УДК 004.6
DOI: 10.26102/2310-6018/2025.51.4.026

Интервальное расширение алгоритма Форда-Фалкерсона поиска максимального потока в транспортной сети и его программная реализация на уровне СУБД

А.И. Мирошников✉, Ю.В. Лубенец

Липецкий государственный технический университет, Липецк, Российская Федерация

Резюме. По своей природе во многих практических задачах исходные данные известны лишь приблизительно или могут изменяться в некоторых пределах. Использование интервальнозначных данных позволяет учитывать существующую неопределенность при моделировании систем, использующих такие значения. Данная работа посвящена разработке и реализации интервального расширения алгоритма Форда-Фалкерсона для решения задачи о максимальном потоке в условиях неопределенности исходных данных. В отличие от классического подхода, где пропускные способности ребер задаются скалярными значениями, предлагаемое решение использует интервальные данные, что позволяет с более высокой точностью моделировать реальные системы с параметрами, известными лишь с определенной точностью. Представлена реализация алгоритма, выполненная на основе реляционной системы управления базами данных PostgreSQL с созданием пользовательского типа данных для работы с интервальными данными. В работе описаны математические основы интервальной арифметики, методы сравнения интервальных величин, а также особенности интеграции алгоритма с системой управления базами данных. Представлены результаты вычислительного эксперимента на тестовом примере. Обсуждаются практические аспекты применения метода в задачах транспортного планирования, управления телекоммуникационными сетями и распределения ресурсов.

Ключевые слова: интервальнозначные данные, интервальная арифметика, алгоритм Форда-Фалкерсона, максимальный поток, потоки в сетях, остаточная пропускная способность, транспортная сеть, пользовательский тип данных, реляционная база данных, система управления базами данных.

Для цитирования: Мирошников А.И., Лубенец Ю.В. Интервальное расширение алгоритма Форда-Фалкерсона поиска максимального потока в транспортной сети и его программная реализация на уровне СУБД. *Моделирование, оптимизация и информационные технологии.* 2025;13(4). URL: https://moitvivt.ru/ru/journal/pdf?id=2071 DOI: 10.26102/2310-6018/2025.51.4.026

Interval extension of the Ford-Fulkerson algorithm for finding the maximum flow in a transport network and its software implementation at the DBMS level

A.I. Miroshnikov✉, Yu.V. Lubenets

Lipetsk State Technical University, Lipetsk, the Russian Federation

Abstract. The initial data in many practical problems are known by their nature only approximately or can vary within certain limits. The use of interval-valued data allows one to take into account the existing uncertainty when modeling systems using such values. The article is devoted to the development and implementation of the interval extension of the Ford-Fulkerson algorithm for solving the maximum flow problem under conditions of uncertainty of the initial data. Unlike the classical approach where the edge capacities are specified by scalar values the proposed solution uses interval data which allows for more accurate modeling of real systems with parameters known only with a certain accuracy. The algorithm

implementation is presented based on the PostgreSQL relational database management system with the creation of a user-defined data type for working with interval data. The paper describes the mathematical foundations of interval arithmetic, methods for comparing interval values, and features of integrating the algorithm with a database management system. The results of a computational experiment on a test example are presented. Practical aspects of the method application in problems of transport planning, telecommunication network management, and resource allocation are discussed.

Keywords: interval-valued data, interval arithmetic, Ford-Fulkerson algorithm, maximum flow, network flows, residual capacity, transport network, user-defined data type, relational database, database management system.

For citation: Miroshnikov A.I., Lubenets Yu.V. Interval extension of the Ford-Fulkerson algorithm for finding the maximum flow in a transport network and its software implementation at the DBMS level. *Modeling, Optimization and Information Technology*. 2025;13(4). (In Russ.). URL: https://moitvivt.ru/ru/journal/pdf?id=2071 DOI: 10.26102/2310-6018/2025.51.4.026

Введение

Задача нахождения максимального потока транспортной сети [1] используется повсеместно в различных областях, где можно анализировать и максимизировать потоки, структуру которых можно описать в виде графа специального вида. Это может быть планирование производства различных товаров или услуг, оптимизация транспортировки грузов или передачи данных в сетях связи.

