Методы поиска кратчайших путей на графах в организационно-экономических системах и их реализация

В.М. Рамзаев[1], И.Н. Хаймович[1,2], И.В. Мартынов[1]

[1]Самарский университет государственного управления «Международный институт рынка», Г.С. Аксакова 21, Самара, Россия, 443030
[2]Самарский национальный исследовательский университет им. академика С.П. Королева, Московское шоссе 34А, Самара, Россия, 443086

Аннотация. В статье реализованы функции, в СУБД Postgre SQL, нахождения кратчайших путей на графах, методом волнового алгоритма, методом Дейкстры и методом Флойда. Экспериментально определены модели зависимостей времени работы реализаций алгоритмов поиска кратчайших путей на графах от количества вершин графа. Проведено сравнение полученных в результате исследования данных, для нахождения наилучших применений реализаций алгоритмов поиска кратчайших путей в СУБД Postgre SQL.

1. Введение

В современных организационно-экономических системах все чаще используются технологии сбора, анализа и обработки данных [1,2] , к таким системам на промышленных предприятиях относятся системы PDM – класса. В этих системах данные располагаются в виде неориентированных графов с различными весами. Основной задачей принятия решений в PDM – системах является поиск оптимального варианта с использованием графовых методов [3,4], также активно данный вид поиска применяется для анализа социальных сетей [5-8]. В настоящее время все более актуальными становятся вопросы эффективности программного обеспечения с открытым исходным кодом и лицензией, которая практически не накладывает ограничений на его использование. Примером такого программного обеспечения является СУБД PostgreSQL, которая по своим возможностям приближается к наиболее распространенной коммерческой СУБД Oracle. Поэтому исследование и анализ эффективности СУБД PostgreSQL является актуальной задачей [9].

В данной работе изучены теоретические основы построения алгоритмов поиска кратчайшего пути и возможности, встроенного в СУБД PostgreSQL языка программирования PL/pgSQL. Для реализации алгоритмов разработаны базы данных и программное обеспечение для поиска кратчайших путей. Исследованы временные характеристики алгоритмов на тестовых графах и проведено их сравнение между собой.

2. Методы поиска кратчайших путей на графах и анализ эффективности их реализации

Рассмотрим волновой алгоритм, алгоритм Дейкстры и алгоритм Флойда для поиска кратчайших путей на графах. Волновой алгоритм (алгоритм волновой трассировки, алгоритм Ли) — алгоритм поиска пути, алгоритм поиска кратчайшего пути на планарном графе. Принадлежит к алгоритмам, основанным на методах поиска в ширину. Волновой алгоритм в

контексте поиска пути в лабиринте был предложен Э. Ф. Муром. Ли независимо открыл этот же алгоритм при формализации алгоритмов трассировки печатных плат в 1961 году.
На шаге 1 алгоритма функция получает номер начальной вершины и вычисляет метки для всех вершин графа, полученные значения записываются в таблицу nodes.
На шаге 2 алгоритма функция получает номер конечной вершины и вычисляет кратчайший маршрут до начальной вершины, результат функция выводит в виде массива с номерами вершин.
Для реализации поиска кратчайшего пути методом волнового алгоритма на заданном графе представим данные в виде таблицы, которая будет содержать следующие поля: номер вершины графа, список соседних вершин, метка пути (№ фронта волны волнового алгоритма), номер предыдущей вершины (Нужен для восстановления пути).
Приведем здесь только текст функции, реализующей шаг 1:

```
create or replace function SetMarks(v0 int)
returns void as $$
declare p int;
declare k int;
declare n int;
begin
update Nodes set mark=0 where num=v0;
k=0;
loop
  p=0;
  for n in (select num from Nodes where mark = k) loop
        p=p+1;
    update Nodes set prev = n, mark = k+1
    where num in (select num from Nodes
                  where mark is null and n = ANY(links));
  end loop;
  exit when p=0;
  k=k+1;
end loop;
end
$$ language plpgsql;
```

Для анализа времени работы волнового алгоритма были проведены расчёты на тестовых графах, генерируемых СУБД PostgreSQL с помощью, написанной авторами функции. Каждый граф представляет собой двумерную прямоугольную таблицу узлов размером N строк и M столбцов. Каждый узел сетки (кроме узлов последнего столбца и последней строки) связан ребром с тремя узлами, лежащими справа, снизу и по диагонали вправо-вниз от текущего узла. Длины всех ребер равны 1.
Для наглядности представим данные в виде графика (рисунок 1).

