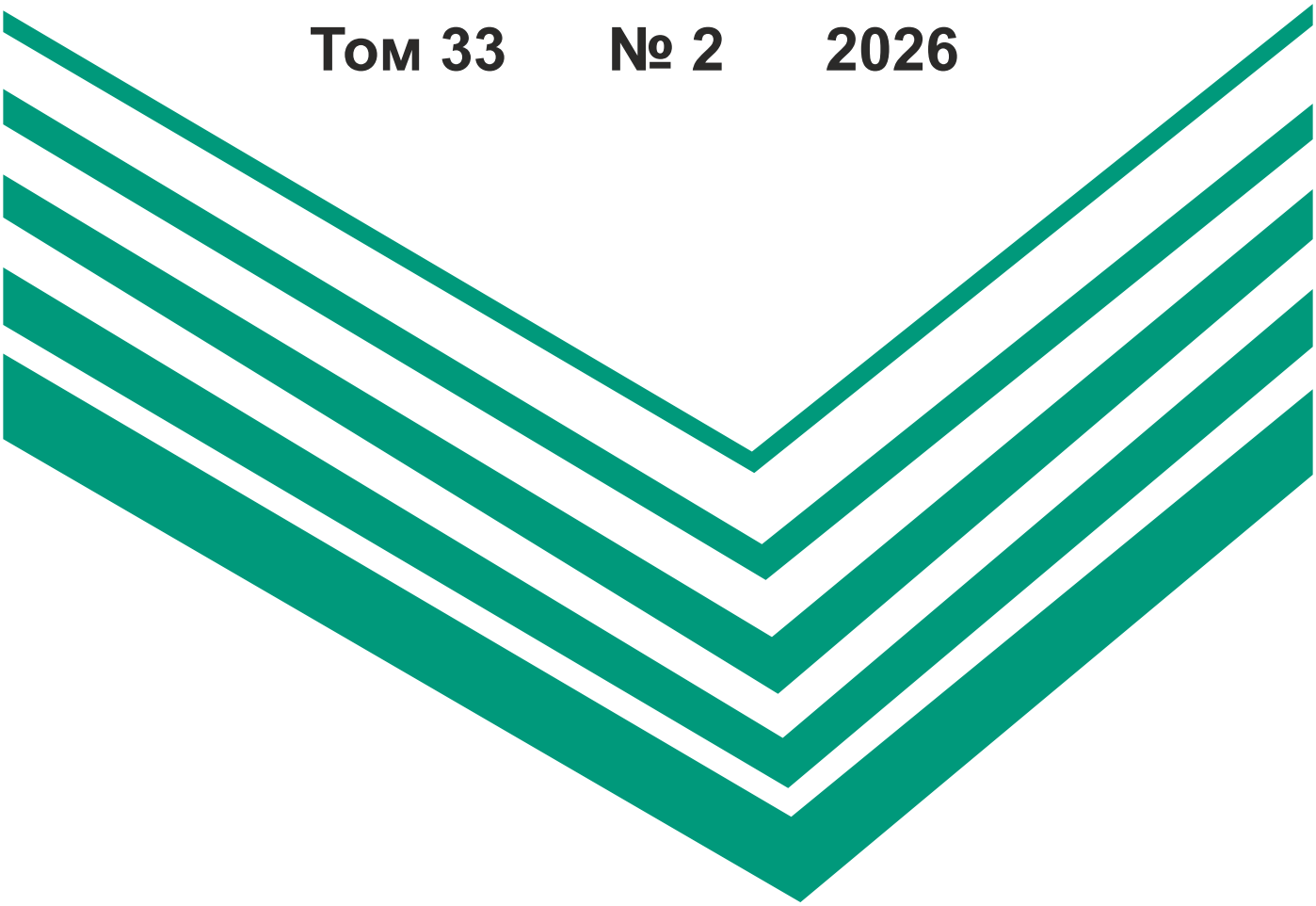
ISSN 2949-5598

ДИСКРЕТНЫЙ АНАЛИЗ И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ

Том 33 № 2 2026



Новосибирск
Издательство Института математики

РЕДАКЦИОННАЯ КОЛЛЕГИЯ

Главный редактор	**В. Л. Береснев**
Зам. главного редактора	**А. А. Евдокимов**
Ответственный секретарь	**Ю. В. Шамардин**

С. В. Августинович	**М. Я. Ковалёв**	**А. В. Пяткин**
Г. П. Агибалов	**А. В. Кононов**	**А. А. Сапоженко**
В. Б. Алексеев	**А. В. Косточка**	**М. Свириденко**
О. В. Бородин	**В. В. Кочергин**	**Б. Я. Рябко**
В. А. Васильев	**Ю. А. Кочетов**	**Н. Н. Токарева**
Э. Х. Гимади	**В. К. Леонтьев**	**Ю. А. Флеров**
А. Ю. Григорьев	**Б. М.-Т. Лин**	**Ф. В. Фомин**
С. Демпе	**В. В. Лозин**	**М. Ю. Хачай**
А. И. Ерзин	**П. Пардалос**	**Я. М. Шафранский**

Учредители журнала Сибирское отделение РАН
Институт математики им. С. Л. Соболева СО РАН

Журнал включён в базу данных Russian Science Citation Index (RSCI) на платформе Web of Science. Переводы статей на английский язык публикуются в *Journal of Applied and Industrial Mathematics* и доступны по ссылке `www.springer.com/mathematics/journal/11754`.

СИБИРСКОЕ ОТДЕЛЕНИЕ РОССИЙСКОЙ АКАДЕМИИ НАУК
ИНСТИТУТ МАТЕМАТИКИ им. С. Л. СОБОЛЕВА СО РАН

ДИСКРЕТНЫЙ АНАЛИЗ И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ

Выпускается с 1994 г. Научный журнал 4 номера в год

Том 33, № 2 (168) **Апрель–июнь 2026**

СОДЕРЖАНИЕ

Гимади Э. Х., Гончаров Е. Н. *Об одном методе ранжирования ресурсов для задачи календарного планирования с ограниченными ресурсами* **5**

Драчев Р. Р., Панин А. А., Плясунов А. В. *Верхние и нижние границы и алгоритмы спуска с чередующимися окрестностями для задачи конкурентного ценообразования* **19**

Жуков Г. А., Ерзин А. И., Плотников Р. В. *Метод ветвей и отсечений для задачи взвешенной 3-раскраски графа* **46**

Монахова Э. А., Монахов О. Г. *Поиск семейств субоптимальных циркулянтных сетей с помощью больших языковых моделей* **64**

Попов В. Ю. *О вычислительной сложности нахождения хребтов булевых формул* **82**

Пяткин А. В., Сафарова А. И. *Четырёхмерный булев куб не 3-представим* **97**

Утюпин С. Ю. *Устойчивость вершинных покрытий в задаче о вечном вершинном покрытии* **115**

Фомичёв В. М. *Локальная нелинейность степеней преобразования* **136**

НОВОСИБИРСК
ИЗДАТЕЛЬСТВО ИНСТИТУТА МАТЕМАТИКИ

В журнале публикуются оригинальные научные статьи и обзоры теоретической и прикладной направленности по следующим разделам дискретного анализа, исследования операций и информатики:

- дискретная оптимизация
- комбинаторика
- контроль и надёжность дискретных устройств
- математические модели и методы принятия решений
- математическое программирование
- модели экономики
- моделирование процессов управления
- построение и анализ алгоритмов
- синтез и сложность управляющих систем
- теория автоматов
- теория графов
- теория игр и её приложения
- теория кодирования
- теория расписаний и размещений

Адрес редакции:
Институт математики
им. С. Л. Соболева СО РАН,
пр. Академика Коптюга, 4,
630090 Новосибирск, Россия
Телефон: +7 (383) 329–75–79
E-mail: discopr@math.nsc.ru

© Институт математики им. С. Л. Соболева СО РАН, 2026

SIBERIAN BRANCH OF THE RUSSIAN ACADEMY OF SCIENCES
SOBOLEV INSTITUTE OF MATHEMATICS

DISKRETNYI ANALIZ I ISSLEDOVANIE OPERATSII

/DISCRETE ANALYSIS AND OPERATIONS RESEARCH/

Published since 1994 | Scientific journal | 4 issues per year

Vol. 33, No. 2 (168) | **April–June, 2026**

CONTENTS

É. Kh. Gimadi and **E. N. Goncharov.** *On a method for ranking resources for the resource-constrained project scheduling problem* **5**

R. R. Drachev, A. A. Panin, and **A. V. Plyasunov.** *Upper and lower bounds and variable neighbourhood descent for competitive pricing problem* **19**

G. A. Zhukov, A. I. Erzin, and **R. V. Plotnikov.** *Branch-and-cut method for the weighted 3-coloring problem* **46**

E. A. Monakhova and **O. G. Monakhov.** *Search for families of suboptimal circulant networks based on AI* **64**

V. Yu. Popov. *On the computational complexity of finding the backbones of Boolean formulas* **82**

A. V. Pyatkin and **A. I. Safarova.** *The four-dimensional Boolean cube is not 3-representable* **97**

S. Yu. Utyupin. *Stability of vertex covers in the eternal vertex cover problem* **115**

V. M. Fomichev. *Local nonlinearity of transformation iterations* **136**

NOVOSIBIRSK
SOBOLEV INSTITUTE PRESS

In this journal we publish original research papers and survey papers of both theoretical and practical importance on the following topics of discrete analysis, operations research and informatics:

- discrete optimization
- combinatorics
- control and reliability of discrete devices
- decision making models and methods
- mathematical programming
- economic models
- management modeling
- design and analysis of algorithms
- synthesis and complexity of control systems
- automata theory
- graph theory
- game theory and its applications
- coding theory
- theory of scheduling and facility location

Editorial office address:
Sobolev Institute of Mathematics,
4 Acad. Koptyug Avenue,
630090 Novosibirsk, Russia
Phone: +7 (383) 329–75–79
E-mail: discopr@math.nsc.ru

© Sobolev Institute of Mathematics SB RAS, 2026

ДИСКРЕТНЫЙ АНАЛИЗ И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ
Апрель–июнь 2026. Т. 33, № 2. С. 5–18

УДК 519.8+518.25 DOI: 10.33048/daio.2026.33.864

ОБ ОДНОМ МЕТОДЕ РАНЖИРОВАНИЯ РЕСУРСОВ ДЛЯ ЗАДАЧИ КАЛЕНДАРНОГО ПЛАНИРОВАНИЯ С ОГРАНИЧЕННЫМИ РЕСУРСАМИ

Э. Х. Гимади[a], *Е. Н. Гончаров*[b]

Институт математики им. С. Л. Соболева,
пр. Акад. Коптюга, 4, 630090 Новосибирск, Россия
E-mail: [a]gimadi@math.nsc.ru, [b]gon@math.nsc.ru

Аннотация. Рассматривается задача календарного планирования с ограниченными ресурсами по критерию минимума длины расписания. Все ресурсы возобновимы, прерывания операций (работ) не допускаются. Эта задача NP-трудна в сильном смысле. Ограниченность ресурсов лежит в основе сложности её решения. К числу перспективных для решения этой задачи относят эвристические алгоритмы. В данной работе предлагается ранжировать ресурсы по степени их критичности (дефицитности). Эвристические алгоритмы могут использовать это ранжирование ресурсов для построения сравнительно лучших эвристических решений. Ранжирование ресурсов получаем из релаксированной задачи, в которой ресурсы складируемы. Предложенный метод ранжирования ресурсов был применён в генетическом алгоритме. Вычислительные эксперименты на примерах из электронной библиотеки PSPLIB показали его эффективность. Для нескольких примеров из PSPLIB найдены новые лучшие (ранее неизвестные) решения. Табл. 1, библиогр. 17.

Ключевые слова: задача календарного планирования с ограниченными ресурсами, возобновимые ресурсы, PSPLIB.

Введение

Рассматривается задача календарного планирования с ограниченными ресурсами (resource-constrained project scheduling problem, RCPSP). Эта задача может быть представлена следующим образом. Дан проект с множеством N операций (работ). Каждая операция j имеет длительность p_j. Прерывание операций не допускается. В связи с технологическими требованиями между некоторыми операциями существуют отношения предшествования. Они задаются наборами непосредственных предшественников P_j, указывающими, что операция j не может быть

© Э. Х. Гимади, Е. Н. Гончаров, 2026

начата до завершения каждого её предшественника из P_j. Имеется множество K типов возобновляемых ресурсов. Для каждого ресурса k в каждый единичный момент времени выделяется одинаковое количество ресурса R_k. Каждая операция j требует r_{jk} единиц ресурса типа k в каждом единичном интервале времени, в течение которого она выполняется. К числу имеющихся добавим две дополнительные операции $j = 0$ и $j = |N| + 1$, представляющие начало и завершение проекта соответственно. Обе являются «фиктивными» операциями с длительностью 0 и без запросов на ресурсы.

Предполагается, что вся информация детерминированная и известна заранее, а параметры неотрицательны и целочисленны. Расписание представляет собой назначение неотрицательного момента начала s_j каждой операции $j = 0, 1, \ldots, |N|+1$. Целью задачи RCPSP является нахождение расписания, которое приводит к кратчайшему возможному моменту завершению проекта. Время завершения проекта в расписании называется также временем выполнения расписания.

По классификации, предложенной в [1], представленная выше задача обозначается через $PS|prec|C_{\max}$, а согласно классификационной схеме из [2] — через $m, 1|cpm|C_{\max}$.

Поскольку задача RCPSP NP-трудна в сильном смысле [3], известные точные методы решения, такие как метод ветвей и границ, не подходят для решения крупномасштабных задач из-за значительного времени вычислений, поэтому точные методы неприемлемы для решения практических задач. Для решения примеров RCPSP большого размера основной интерес для исследований представляют эвристические процедуры и метаэвристики. Многие эвристические процедуры основаны на правилах приоритета и методах forward-backward improvement (FBI). К числу метаэвристических алгоритмов, предложенных для RCPSP, могут быть отнесены генетический алгоритм (ГА), поиск с запретами, имитация отжига, оптимизация муравьиной колонии, перелинковка путей (the path re-linking) и разнообразные гибридные алгоритмы. Численные эксперименты, проведённые на стандартных наборах примеров PSPLIB [4], показали высокую эффективность этих метаэвристических методов.

Задача RCPSP имеет обширное применение на практике, поэтому она привлекала к себе внимание многих исследователей на протяжении нескольких последних десятилетий. В прошлом опубликован ряд обзоров, связанных с RCPSP. Детерминированные варианты задачи и методы их решения были рассмотрены в [1]. Перечисление детерминированных вариантов и расширение базовой модели приведено в [5], но без методов решения. Эвристические методы для базового варианта рассмотрены в [6]. Обзор робастных и реактивных процедур планирования дан

в [7]. Обзор по задачам планирования проектов с ограниченными ресурсами в условиях неопределённости представлен в [8], а обстоятельный обзор гибридных метаэвристик для RCPSP — в [9]. Упомянем также обзоры [10–12].

Как утверждается в [13], решение ресурсных конфликтов лежит в основе задачи RCPSP, и многие исследователи также отмечали, что наличие ограниченных ресурсов является определяющим фактором для сложности задачи. В работе [13] авторы анализировали сложность задачи RCPSP с экспериментальной точки зрения и предложили подход к преобразованию простых задач RCPSP в очень сложные. При этом количество операций и ресурсов в таких примерах должно быть как можно меньше. Одним из шагов указанного преобразования является сокращение количества ресурсов, необходимое для поиска узких мест, определяющих сложность задачи.