При транспортировке грузов решается задача определения маршрутов перемещения грузов между складами и торговыми точками. Задача в этом случае заключается в определении максимального потока грузов по дорожной сети при наличии ограничений по пропускной способности для каждого участка дороги.

При планировании устройства компьютерных сетей в телекоммуникационных и вычислительных сетях важной целью является максимальная скорость передачи данных между узлами сети и минимизация задержек. Это достигается перераспределением трафика с учетом пропускной способности каналов таким образом, чтобы минимизировать трафик на отдельных участках каналов связи и избежать перегрузки в узлах связи.

Алгоритмы нахождения максимального потока также могут быть полезны при решении задачи оптимального распределения таких ресурсов, как водные ресурсы, топливо, электроэнергия и т. д. в системах, где имеются ограничения на количество единиц ресурса, передаваемых в единицу времени.

При моделировании транспортных потоков задача нахождения максимального потока транспортной сети состоит в управлении автомобильным движением в условиях ограниченной пропускной способности участков дорог, наличия платных автомагистралей, перекрестков и развязок. Решение этой задачи позволяет, в частности, сократить продолжительность пробок.

Задача о назначениях, которая по сути является частным случаем транспортной задачи, используется, например, в управлении персоналом. Целью решения этой задачи является нахождение оптимального способа распределения задач между сотрудниками или должностями так, чтобы все должности были заняты, а каждая задача выполнялась одним сотрудником. Эта задача решается с помощью алгоритма поиска максимального паросочетания, который является частным случаем задачи о максимальном потоке.

В производственном планировании и управлении задача нахождения максимального потока может заключаться в организации производства таким образом, чтобы максимизировать объем выпускаемой продукции при заданных ограничениях на ресурсы, материалы, сотрудников и производственные линии.

Одним из классических методов решения задачи нахождения максимального потока является алгоритм Форда-Фалкерсона. В данной статье рассматривается расширение этого алгоритма для работы с интервальнозначными данными [2, 3]. Его применение позволит повысить точность расчетов в случае интервального характера исходных данных за счет того, что позволит избежать необходимости усреднения интервалов для использования скалярного представления данных в «классических» расчетах.

Применение интервального расширения алгоритма Форда-Фалкерсона в среде реляционной системы управления базами данных, позволит более эффективно работать с большими объемами данных и использовать преимущества СУБД для хранения и обработки графовых структур. Одним из преимуществ, в частности, является возможность индексирования интервальнозначных данных, которая становится доступной при реализации нового пользовательского интервального типа данных и операций над ним.

Целью работы является разработка интервального расширения алгоритма Форда-Фалкерсона и его реализация в среде СУБД PostgreSQL, обеспечивающей возможность работы с интервальнозначным типом данных.

Материалы и методы

Состояние проблемы. Задача о максимальном потоке и алгоритм Форда-Фалкерсона хорошо изучены в литературе [4, 5]. Классический алгоритм предполагает работу с детерминированными значениями пропускных способностей ребер. Однако в реальных задачах часто встречается неопределенность данных, что требует разработки специальных методов.

В [6] рассматриваются потоки, для которых пропускные способности ребер являются случайными величинами с известными распределениями вероятностей. Этот подход требует знания законов распределения и может быть вычислительно затратным для больших сетей.

Еще один подход к учету неопределенности основан на применении теории нечетких множеств [7]. В данном случае пропускные способности представляются в виде нечетких чисел, а алгоритм оперирует функциями принадлежности. Этот подход позволяет учитывать качественную неопределенность, но может быть сложен в реализации и интерпретации результатов.