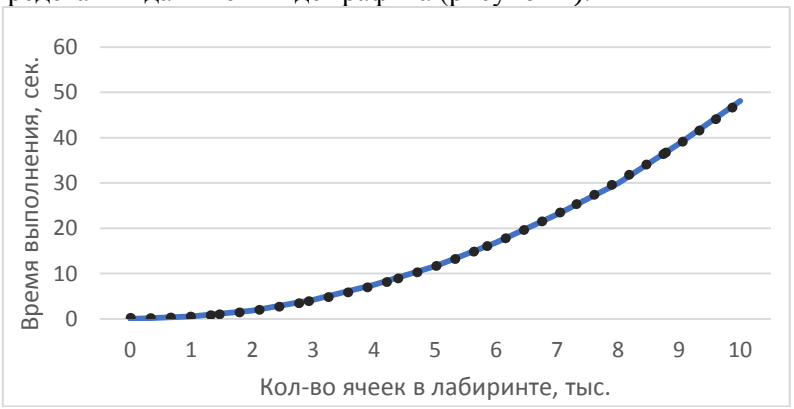


Рисунок 1. Зависимость времени работы волнового алгоритма от количества вершин.

Из графика видно, что зависимость получилось нелинейная. Аппроксимируем, многочленом второго порядка, полученные данные, для нахождения функции зависимости времени работы алгоритма от количества ячеек в лабиринте.
Запишем полученную функцию:

$$y = 0{,}47x^2 - 0{,}19x + 0{,}19.$$

Следующим алгоритмом поиска кратчайших путей является алгоритм Дейкстры. Этот алгоритм предложил голландский ученый Дейкстра в 1959 г. Описание алгоритма приводится во многих источниках [9].
Алгоритм Дейкстры применим к ориентированному взвешенному графу с неотрицательными весами [10].
На шаге 1 алгоритма начальной вершине S присваивается статус постоянной метки с нулевым значением d(S). Эту вершину назовем базовой и обозначим через V.
На шаге 2 вершинам, непосредственно достижимым из базовой вершины V, присваиваются временные значения меток, равные d(V)+w(V→U), где w(V→U) есть вес ребра из вершины V в вершину U.
На шаге 3 среди помеченных временными значениями вершин выбирается вершина U с минимальной меткой и ей присваивается статус постоянной метки. Значение постоянной метки будет равно длине кратчайшего пути из начальной вершины S в вершину U.
На шаге 4 в качестве базовой вершины V принимается вершина U и происходит переход к шагу 2.
Работа алгоритма завершается, когда все достижимые вершины будут помечены постоянными метками.
Для реализации алгоритма в СУБД PostgreSQL создадим таблицу Edge в которой для каждого ребра будем хранить, номер начальной вершины, номер конечной вершины, вес ребра. Для восстановления пути добавим таблицу Node в которой будем хранить
Для реализации алгоритма используем встроенный в СУБД язык программирования PL/pgSQL [10], в котором кроме выполнения SQL-запросов можно использовать стандартные для языков высокого уровня инструкции управления.
Приведем здесь только текст функции, реализующей шаги 2 и 3 описанного выше алгоритма.

```
        -- Запустите алгоритм, пока мы не решим, что мы закончили
      LOOP
    -- Сбросьте переменную, чтобы мы могли обнаружить отсутствие
записей на следующем шаге.
          currentfromnode := NULL;
              -- Выберите Идентификатор и текущую оценку для узла,
который не был выполнен, с самой низкой оценкой.
          SELECT nodeestimate.id, estimate INTO currentfromnode,
currentestimate
             FROM nodeestimate WHERE done = FALSE AND estimate <
  999999999
             ORDER BY estimate LIMIT 1;
          -- Остановитесь, если у нас больше нет невидимых
достижимых узлов.
          IF currentfromnode IS NULL OR currentfromnode = endnode
THEN EXIT; END IF;
          -- Сейчас мы закончили с этим узлом.
```

Для анализа времени работы алгоритма Дейкстры были проведены расчёты на тестовых графах. Длины всех ребер выбирались случайным образом в диапазоне от 1 до 1000000.
Для наглядности представим данные в виде графика (рисунок 2).
Из графика видно, что зависимость получилось нелинейная. Аппроксимируем, многочленом второго порядка, полученные из данные, для нахождения функции зависимости времени работы алгоритма от количества ячеек в лабиринте.

