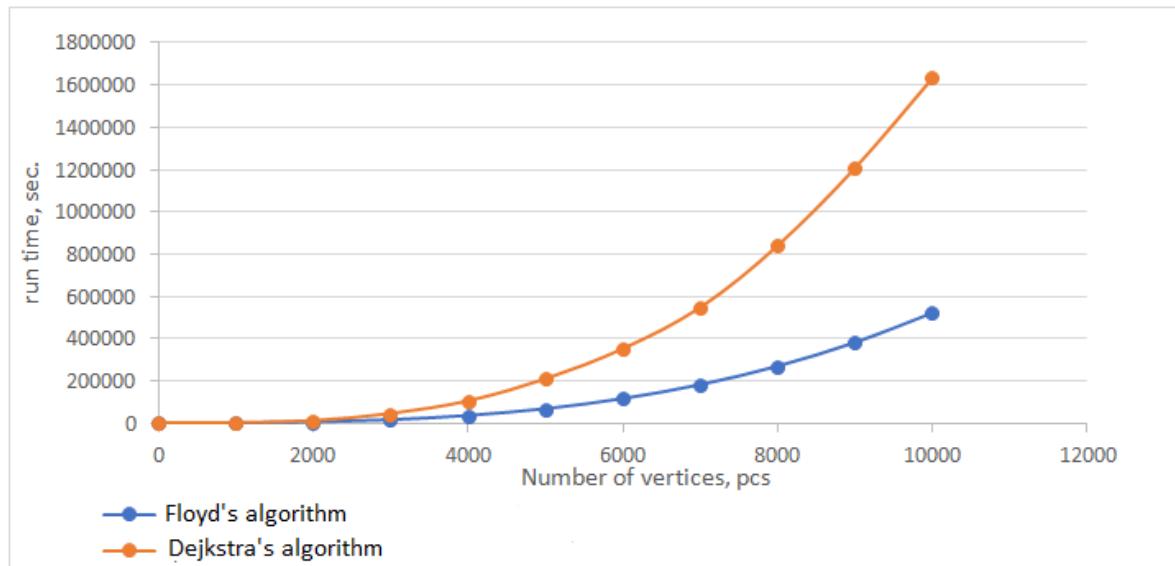


Figure 5. Comparison of the Floyd's algorithm time and the Dijkstra's algorithm time.

Let us write the equation of the dependence of time of the proposed implementation of the Floyd's algorithm on the number of vertices in the graph:

$$y = 5 \cdot 10^{-7}x^3 + 0.0002x^2 - 0.0684x + 0.5874.$$

The R-squared value (determination coefficient) of the current model is 0.9235, which corresponds to a model of good quality. Let us write the equation of the dependence of time of the proposed implementation of the Dijkstra's algorithm on the number of vertices in the graph:

$$y = 2 \cdot 10^{-6}x^3 - 2 \cdot 10^{-5}x^2 - 0.0247x + 47.552.$$

The R-squared value (determination coefficient) of the current model is 0.9242, which corresponds to a model of good quality. To calculate the difference in the operation time of the proposed implementations of these algorithms, compare the coefficients at the highest powers in the obtained equations:

$$a = 2 \cdot 10^{-6} / 5 \cdot 10^{-7} = 4$$

It turns out that the proposed implementation of the Floyd's algorithm copes with the task of finding the shortest path from all the vertices at 4 times faster than the proposed implementation of the Dijkstra's algorithm. From which we can conclude that in such problems it is better to use the Floyd's algorithm. Let us compare the runtime of the Dijkstra's algorithm implementations on the example of implementations in PostgreSQL and C++.

The result of the analysis of the running time of the implementation of the Dijkstra's algorithm in PostgreSQL and C++ is shown in table 1.

For clarity, the test results are illustrated in the Figure. 6.

Table 1. The dependence of processing time on the number of vertices on the graph.

Number of vertices/pcs.	Processing time/sec.	
	PostgreSQL	C++
0	0	0
100	0.3	0
200	0.6	0
500	1.1	0
1000	1.8	0.04
2000	5.7	0.13
5000	42.1	0.75
10000	163.5	40
10500	180.23	73
11000	197.78	115.2
12000	235.49	182.3
14000	320.65	549.9
15000	368.15	809