Интервальный подход, предлагаемый в данной работе, занимает промежуточное положение между детерминированным и стохастическим подходами. Он не требует знания распределений вероятностей и, в то же время, позволяет количественно оценить неопределенность результата. Интервальные методы хорошо исследованы для задач оптимизации [8, 9], но их применение к задаче о максимальном потоке изучено недостаточно.

Что касается реализации в СУБД, то в литературе описаны различные подходы к хранению и обработке графовых данных в реляционных базах данных [10, 11]. Однако вопросы реализации алгоритмов нахождения максимального потока, в частности, с интервальнозначными данными в среде СУБД практически не освещены.

Интервальная алгебра – это математическая структура, определяющая операции, аналогичные арифметическим операциям для действительных интервалов.

Над интервалами $[a] = [\underline{a};\ \overline{a}]$ и $[b] = [\underline{b};\ \overline{b}]$ определены следующие основные операции (1)–(4):

$$[a] + [b] = \left[\underline{a} + \underline{b};\ \overline{a} + \overline{b}\right], \quad (1)$$

$$[a]-[b]=\left[\underline{a}-\overline{b};\ \overline{a}-\underline{b}\right], \quad (2)$$

$$[a][b]=\left[\min\left\{\underline{ab},\underline{a}\overline{b},\overline{a}\underline{b},\overline{ab}\right\},\max\left\{\underline{ab},\underline{a}\overline{b},\overline{a}\underline{b},\overline{ab}\right\}\right], \quad (3)$$

$$\frac{[a]}{[b]}=[a]\left[\frac{1}{[\overline{b}]};\ \frac{1}{[\underline{b}]}\right],\{0\}\notin[b]. \quad (4)$$

Особое значение в интервальной алгебре имеет вопрос сравнения двух интервальных значений. Оно может проводиться с использованием численного показателя $R([x]\le[y])=\frac{mid[y]-mid[x]}{rad[x]+rad[y]}$, где $mid[x]=\frac{\underline{x}+\overline{x}}{2}$ – медиана интервала; $rad[x]=\frac{\overline{x}-\underline{x}}{2}$ – радиус интервала.

Некоторые параметры, характеризующие взаимное расположение сравниваемых интервалов:

– интервал $[a]$ целиком лежит справа от интервала $[b]$;

– интервал $[a]$ целиком лежит слева от интервала $[b]$;

– середина интервала $[a]$ находится справа от середины интервала $[b]$;

– середина интервала $[a]$ находится слева от середины интервала $[b]$.

Используя только эти данные о взаимном расположении границ двух интервальных значений, необходимо определить признак, характеризующий событие «выбранное из интервала $[a]$ значение больше или равно выбранному из интервала $[b]$».

Это можно сделать, задав граничное значение вероятности, которое определяется в зависимости от значения показателя *R*:

$$P([x]\le[y])=\begin{cases}\dfrac{(rad[x]+rad[y])^2}{8rad[x]rad[y]}(1+R([x]\le[y]))^2,\\ \text{если } R([x]\le[y])<-\dfrac{|rad[x]-rad[y]|}{rad[x]+rad[y]};\\ \dfrac{1}{2}+\dfrac{rad[x]+rad[y]}{2rad[y]}R([x]\le[y]),\\ \text{если } |R([x]\le[y])|\le\dfrac{|rad[x]-rad[y]|}{rad[x]+rad[y]};\\ 1-\dfrac{(rad[x]+rad[y])^2}{8rad[x]rad[y]}(1-R([x]\le[y]))^2;\\ \text{если } R([x]\le[y])>\dfrac{|rad[x]-rad[y]|}{rad[x]+rad[y]}.\end{cases}$$

В этом случае алгоритм сравнения двух интервалов принимает следующий вид:

1) задание вероятности *p*;

2) расчет показателя $R([x]\le[y])$;

3) вычисление значения вероятности $P([x]\le[y])$;

4) если $P([x]\le[y])>p$, то результат операции сравнения TRUE, иначе – FALSE.