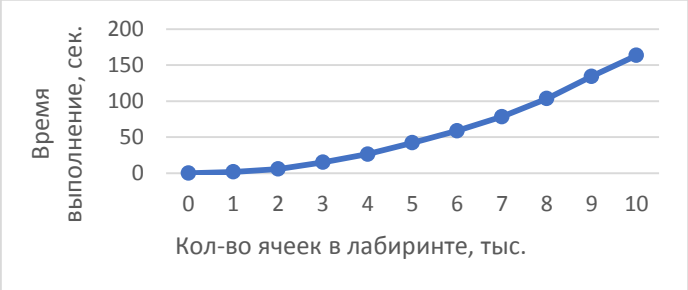


Рисунок 2. Зависимость времени работы алгоритма Дейкстры от количества вершин.

Запишем полученную функцию:

$$y = 1{,}64x^2 - 0{,}06x + 0{,}05.$$

Следующим алгоритм поиска кратчайших путей на графах является алгоритм Флойда. Данный алгоритм — динамический алгоритм для нахождения кратчайших расстояний между всеми вершинами взвешенного ориентированного графа. Разработан в 1962 году Робертом Флойдом и Стивеном Уоршеллом.

На шаге 1 алгоритм вычисляет и записывает в таблицу ADS матрицы mD – матрица длин кратчайших путей; mS – вспомогательная матрица для восстановления пути.

На шаге 2 алгоритм будет вычислять оптимальный маршрут из вершины u в вершину v графа.

Для реализации алгоритма в СУБД PostgreSQL создадим таблицу Edge в которой для каждого ребра будем хранить,номер записи, а фактически номер задачи; количество вершин графа; матрица смежности; матрица длин кратчайших путей; вспомогательная матрица для восстановления пути.

Для реализации алгоритма используем встроенный в СУБД язык программирования PL/pgSQL [10], в котором кроме выполнения SQL-запросов можно использовать стандартные для языков высокого уровня инструкции управления.

Приведем здесь только текст функции, реализующей инициализацию и определение длин кратчайших путей.

```
-- инициализация --
  n = (SELECT nn FROM ADS WHERE id=idnum);
  a = (SELECT mA FROM ADS WHERE id=idnum);
  d = a;
  s = a;
   for i in 1..n loop
    for j in 1..n loop
       s[i][j] = i;
    end loop;
  end loop;
-- определение длин кратчайших путей --
  for i in 1..n loop
    for j in 1..n loop
      for k in 1..n loop
        if d[i][j]>d[i][k]+d[k][j]
        then
           d[i][j]=d[i][k]+d[k][j];
           s[i][j]=s[k][j];
        end if;
      end loop;
    end loop;
  end loop;
```

Для анализа времени работы алгоритма Флойда были проведены расчёты на тестовых графах. Длины всех ребер выбирались случайным образом в диапазоне от 1 до 1000000.

Для наглядности представим данные в виде графика (рисунок 3).

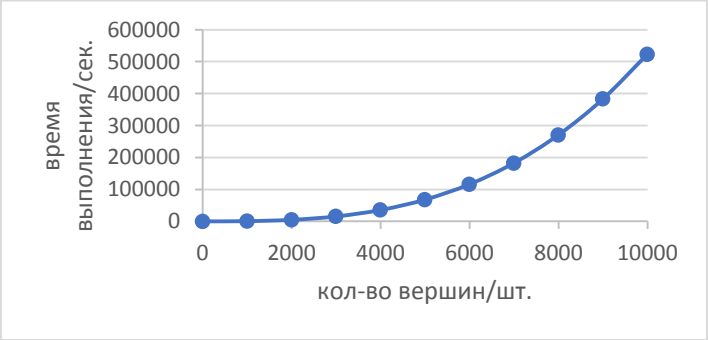
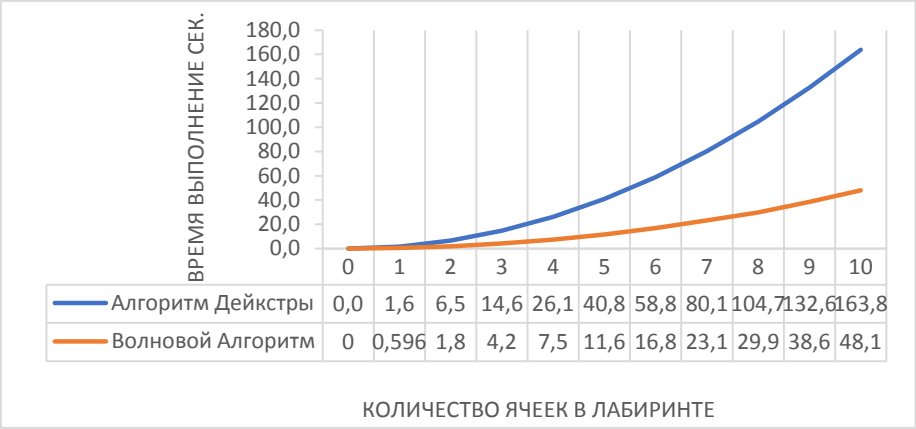