Поиск узких мест среди ограниченных ресурсов являлся главной темой в работе [14]. В отличие от [13], где обнаруженные ресурсные узкие места использовались для построения трудных примеров, в [14] производилось ранжирование ресурсов по степени их дефицитности. Оба эти подхода в своей основе используют поиск узких мест задачи по использованию ресурсов, и в этом они перекликаются. Полученное ранжирование применялось далее для построения эвристического приближённого решения жадным алгоритмом. На каждом шаге этого алгоритма учитывалось применение более приоритетных ресурсов и рассматривались в первую очередь использующие их операции. В генетическом алгоритме [15] использовался оператор скрещивания родительских хромосом, а для выявления перспективных генов в этих хромосомах также применялся этот способ ранжирования ресурсов. Численные эксперименты показали эффективность генетического алгоритма с применением ранжирования ресурсов по степени их дефицитности.

В настоящей статье предлагается новый способ ранжирования ресурсов по степени их дефицитности. Для этого рассматривается релаксированная задача RCPSP со складируемыми (cumulative) ресурсами. Эта задача полиномиально разрешима [16]. В [17] был предложен модифицированный алгоритм задачи с улучшенной оценкой временной сложности. После нахождения решения релаксированной задачи мы проведём детальный анализ динамики потребления ресурсов, сравнивая её с динамикой выделения этих ресурсов. Таким образом фиксируются узкие места задачи, связанные с ресурсами, и выявляются ресурсы в этой релаксированной задаче, критически важные в каждый конкретный момент времени. В результате мы ранжируем ресурсы по степени их дефицитности, а затем делаем предположение, что такое ранжирование ресурсов

для релаксированной задачи будет также сохраняться и для исходной задачи.

Фактически предлагаемый метод ранжирования ресурсов развивает идеи из [15]. Если ранее обращалось внимание лишь на интегральные остатки неиспользованных ресурсов по окончании решения релаксированной задачи, то теперь изучается характер потребления ресурсов в динамике, что позволяет более аккуратно выявлять ресурсные коллизии.

Качество предложенного метода ранжирования ресурсов для RCPSP протестировано на серии численных экспериментов. Для их проведения использован генетический алгоритм [15], в операции кроссовера которого задействовано эвристическое правило ранжирования ресурсов. Тестовые примеры для эксперимента взяты из электронной библиотеки тестовых примеров PSPLIB [4]. Для тестирования использованы примеры из тестовых множеств j90 и j120 размерностью 90 и 120 операций соответственно.

Ниже показано, что с помощью предложенного метода ранжирования ресурсов получены лучшие решения в сравнении с алгоритмами, не использующими ранжирование ресурсов, а также использующими применённый ранее [14] интегральный способ ранжирования ресурсов. Генетический алгоритм с предложенным правилом ранжирования ресурсов получил лучшие решения для нескольких десятков примеров из j90 и j120. Для одного примера из j90 и четырёх примеров из j120 найдены лучшие (ранее неизвестные) эвристические решения. Эти решения внесены в таблицу лучших эвристических решений, расположенную на сайте PSPLIB.

1. Постановка задачи

Задачу RCPSP можно сформулировать следующим образом. Рассмотрим проект как ориентированный ациклический граф $G = (N, A)$, в котором N — множество операций проекта, а $A \subseteq N \times N$ — множество пар, отражающее отношение предшествования между операциями. Положим $N = \{1, \ldots, n\} \cup \{0, n+1\}$, при этом фиктивные операции 0 и $n+1$ задают начало и завершение всего комплекса операций. Операции $1, \ldots, n$ будем называть фактическими. Отношение предшествования на N определим множеством пар $A = \{(i, j) \mid i \in P_j\}$. Если $(i, j) \in A$, то операция j не может начаться раньше завершения операции i.

Через K обозначим множество типов ресурсов, необходимых для выполнения операций. Все ресурсы предполагаются возобновимыми. В каждый единичный временной интервал $[t-1, t)$, $t \in \mathbb{Z}_+$, для каждого типа ресурсов выделяется фиксированный объём этого ресурса $R_k \in \mathbb{Z}_+$.

Фактическая операция j имеет определённую длительность своего выполнения $p_j \in \mathbb{Z}_+$ и требует в каждую единицу $[t-1, t)$, $t \in \mathbb{Z}_+$, времени своего выполнения $r_{jk} \geqslant 0$ единиц ресурса типа $k \in K$. С целью избежать неразрешимости задачи будем предполагать, что $r_{jk} \leqslant R_k$, $j \in N$, $k \in K$.

Фиктивные операции 0 и $n+1$ имеют нулевую длительность и нулевое потребление ресурсов. Фактические операции имеют ненулевую целочисленную длительность.

Введём переменные задачи. Через $s_j \geqslant 0$ обозначим момент начала выполнения j-й операции. Так как операции выполняются без прерывания, момент окончания j-й операции определяется равенством $c_j = s_j + p_j$. Определим расписание S как $(n+2)$-вектор $(s_0, \ldots, s_{n+1})$. Время завершения проекта $T(S)$ соответствует моменту завершения последней операции проекта, т. е. $T(S) = c_{n+1}$. Через $J(t, S) = \{j \in N \mid s_j \leqslant t < c_j\}$ обозначим множество операций, выполняемых в единичном интервале времени $[t-1, t)$, $t \in \mathbb{Z}_+$ при расписании S. Задача заключается в нахождении допустимого расписания $S = \{s_j\}$ с минимальным временем завершения проекта $T(S)$. Её можно формализовать следующим образом: минимизировать время завершения проекта

$$T(S) = c_{n+1} \tag{1}$$

при ограничениях

$$s_i + p_i \leqslant s_j, \quad (i, j) \in A, \tag{2}$$

$$\sum_{j \in J(t,S)} r_{jk} \leqslant R_k, \quad k \in K,\ t \in \mathbb{Z}_+ \colon J(t, S) \neq \varnothing, \tag{3}$$

$$s_j \in \mathbb{Z}_+, \quad j \in N. \tag{4}$$

Неравенства (2) определяют ограничения предшествования операций, а (3) обеспечивают соблюдение ресурсных ограничений с возобновимыми ресурсами: суммарное количество ресурса типа k, потребляемое всеми операциями, выполняемыми в единичном интервале времени $[t-1, t)$, не должно превышать имеющегося в наличии количества этого ресурса в данный момент времени. Наконец, (4) определяет искомые переменные.

2. Ранжирование ресурсов

Идея нахождения относительно более перспективных ресурсов (их ранжирования по степени дефицитности) состоит в использовании релаксированной задачи для (1)–(4). Релаксация состоит в том, что теперь все ресурсы будем считать складируемыми (cumulative). По решению релаксированной задачи определяем для неё ранжирование ресурсов. Далее делаем предположение, что ранжирование ресурсов для релаксированной задачи будет также сохраняться и для исходной задачи.

Подобная идея была реализована в [14, 15]. В отличие от предлагаемого в данной работе метода ранжирования, там применялся интегральный способ определения дефицитных ресурсов, исходя из общих остатков неиспользованных ресурсов. Этот метод показал свою эффективность

при проведении численных экспериментов. Однако, он имеет и слабые стороны. Вполне возможно, что некий ресурс интегрально не проявил себя критически дефицитным, но на локальных отрезках времени именно он оказался тем узким местом, которое не позволило сократить общую длину расписания. Новый подход рассматривает процесс построения расписания в динамике, и тем самым он лишён этого недостатка.

Для задачи (1)–(4) рассмотрим её релаксацию, в которой условие возобновимости ресурсов ослаблено до условия их складируемости: минимизировать время завершения проекта

$$T(S) = c_{n+1} \tag{5}$$

при ограничениях

$$s_i + p_i \leqslant s_j, \quad (i,j) \in A, \tag{6}$$

$$\sum_{t'=1}^{t} \sum_{j \in J(t',S)} r_{jk} \leqslant \sum_{t'=1}^{t} R_k, \quad k \in K,\ t \in \mathbb{Z}_+ \colon J(t,S) \neq \varnothing, \tag{7}$$

$$s_j \in \mathbb{Z}_+, \quad j \in N. \tag{8}$$

Неравенства (7) обеспечивают соблюдение ограничений для складируемых ресурсов, а соотношения (5)–(8) представляют собой задачу календарного планирования со складируемыми ресурсами.

Отметим, что задача RCPSP с ограничениями на ресурсы складируемого типа (7) по трудоёмкости решения отличается от задачи с ограничениями с возобновимыми ресурсами (3). Если все ресурсы только складируемого типа, то для довольно широкого класса задач (с целочисленными длительностями операций) задача полиномиально разрешима [16]. В [17] был предложен модифицированный алгоритм для релаксированной задачи (5)–(8) с улучшенной оценкой временной сложности, равной $O(|A| + n \log_2 n + n|K|))$, где n — число операций, K — множество типов ресурсов, $|A|$ — число пар операций, между которыми установлены отношения предшествования. Применительно к рассматриваемой постановке задачи мощность множества A имеет оценку $O(n^2)$, и тогда оценку временной сложности задачи (5)–(8) можно представить как $O(n(n + |K|))$.

Опишем алгоритм ранжирования ресурсов для релаксированной задачи (5)–(8).

Шаг 1. Решаем задачу (5)–(8). В результате решения получаем оптимальную длину расписания T^* и моменты начала выполнения операций в оптимальном расписании $S^* = \{s_j^*\}$.

Шаг 2. Для оптимального расписания S^* находим остатки неиспользованных ресурсов в каждый момент времени:

$$\rho_{kt} = \sum_{t'=1}^{t} R_k - \sum_{t'=1}^{t} \sum_{j \in J^*(t', S^*)} r_{jk}, \quad k \in K,\ t \in \overline{1, T^*}.$$

Здесь $J^*(t, S^*)$ — множество операций, выполняемых в единичном интервале времени $[t-1, t)$ при расписании S^*:

$$J^*(t, S^*) = \{j \in N \mid s_j^* \leqslant t < s_j^* + p_j\}.$$

Шаг 3. Для каждого ресурса k вычисляем минимальную по всем моментам времени t величину $\rho_k^{\min}$ минимальных остатков неизрасходованных ресурсов:

$$\rho_k^{\min} = \min_{t \in \overline{1, T^*}} \rho_{kt}, \quad k \in K.$$

Шаг 4. Сортируем по неубыванию $\rho_k^{\min}$, $k \in K$. Результат сортировки является искомым ранжированием ресурсов: ресурс с минимальным значением $\rho_k^{\min}$ будем считать наиболее дефицитным, а с максимальным значением — наименее дефицитным. В случае равенства минимальных остатков более дефицитным будем считать ресурс с меньшим интегральным значением неизрасходованных ресурсов ρ_{kT^*}.

Заметим, что значение величин ρ_{kt} получается в ходе работы алгоритма на шаге 1 в виде разности двух следующих величин: суммарного количества выделенных и суммарного количества потреблённых ресурсов. Это же замечание справедливо и для шага 3 алгоритма: минимальное количество неиспользованных ресурсов для всех моментов времени также может быть получено в ходе решения релаксированной задачи. Наконец, само ранжирование ресурсов на шаге 4 требует сортировки массива $\rho_k^{\min}$, $k \in K$, по неубыванию, оно выполняется за $O(|K| \log |K|)$ действий. Следовательно, общая временна́я сложность алгоритма равна $O(n(n + |K|) + |K| \log |K|))$.

3. Численные эксперименты

Для исследования эффективности предложенного метода ранжирования ресурсов использованы наборы тестовых примеров из электронной библиотеки PSPLIB [4], доступной по ссылке `www.om-db.wi.tum.de/psplib`. Мы использовали множества тестовых примеров j60, j90 и j120. Наборы j60 и j90 содержат примеры с 60 и 90 операциями в каждом соответственно, по 48 серий примеров в каждом наборе и по 10 экземпляров в каждой серии — всего по 480 примеров в каждом из этих наборов. Набор данных j120 содержит 60 серий тестовых примеров по 10 экземпляров в каждой серии — всего 600 экземпляров.

Пусть R_{din} — массив ресурсов в порядке их ранжирования предложенным в данной работе динамическим методом, а R_{int} — соответствующий массив ранжирования, полученный интегральным методом. Понятно, что на некоторых примерах ранжирование может совпадать, и в таких случаях у нас не будет предпосылок для обнаружения новых лучших решений за счёт более аккуратного ранжирования. В первом численном эксперименте находим на примерах из PSPLIB оба ранжирования и производим их сравнение по числу возможных совпадений.

Таблица 1

Сравнение ранжирований R_{din} и R_{int} на примерах из PSPLIB

Набор примеров из PSPLIB	Общее число примеров	Число примеров с $R_{\text{din}} = R_{\text{int}}$	Процент совпадений R_{din} и R_{int}	Число примеров с $R_{\text{din}} \neq R_{\text{int}}$	Процент несовпадений R_{din} и R_{int}
j60	480	116	24,2	364	75,8
j90	480	137	28,5	343	71,5
j120	600	152	25,3	448	74,5

Как видно из табл. 1, около трёх четвертей всех примеров из PSPLIB имеют другое, более тонко настроенное ранжирование ресурсов.

В следующем численном эксперименте исследуем эффект влияния предложенного метода ранжирования ресурсов на решения, получаемые использующим его метаэвристическим алгоритмом, в качестве которого применялся генетический алгоритм [15]. Были использованы примеры из наборов j90 и j120 электронной библиотеки PSPLIB.

Оператор кроссовера этого алгоритма действует по следующей схеме. В обоих родительских хромосомах находятся перспективные участки (гены) по принципу наилучшего использования ресурсов. Под наилучшим использованием ресурсов понимается минимальный объём неиспользованных ресурсов (потерь) среди возобновимых ресурсов. Затем эти перспективные гены передаются дочерней хромосоме. Теперь, имея ранжирование ресурсов по степени их дефицитности, стремимся передать дочерней хромосоме те гены, которые по возможности используют наилучшим способом именно более дефицитные ресурсы. Генетический алгоритм совершил две серии экспериментов по числу шагов алгоритма: 50 000 шагов и 500 000 шагов.

Численные эксперименты показали, что на примерах из набора j90 генетический алгоритм при 50 000 шагов отыскал решения с меньшей длиной расписания для 46 примеров. На примерах из j120 удалось найти 63 лучших решений. При 500 000 шагах алгоритма было найдено 15 и 29 лучших решений соответственно.

Для одного примера из j90 (пример j9013_3) удалось получить новое рекордное (неизвестное ранее) эвристическое решение. Для множества примеров j120 были обновлены решения для четырёх примеров: X11_5, X16_2, X16_3 и X51_1. Эти обновлённые решения размещены в таблице лучших эвристических решений на сайте PSPLIB.