Вывод и обоснование применимости подхода приведены в [12].

Сеть определяется как (*G, u, s, t*), где *G* – ориентированный простой граф, *u* – пропускная способность дуг, вершины *s* и *t* являются источником (начальным узлом) и стоком (конечным узлом) графа.

В классическом определении сетей значения пропускных способностей дуг принадлежат $\mathbb{R}$. В предлагаемом подходе они представляют собой интервальные значения и, следовательно, результирующий максимальный поток также будет

представлен в интервальном виде. В связи с этим, пропускная способность потоковой сети определяется как:

$$u: E(G) \to \left[\underline{a};\ \overline{a}\right], \text{где } \{\underline{a};\ \overline{a}\} \in \mathbb{R}. \quad (5)$$

Поток графа определяется как функция:

$$f: E(G) \to \left[\underline{a};\ \overline{a}\right], \text{где } \{\underline{a};\ \overline{a}\} \in \mathbb{R}, \quad (6)$$

такая что $f(e) \le u(e)$ для всех $e \in \mathrm{E}(G)$.

Таким образом, в случае интервальных значений, *u* и *f* определяются как функции, отображающие *E*(*G*) на $\left[\underline{a};\ \overline{a}\right]$.

Избыточный поток *f* в вершине $v \epsilon V(G)$ равен:

$$val(f) = ex_f(v) = \sum_{e\in\delta^-(v)} f(e) - \sum_{e\in\delta^+(v)} f(e). \quad (7)$$

Тогда задача о максимальном потоке может быть сформулирована как максимизация значения $val(f)$ при условии $f(e) \le u(e),\ \forall e \in \mathrm{E}(G)$.

Остаточная пропускная способность определяется как:

$$u_f: E(G) \to \left[\underline{a};\ \overline{a}\right], \text{где } \{\underline{a};\ \overline{a}\} \in \mathbb{R},$$

$$u_f(e) = u(e) - f(e).$$

Функция u_f также отображает *E*(*G*) на $\left[\underline{a};\ \overline{a}\right]$.

Увеличение потока *f* вдоль пути *P* на величину *γ*:

$$\forall e\ \in P \quad f(e) = f(e_0) + \gamma_f. \quad (8)$$

Для соответствующего обратного ребра:

$$\forall e^{\leftarrow} \in P \quad f(e^{\leftarrow}) = f(e_0^{\leftarrow}) - \gamma_f. \quad (9)$$

Путь *P* будет называться увеличивающим (дополняющим) путем. В этом пути остаточная пропускная способность $u_f(e) > 0, \forall e\ \in P$.

Таким образом, интервальное расширение алгоритма Форда-Фалкерсона получения максимального *s*-*t* потока определяется следующим образом:

Шаг 1. Для каждого ребра $e \in E(G)$ выполняется инициализация нулевым интервалом $f(e) = [0]$. Вид нулевого интервала определяется с учетом метрики.

Для обратных ребер: $f(e^{\leftarrow}) = u(e)$.

Общий поток: $val(f) = [0]$.

Шаг 2. Поиск увеличивающего пути *P* от источника к стоку, по которому можно пропустить ненулевой поток. Если такого пути не существует, алгоритм завершается. Поиск пути может осуществляться любым алгоритмом, например, обхода в глубину.

Шаг 3. Выбор наименьшего ребра пути («bottleneck edge» – бутылочное горлышко, узкое место):

$$\gamma = min_{e\in E(P)} u_f(e).$$

Функция поиска минимума должна учитывать метрику.

Шаг 4. Изменение остаточной пропускной способности ребер пути:

$$u_f(e) = u_f(e_0) - \gamma,$$

$$u_f(e^{\leftarrow}) = u_f(e_0^{\leftarrow}) + \gamma.$$

Шаг 5. Изменение потока ребер пути:

$$f(e) = f(e_0) + \gamma,$$
$$f(e^{\leftarrow}) = f(e_0^{\leftarrow}) - \gamma.$$

Шаг 6. Обновление общего потока:

$$val(f) = val(f_0) + \gamma.$$

Шаг 7. Переход к шагу 2.