Рисунок 3. Зависимость времени работы алгоритма Флойда от количества вершин.

Из графика видно, что зависимость получилось нелинейная. Аппроксимируем, многочленом второго порядка, полученные из данных, для нахождения функции зависимости времени работы алгоритма от количества ячеек в лабиринте.
Запишем полученную функцию:

$$y = 5 * 10^{-7} * x^3 + 0{,}0003x^2 - 0{,}0735x + 2{,}8405.$$

Проведем анализ времени работы волнового алгоритма и алгоритма Дейкстры на графах с единичными весами содержащие от одной тысячи до десяти тысяч вершин.



	0	1	2	3	4	5	6	7	8	9	10
Алгоритм Дейкстры	0,0	1,6	6,5	14,6	26,1	40,8	58,8	80,1	104,7	132,6	163,8
Волновой Алгоритм	0	0,596	1,8	4,2	7,5	11,6	16,8	23,1	29,9	38,6	48,1

Рисунок 4. Сравнение времени работы волнового алгоритма и алгоритма Дейкстры.

Запишем уравнение зависимости времени работы предложенной реализации волнового алгоритма от количества вершин в графе:

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

Запишем уравнение зависимости времени работы предложенной реализации алгоритма Дейкстры от количества вершин в графе:

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

Для вычисления разницы во времени работы предложенных реализаций данных алгоритмов сравним коэффициенты при старших степенях в полученных уравнениях:

$$a = \frac{1.64}{0.47} = 3.48.$$

Получается, что предложенная реализация волнового алгоритма справляется с поставленной задачей быстрее в 3.5 раза чем предложенная реализация алгоритма Дейкстры. Из чего можно сделать вывод, что в неориентированных графах с единичными весами лучше всего использовать волновой алгоритм.
Рассмотрим сравнение времени работы алгоритма Флойда и Дейкстры на тестовых графах.
В связи с тем, что алгоритм Флойда построен на нахождении кратчайших расстояний между всеми вершинами графа, а алгоритм Дейкстры для нахождения кратчайшего расстояния от одной из вершин графа до всех остальных, для сравнения их времени работы будем

пользоваться следующим методом. Будем сравнивать время работы алгоритма Флойда и время работы алгоритма Дейкстры умноженное на n, где $n -$ количество вершин графа.
Представим данные в виде графика 5.

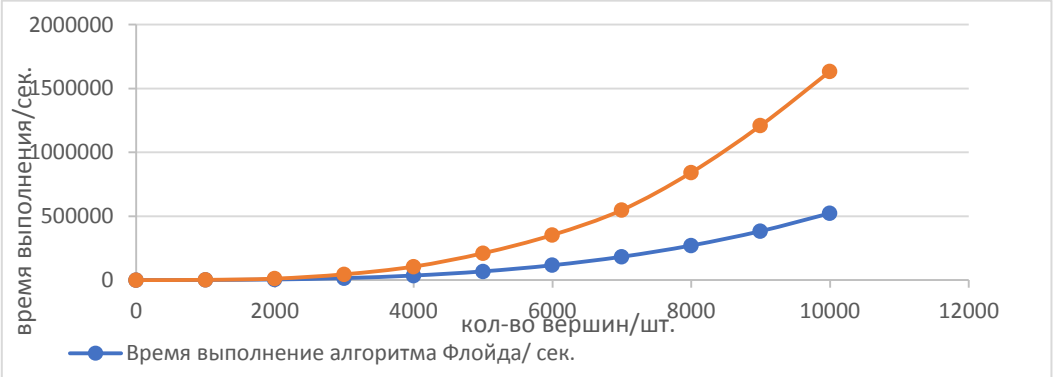


Рисунок 5. Сравнение времени работы алгоритма Флойда и алгоритма Дейкстры.