Заключение

Для задачи календарного планирования с ограниченными ресурсами по критерию минимизации длины расписания предложен новый метод ранжирования ресурсов по степени их дефицитности. Наличие такого ранжирования ресурсов позволяет эвристическим алгоритмам и метаэвристикам находить сравнительно лучшие эвристические решения. Для определения более дефицитных ресурсов рассматривается релаксированная задача RCPSP со складируемыми ресурсами. Эта задача полиномиально разрешима. Получив решение релаксированной задачи, мы производим анализ динамики потребления ресурсов, фиксируем узкие места задачи, связанные с ресурсами, и ранжируем ресурсы по степени их дефицитности. Затем делается предположение, что такое ранжирование ресурсов для релаксированной задачи будет также сохраняться и для исходной задачи. Этот метод опробован в рамках численных экспериментов. Для проведения экспериментов использован генетический алгоритм, в операции кроссовера которого задействовано разработанное эвристическое правило ранжирования ресурсов. Тестовые примеры взяты из электронной библиотеки тестовых примеров PSPLIB [4]. Численные эксперименты показали эффективность предложенного метода.

Финансирование работы

Работа выполнена в рамках государственного задания Института математики им. С. Л. Соболева (проект № FWNF–2026–0021). Дополнительных грантов на проведение или руководство этим исследованием получено не было.

Конфликт интересов

Авторы заявляют, что у них нет конфликта интересов.

Литература

1. **Brucker P., Drexl A., Möhring R., Neumann K., Pesch E.** Resource-constrained project scheduling: Notation, classification, models, and methods // Eur. J. Oper. Res. 1999. V. 112, No. 1. P. 3–41.
2. **Herroelen W., Demeulemeester E., De Reyck B.** A classification scheme for project scheduling // Project scheduling: Recent models, algorithms and applications. Dordrecht: Kluwer Acad. Publ., 1999. P. 1–26.

3. **Błażewicz J., Lenstra J. K., Rinnoy Kan A. H. G.** Scheduling subject to resource constraints: Classification and complexity // Discrete Appl. Math. 1983. V. 5, No. 1. P. 11–24.
4. **Kolisch R., Sprecher A.** PSPLIB — A project scheduling problem library // Eur. J. Oper. Res. 1996. V. 96. P. 205–216.
5. **Hartmann S., Briskorn D.** A survey of variants and extentions of the resource-constrained project scheduling problem // Eur. J. Oper. Res. 2010. V. 207. P. 1–14.
6. **Kolisch R., Hartmann S.** Experimental investigation of heuristics for resource-constrained project scheduling: An update // Eur. J. Oper. Res. 2006. V. 174, No. 1. P. 23–37.
7. **Herroelen W., Leus R.** Robust and reactive project scheduling: A review and classification of procedures // Int. J. Prod. Res. 2004. V. 42, No. 8. P. 1599–1620.
8. **Golab A., Sedgh Gooya E., Al Falou A., Cabon M.** Review of conventional metaheuristic techniques for resource-constrained project scheduling problem // J. Proj. Manag. 2022. V. 7, No. 2. P. 95–110. DOI: 10.5267/j.jpm.2021.10.002.
9. **Pellerin R., Perrier N., Berthaut F.** A survey of hybrid metaheuristics for the resource-constrained project scheduling problem // Eur. J. Oper. Res. 2020. V. 280, No. 2. P. 395–416. DOI: 10.1016/j.ejor.2019.01.063.
10. **Abdolshah M.** A review of resource-constrained project scheduling problems (RCPSP) approaches and solutions // Int. Trans. J. Eng. Manag. Appl. Sci. Technol. 2014. V. 5, No. 4. P. 253–286.
11. **Hartmann S., Briskorn D.** An updated survey of variants and extensions of the resource-constrained project scheduling problem // Eur. J. Oper. Res. 2022. V. 297, No. 1. P. 1–14. DOI: 10.1016/j.ejor.2021.05.004.
12. **Khajesaeedi S., Sadjadi S., Barzinpour F., Tavakkoli-Moghaddam R.** Resource-constrained project scheduling problem: Review of recent developments // J. Proj. Manag. 2025. V. 10, No. 1. P. 1–26. DOI: 10.5267/j.jpm.2024.12.002.
13. **Coelho J., Vanhoucke M.** Going to the core of hard resource-constrained project scheduling instances // Comput. Oper. Res. 2020. V. 121. Article ID 104976. 13 p. DOI: 10.1016/j.cor.2020.104976.
14. **Гончаров Е. Н.** Жадный алгоритм для задачи календарного планирования с ограниченными ресурсами // Дискрет. анализ и исслед. операций. 2024. Т. 31, № 4. С. 27–39. DOI: 10.33048/daio.2024.31.796.
15. **Гончаров Е. Н., Леонов В. В.** Генетический алгоритм для задачи календарного планирования с ограниченными ресурсами // Автоматика и телемеханика. 2017. № 6. С. 173–189.
16. **Гимади Э. Х., Залюбовский В. В., Севастьянов С. В.** Полиномиальная разрешимость задач календарного планирования со складируемыми ресурсами и директивными сроками // Дискрет. анализ и исслед. операций. Сер. 2. 2000. Т. 7, № 1. С. 9–34.

17. Гимади Э. Х., Гончаров Е. Н., Штепа А. А. Быстрый алгоритм вычисления нижней оценки для решения задачи ресурсно-календарного планирования с тестированием на примерах библиотеки PSPLIB // Тр. Ин-та математики и механики. 2021. Т. 27, № 1. С. 22–36. DOI: 10.21538/0134-4889-2021-27-1-22-36.

Гимади Эдуард Хайрутдинович
Гончаров Евгений Николаевич

Статья поступила
8 апреля 2026 г.
После доработки —
15 апреля 2026 г.
Принята к публикации
22 апреля 2026 г.

DISKRETNYI ANALIZ I ISSLEDOVANIE OPERATSII
/DISCRETE ANALYSIS AND OPERATIONS RESEARCH/
April–June 2026. Vol. 33, No. 2. P. 5–18

UDC 519.8+518.25 DOI: 10.33048/daio.2026.33.864

ON A METHOD FOR RANKING RESOURCES FOR THE RESOURCE-CONSTRAINED PROJECT SCHEDULING PROBLEM

É. Kh. Gimadi[a] and *E. N. Goncharov*[b]

Sobolev Institute of Mathematics,
4 Acad. Koptyug Avenue, 630090 Novosibirsk, Russia
E-mail: [a] gimadi@math.nsc.ru, [b] gon@math.nsc.ru

Abstract. We consider a resource-constrained project scheduling problem (RCPSP) with respect to the makespan minimization criterion. The problem accounts for technological constraints on activities precedence together with resource constraints. Activities preemptions are not allowed. The problem with renewable resources is NP-hard in the strong sense. Metaheuristics are one of the promising methods for solving this problem. We propose a new method for identifying the most scarce resources, which helps metaheuristics to overcome these resource conflicts and find better solutions. The proposed resource ranking method is applied in a genetic algorithm. Computational experiments with instances from the PSPLIB electronic library showed its effectiveness. New solutions are obtained for several instances, while new best (previously unknown) solutions are found for one instance from j90 and four instances from j120 datasets. Tab. 1, bibliogr. 17.

Keywords: project management, resource-constrained project scheduling problem, renewable resources, PSPLIB.

References

1. **P. Brucker, A. Drexl, R. Möhring, K. Neumann,** and **E. Pesch,** Resource-constrained project scheduling: Notation, classification, models, and methods, *Eur. J. Oper. Res.* **112** (1), 3–41 (1999).
2. **W. Herroelen, E. Demeulemeester,** and **B. De Reyck,** A classification scheme for project scheduling, in *Project Scheduling: Recent Models, Algorithms and Applications* (Kluwer Acad. Publ., Dordrecht, 1999), pp. 1–26.

English transl.: *Journal of Applied and Industrial Mathematics* **20** (2) (2026).

3. **J. Błażewicz, J. K. Lenstra,** and **A. H. G. Rinnoy Kan,** Scheduling subject to resource constraints: Classification and complexity, *Discrete Appl. Math.* **5** (1), 11–24 (1983).
4. **R. Kolisch** and **A. Sprecher,** PSPLIB — A project scheduling problem library, *Eur. J. Oper. Res.* **96**, 205–216 (1996).
5. **S. Hartmann** and **D. Briskorn,** A survey of variants and extentions of the resource-constrained project scheduling problem, *Eur. J. Oper. Res.* **207**, 1–14 (2010).
6. **R. Kolisch** and **S. Hartmann,** Experimental investigation of heuristics for resource-constrained project scheduling: An update, *Eur. J. Oper. Res.* **174** (1), 23–37 (2006).
7. **W. Herroelen** and **R. Leus,** Robust and reactive project scheduling: A review and classification of procedures, *Int. J. Prod. Res.* **42** (8), 1599–1620 (2004).
8. **A. Golab, E. Sedgh Gooya, A. Al Falou,** and **M. Cabon,** Review of conventional metaheuristic techniques for resource-constrained project scheduling problem, *J. Proj. Manag.* **7** (2), 95–110 (2022), DOI: 10.5267/j.jpm.2021.10.002.
9. **R. Pellerin, N. Perrier,** and **F. Berthaut,** A survey of hybrid metaheuristics for the resource-constrained project scheduling problem, *Eur. J. Oper. Res.* **280** (2), 395–416 (2020), DOI: 10.1016/j.ejor.2019.01.063.
10. **M. Abdolshah,** A review of resource-constrained project scheduling problems (RCPSP) approaches and solutions, *Int. Trans. J. Eng. Manag. Appl. Sci. Technol.* **5** (4), 253–286 (2014).
11. **S. Hartmann** and **D. Briskorn,** An updated survey of variants and extensions of the resource-constrained project scheduling problem, *Eur. J. Oper. Res.* **297** (1), 1–14 (2022), DOI: 10.1016/j.ejor.2021.05.004.
12. **S. Khajesaeedi, S. Sadjadi, F. Barzinpour,** and **R. Tavakkoli-Moghaddam,** Resource-constrained project scheduling problem: Review of recent developments, *J. Proj. Manag.* **10** (1), 1–26 (2025), DOI: 10.5267/j.jpm.2024.12.002.
13. **J. Coelho** and **M. Vanhoucke,** Going to the core of hard resource-constrained project scheduling instances, *Comput. Oper. Res.* **121**, ID 104976 (2020), DOI: 10.1016/j.cor.2020.104976.
14. **E. N. Goncharov,** A greedy algorithm for the resource-constrained project scheduling problem, *Diskretn. Anal. Issled. Oper.* **31** (4), 27–39 (2024) [Russian], DOI: 10.33048/daio.2024.31.796 [*J. Appl. Ind. Math.* **18** (4), 679–685 (2024)].
15. **E. N. Goncharov** and **V. V. Leonov,** A genetic algorithm for the resource-constrained project scheduling problem, *Avtom. Telemekh.*, No. 6, 173–189 (2017) [Russian] [*Autom. Remote Control* **78** (6), 1101–1114 (2017)].
16. **É. Kh. Gimadi, V. V. Zalyubovskiy,** and **S. V. Sevast'yanov,** Polynomial solvability of scheduling problems with storable resources and directive deadlines, *Diskretn. Anal. Issled. Oper., Ser. 2*, **7** (1), 9–34 (2000) [Russian].

17. É. Kh. Gimadi, E. N. Goncharov, and **A. A. Shtepa,** A fast algorithm for finding a lower bound of the solution to the resource-constrained project scheduling problem tested on PSPLIB instances, *Tr. Inst. Mat. Mekh.* **27** (1), 22–36 (2021), DOI: 10.21538/0134-4889-2021-27-1-22-36 [Russian].

Éduard Kh. Gimadi
Evgeny N. Goncharov

Received April 8, 2026
Revised April 15, 2026
Accepted April 22, 2026

ДИСКРЕТНЫЙ АНАЛИЗ И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ
Апрель–июнь 2026. Т. 33, № 2. С. 19–45

УДК 519.8+518.25 DOI: 10.33048/daio.2026.33.850

ВЕРХНИЕ И НИЖНИЕ ГРАНИЦЫ И АЛГОРИТМЫ СПУСКА С ЧЕРЕДУЮЩИМИСЯ ОКРЕСТНОСТЯМИ ДЛЯ ЗАДАЧИ КОНКУРЕНТНОГО ЦЕНООБРАЗОВАНИЯ

Р. Р. Драчев [a], *А. А. Панин* [b], *А. В. Плясунов* [c]

Институт математики им. С. Л. Соболева,
пр. Акад. Коптюга, 4, 630090 Новосибирск, Россия

E-mail: [a] rrdrachev@gmail.com,
[b] aapanin1988@gmail.com, [c] apljas@math.nsc.ru

Аннотация. Рассматривается трёхуровневая задача конкурентного ценообразования. Две компании, производящие однородный продукт, конкурируют за потребительский спрос. На первом уровне компания-лидер, а на втором уровне компания-последователь устанавливают цены на каждом из своих предприятий с целью максимизировать собственный доход. На третьем уровне каждый потребитель выбирает предприятие с наименьшими общими расходами на покупку и транспортировку товара и совершает покупку в том случае, если расходы не превышают его бюджета. Цель игры — найти такие цены лидера, при которых его доход максимален. Известно, что проблема Σ_2^p-трудна. Предложены нижние и верхние границы оптимума, и разработаны приближённые алгоритмы решения, основанные на спуске с чередующимися окрестностями. Проведён вычислительный эксперимент с использованием примеров различной размерности и сложности. Табл. 7, ил. 2, библиогр. 20.

Ключевые слова: трёхуровневое программирование, игра Штакельберга, конкурентное ценообразование, локальный поиск, спуск с чередующимися окрестностями.

Введение

В работе исследуется обобщение двухуровневой задачи ценообразования, впервые рассмотренной в [1, 2]. Эта задача характеризуется единичным объёмом спроса, отдельными ценами на каждом пункте обслуживания одного производителя и ненулевыми затратами на транспортировку продукции из пункта обслуживания до потребителя, которые полностью

© Р. Р. Драчев, А. А. Панин, А. В. Плясунов, 2026

оплачиваются самим потребителем. В [3] рассмотрена близкая постановка, в которой транспортные расходы нулевые, а спрос не равен единице. Другие вариации задачи также можно найти в литературе [4–12].

Для моделирования рыночных процессов требуется учитывать интересы других производителей (конкурентов), что достаточно хорошо описывается игрой Штакельберга или многоуровневой постановкой, когда задача решается от лица производителя-лидера, принимающего решение первым, а вторым шагом на его решение реагируют производители-последователи. Основная проблема таких постановок — сложность или даже невозможность построения одноуровневых моделей с полиномиальным числом ограничений, что приводит к невозможности применения классических методов решения оптимизационных задач и требует разработки новых алгоритмических подходов для построения как точного, так и приближённого решений.

Содержательно рассматриваемую в данной работе задачу можно описать следующей игрой Штакельберга. Две компании, производящие однородный продукт, конкурируют за потребительский спрос. На первом уровне лидер устанавливает цены на своих предприятиях, следуя некоторой стратегии ценообразования. На втором уровне последователь устанавливает цены на каждом из своих предприятий с учётом действий лидера на первом ходу и с целью максимизировать свой доход. На третьем уровне каждый клиент выбирает предприятие с наименьшими общими расходами на покупку и транспортировку и совершает покупку только в том случае, если эти расходы не превышают его бюджета.

Различают несколько стратегий ценообразования [13]: равномерная стратегия, когда на всех предприятиях устанавливается единая цена; фабричная стратегия, когда на каждом предприятии устанавливается своя цена; дискриминационная стратегия, когда на каждом предприятии устанавливается своя цена для каждого клиента.