Результаты

В [13] описана программная реализация интервального типа данных на языке C# применительно к СУБД MS SQL Server. В этой СУБД реализация пользовательского типа данных осуществляется в динамически подключаемой библиотеке классов, которая затем подключается к базе данных как сборка и регистрируется в рамках этой базы данных. Такой подход позволяет использовать тип данных «интервал» наравне со встроенными типами данных, включая их индексацию, при переопределении операции сравнения двух интервалов, а также учитывать все его особенности с точки зрения интервальной алгебры, которые встроенные типы данных не позволяют.

На Рисунке 1 представлен один из конструкторов базового класса «iInterval», использование которого позволяет определять операции (1)–(4), операции сравнения интервалов, агрегатные функции и настраивать индексацию по столбцу этого типа в СУБД.

```
public iInterval(double down, double up): this()
{
    if (down >= up) {
        this = Null;
        isNull = true;
    }
    else {
        isNull = false;
        this.down = Math.Round(down, 14, MidpointRounding.ToEven);
        this.up = Math.Round(up, 14, MidpointRounding.AwayFromZero);
    }
}
```

Рисунок 1 – Объявление нового пользовательского интервальнозначного типа данных в среде СУБД MS SQL Server
Figure 1 – Declaration of a new user-defined interval-valued data type in the MS SQL Server DBMS environment

Реализация интервального расширения алгоритма Форда-Фалкерсона на уровне СУБД учитывает особенности PostgreSQL:

– интервальнозначный тип данных и операции над ним определены в динамически подключаемой библиотеке, написанной на языке программирования C;

– для использования нового типа данных все функции импортируются SQL-скриптом, далее определяется тип данных и его поведение;

– алгоритм Форда-Фалкерсона реализуется с помощью процедурного расширения языка SQL – PL/pgSQL.

Граф хранится в базе данных, как список смежности. Каждая запись представляет собой пару вершин и величину пропускной способности ребра, представленную в интервальном виде.

Для выполнения расчетов создается копия графа, в которой значения обновляются на каждой итерации алгоритма. Для построения пути используется обход в ширину. Возможно использование других алгоритмов поиска пути.

Путь хранится во временной таблице в поле типа «массив» для вершин и остаточной пропускной способности соответствующих вершин. Это позволяет зафиксировать количество столбцов для каждой записи.

Создание нового пользовательского типа данных «iInterval» в СУБД PostgreSQL имеет свои особенности и представлено на Рисунке 2.

```
CREATE TYPE iInterval(
    internallength = 16,
    input = iInterval_in,
    output = iInterval_out,
    receive = iInterval_recv,
    send = iInterval_send,
    alignment = double
);
```

Рисунок 2 – Объявление нового пользовательского интервального типа средствами SQL в СУБД PostgreSQL

Figure 2 – Declaring a new user-defined interval type using SQL in PostgreSQL DBMS

Для его объявления требуется указать:

– длину внутреннего хранения типа (16 байт);

– имя функции для преобразования входных данных (из строки) во внутреннее представление (interval_in);

– имя функции для преобразования внутреннего представления в выходной формат (в строку) (interval_out);

– функцию для преобразования из двоичного формата;

– функцию для преобразования в двоичный формат;

– значение выравнивания в памяти (double – 8 байт).

Обсуждение

Рассмотрим граф *G*, представленный на Рисунке 4, содержащий 9 вершин и 14 дуг. Вершина «1» является начальной (источник), а вершина «9» – конечной (сток). Веса графа представлены в виде интервальных значений.