Запишем уравнение зависимости времени работы предложенной реализации алгоритма Флойда от количества вершин в графе:

$$y = 5 * 10^{-7}x^3 + 0{,}0002x^2 - 0{,}0684x + 0{,}5874.$$

Запишем уравнение зависимости времени работы предложенной реализации алгоритма Дейкстры от количества вершин в графе:

$$y = 2 * 10^{-6}x^3 - 2 * 10^{-5}x^2 - 0.0247x + 47{,}552.$$

Для вычисления разницы во времени работы предложенных реализаций данных алгоритмов сравним коэффициенты при старших степенях в полученных уравнениях:

$$a = \frac{2*10^{-6}}{5*10^{-7}} = 4.$$

Получается, что предложенная реализация алгоритма Флойда справляется с задачей нахождения кратчайшего пути из всех вершин во все быстрее в 4 раза чем предложенная реализация алгоритма Дейкстры. Из чего можно сделать вывод, что в подобных задачах лучше пользоваться алгоритмом Флойда. Проведем сравнение времени работы реализаций алгоритма Дейкстры на примере реализаций в PostgreSQL и C++.
Результат анализа времени работы реализации алгоритма Дейкстры в PostgreSQL и C++ показан в таблице1.

Таблица 1. Зависимость времени обработки от количества вершин на графе.

Количество вершин/шт.	Время работы/сек.	
	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

Для наглядности результаты тестирования проиллюстрированы на графике 6.

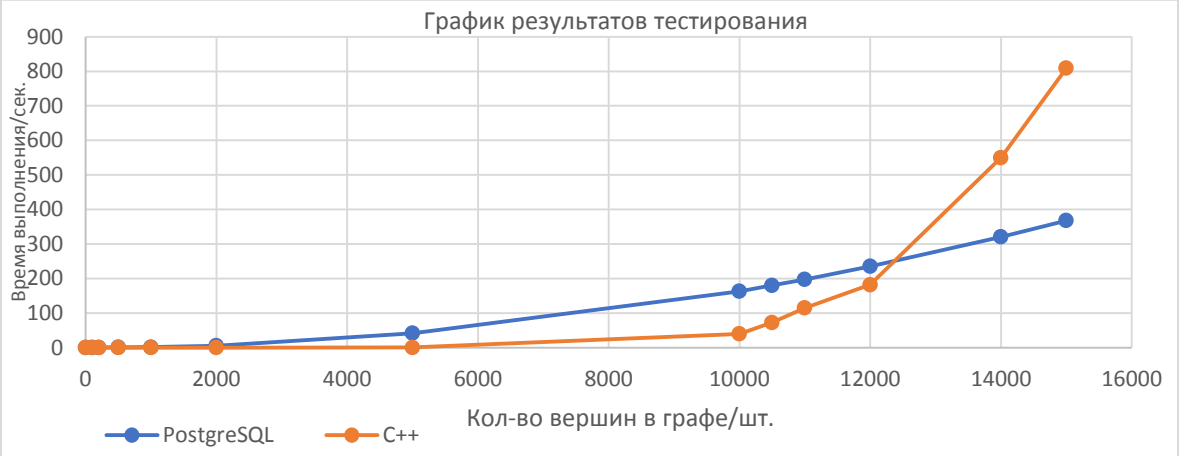


Рисунок 6. Сравнение времени работы реализации алгоритма Дейкстры на C++ и PostgreSQL.

3. Анализ полученных результатов

Из результатов исследований можно сделать следующие выводы:

- предложенная реализация волнового алгоритма лучше всего подходит, для решения задач нахождения кратчайшего пути в неориентированных графах с единичными весами;
- предложенная реализация алгоритма Дейкстры лучше всего подходит, для решения задач нахождения кратчайшего пути в неориентированных графах с различными весами;
- предложенная реализация алгоритма Флойда лучше всего подходит, для решения задач нахождения кратчайшего пути в неориентированных графах с любыми весами, где граф остается неизменным длительное время, например граф дорог или система метро.