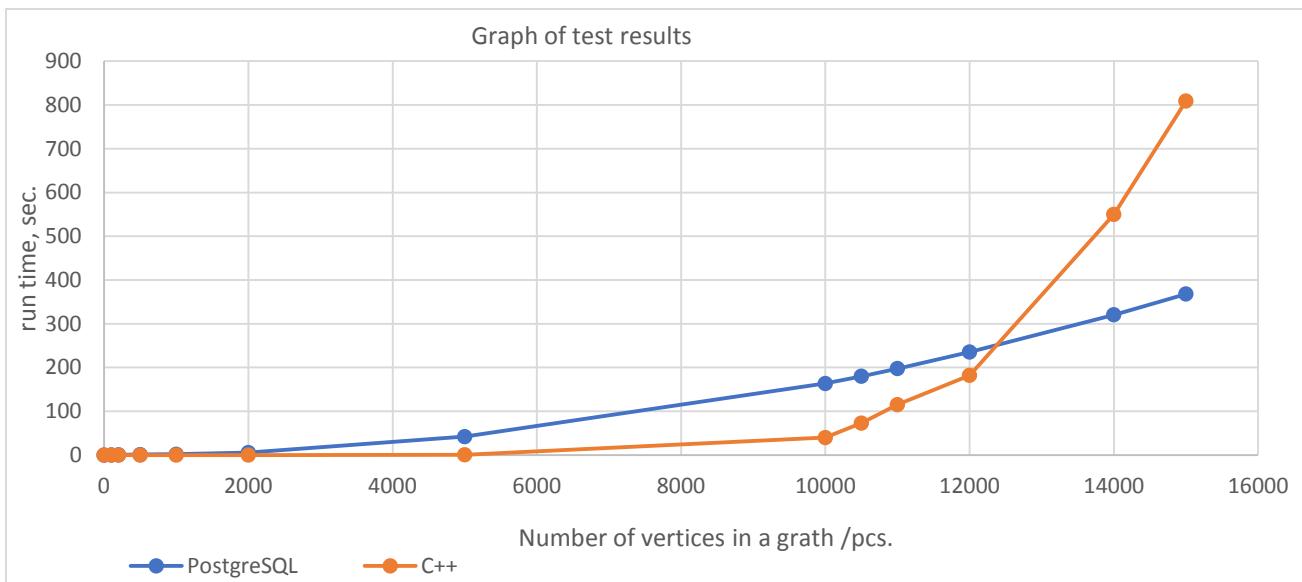


Figure 6. Comparison of the operating time of the implementation of the Dijkstra's algorithm in C ++ and PostgreSQL .

3. Analysis of obtained results

From the research results we can draw the following conclusions:

- the proposed implementation of the wave algorithm is best suited for solving the problems of finding the shortest path in not oriented graphs with unit weights;
- the proposed implementation of the Dijkstra's algorithm is best suited for solving problems of finding the shortest path in not oriented graphs with different weights;
- the proposed implementation of the Floyd's algorithm is best suited for solving problems of finding the shortest path in not oriented graphs with any weights, where the graph remains unchanged for a long time, for example, a graph of roads or a metro system.

Of the algorithmic advantages of solving the problem using the database tools, two points are most significant: specification of the distances to unvisited adjacent vertices within a single UPDATE query and accelerated search for vertices using the indexing tools built into the database engine.

The main disadvantage of this implementation is the relatively low execution speed of single queries to the database, which is noticeable when solving organizational and economic problems of

small and medium dimensions [11 - 14]. The application of data of graph search acceleration algorithms is particularly important in life cycle management systems of industrial products, such as PDM systems. In these systems, new modules have emerged related to the analysis of big data, as in industrial enterprises there are opportunities to integrate design and technological data on all parts and assembly elements of complex innovative products, such as aircraft products and products of the rocket-space complex (about 10 million parts and assembly elements in one product). For complex technological products, we create certifications for parts and working operations associated with loading photographs and measurements into the databases for quality control and destruction during the operation of these products. Such modules in PDM - systems include modules: Siemens TeamCenter 4G Big DATA, Enovia SmarTeam 4G Big Data.

4. References

- [1] Chemodanov D A, Esposito F, Calyam P and Sukhov A 2018 Constrained Shortest Path Scheme for Virtual Network Service Management *IEEE Transactions on Network and Service Management* p 17
- [2] Chemodanov D A, Calyam P, Esposito F and Sukhov A 2016 General Constrained Shortest Path Approach for Virtual Path Embedding *The 22nd IEEE international symposium on local and metropolitan area networks* (Rome, Italy) 1-7
- [3] Khaimovich I N, Ramzaev V M and Chumak V G 2016 Use of big data technology in public and municipal management *CEUR Workshop Proceedings* **1638** 864-872
- [4] Khaimovich I N, Ramzaev V M and Chumak V G 2015 Challenges of data access in economic research based on Big Data technology *CEUR Workshop Proceedings* **1490** 327-337
- [5] Wang Y 2016 A branch-and-price framework for optimal virtual network embedding *Computers Networks* **94** 318-326
- [6] Chowdhury M, Rahman M and Boutaba R 2012 Vineyard: Virtual network embedding algorithms with coordinated node and link mapping *IEEE ACM Trans. Netw.* **20(1)** 206-219
- [7] Esposito F, Paola D and Matta I 2014 On Distributed Virtual Network Embedding with Guarantees *IEEE ACM Trans. Netw.* **24(1)** 569-582
- [8] Danna E, Mandal S and Singh A 2012 A practical algorithm for balancing the max-min fairness and throughput objectives in traffic engineering *Proc. of IEEE INFOCOM* 846-854
- [9] Cormen T, Leiserson C and Rivest R 1990 *Introduction to Algorithms* (Cambridge) p 1091
- [10] Worsley J, Drake J 2003 *Practical PostgreSQL* (O'Reilly Media Inc.) p 640
- [11] Chumak P V, Ramzaev V M and Khaimovich I N 2015 Models for forecasting the competitive growth of enterprises due to energy modernization *Studies on Russian Economic Development* **26(1)** 49-54
- [12] Khaimovich A I, Grechnikov F V 2015 Development of the requirements template for the information support system in the context of developing new materials involving Big Data *CEUR Workshop Proceedings* **1490** 364-375
- [13] Geras'kin M I, Chkhartishvili A G 2017 Analysis of Game-Theoretic Models of an Oligopoly Market under Constraints on the Capacity and Competitiveness of Agents *Automation and Remote Control* **78(11)** 2025-2038
- [14] Kazanskiy N L, Stepanenko I S, Khaimovich A I, Kravchenko S V, Byzov E V and Moiseev M A 2016 Injectional multilens molding parameters optimization *Computer Optics* **40(2)** 203-214 DOI: 10.18287/ 2412-6179-2016-40-2-203-214