В качестве типа данных для столбца «capacity» применяется пользовательский тип данных «iInterval» (Рисунок 3). На Рисунке 5 показан формат хранения графа *G* в виде таблицы в базе данных под управлением СУБД PostgreSQL.

```
CREATE TABLE edge_list (
    source int NOT NULL,
    target int NOT NULL,
    capacity iInterval NOT NULL
);
```

Рисунок 3 – Использование нового пользовательского типа данных «iInterval» при создании таблицы

Figure 3 – Using a new user defined interval data type when creating a table

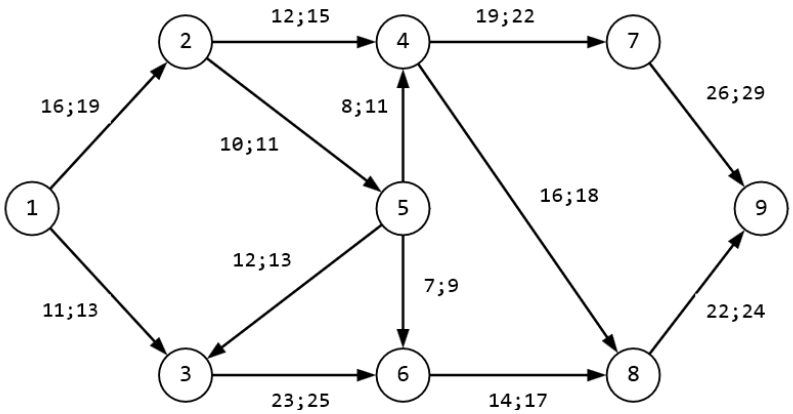

Рисунок 4 – Ориентированный граф с весами, представленными интервальными значениями
Figure 4 – Directed graph with weights represented by interval values

```
SELECT * FROM edge_list;
postgres   |  source | target | capacity
postgres   | --------+--------+----------
postgres   |       1 |      2 | 16;19
postgres   |       1 |      3 | 11;13
postgres   |       2 |      4 | 12;15
postgres   |       2 |      5 | 10;11
postgres   |       3 |      6 | 23;25
postgres   |       4 |      7 | 19;22
postgres   |       4 |      8 | 16;18
postgres   |       5 |      3 | 12;13
postgres   |       5 |      4 | 8;11
postgres   |       5 |      6 | 7;9
postgres   |       6 |      8 | 14;17
postgres   |       7 |      9 | 26;29
postgres   |       8 |      9 | 22;24
postgres   | (13 rows)
```

Рисунок 5 – Формат хранения ориентированного взвешенного графа в таблице СУБД PostgreSQL
Figure 5 – Directed weighted graph storage format in PostgreSQL DBMS table

Суммарный максимальный поток, полученный в результате применения интервального расширения алгоритма Форда-Фалкерсона для указанного графа, равен (24; 35).

Заключение

Статья посвящена разработке интервального расширения алгоритма Форда-Фалкерсона. Проанализированы существующие подходы к решению задачи о поиске максимального потока в сети в условиях неопределенности исходных данных. Предложен подход, учитывающий эту неопределенность с использованием интервальных значений.

Возможности СУБД PostgreSQL позволяют реализовать алгоритм, используя определенный пользователем интервальнозначный тип данных. Реляционные базы данных обеспечивают удобство хранения значений и высокую скорость доступа к ним, что позволяет решать задачи большой размерности. Разработан пользовательский тип данных «iInterval» и реализована интервальная модификация алгоритма Форда-Фалкерсона на уровне СУБД.