Для трёхуровневых задач необходимо уточнить характер взаимодействия (кооперации) уровней [14]. Если некоторый уровень имеет несколько оптимальных решений, то уровни выше должны точно знать, какое именно решение выберет игрок, иначе задачу можно считать некорректной. Так, последователь может либо кооперироваться с лидером, т. е. из нескольких своих оптимальных решений выбирать то, которое максимизирует доход лидера, либо не кооперироваться, т. е. из нескольких своих оптимальных решений выбирать то, которое минимизирует доход лидера. В свою очередь, клиенты могут кооперироваться либо с лидером, либо с последователем, т. е. из нескольких своих оптимальных решений выбирать то, которое максимизирует доход лидера или последователя соответственно.

В работе рассматривается задача с фабричным ценообразованием как у лидера, так и у последователя и с кооперацией всех нижних уровней с лидером. Известно, что задача принадлежит классу $\Sigma_2^P O$ и Σ_2^P-трудна [15]. Для её решения будем использовать подход, впервые предложенный в [16] для двухуровневой задачи размещения производства и ценообразования, в котором для оценки эффективности фиксированного размещения подзадача ценообразования решается приближённо. В нашем случае для оценки качества выбранных цен лидера используем приближённый эвристический алгоритм решения задачи последователя.

В разд. 1 приводится математическая модель задачи и обсуждаются её свойства. В разд. 2 представлены верхние и нижние границы оптимумов, используемые для оценки эффективности алгоритмов. В разд. 3 представлены алгоритмы решения исследуемой задачи, а в разд. 4 приведены результаты численных экспериментов.

1. Математическая модель

Введём следующие обозначения:

- $I_L = \{1, \ldots, n_L\}$ — множество предприятий лидера;
- $I_F = \{n_L{+}1, \ldots, n_L{+}n_F\}$ — множество предприятий последователя;
- $I = I_L \cup I_F$;
- $J = \{1, \ldots, m\}$ — множество клиентов;
- $b_j \in \mathbb{Z}_{\geqslant}$ — бюджет j-го клиента;
- $c_{ij} \in \mathbb{Z}_{\geqslant}$ — транспортные затраты j-го клиента в i-м предприятии.

Введём следующие переменные:

- $x_{ij} = \begin{cases} 1, & \text{если } j\text{-й клиент обслуживается на } i\text{-м предприятии,} \\ 0 & \text{иначе;} \end{cases}$
- $p_i \in \mathbb{Z}_{\geqslant}$ — цена продукта лидера на i-м предприятии;
- $q_i \in \mathbb{Z}_{\geqslant}$ — цена продукта последователя на i-м предприятии.

Используя введённые обозначения, запишем трёхуровневую задачу фабричного ценообразования в виде трёхуровневой модели квадратичного программирования с целочисленными и булевыми переменными:

$$\sum_{i \in I_L} \sum_{j \in J} p_i x_{ij} \to \max_{p,q,x}, \tag{1}$$

$$p_i \in \mathbb{Z}_{\geqslant}, \tag{2}$$

$$(q, x) \in \mathcal{F}^*_{F_l}(p), \tag{3}$$

где $\mathcal{F}^*_{F_l}(p)$ — множество оптимальных решений задачи последователя при кооперации с лидером:

$$\sum_{i \in I_F} \sum_{j \in J} q_i x_{ij} \to \max_{q,x}, \tag{4}$$

$$q_i \in \mathbb{Z}_{\geqslant}, \tag{5}$$

$$(x) \in \mathcal{F}^*_{C_l}(p, q), \tag{6}$$

где $\mathcal{F}^*_{C_l}(p, q)$ — множество оптимальных решений задачи клиентов при кооперации с лидером:

$$\sum_{j \in J} \Big(\sum_{i \in I_L} (b_j - c_{ij} - p_i) x_{ij} + \sum_{i \in I_F} (b_j - c_{ij} - q_i) x_{ij} \Big) \to \max_x, \tag{7}$$

$$\sum_{i \in I} x_{ij} \leqslant 1, \quad j \in J, \tag{8}$$

$$x_{ij} \in \{0, 1\}, \quad i \in I_L \cup I_F,\ j \in J. \tag{9}$$

Целевая функция верхнего уровня (1) определяет доход лидера, целевая функция второго уровня (4) — доход последователя. Целевая функция третьего уровня (7) отражает стремление каждого клиента минимизировать свои расходы или, другими словами, максимизировать остаток бюджета. Ограничения (2) и (5) устанавливают ненулевые цены на продукцию лидера и последователя, а (8) гарантирует, что каждый клиент обслуживается не более чем на одном предприятии. Ограничения (3) и (6) делают модель трёхуровневой.

В работе [15] для рассматриваемой задачи, а также её модификаций, полученных заменой стратегии ценообразования лидера или последователя, введены обозначения, которые будем использовать ниже:

- MMP_L — исходная задача с фабричным ценообразованием у лидера и последователя;
- DMP_L — в задаче MMP_L стратегия ценообразования лидера заменяется на дискриминационную;
- UMP_L — в задаче MMP_L стратегия ценообразования лидера заменяется на равномерную;
- MDP_L — в задаче MMP_L стратегия ценообразования последователя заменяется на дискриминационную;
- MUP_L — в задаче MMP_L стратегия ценообразования последователя заменяется на равномерную.

1.1. Одноуровневая модель для задачи последователя. Аналогично обычной двухуровневой задаче фабричного ценообразования [1, 2] задача последователя может быть переписана в виде задачи одноуровневого целочисленного линейного программирования. Для этого введём верхние границы на оптимальные цены:

$$\overline{p}_i = \max_{j \in J} \{b_j - c_{ij}\},\ i \in I_L, \quad \overline{q}_i = \max_{j \in J} \{b_j - c_{ij}\},\ i \in I_F.$$

Целевые функции и ограничения (4)–(9) перепишем в следующем виде:

$$\sum_{i\in I_F}\sum_{j\in J} z_{ij} \to \max_{q,x,z}, \tag{10}$$

$$\sum_{i\in I}(b_j - c_{ij})x_{ij} - \sum_{i\in I} z_{ij} \geqslant 0, \quad j \in J, \tag{11}$$

$$c_{kj} + p_k - \sum_{i\in I} c_{ij}x_{ij} - \sum_{i\in I} z_{ij} \geqslant 0, \quad k \in I_L,\ j \in J, \tag{12}$$

$$c_{kj} + q_k - \sum_{i\in I} c_{ij}x_{ij} - \sum_{i\in I} z_{ij} \geqslant 0, \quad k \in I_F,\ j \in J, \tag{13}$$

$$(1 - x_{ij})\overline{p}_i - z_{ij} + p_i \geqslant 0, \quad i \in I_L,\ j \in J, \tag{14}$$

$$(1 - x_{ij})\overline{p}_i + z_{ij} - p_i \geqslant 0, \quad i \in I_L,\ j \in J, \tag{15}$$

$$(1 - x_{ij})\overline{q}_i - z_{ij} + q_i \geqslant 0, \quad i \in I_F,\ j \in J, \tag{16}$$

$$(1 - x_{ij})\overline{q}_i + z_{ij} - q_i \geqslant 0, \quad i \in I_F,\ j \in J, \tag{17}$$

$$z_{ij} \leqslant \overline{p}_i x_{ij}, \quad i \in I_L,\ j \in J, \tag{18}$$

$$z_{ij} \leqslant \overline{q}_i x_{ij}, \quad i \in I_F,\ j \in J, \tag{19}$$

$$\sum_{i\in I} x_{ij} \leqslant 1, \quad j \in J, \tag{20}$$

$$q_i \in Z_{\geqslant},\ i \in I_F, \quad x_{ij} \in \{0,1\},\ z_{ij} \in Z_{\geqslant},\ i \in I,\ j \in J. \tag{21}$$

Здесь переменные z_{ij} обозначают доход, полученный на i-м предприятии от j-го клиента. Ограничения (12)–(14) гарантируют, что клиенты выберут оптимальное решение, а условия (15)–(19) гарантируют, что доход на i-м предприятии от j-го клиента равен цене на i-м предприятии, если клиент обслуживается на этом предприятии.

Заметим, что при $I = I_F$ получаем двухуровневую задачу фабричного ценообразования (MP) [2].

1.2. Сравнение кооперативной и некооперативной постановок. Для поиска кооперативного решения приходится решать NP-трудную задачу последователя дважды. Сначала находим оптимум opt_F^* задачи (10)–(21). Затем решаем новую модель, содержащую ограничения (11)–(21), целевую функцию, определяющую доход лидера:

$$\sum_{i\in I_L}\sum_{j\in J} z_{ij} \to \max_{q,x,z}$$

и новое ограничение, гарантирующее, что последователь выберет своё оптимальное решение:

$$\sum_{i\in I_F}\sum_{j\in J} z_{ij} = \mathrm{opt}_F^*.$$

Ключевая особенность многоуровневых алгоритмов локальных улучшений, описанных ниже, заключается в упрощении процедуры вычисления дохода лидера. Например, при определении лучшего в некоторой окрестности вектора цен лидера можно отказаться от принципа кооперативного взаимодействия лидера и последователя. Следующее утверждение говорит о том, что пренебрегать кооперацией неразумно.

Теорема 1. *Зафиксируем оптимальное решение задачи* MMP_L. *При смене решения последователя с кооперативного на произвольное оптимальное решение* (*цены лидера остаются прежними*) *доход лидера может измениться во сколько угодно раз.*

Утверждение теоремы останется справедливым даже при выполнении одного из условий:

(1°) $|J| \leqslant 2$;

(2°) $b_j \leqslant 2 \ \ \forall j \in J$.

Доказательство. Рассмотрим условие (1°). Пусть у лидера и последователя есть по одному предприятию с номерами 1 и 2 соответственно. Предположим, что есть два клиента с номерами 1 и 2 соответственно, причём клиент 2 может обслуживаться только последователем:

$$c_{11} = 0, \quad c_{21} = 0,$$
$$c_{12} = 2k, \quad c_{22} = 0,$$
$$b_1 = k, \quad b_2 = 2k.$$

Доход лидера при любом решении последователя не превосходит k. Максимальный доход может быть достигнут, если обслужить клиента 1, установив цену $p_1 = k$, а последователь назначит цену $q_2 = 2k$ и обслужит клиента 2, что соответствует кооперативному оптимальному решению. Однако последователь может выбрать другое оптимальное решение $q_2 = k$, при котором он обслуживает всех клиентов.

Рассмотрим условие (2°). Пусть у лидера и последователя по-прежнему есть по одному предприятию с номерами 1 и 2 соответственно. Предположим, что есть $2d$ клиентов, половина из которых могут обслуживаться только последователем:

$$c_{1j} = 0, \quad c_{2j} = 0, \quad j \in \{1, \ldots, k\},$$
$$c_{1j} = 2, \quad c_{2j} = 0, \quad j \in \{k+1, \ldots, 2k\},$$
$$b_j = 1, \quad j \in \{1, \ldots, k\},$$
$$b_j = 2, \quad j \in \{k+1, \ldots, 2k\}.$$

Доход лидера при любом решении последователя не превосходит d, а достигает этой верхней границы при обслуживании d первых клиентов

с ценой продукции $p_1 = 1$ и при кооперативной оптимальной цене последователя $q_2 = 2$. При этом последователь обслуживает d последних клиентов, а при другой оптимальной цене $q_2 = 1$ последователь обслуживает всех клиентов. Теорема 1 доказана.

2. Верхние и нижние границы

Простейшей верхней оценкой, которая может быть построена для любой многоуровневой задачи, является оценка, полученная релаксацией экстремального ограничения (high-point relaxation [17]). Соответствующая релаксированная задача получается из исходной постановки удалением целевой функции последователя и объединением оставшихся ограничений лидера и последователя. В рассматриваемой в данной работе задаче в результате релаксации экстремального ограничения получается задача фабричного ценообразования [1, 2]. На практике релаксация экстремального ограничения оказывается довольно грубой оценкой. Оценку, полученную релаксацией экстремального ограничения, обозначим через $\mathrm{UB}_{\mathrm{hp}}$.

В работе [2] описаны верхняя и нижняя границы для дохода лидера в задаче фабричного ценообразования, полученные путём смены стратегии ценообразования. Возникает вопрос: можно ли использовать метод смены стратегии ценообразования с целью получения оценок для задачи MMP_L? Следующие утверждения говорят о том, что в некоторых случаях это возможно.

Теорема 2. *Оптимум задачи* DMP_L *является верхней границей оптимума задачи* MMP_L.

Доказательство. Заметим, что фабричное ценообразование вкладывается в дискриминационное. Следовательно, множество допустимых решений задачи MMP_L представляет собой подмножество множества допустимых решений задачи DMP_L. Теорема 2 доказана.

Задача DMP_L принадлежит классу NPO [15]. Стоит отметить, что для задачи DMP_L в рамках данной работы не было найдено эффективных способов решения. Это приводит к необходимости полного перебора цен лидера, коих существенно больше, чем в задаче MMP_L. В результате практическая ценность теоремы 2 в настоящий момент остаётся низкой.

Теорема 3. *Оптимум задачи* UMP_L *является нижней границей оптимума задачи* MMP_L.

Доказательство. Достаточно заметить, что равномерное ценообразование вкладывается в фабричное, поэтому множество допустимых решений задачи UMP_L представляет собой подмножество множества допустимых решений задачи MMP_L. Теорема 3 доказана.

Задача UMP_L принадлежит классу NPO [15]. Оценку из теоремы 3 обозначим через $\mathrm{LB}_{\mathrm{um}}$. Для задачи UMP_L в рамках данной работы использован полный перебор решений. Однако их число существенно меньше, чем в задаче MMP_L. Как показывают проведённые эксперименты, теорема 3 вполне применима на практике.

Теорема 4. *Оптимум задачи* MDP_L *является нижней границей для оптимума задачи* MMP_L.

Доказательство. Рассмотрим задачу MDP_L. В ней последователь устанавливает на каждом предприятии для каждого клиента свою цену. Таким образом, последователь для любого клиента всегда может установить такую цену, чтобы затраты на обслуживание у последователя были на единицу меньше, чем затраты на обслуживание у лидера, исключая случаи, когда для этого нужно установить нулевую или отрицательную цену. Таким образом, клиент j обслуживается на предприятии лидера i тогда и только тогда, когда $c_{ij} + p_i \leqslant \min\{\min\limits_{k \in I_F}\{c_{kj}\} + 1, b_j\}$.

Если в задаче MMP_L имеем $c_{ij}+p_i \leqslant \min\{\min\limits_{k \in I_F}\{c_{kj}\}+1, b_j\}$, то клиент j обслуживается на предприятии лидера i.

Пусть $\overline{p}^{\,\mathrm{opt}}$ — оптимальный вектор цен лидера в задаче MDP_L, а J_L^{opt} — множество клиентов, обслуживаемых лидером при указанном векторе цен $\overline{p}^{\,\mathrm{opt}}$ в некотором оптимальном решении. По приведённым выше свойствам в задаче MMP_L при ценах $\overline{p}^{\,\mathrm{opt}}$ лидер точно обслужит всех клиентов из множества J_L^{opt}. Значит, доход лидера в задаче MMP_L при оптимальных ценах из задачи MDP_L будет не меньше, чем оптимум MDP_L. Теорема 4 доказана.