Из алгоритмических преимуществ решения задачи средствами БД наиболее существенны два момента: уточнение расстояний до непосещенных смежных вершин в рамках единственного запроса UPDATE и ускоренный поиск вершин посредством встроенных в ядро СУБД инструментов индексирования.

Основным же недостатком данной реализации является сравнительно низкая скорость выполнения единичных запросов к базе данных, что заметно при решении организационно-экономических задач малой и средней размерности [11,12,13].

4. Литература

[1] Chemodanov, D. A Constrained Shortest Path Scheme for Virtual Network Service Management / D. Chemodanov, F. Esposito, P. Calyam, A. Sukhov // IEEE Transactions on Network and Service Management, 2018. – 17 p.

[2] Chemodanov, D. A General Constrained Shortest Path Approach for Virtual Path Embedding / D. Chemodanov, P. Calyam, F. Esposito, A. Sukhov // The 22nd IEEE international symposium on local and metropolitan area networks. – Rome, Italy, 2016. – P. 1-7.

[3] Khaimovich, I.N. Use of big data technology in public and municipal management/ I.N. Khaimovich, V.M. Ramzaev, V.G. Chumak // CEUR Workshop Proceedings. – 2016. –Vol. 1638. – P. 864-872.

[4] Khaimovich, I.N. Challenges of data access in economic research based on Big Data technology / I.N. Khaimovich, V.M. Ramzaev, V.G. Chumak // CEUR Workshop Proceedings. – 2015. –Vol. 1490. – P. 327-337.

[5] Wang, Y. A branch-and-price framework for optimal virtual network embedding / Y. Wang // Computers Networks. – 2016. – Vol. 94. – P. 318-326.

[6] Chowdhury, M. Vineyard: Virtual network embedding algorithms with coordinated node and link mapping / M. Chowdhury, M. Rahman, R. Boutaba // IEEE ACM Trans. Netw. – 2012. – Vol. 20(1). – P. 206-219.

[7] Esposito, F. On Distributed Virtual Network Embedding with Guarantees / F. Esposito, D. Paola, I. Matta // IEEE ACM Trans. Netw. – 2014. – Vol. 24(1). – P. 569-582.
[8] Danna, E. A practical algorithm for balancing the max-min fairness and throughput objectives in traffic engineering / E. Danna, S. Mandal, A. Singh // Proc. of IEEE INFOCOM. – 2012. – P.846-854.
[9] Cormen, T. Introduction to Algorithms / T. Cormen, C. Leiserson, R. Rivest. – Cambridge,1990. – 1091 p.
[10] Worsley, J. Practical PostgreSQL / J. Worsley, J. Drake // O'Relly Media Inc., 2003. – 640 p.
[11] Chumak, P.V. Models for forecasting the competitive growth of enterprises due to energy modernization / P.V. Chumak, V.M. Ramzaev, I.N. Khaimovich // Studies on Russian Economic Development. – 2015. – Vol. 26(1). – P. 49-54.
[12] Khaimovich , A.I. Development of the requirements template for the information support system in the context of developing new materials involving Big Data / F.V. Grechnikov, A.I. Khaimovich // CEUR Workshop Proceedings. – 2015. – Vol. 1490. – P. 364-375.
[13] Geras'kin, M.I. Analysis of Game-Theoretic Models of an Oligopoly Market under Constrains on the Capacity and Competitiveness of Agents / M.I. Geras'kin, A.G. Chkhartishvili // Automation and Remote Control. – 2017. – Vol. 78(11). – P. 2025-2038.

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

V. Ramzaev[1], I. Khaimovich[1,2], I. Martynov[1]

[1]International Market Institute, G.S. Aksakova 21, Samara, Russia, 443030
[2]Samara National Research University, Moskovskoe Shosse 34A, Samara, Russia, 443086

Abstract. The article implements the functions for PostgreSQL DBMS, finding the shortest paths in graphs, using the wave algorithm method, the Dijkstra's method and the Floyd method. The authors determined experimentally the models that show the dependency of the running time of the graph-based shortest-path search algorithms implementation on the number of graph nodes. The authors carried out the comparison of the data obtained as a result of the study to find the best applications of the shortest-path search algorithms implementation in the PostgreSQL DBMS.