СПИСОК ИСТОЧНИКОВ / REFERENCES

1. Cormen Th.H., Leiserson Ch.E., Rivest R.L., Stein C. *Introduction to Algorithms*. Cambridge: MIT Press; 2022. 1332 p.
2. Шарый С.П. *Конечномерный интервальный анализ*. Новосибирск: XYZ; 2025. 671 с.
3. Allahviranloo T., Pedrycz W., Esfandiari A. *Advances in Numerical Analysis Emphasizing Interval Data*. Boca Raton: CRC Press; 2022. 218 p. https://doi.org/10.1201/9781003218173
4. Ahuja R.K., Magnanti Th.L., Orlin J.B. *Network Flows: Theory, Algorithms, and Applications*. New Jersey: Prentice Hall; 1993. 846 p.
5. Korte B., Vygen J. *Combinatorial Optimization*: *Theory and Algorithms.* Berlin, Heidelberg: Springer; 2012. 660 p. https://doi.org/10.1007/978-3-642-24488-9
6. Ding S. The α-Maximum Flow Model with Uncertain Capacities. *Applied Mathematical Modelling*. 2015;39(7):2056–2063. http://doi.org/10.1016/j.apm.2014.10.021
7. Liu B. *Theory and Practice of Uncertain Programming*. Berlin, Heidelberg: Springer; 2009. 202 p. https://doi.org/10.1007/978-3-540-89484-1
8. Hansen E., Walster G.W. *Global Optimization Using Interval Analysis*. New York: Marcel Dekker; 2004. 489 p.
9. Mayer G. *Interval Analysis and Automatic Result Verification*. Berlin, Boston: Walter de Gruyter GmbH; 2017. 532 p.
10. Sakr Sh. Storing and Querying Graph Data Using Efficient Relational Processing Techniques. In: *Information Systems: Modeling, Development, and Integration: Third International United Information Systems Conference: Volume 20, 21–24 April 2009, Sydney, Australia.* Berlin, Heidelberg: Springer; 2009. P. 379–392. http://doi.org/10.1007/978-3-642-01112-2_39
11. Angles R., Gutierrez C. Survey of Graph Database Models. *ACM Computing Surveys*. 2008;40(1). https://doi.org/10.1145/1322432.1322433
12. Погодаев А.К., Галкин А.В., Сараев П.В., Мирошников А.И. Сравнение интервальных типов данных в системе MS SQL Server. *Системы управления и информационные технологии*. 2018;(1):68–72.
    Pogodaev A.K., Galkin A.V., Saraev P.V., Miroshnikov A.I. Interval Data Type Comparison in MS SQL Server. *Sistemy upravleniya i informatsionnye tekhnologii*. 2018;(1):68–72. (In Russ.).
13. Pogodaev A.K., Galkin A., Saraev P., Miroshnikov A.I. The Development of Interval Data Type for Analytical Information Processing. In: *Recent Developments and the New Direction in Soft-Computing Foundations and Applications: Selected Papers from the 7th World Conference on Soft Computing, 29–31 May 2018, Baku, Azerbaijan*. Cham: Springer; 2021. P. 509–520. https://doi.org/10.1007/978-3-030-47124-8_41

ИНФОРМАЦИЯ ОБ АВТОРАХ / INFORMATION ABOUT THE AUTHORS

Мирошников Артем Игоревич, кандидат технических наук, доцент кафедры прикладной математики и системного анализа, Липецкий государственный технический университет, Липецк, Российская Федерация.
e-mail: a.i.miroshnikov@yandex.ru
ORCID: 0000-0003-0639-105X

Artem I. Miroshnikov, Candidate of Engineering Sciences, Associate Professor at the Department of Applied Mathematics and Systems Analysis, Lipetsk State Technical University, Lipetsk, the Russian Federation.

Лубенец Юрий Владимирович, кандидат физико-математических наук, доцент, доцент кафедры прикладной математики и системного анализа, Липецкий государственный технический университет, Липецк, Российская Федерация.
e-mail: yuv791@gmail.com
ORCID: 0000-0002-6951-2536

Yuri V. Lubenets, Candidate of Physical and Mathematical Sciences, Docent, Associate Professor at the Department of Applied Mathematics and Systems Analysis, Lipetsk State Technical University, Lipetsk, the Russian Federation.

Статья поступила в редакцию 08.09.2025; одобрена после рецензирования 11.10.2025; принята к публикации 24.10.2025.

The article was submitted 08.09.2025; approved after reviewing 11.10.2025; accepted for publication 24.10.2025.