Задача MDP_L принадлежит классу NPO [15]. Оценку из теоремы 4 будем обозначать через $\mathrm{LB}_{\mathrm{md}}$. Для данной задачи удалось построить одноуровневую модель линейного смешанного целочисленного программирования. Введём величину $d_{ij} = \min\{\min\limits_{k \in I_F} c_{kj} + 1 - c_{ij}, 0\}$. Модель получается путём добавления к модели двухуровневой задачи фабричного ценообразования (MP) из п. 1.1, записанной для лидера, нового ограничения:

$$z_{ij} \leqslant d_{ij}, \quad i \in I_L,\, j \in J,$$

которое гарантирует, что лидер не сможет обслужить клиента, которого может обслужить последователь.

В задаче MUP_L мы сужаем множество решений последователя и, казалось бы, должны ожидать увеличения дохода лидера. На самом деле, оптимум MUP_L не является ни нижней, ни верхней границей.

Пример 1. Пусть у лидера есть одно предприятие с номером 1, а у последователя есть два предприятия с номерами 2, 3. Пусть имеется 4 клиента со следующими бюджетами и транспортными затратами:

$$b_1 = b_2 = 3, \quad b_3 = 4, \quad b_4 = 9,$$
$$c_{11} = c_{12} = 3, \quad c_{13} = 0, \quad c_{14} = 9,$$
$$c_{21} = c_{22} = c_{23} = c_{24} = 0,$$
$$c_{31} = c_{32} = c_{33} = c_{34} = 0.$$

На рис. 1 дана иллюстрация этого примера. Ромбом и квадратами обозначены предприятия лидера и последователя соответственно, кругами — клиенты. Линиями обозначены нулевые транспортные затраты, отсутствие линий означает, что транспортные затраты равны бюджету клиента. Здесь лидер может обслужить только клиента 3, значит, его максимальный доход не может превышать 4. Более того, если лидер обслуживает клиента 3, то его доход равен установленной цене.

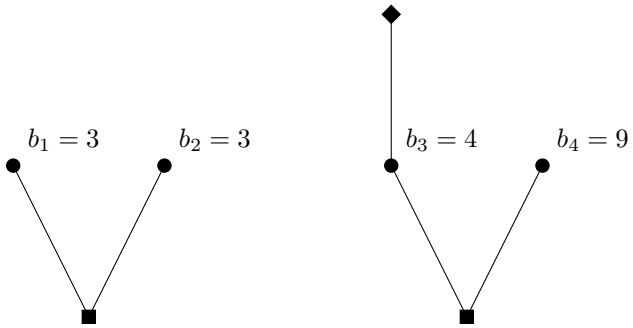


Рис. 1. Пример задачи конкурентного ценообразования

Рассмотрим исходную задачу с фабричным ценообразованием у последователя. Пусть лидер установил цену $p_1 = 4$, в таком случае последователю выгодно на предприятии 2 обслужить клиентов 1 и 2 по цене 3, а на предприятии 3 обслужить клиента 4 по цене 9, но не клиента 3 по более низкой цене. Таким образом, лидер получит доход 4.

Теперь поменяем стратегию ценообразования последователя на равномерную. Это означает, что последователь устанавливает единую цену на обоих своих предприятиях. Доход лидера не меньше установленной цены. Следовательно, чтобы получить доход 4 (такой же, как при фабричном ценообразовании у последователя), он вынужден установить цену $p_1 = 4$. Тогда последователь установит цену $q = 3$ и обслужит при этом всех клиентов.

Таким образом, оптимальное значение целевой функции лидера при фабричном ценообразовании у последователя может быть выше, чем при равномерном.

Пример 2. Пусть у лидера есть одно предприятие с номером 1, а у последователя — два предприятия с номерами 2, 3. Пусть имеется два клиента со следующими бюджетами и транспортными затратами:

$$b_1 = b_2 = 2,$$
$$c_{11} = 1, \quad c_{12} = 2,$$
$$c_{21} = 0, \quad c_{22} = 2,$$
$$c_{31} = 2, \quad c_{32} = 0.$$

На рис. 2 дана иллюстрация этого примера. Ромбом и квадратами обозначены предприятия лидера и последователя соответственно, кругами — клиенты. Отсутствие линий означает, что транспортные затраты равны бюджету клиента. Здесь лидер может получить доход не больше 1. Максимум достижим только при цене $p_1 = 1$.

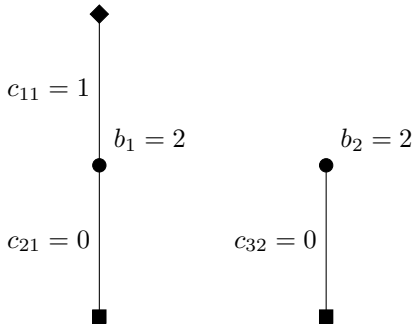


Рис. 2. Пример задачи конкурентного ценообразования.

Рассмотрим исходную задачу с фабричным ценообразованием у последователя. Пусть лидер установил цену $p_1 = 1$. Последователь назначит цены $q_2 = 1$, $q_3 = 2$ и получит максимальный доход, обслужив всех клиентов.

Теперь поменяем стратегию ценообразования последователя на равномерную. Лидер по-прежнему установит цену $p_1 = 1$. Последователь имеет два оптимальных решения $q = 1$ и $q = 2$. В силу кооперации он выберет второе решение и обслужит только второго клиента. Доход лидера составит 1.

Следовательно, оптимальное значение целевой функции лидера при фабричном ценообразовании у последователя меньше, чем при равномерном.

Эти два примера приводят нас к следующему утверждению.

Теорема 5. *Оптимум задачи* MUP_L *не является ни верхней, ни нижней границей для оптимума задачи* MMP_L.

3. Алгоритмы решения

Для решения задачи MMP_L предлагаем подходы, основанные на приближённом алгоритме решения одноуровневой задачи размещения и фабричного ценообразования [16]. Все подходы используют принцип локальных улучшений вектора цен лидера, тогда как параметрическая задача последователя при фиксированных ценах лидера решается с помощью эвристики, также использующей локальные улучшения, что существенно сокращает время счёта. По решению последователя, построенному эвристикой, во многих случаях можно сделать довольно точное предположение об ожидаемом доходе лидера для заданного вектора цен. Другими словами, эвристический алгоритм даёт хорошую оценку значения целевой функции на допустимом решении с заданным вектором цен лидера. Чтобы найти это допустимое решение, нужно решить NP-трудную задачу последователя. Сокращение числа обращений к точному алгоритму решения задачи последователя, чего и пытаемся добиться применением эвристики, существенно ускоряет время работы алгоритма решения задачи MMP_L.

Стоит отметить, что иногда оценка, полученная эвристическим алгоритмом решения задачи последователя, оказывается слишком грубой, и локальный поиск по ценам лидера может зайти в бесперспективную окрестность и остановиться на достаточно плохом решении всего из-за одной ошибки. Тем самым для корректировки ошибок в предсказании дохода время от времени полученное эвристикой решение корректируется путём точного решения задачи последователя. Однако тут нужно соблюдать баланс: при частой корректировке решений быстро растёт время счёта, при недостаточной — алгоритм часто приводит к плохому решению.

3.1. Окрестностные структуры. Опишем используемые окрестностные структуры в алгоритмах локального поиска. Приведённые ниже окрестности хорошо известны [16].

Через k-Flip(p) обозначается окрестность вектора цен лидера p, содержащая все векторы цен, отличающиеся от p не более чем в k компонентах.

Через k-Flip$(q \,|\, p)$ обозначается окрестность вектора цен последователя q, где все соседи отличаются от q не более чем в k компонентах. Вектор p в названии окрестности соответствует ценам лидера, которые используются при определении дохода последователя.

3.2. Алгоритмы решения задачи последователя. Опишем алгоритмы, используемые в данной работе для нахождения решений задачи последователя.

Для нахождения точного решения задачи последователя при фиксированном векторе цен лидера используется схема, описанная в п. 1.1, причём все возникающие задачи ЦЛП решаются MILP-решателем. Обозначим эту процедуру через EX(p).

Перейдём к описанию приближённых алгоритмов. Базовым алгоритмом является локальный поиск [16], в котором, начиная с некоторого стартового решения, на каждой итерации происходит переход в соседнее (относительно некоторой заданной окрестности текущего решения) решение с лучшим значением целевой функции. В данном случае использована окрестность 1-Flip($q \,|\, p$). Для этого алгоритма различают две стратегии: первое улучшение (first improve [19]), при котором переходим в новую точку сразу, как только нашли первое решение с лучшим значением целевой функции, и наибольшее улучшение (best improve [19]), при котором переходим в лучшее решение в окрестности после её полного просмотра. В данной работе для решения задачи последователя преимущественно использовалась стратегия первого улучшения, так как в вычислительных экспериментах она занимала меньшее время при схожем качестве решений (см. табл. 7).

Алгоритм локального спуска с чередующимися окрестностями (VND [16, 18]) основан на локальном поиске и после нахождения локального оптимума позволяет выйти из окрестности локального оптимума с помощью детерминированной процедуры k-Improve($q \,|\, p$) и продолжить локальный поиск со следующей точки. Сама процедура k-Improve($q \,|\, p$) последовательно просматривает окрестности 2-Flip($q \,|\, p$), ..., k-Flip($q \,|\, p$) до того момента, пока не найдёт решение лучше текущего [16].

3.3. Алгоритмы решения задачи лидера. Опишем алгоритмы решения задачи лидера. На верхнем уровне также используется простой локальный поиск или локальный спуск с чередующимися окрестностями со стратегией наибольшего улучшения. Для нахождения решения задачи

Алгоритм 1. LS | EX

Выход: Вектор цен лидера p^*

1: $p \leftarrow$ random
2: $p^* \leftarrow$ undefined
3: **while** $f^L(p) > f^L(p^*)$ **do**
4: $\quad p^* \leftarrow p$
5: $\quad p \leftarrow \underset{p' \in 1\text{-Flip}(p)}{\arg\max} f^L(p', EX(p'))$

последователя первой очевидной идеей является использование точной процедуры. Описание реализации эвристики приведено в алгоритме 1. Для удобства описания алгоритмов введём $f^L(p, q)$ — значение целевой функции лидера при ценах p и q лидера и последователя соответственно. Аргумент q опускается, если используется оптимальная цена последователя.

В алгоритме 1 необходимо решать линейную задачу целочисленного программирования для каждого просмотренного элемента окрестности, что может занимать большое вычислительное время.

Следующая идея основана на замене процедуры точного решения задачи последователя процедурой локального поиска, а точнее, процедурой VND_k: применим процедуру VND, снабжённую процедурой улучшения k-Improve с периодической корректировкой в конце каждой итерации локального поиска. Другими словами, после просмотра очередной окрестности цен лидера вычисляем доход лидера, используя точный алгоритм решения задачи последователя. При этом рассматривать стратегию первого улучшения для лидера в данном случае не имеет смысла, так как решение, полученное таким методом в конце итерации, ещё с большей вероятностью окажется хуже рекорда. Описание реализации эвристики со стратегией лучшего улучшения приведено в алгоритме 2.

Алгоритм 2. LS | VND_k

Вход: Параметр k
Выход: Вектор цен лидера p^*

1: $p \leftarrow$ random
2: $p^* \leftarrow$ undefined
3: **while** $f^L(p) > f^L(p^*)$ **do**
4: $\quad p^* \leftarrow p$
5: $\quad p' \leftarrow \underset{p' \in 1\text{-Flip}(p)}{\arg\max} f^L(p', VND_k(p'))$

Алгоритмы LS | LS, VND_k | LS определяются аналогично.

Приведённые алгоритмы исполняются существенно быстрее по времени, так как точная процедура запускается лишь в конце просмотра окрестности. Однако у них есть существенный недостаток: часто возникает ситуация, когда лучшее решение, найденное приближёнными процедурами, при проверке точным алгоритмом оказывается хуже, чем текущий рекорд, и, несмотря на то, что в окрестности есть решения лучше текущего рекорда, алгоритм заканчивает свою работу на довольно плохом решении, которое иногда имеет значение целевой функции даже меньше нижних оценок из разд. 2.

Для исправления такой ситуации разумно запоминать не только рекорд текущей итерации, полученный с помощью приближённой процедуры, но и h лучших решений на текущей итерации, проверяя их с помощью точной процедуры в порядке убывания значений целевой функции, полученных приближённым алгоритмом. Описание реализации эвристики со стратегией лучшего улучшения приведено в алгоритме 3.

Алгоритм 3. $(\mathrm{LS} \mid \mathrm{VND}_k)^h$

Вход: Параметры h, k
Выход: Вектор цен лидера p^*

$p \leftarrow$ random
$p^* \leftarrow$ undefined
$p^* \leftarrow p$
$P \leftarrow$ массив, упорядоченный по убыванию
$i \leftarrow 1$
while $i \leqslant h$ **do**
    $p' \leftarrow \underset{p' \in 1\text{-Flip}(p) \setminus P}{\arg\max} f^L(p', \mathrm{VND}_k(p'))$
    поместить p' в P
    $i \leftarrow i + 1$
$i \leftarrow 1$
while $i \leqslant h$ **do**
    if $f^L(P[i]) > f^L(p^*)$ **then**
        $p \leftarrow P[i]$
        go to шаг 3
    $i \leftarrow i + 1$

Алгоритмы $(\mathrm{LS} \mid \mathrm{LS})^h$, $(\mathrm{VND}_k \mid \mathrm{LS})^h$ определяются аналогично.

Число проверяемых решений разумно выбирать в зависимости от размерности задачи. Заметим, что при $h = \infty$, мы в определённом порядке просматриваем все элементы окрестности, пока не найдём решение лучше текущего, т. е. это алгоритм LS | EX со стратегией первого улучшения и приоритетами просмотра элементов окрестности. Назовём такой алгоритм LS | (LS | EX).

4. Вычислительные эксперименты

В этом разделе приведены результаты численных экспериментов, проведённых на персональном компьютере, оснащённом процессором AMD Ryzen 2200 3,4 ГГц с 4 ядрами и оперативной памятью DDR4 3200 МГц 32 ГБ.

В качестве MILP-решателя применён Gurobi. В табл. 1 приведено сравнение времени работы решателей Gurobi 12.0.1 и SCIP 9.2.4 на некоторых примерах.

Таблица 1

Сравнение времени работы решателей Gurobi и SCIP

m	n^L	n^F	LS | (LS | EX)			
			Gurobi		SCIP	
			gap	время	gap	время
30	2	2	0,00	0,49	0,00	4,10
		5	0,00	6,02	0,00	33,87
		10	0,02	33,44	0,02	307,88
	5	2	0,01	0,85	0,01	4,03
		5	0,03	52,42	0,03	230,05
		10	0,03	134,87	0,03	1109,88
	10	2	0,04	3,05	0,04	8,73
		5	0,05	35,36	0,05	151,25
		10	0,03	93,46	0,03	450,17
	20	2	0,05	8,35	0,05	29,22
		5	0,03	47,61	0,03	148,39
		10	0,02	127,27	0,02	485,56
60	50	2	0,04	118,83	0,04	365,98
		5	0,00	1399,64	0,02	2894,86
		10	0,03	2937,12	0,14	3600,00
Среднее			0,03	333,25	0,03	654,93

Основная сложность проведения вычислительного эксперимента заключается в невозможности сравнения полученных приближённых решений с точными ввиду того, что для задачи MMP_L неизвестно точных алгоритмов, способных решать задачу за приемлемое время, поэтому для оценки эффективности алгоритмов подготовлены две группы тестов и вычислены нижние и верхние оценки из разд. 2. Время во всех таблицах статьи указано в секундах.

4.1. Библиотека тестов. Для исследования алгоритмов используются две группы тестовых примеров. Первая группа основана на открытой библиотеке примеров задач размещения и ценообразования [20]. В этой библиотеке предприятиям и клиентам соответствуют точки плоскости, транспортные затраты j-го клиента до i-го предприятия берутся равными евклидову расстоянию между точками, соответствующими j-му клиенту и i-му предприятию. Координаты точек берутся из равномерного распределения на квадрате $[0, 100] \times [0, 100]$, бюджеты клиентов берутся из равномерного распределения на отрезке $[0, 100]$. На основе каждого примера этой библиотеки составлены примеры различной

Таблица 2

Результаты вычислений верхних и нижних оценок для первой группы тестов

m	n^L	n^F	$\frac{\mathrm{UB_{hp}}}{\mathrm{LB}}$	$\frac{\mathrm{LB_{um}}}{\mathrm{LB_{md}}}$	$\mathrm{gap_{UB}}$	$\mathrm{gap_{LB}}$	Время		
							$\mathrm{UB_{hp}}$	$\mathrm{LB_{um}}$	$\mathrm{LB_{md}}$
30	2	2	1,86	1,38	0,39	0,09	0,05	5,37	0,02
		5	3,24	2,73	0,63	0,07	0,05	38,21	0,01
		10	3,93	3,00	0,69	0,08	0,05	147,98	0,02
	5	2	1,80	1,24	0,34	0,15	0,45	5,35	0,11
		5	3,15	1,45	0,52	0,21	0,46	32,69	0,07
		10	3,93	1,66	0,60	0,24	0,45	134,18	0,04
	10	2	1,91	1,04	0,35	0,19	1,19	7,01	0,54
		5	3,05	1,23	0,55	0,21	1,21	37,07	0,27
		10	4,07	1,37	0,66	0,24	1,24	128,37	0,08
	20	2	1,88	0,92	0,35	0,18	7,55	9,62	1,13
		5	2,94	1,06	0,52	0,26	7,54	44,19	0,40
		10	3,81	1,21	0,62	0,27	7,66	131,75	0,23
60	50	2	—	0,89	—	0,12	—	84,50	121,44
		5	—	0,92	—	0,18	—	294,23	20,04
		10	—	1,02	—	0,16	—	1994,43	7,73
Среднее			2,92	1,41	0,51	0,18	133,89	206,33	10,14

размерности для рассматриваемой задачи. Каждый пример размерности $m \times n^L \times n^F$ состоит из m первых клиентов, n^L первых предприятий (лидера) и n^F последних предприятий (последователя). Примеры больших

Таблица 3

Результаты вычислений верхних и нижних оценок для второй группы тестов

m	n^L	n^F	$\frac{\mathrm{UB_{hp}}}{\mathrm{LB}}$	$\frac{\mathrm{LB_{um}}}{\mathrm{LB_{md}}}$	$\mathrm{gap_{UB}}$	$\mathrm{gap_{LB}}$	Время		
							$\mathrm{UB_{hp}}$	$\mathrm{LB_{um}}$	$\mathrm{LB_{md}}$
30	2	2	2,53	2,34	0,54	0,05	0,12	0,80	0,01
		5	4,08	4,53	0,72	0,10	0,11	7,58	0,01
		10	6,10	6,00	0,83	0,03	0,12	21,89	0,01
	5	2	2,45	1,55	0,48	0,16	1,30	0,75	0,04
		5	3,84	2,81	0,68	0,16	1,33	5,63	0,02
		10	4,84	3,58	0,77	0,10	1,37	13,90	0,01
	10	2	2,40	1,19	0,46	0,20	7,60	0,84	0,11
Среднее			3,75	2,73	0,64	0,11	1,71	7,34	0,03

размеров формировались из примеров меньших размеров путём добавления новых предприятий к уже существующим, что позволяет проанализировать изменения в поведении игроков при увеличении возможностей лидера или последователя.

Вторая группа тестовых примеров была сформирована аналогично, с тем лишь отличием, что бюджеты клиентов и координаты предприятий и клиентов брались из равномерного распределения на отрезке $[0, 10]$, что позволило для небольших размерностей найти точное решение полным перебором цен.

В экспериментах использовалось по 4 примера каждой представленной ниже размерности в каждой группе тестов.

4.2. Оценки оптимумов. В табл. 2 и 3 представлены усреднённые характеристики верхних и нижних границ из разд. 2 для первой и второй групп тестов соответственно:

$$\mathrm{gap}_{\mathrm{UB}} = \frac{\mathrm{UB}_{\mathrm{hp}} - \mathrm{best}}{\mathrm{UB}_{\mathrm{hp}}}, \quad \mathrm{gap}_{\mathrm{LB}} = \frac{\mathrm{best} - \max\{\mathrm{LB}_{\mathrm{md}}, \mathrm{LB}_{\mathrm{um}}\}}{\mathrm{best}},$$

где best — значение целевой функции на лучшем известном решении, $\mathrm{LB} = \max\{\mathrm{LB}_{\mathrm{um}}, \mathrm{LB}_{\mathrm{md}}\}$. Для части примеров не удалось вычислить верхнюю оценку $\mathrm{UB}_{\mathrm{hp}}$ при ограничении на время счёта 8 часов.

Из результатов следует, что оценка $\mathrm{LB}_{\mathrm{md}}$ вычисляется намного быстрее оценки $\mathrm{LB}_{\mathrm{um}}$. При малом количестве предприятий лидера оценка $\mathrm{LB}_{\mathrm{um}}$ даёт более точную аппроксимацию, при увеличении же количества предприятий становится более точной оценка лидера $\mathrm{LB}_{\mathrm{md}}$.

Из табл. 3 следует, что релаксация экстремального ограничения оказывается на практике довольно грубой верхней оценкой оптимума, причём с возрастанием количества предприятий лидера и количества клиентов наблюдается существенный рост затраченного на вычисление времени.

4.3. Разработанные алгоритмы. Для оценки эффективности алгоритмов сравнивались средние по размерностям время, указанное в таблицах в секундах, и величина

$$\mathrm{gap} = \frac{\mathrm{best} - \mathrm{val}}{\mathrm{best}},$$

где best — лучшая нижняя оценка, достигнутая представленными в статье алгоритмами, а val — оценка, полученная рассматриваемым алгоритмом.

В табл. 4 и 5 представлены результаты алгоритмов со стратегией первого улучшения у последователя для первой и второй группы тестов. В табл. 6 представлены абсолютные значения для некоторых примеров.

Таблица 4

Результаты вычислений на первой группе тестов

m	n^L	n^F	LS | EX		LS | LS		LS | VND_2		VND_2 | LS	
			gap	время	gap	время	gap	время	gap	время
30	2	2	0,00	33,8	0,00	0,5	0,00	0,8	0,00	1,3
		5	0,06	220,8	0,00	3,6	0,15	12,7	0,00	9,9
		10	0,02	734,6	0,31	8,6	0,03	90,5	0,29	35,0
	5	2	0,01	138,6	0,04	0,5	0,01	2,1	0,03	8,8
		5	0,03	727,8	0,13	3,0	0,08	30,2	0,10	44,9
		10	0,02	2602,4	0,36	7,9	0,15	275,8	0,26	260,6
	10	2	0,04	489,5	0,15	0,6	0,04	5,6	0,02	46,8
		5	0,05	2325,3	0,12	4,6	0,08	74,4	0,07	160,4
		10	0,11	3571,3	0,34	7,6	0,27	356,8	0,24	1109,1
	20	2	0,03	1826,7	0,17	1,6	0,04	16,7	0,06	260,9
		5	0,05	3600,0	0,26	2,7	0,14	239,3	0,11	781,4
		10	0,18	3600,0	0,41	8,7	0,20	1474,8	0,33	2707,9
60	50	2	0,79	3600,0	0,15	19,9	0,04	176,1	0,06	2899,5
		5	1,00	3600,0	0,21	15,5	0,16	1158,6	0,21	3600,0
		10	1,00	3600,0	0,35	79,1	0,29	3469,6	0,35	3600,0
Среднее			0,23	2044,7	0,20	11,0	0,11	492,3	0,14	1035,1

m	n^L	n^F	(LS | (LS | EX))		(LS | LS)$^{1,5n^L}$		(LS | VND_2)$^{1,5n^L}$		(VND_2 | LS)$^{1,5n^L}$	
			gap	время	gap	время	gap	время	gap	время
30	2	2	0,00	0,5	0,00	0,5	0,00	0,8	0,00	1,3
		5	0,00	6,0	0,10	4,3	0,15	13,4	0,10	11,9
		10	0,02	33,4	0,27	13,5	0,02	94,9	0,01	55,6
	5	2	0,01	0,9	0,01	0,8	0,01	2,1	0,00	9,7
		5	0,03	52,4	0,04	10,6	0,02	42,6	0,04	45,9
		10	0,03	134,9	0,03	56,3	0,05	404,1	0,02	289,7
	10	2	0,04	3,0	0,04	3,0	0,04	5,5	0,00	36,1
		5	0,05	35,4	0,08	11,6	0,05	91,3	0,07	153,5
		10	0,03	93,5	0,04	64,7	0,08	782,9	0,04	787,9
	20	2	0,05	8,4	0,05	4,5	0,04	16,7	0,02	261,2
		5	0,03	47,6	0,03	47,4	0,00	336,7	0,02	472,5
		10	0,02	127,3	0,02	125,6	0,05	2578,7	0,02	2504,0
60	50	2	0,04	118,8	0,04	161,9	0,04	182,9	0,00	3600,0
		5	0,00	1399,6	0,00	1184,6	0,01	3235,3	0,00	3600,0
		10	0,03	2937,1	0,06	2591,0	0,28	3600,0	0,06	3600,0
Среднее			0,03	333,3	0,05	285,4	0,06	759,2	0,03	1028,6

Таблица 5

Результаты вычислений на второй группе тестов

m	n^L	n^F	LS | EX		LS | LS		LS | VND_2		VND_2 | LS	
			gap	время	gap	время	gap	время	gap	время
30	2	2	0,00	3,2	0,10	0,1	0,07	0,2	0,07	0,2
		5	0,00	20,6	0,04	1,2	0,04	1,3	0,04	1,5
		10	0,00	72,3	0,03	2,8	0,03	3,0	0,03	3,0
30	5	2	0,00	5,9	0,07	0,2	0,05	0,2	0,06	0,3
		5	0,09	40,6	0,09	1,2	0,09	1,3	0,00	1,5
		10	0,00	83,1	0,00	2,2	0,00	2,9	0,00	2,7
30	10	2	0,01	19,1	0,03	0,2	0,03	0,2	0,01	0,4
Среднее			0,01	35,0	0,05	1,1	0,04	1,3	0,03	1,4

m	n^L	n^F	(LS | (LS | EX)		(LS | LS)$^{1,5n^L}$		(LS | VND_2)$^{1,5n^L}$		(VND_2 | LS)$^{1,5n^L}$	
			gap	время	gap	время	gap	время	gap	время
30	2	2	0,10	0,1	0,10	0,4	0,07	0,1	0,07	0,2
		5	0,04	1,2	0,04	1,3	0,04	1,1	0,04	1,5
		10	0,03	2,8	0,03	2,9	0,03	3,2	0,03	3,2
30	5	2	0,07	0,2	0,07	0,2	0,05	0,2	0,06	0,2
		5	0,09	1,0	0,09	1,2	0,09	1,2	0,00	1,6
		10	0,00	2,0	0,00	2,3	0,00	2,7	0,00	2,7
30	10	2	0,03	0,3	0,04	0,3	0,04	0,2	0,00	0,4
Среднее			0,05	1,1	0,05	1,2	0,04	1,3	0,03	1,4

На основе результатов, полученных из экспериментов, можно заключить, что алгоритмы, использующие большее количество окрестностных структур, в среднем работают лучше. Когда VND-метаэвристика используется на верхнем уровне, просматривается большее количество потенциально оптимальных решений, чем когда используется LS-процедура. Когда же VND-метаэвристика используется на нижнем уровне, делается более точное предположение о решении последователя, следовательно, верхний уровень принимает более разумное решение, чем при использовании LS-процедуры. Алгоритмы, проверяющие большее количество решений последователя точной процедурой, дают в среднем лучшие решения. Отдельно можно выделить алгоритм LS | (LS | EX), который имеет низкий gap и сравнительно небольшое время счёта, что делает его предпочтительным в приложениях.

При анализе результатов экспериментов установлено, что на используемых примерах алгоритм LS | LS со стратегией первого улучшения доминируется алгоритмами VND_2 | LS, LS | (LS | EX) и (VND_2 | LS)$^{1,5n^L}$ со стратегией первого улучшения, алгоритм (LS | LS)$^{1,5n^L}$ доминируется

Таблица 6

Абсолютные значения на первой группе тестов

m	n^L	n^F	LS | EX		LS | LS		LS | VND_2		VND_2 | LS	
			знач.	время	знач.	время	знач.	время	знач.	время
30	2	2	202	43,0	202	0,8	202	0,9	202	1,1
		5	87	197,8	87	4,6	87	16,4	87	11,7
		10	72	549,5	61	10,0	72	133,7	61	45,9
	5	2	319	178,5	314	0,6	319	2,0	314	3,3
		5	186	700,2	165	4,0	185	29,1	184	85,4
		10	183	1640,1	118	7,5	165	307,7	118	148,5
	10	2	498	828,9	472	1,1	498	6,2	492	25,3
		5	266	2219,0	262	6,0	266	63,6	262	131,6
		10	240	3485,2	145	5,6	198	372,3	177	967,3
	20	2	537	1810,1	479	1,2	537	12,0	479	54,6
		5	348	3600,0	285	3,6	202	41,7	351	971,7
		10	243	3600,0	102	6,3	283	1766,8	197	3600,0
60	50	2	0	3600,0	775	9,6	1044	205,0	1122	3600,0
		5	0	3600,0	601	20,9	651	1176,9	601	3600,0
		10	0	3600,0	381	133,7	385	3600,0	381	3600,0

m	n^L	n^F	(LS | (LS | EX)		(LS | LS)$^{1,5n^L}$		(LS | VND_2)$^{1,5n^L}$		(VND_2 | LS)$^{1,5n^L}$	
			знач.	время	знач.	время	знач.	время	знач.	время
30	2	2	202	0,8	202	0,8	202	0,9	202	1,1
		5	87	5,2	87	5,4	87	16,4	87	13,7
		10	72	21,3	72	20,8	72	130,1	72	62,6
	5	2	319	1,3	319	1,4	319	2,0	319	5,3
		5	186	14,0	186	14,1	186	35,0	186	59,3
		10	189	37,9	189	39,6	183	469,9	189	219,8
	10	2	498	7,6	498	7,5	498	5,8	498	23,9
		5	266	12,0	266	11,6	266	62,0	266	149,2
		10	233	124,3	224	53,1	228	596,4	224	587,8
	20	2	537	21,9	534	6,4	537	12,0	534	82,5
		5	335	25,0	335	24,0	357	245,8	335	337,2
		10	276	174,0	276	175,2	293	2461,5	276	1747,9
60	50	2	1045	27,6	1049	30,7	1049	213,7	1120	3600,0
		5	843	587,8	843	630,9	817	3520,4	843	3600,0
		10	486	3600,0	486	3600,0	385	3600,0	486	3600,0

алгоритмом (VND_2 | LS)$^{1,5n^L}$ со стратегией первого улучшения, а алгоритм LS | LS со стратегией наибольшего улучшения доминируется алгоритмом (VND_2 | LS)$^{1,5n^L}$ со стратегией наибольшего улучшения. Других доминируемых алгоритмов нет.

Таблица 7

Результаты вычислений на первой группе тестов со стратегией наибольшего улучшения

m	n^L	n^F	LS | LS		LS | VND_2		VND_2 | LS	
			gap	время	gap	время	gap	время
30	2	2	0,00	0,48	0,00	0,88	0,00	0,78
		5	0,00	3,27	0,15	11,00	0,00	6,61
		10	0,30	8,83	0,03	79,34	0,30	19,44
	5	2	0,04	0,48	0,01	2,00	0,03	3,25
		5	0,21	2,16	0,07	32,79	0,21	16,00
		10	0,43	5,36	0,23	217,73	0,25	175,70
	10	2	0,15	0,64	0,04	5,52	0,04	20,61
		5	0,12	4,14	0,08	75,17	0,03	101,74
		10	0,39	6,02	0,34	328,65	0,28	578,97
	20	2	0,17	1,51	0,04	16,53	0,06	156,65
		5	0,26	2,67	0,14	236,26	0,12	503,92
		10	0,41	6,45	0,31	1140,42	0,33	1334,61
Среднее			0,21	3,50	0,12	178,86	0,14	243,19

m	n^L	n^F	$(LS \mid LS)^{1,5n^L}$		$(LS \mid VND_2)^{1,5n^L}$		$(VND_2 \mid LS)^{1,5n^L}$	
			gap	время	gap	время	gap	время
30	2	2	0,00	0,42	0,00	0,77	0,00	0,80
		5	0,10	3,79	0,15	12,12	0,10	8,90
		10	0,25	16,19	0,02	88,48	0,25	32,56
	5	2	0,01	0,83	0,01	1,97	0,00	4,17
		5	0,05	12,89	0,02	40,93	0,05	32,78
		10	0,07	48,53	0,06	336,63	0,06	172,41
	10	2	0,04	2,38	0,04	5,35	0,00	18,14
		5	0,02	12,33	0,05	95,29	0,00	91,12
		10	0,08	57,86	0,08	913,87	0,08	383,02
	20	2	0,05	4,03	0,04	16,49	0,02	156,57
		5	0,03	49,76	0,00	330,36	0,03	322,49
		10	0,02	161,60	0,09	2416,76	0,01	1357,19
Среднее			0,06	30,88	0,05	354,92	0,05	215,01

Отметим, что некоторые алгоритмы не нашли допустимых решений с ненулевым доходом лидера для части примеров (например, алгоритм LS | LS в случае размерности $m = 30$, $n^L = 2$, $n^F = 10$ первой группы тестов). В таких примерах стартовое решение содержит нулевой доход, а процедура локального поиска не может улучшить решение в силу того, что предполагаемый доход существенно отличается от реального,

поскольку решение задачи последователя приближённое. В связи с этим проведён эксперимент, в котором при фиксированной цене лидера сравниваются доходы лидера при оптимальном кооперативном решении последователя и решении последователя с отсечённым оптимальным решением. В результате в 9% случаев доходы лидера не совпали.

Также тестировались алгоритмы локального поиска для задачи последователя, использующие помимо стратегии первого улучшения стратегию наибольшего улучшения, но при схожих значениях целевой функции они тратят большее вычислительное время, что видно из результатов, представленных в табл. 7.

Заключение

Результаты экспериментов демонстрируют работоспособность предложенных алгоритмов для решения трёхуровневых задач ценообразования. Можно сделать вывод, что для прикладных задач предпочтительно использовать алгоритм LS | (LS | EX), который за разумное время находит (в среднем) лучшие решения. Однако для ситуации, когда требуется найти наилучшее решение за разумное время, экспериментально выявлен класс недоминируемых алгоритмов.

Также построены алгоритмы вычисления верхних и нижних границ. Как показал численный эксперимент, на практике они дают плохое представление об оптимуме задачи.

С практической точки зрения исследование может быть продолжено, поскольку имеются возможности для существенного улучшения предложенных алгоритмов. В частности, можно разработать методы отсечения бесперспективных элементов окрестностей, рассмотреть различные способы построения стартовых решений и исследовать другие эвристические подходы для приближённого решения задачи последователя.

Финансирование работы

Исследование выполнено в рамках государственного задания Института математики им. С. Л. Соболева (проект № FWNF–2026–0021). Дополнительных грантов на проведение или руководство этим исследованием получено не было.

Конфликт интересов

Авторы заявляют, что у них нет конфликта интересов.

Литература

1. **Панин А. А., Плясунов А. В.** Задача ценообразования. Часть 1. Точные и приближённые алгоритмы решения // Дискрет. анализ и исслед. операций. 2012. Т. 19, № 5. С. 83–100.

2. **Плясунов А. В., Панин А. А.** Задача ценообразования. Часть 2. Вычислительная сложность // Дискрет. анализ и исслед. операций. 2013. Т. 19, № 6. С. 56–71.
3. **Дементьев В. Т., Шамардин Ю. В.** Задача о выборе цен на продукцию при условии обязательного удовлетворения спроса // Дискрет. анализ и исслед. операций. Сер. 2. 2002. Т. 9, № 2. С. 31–40.
4. **Lederes Ph. J., Thisse J.-F.** Competitive location on network under delivered pricing // Oper. Res. Lett. 1990. V. 9, No. 3. P. 147–154. DOI: 10.1016/0167-6377(90)90012-T.
5. **Aboolian R., Berman O., Krass D.** Optimizing pricing and location decisions for competitive service facilities charging uniform price // J. Oper. Res. Soc. 2008. V. 59, No. 11. P. 1506–1519. DOI: 10.1057/palgrave.jors.2602493.
6. **Luer-Villagra A., Marianov V.** A competitive hub location and pricing problem // Eur. J. Oper. Res. 2013. V. 231, No. 3. P. 734–744. DOI: 10.1016/j.ejor.2013.06.006.
7. **Serra D., ReVelle Ch.** Competitive locations and pricing on networks // Geogr. Anal. 1999. V. 31, No. 2. P. 109–129. DOI: 10.1111/gean.1999.31.1.109.
8. **Aboolian R., Berman O., Krass D.** Competitive facility location model with concave demand // Eur. J. Oper. Res. 2007. V. 181, No. 2. P. 598–619. DOI: 10.1016/j.ejor.2005.10.075.
9. **Bouhtou M., Grigoriev A., van Hoesel S., van der Kraaij A. F., Spieksma F. C. R., Uetz M.** Pricing bridges to cross a river // Nav. Res. Logist. 2007. V. 54, No. 4. P. 411–420. DOI: 10.1002/nav.20216.
10. **Diakova Z. S., Kochetov Yu. A.** A double VNS heuristic for the facility location and pricing problem // Electron. Notes Discrete Math. 2012. V. 39, No. 4. P. 29–34. DOI: 10.1016/j.endm.2012.10.005.
11. **Ahmadi-Javid A., Amire E., Meskar M.** A profit-maximization location-routing-pricing problem: A branch-and-price algorithm // Eur. J. Oper. Res. 2018. V. 271, No. 3. P. 866-881. DOI: 10.1016/j.ejor.2018.02.020.
12. **Lin Y. H., Tian Q.** Facility location and pricing problem: Discretized mill price and exact algorithms // Eur. J. Oper. Res. 2023. V. 308, No. 2. P. 568–580. DOI: 10.1016/j.ejor.2022.11.052.
13. **Hansen P., Hanjoul P., Thisse J.-F., Peeters D.** Uncapacitated plant location under alternative spatial price policies // Manag. Sci. 1990. V. 36, No. 1. P. 41–57. DOI: 10.1287/mnsc.36.1.41.
14. **Garcia-Herreros P., Florensa C., Pratik M.** Capacity planning with competitive decision-makers: Trilevel milp formulation, degeneracy and solution approaches // Eur. J. Oper. Res. 2017. V. 262, No. 2. P. 449–463. DOI: 10.1016/j.ejor.2017.04.013.
15. **Plyasunov A. V., Panin A. A.** The multilevel facility location and pricing problems: the computational complexity and the stability analysis // Optim. Lett. 2022. V. 17, No. 6. P. 1295–1315. DOI: 10.1007/s11590-022-01924-3.

16. Кочетов Ю. А., Панин А. А., Плясунов А. В. Сравнение метаэвристик для решения двухуровневой задачи размещения предприятий и фабричного ценообразования // Дискрет. анализ и исслед. операций. 2015. Т. 22, № 3. С. 36–54. DOI: 10.17377/daio.2015.22.480.

17. Sinnl M., Fischetti M., Ljubić I., Monaci M. Intersection cuts for bilevel optimization // Integer programming and combinatorial optimization. Proc. 18th Int. Conf. (Liège, Belgium, June 1–3, 2016). Cham: Springer, 2016. P. 77–88. (Lect. Notes Comput. Sci.; V. 9682.) DOI: 10.1007/978-3-319-33461-5_7.

18. Hansen P. N., Mladenović N. Variable neighborhood search // Eur. J. Oper. Res. 2001. V. 130, No. 3. P. 449–467. DOI: 10.1016/S0305-0548(97)00031-2.

19. Mladenović N., Hansen P. First vs. best improvement: An empirical study // Discrete Appl. Math. 2006. V. 154, No. 5. P. 802–817. DOI: 10.1016/j.dam.2005.05.020.

20. Задача размещения и ценообразования // Дискретные задачи размещения. Новосибирск: ИМ СО РАН, 2025. URL: old.math.nsc.ru/AP/benchmarks/Pricing/price.html (accessed: 20.10.2025).

Драчев Руслан Романович
Панин Артём Александрович
Плясунов Александр Владимирович

Статья поступила
24 декабря 2025 г.
После доработки —
12 февраля 2026 г.
Принята к публикации
23 марта 2026 г.

DISKRETNYI ANALIZ I ISSLEDOVANIE OPERATSII
/DISCRETE ANALYSIS AND OPERATIONS RESEARCH/
April–June 2026. Vol. 33, No. 2. P. 19–45

UDC 519.8+518.25 DOI: 10.33048/daio.2026.33.850

UPPER AND LOWER BOUNDS AND VARIABLE NEIGHBOURHOOD DESCENT FOR COMPETITIVE PRICING PROBLEM

R. R. Drachev[a], *A. A. Panin*[b], and *A. V. Plyasunov*[c]

[1] Sobolev Institute of Mathematics,
4 Acad. Koptyug Avenue, 630090 Novosibirsk, Russia
E-mail: [a] rrdrachev@gmail.com,
[b] aapanin1988@gmail.com, [c] apljas@math.nsc.ru

Abstract. This article considers a three-level competitive pricing problem. Two companies producing a homogeneous product compete for consumer demand. At the first level, the leader and the follower set prices at each of their facilities to maximize their own revenue. At the third level, each client chooses the facility with the lowest total purchasing and transportation costs and makes a purchase if these costs do not exceed their budget. The goal of the game is to find the leader's prices that maximize their revenue. The problem is known to be Σ_2^p-hard. Lower and upper bounds for the optimum are proposed, and approximate solution algorithms based on descent with alternating neighborhoods are developed. A computational experiment is conducted using examples of varying dimensions and complexity. Tab. 7, illustr. 2, bibliogr. 20.

Keywords: three-level programming, Stackelberg game, competitive pricing, local search, VND.

References

1. **A. V. Plyasunov** and **A. A. Panin,** The pricing problem. Part I: Exact and approximate algorithms, *Diskretn. Anal. Issled. Oper.* **19** (5), 83–100 (2012) [Russian] [*J. Appl. Ind. Math.* **7** (2), 241–251 (2013), DOI: 10.1134/S1990478913020142].
2. **A. V. Plyasunov** and **A. A. Panin,** The pricing problem. Part II: Computational complexity, *Diskretn. Anal. Issled. Oper.* **19** (6), 56–71 (2013) [Russian] [*J. Appl. Ind. Math.* **7** (3), 420–430 (2013), DOI: 10.1134/S1990478913030150].

English transl.: *Journal of Applied and Industrial Mathematics* **20** (2) (2026).

3. **V. T. Dementyev** and **Yu. V. Shamardin,** The problem of price selection for production under the condition of obligatory satisfaction of demand, *Diskretn. Anal. Issled. Oper., Ser. 2*, **9** (2), 31–40 (2002) [Russian].
4. **Ph. J. Lederes** and **J.-F. Thisse,** Competitive location on network under delivered pricing, *Oper. Res. Lett.* **9** (3), 147–154 (1990), DOI: 10.1016/0167-6377(90)90012-T.
5. **R. Aboolian, O. Berman,** and **D. Krass,** Optimizing pricing and location decisions for competitive service facilities charging uniform price, *J. Oper. Res. Soc.* **59** (11), 1506–1519 (2008), DOI: 10.1057/palgrave.jors.2602493.
6. **A. Luer-Villagra,** and **V. Marianov,** A competitive hub location and pricing problem, *Eur. J. Oper. Res.* **231** (3), 734–744 (2013), DOI: 10.1016/j.ejor.2013.06.006.
7. **D. Serra** and **Ch. ReVelle,** Competitive locations and pricing on networks, *Geogr. Anal.* **31** (2), 109–129 (1999), DOI: 10.1111/gean.1999.31.1.109.
8. **R. Aboolian, O. Berman,** and **D. Krass,** Competitive facility location model with concave demand, *Eur. J. Oper. Res.* **181** (2), 598–619 (2007), DOI: 10.1016/j.ejor.2005.10.075.
9. **M. Bouhtou, A. Grigoriev, S. van Hoesel, A. F. van der Kraaij, F. C. R. Spieksma,** and **M. Uetz,** Pricing bridges to cross a river, *Nav. Res. Logist.* **54** (4), 411–420 (2007), DOI: 10.1002/nav.20216.
10. **Z. S. Diakova** and **Yu. A. Kochetov,** A double VNS heuristic for the facility location and pricing problem, *Electron. Notes Discrete Math.* **39** (4), 29–34 (2012), DOI: 10.1016/j.endm.2012.10.005.
11. **A. Ahmadi-Javid, E. Amire,** and **M. Meskar,** A profit-maximization location-routing-pricing problem: A branch-and-price algorithm, *Eur. J. Oper. Res.* **271** (3), 866-881 (2018), DOI: 10.1016/j.ejor.2018.02.020.
12. **Y. H. Lin** and **Q. Tian,** Facility location and pricing problem: Discretized mill price and exact algorithms, *Eur. J. Oper. Res.* **308** (2), 568–580 (2023), DOI: 10.1016/j.ejor.2022.11.052.
13. **P. Hansen, P. Hanjoul, J.-F. Thisse,** and **D. Peeters,** Uncapacitated plant location under alternative spatial price policies, *Manag. Sci.* **36** (1), 41–57 (1990), DOI: 10.1287/mnsc.36.1.41.
14. **P. Garcia-Herreros, C. Florensa,** and **M. Pratik,** Capacity planning with competitive decision-makers: Trilevel milp formulation, degeneracy and solution approaches, *Eur. J. Oper. Res.* **262** (2), 449–463 (2017), DOI: 10.1016/j.ejor.2017.04.013.
15. **A. V. Plyasunov** and **A. A. Panin,** The multilevel facility location and pricing problems: the computational complexity and the stability analysis, *Optim. Lett.* **17** (6), 1295–1315 (2022), DOI: 10.1007/s11590-022-01924-3.
16. **Yu. A. Kochetov, A. A. Panin,** and **A. V. Plyasunov,** Comparison of metaheuristics for the bilevel facility location and mill pricing problem, *Diskretn. Anal. Issled. Oper.* **22** (3), 36–54 (2015), DOI: 10.17377/daio.2015.22.480 [Russian] [*J. Appl. Ind. Math.* **9** (3), 392–401 (2015), DOI: 10.1134/S1990478915030102].

17. M. Sinnl, M. Fischetti, I. Ljubić, and **M. Monaci,** Intersection cuts for bilevel optimization, in *Integer Programming and Combinatorial Optimization*, Proc. 18th Int. Conf. (Liège, Belgium, June 1–3, 2016) (Springer, Cham, 2016), pp. 77–88 (Lect. Notes Comput. Sci., Vol. 9682.) DOI: 10.1007/978-3-319-33461-5_7

18. P. N. Hansen and **N. Mladenović,** Variable neighborhood search, *Eur. J. Oper. Res.* **130** (3), 449–467 (2001), DOI: 10.1016/S0305-0548(97)00031-2.

19. N. Mladenović and **P. Hansen,** First vs. best improvement: An empirical study, *Discrete Appl. Math.* **154** (5), 802–817 (2006), DOI: 10.1016/j.dam.2005.05.020.

20. Facility location and pricing problem, in *Discrete Location Problems* (IM SO RAN, Novosibirsk, 2025) [Russian], URL: old.math.nsc.ru/AP/benchmarks/Pricing/price.html (accessed: 20.10.2025).

Ruslan R. Drachev
Artem A. Panin
Aleksandr V. Plyasunov

Received December 24, 2025
Revised February 12, 2026
Accepted March 23, 2026

ДИСКРЕТНЫЙ АНАЛИЗ И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ
Апрель–июнь 2026. Т. 33, № 2. С. 46–63

УДК 519.852+519.856+519.6 DOI: 10.33048/daio.2026.33.848

МЕТОД ВЕТВЕЙ И ОТСЕЧЕНИЙ ДЛЯ ЗАДАЧИ ВЗВЕШЕННОЙ 3-РАСКРАСКИ ГРАФА

Г. А. Жуков[1, a], *А. И. Ерзин*[1, 2, b], *Р. В. Плотников*[3, c]

[1] Новосибирский гос. университет,
ул. Пирогова, 2, 630090 Новосибирск, Россия
[2] Институт математики им. С. Л. Соболева,
пр. Акад. Коптюга, 4, 630090 Новосибирск, Россия
[3] ООО «Ледас»,
ул. Николаева, 11/5, 630090 Новосибирск, Россия
E-mail: [a] g.zhukov@g.nsu.ru,
[b] adilerzin@math.nsc.ru, [c] rvplotnikov@gmail.com

Аннотация. Для рёберно-взвешенного неориентированного графа рассматривается NP-трудная задача раскраски вершин в три цвета при условии минимизации суммарного веса рёбер, соединяющих вершины одного цвета. Данная задача математически эквивалентна известной задаче о максимальном разрезе на k частей (о максимальном k-разрезе, maximum k-cut), где требуется максимизировать вес рёбер между вершинами из различных подмножеств. В настоящей работе представлен точный алгоритм для случая $k = 3$, реализованный в рамках схемы ветвей и отсечений (branch-and-cut). В то время как большинство существующих исследований сфокусированы на классах допустимых неравенств, ключевым вкладом данной работы является метод уменьшения глубины дерева ветвления на основе декомпозиции вершин, интегрированный в схему branch-and-cut. Вычислительные эксперименты демонстрируют высокую эффективность разработанного алгоритма. Табл. 4, ил. 1, библиогр. 28.

Ключевые слова: 3-раскраска, метод ветвей и отсечений, задача о максимальном k-разрезе.

1. Обзор литературы

В настоящей работе исследуется следующая NP-трудная задача. Дан взвешенный неориентированный граф $G = (V, E)$, каждому ребру $e \in E$ приписан положительный вес w_e. Требуется раскрасить вершины графа в три цвета так, чтобы сумма весов рёбер, соединяющих вершины

© Г. А. Жуков, А. И. Ерзин, Р. В. Плотников, 2026

одного цвета (одноцветных рёбер), была минимальной. Будем называть такую постановку задачей о минимальном 3-разбиении, поскольку в англоязычной литературе она известна как minimum 3-partition problem. В эквивалентной формулировке максимизации суммарного веса рёбер разреза, т. е. рёбер, соединяющих вершины разных кластеров (цветов), имеем задачу о максимальном 3-разрезе — maximum 3-cut problem.

1.1. Задачи раскраски графов. Исследуемая задача тесно связана с классическими проблемами раскраски графов: 1) определить хроматическое число $\chi(G)$ графа G, т. е. минимальное число цветов для правильной раскраски, в которой смежные вершины имеют разные цвета; 2) установить, допускает ли граф правильную раскраску в три цвета.

Задача определения $\chi(G)$ входит в список из 21 NP-полной задачи Карпа [1]. Много работ посвящено точным и приближённым методам её решения; фундаментальные обзоры представлены в [2–4]. Классической конструктивной эвристикой является алгоритм DSATUR [5], который динамически упорядочивает вершины, выбирая на каждом шаге ту, что имеет наибольшее число разноцветных соседей (степень насыщения).

В классе точных методов ключевая роль за работой [6], где предложен подход branch-and-price, моделирующий задачу как разбиение на независимые множества и решающий её с помощью процедуры генерации столбцов. Для больших графов более целесообразно применение метаэвристических подходов. В частности, гибридный эволюционный алгоритм из [7] показывает хорошие результаты за счёт применения специализированных операторов скрещивания и интеграции метаэвристики поиска с запретами, а в [8] авторы используют декомпозицию вершин для итеративного упрощения графа и сужения пространства поиска.

Значительная часть теоретических исследований обращена к гипотезе Бородина — Косточки, согласно которой хроматическое число удовлетворяет неравенству $\chi(G) \leqslant \max\{\omega, \Delta - 1\}$ при максимальной степени графа $\Delta \leqslant 9$, где ω — размер максимальной клики. Так, в [9] доказано, что некоторые гипотезы, на первый взгляд кажущиеся более слабыми, чем гипотеза Бородина — Косточки, на самом деле ей эквивалентны. В [10] получен аналогичный результат для списочной раскраски: неравенство $\chi < \Delta$ выполняется для всех графов с достаточно большим Δ при условии $\omega < \Delta$. В [11] представлен детерминированный алгоритм, строящий $O(k\Delta)$-раскраску вершин за $O(\Delta/k) + \log n$ раундов, где $k \in [1, O(\Delta)]$.

Задача проверки существования правильной раскраски графа в три цвета NP-полна даже для планарных 4-регулярных графов [12]. Лучший известный точный алгоритм для общего случая имеет сложность $O(1{,}3217^n)$ [13], а для планарных графов без треугольников существует линейный алгоритм 3-раскраски [14].

1.2. Аппроксимационные алгоритмы для задачи о максимальном k-разрезе. Задача о максимальном k-разрезе являет собой обобщение классической задачи о максимальном разрезе (maximum 2-cut). В [15] предложен приближённый алгоритм на основе полуопределённого программирования (semidefinite programming, SDP), гарантирующий коэффициент аппроксимации 0,87856 для $k = 2$. В [16] атворы обобщили этот результат для задачи о максимальном k-разрезе, предложив SDP-релаксацию с гарантированной точностью, зависящей от k (для $k = 3$ коэффициент составлял 0,800207). Впоследствии данные оценки были уточнены: в частности, для $k = 3$ соответствующий коэффициент был улучшен до 0,836008 [17].

1.3. Полиэдральная комбинаторика и метод ветвей и отсечений. Один из наиболее эффективных подходов для точного решения задачи о максимальном k-разрезе — это метод ветвей и отсечений (branch-and-cut), использующий свойства полиэдра задач.

Существует две основные формулировки задачи: вершинная и рёберная. Вершинные формулировки используют булевы переменные x_{ik}, указывающие, принадлежит ли вершина i кластеру k (покрашена ли вершина i в цвет k). Такие модели обладают недостатком — симметрией (перестановка цветов меняет решение, но не изменяет значения целевой функции). Для борьбы с симметрией применяются специальные неравенства и методы изоморфного отсечения [18].

Рёберные формулировки более перспективны для метода ветвей и отсечений. Они используют переменные y_{ij}, которые ассоциированы с рёбрами и равны 1, если вершины i и j лежат в разных кластерах (имеют разные цвета). Однако такие формулировки содержат большое число неравенств (порядка $\binom{n}{k+1}$). Для получения верхних (для задачи на максимум) оценок также используют формулировку полуопределённого программирования.

Статья [19] — одна из ключевых работ в области полиэдрального анализа задачи о k-разбиении. Её авторы детально изучили полиэдр задачи для произвольного k и описали несколько важнейших классов отсечений: общие кликовые неравенства (general clique inequalities), неравенства типа «колесо» (wheel inequalities), неравенства типа «велосипедное колесо» (bicycle wheel inequalities). Эти классы неравенств широко применяются в современных алгоритмах ветвей и отсечений.

Автор [20] через анализ гиперметрических неравенств объясняет высокое качество нижних оценок, получаемых с помощью релаксации модели полуопределённого программирования для задачи о k-разбиении. При этом доказано, что полуопределённая релаксация неявно содержит ослабленные версии этих ограничений, и установлены точные границы

максимального нарушения фасетоопределяющих неравенств (треугольных и кликовых) её допустимыми решениями.

Во многих работах метод ветвей и отсечений применяется для решения задач о минимальном k-разбиении и максимальном k-разрезе. Фундаментальный обзор математических формулировок, допустимых неравенств и ключевых элементов метода ветвей и отсечений для задачи о максимальном k-разрезе представлен в работе [21]. В [22] для задачи о минимальном k-разбиении предложен алгоритм ветвей и отсечений, использующий для вычисления нижних оценок формулировку полуопределённого программирования, а для поиска допустимых решений — новую эвристику итеративной кластеризации. Вычислительный эксперимент показал, что данная эвристика находит лучшие допустимые решения, чем классические техники округления с помощью гиперплоскостей.

Авторы [23] провели сравнительный анализ четырёх постановок задачи о минимальном k-разбиении: двух вершинных и двух рёберных. Они исследовали структуру политопов для рёберных моделей и описали новый класс фасетоопределяющих неравенств. Эксперименты показали, что вершинные формулировки обладают слабой линейной релаксацией, тогда как рёберные обеспечивают значительно более качественные оценки. На основе рёберной формулировки была предложена эффективная реализация метода ветвей и отсечений.

В работе [24] изучается обобщение задачи о максимальном k-разрезе, в котором дополнительно требуется, чтобы каждый из получаемых k кластеров индуцировал связный подграф. Авторы предложили новую модель целочисленного линейного программирования, а также метод ветвей и отсечений. Для повышения вычислительной эффективности применён анализ изоморфизма графов, позволяющий учитывать структуру рассматриваемых экземпляров.

2. Постановка задачи

Сформулируем задачу взвешенной 3-раскраски графа. Рассмотрим простой неориентированный граф $G = (V, E)$, где V — множество вершин, а E — множество рёбер. Каждому ребру $e \in E$ сопоставлен положительный вес $w(e) > 0$. Задача заключается в раскраске вершин графа в три цвета так, чтобы минимизировать суммарный вес рёбер, соединяющих вершины одного цвета.

Эта задача NP-трудна. Действительно, известно, что задача 3-раскрашиваемости графа NP-полна даже для планарных 4-регулярных графов [12]. Покажем сводимость этой классической задачи к рассматриваемой. Положим веса всех рёбер равными единице, т. е. $w(e) = 1$ для любого $e \in E$. В рассматриваемой постановке оптимальное значение целевой функции равно нулю тогда и только тогда, когда граф допускает

правильную 3-раскраску. Следовательно, существование полиномиального алгоритма для задачи взвешенной 3-раскраски влекло бы за собой полиномиальную разрешимость NP-полной задачи 3-раскрашиваемости графа, что невозможно в предположении $\mathrm{P} \neq \mathrm{NP}$.

Существует несколько постановок этой задачи. Рассмотрим следующую линейную модель, которую будем использовать в методе ветвей и отсечений. Введём булевы переменные:

- $x_{ik} = 1$, если вершина i имеет цвет k, и 0 иначе ($i \in V$, $k \in \{1, 2, 3\}$);
- $y_{ij} = 1$, если вершины i и j имеют одинаковый цвет, и 0 иначе ($(i, j) \in E$).

Задачу взвешенной 3-раскраски можно сформулировать в виде следующей модели целочисленного линейного программирования:

$$\min \sum_{(i,j)\in E} w_{ij} y_{ij},$$

$$\sum_{k=1}^{3} x_{ik} = 1, \quad i \in V,$$

$$y_{ij} \geqslant x_{ik} + x_{jk} - 1, \quad (i, j) \in E,\ k \in \{1, 2, 3\},$$

$$x_{ik} \in \{0, 1\}, \quad i \in V,\ k \in \{1, 2, 3\},$$

$$0 \leqslant y_{ij} \leqslant 1, \quad (i, j) \in E.$$

Правая часть ограничения $y_{ij} \geqslant x_{ik} + x_{jk} - 1$ всегда принимает целочисленные значения, поскольку переменные x_{ik} булевы. Так как целевая функция линейно зависит от y_{ij} (с положительными весами) и минимизируется, в оптимальном решении переменные y_{ij} будут принимать минимально допустимые значения, удовлетворяющие системе ограничений. Следовательно, при фиксированных x_{ik} переменные y_{ij} примут значения 0 или 1. Это позволяет опустить требование целочисленности y_{ij} и рассматривать их как непрерывные переменные на отрезке $[0, 1]$.

3. Метод ветвей и отсечений

В современной практике решения задач оптимизации широко применяются коммерческие решатели: CPLEX, Gurobi и др., которые предоставляют обширные возможности для настройки алгоритма ветвей и отсечений. В данном исследовании использовался решатель SCIP — свободно распространяемое программное обеспечение с открытым исходным кодом. Выбор SCIP обусловлен его высокой гибкостью в настройке метода ветвей и отсечений: пакет предоставляет широкий спектр точек настройки, детально описанных в официальной документации. В настоящей работе требуются следующие ключевые параметры настройки алгоритма.

1. Управление процессом ветвления — стратегия выбора переменной для фиксации её значения.

2. Стратегия обхода дерева поиска — порядок обработки узлов (в глубину, в ширину и другие альтернативные подходы).

3. Генерация отсечений — добавление ограничений для сужения пространства поиска.

Далее опишем метод декомпозиции вершин графа. Алгоритм декомпозиции основывается на построении разбиения множества вершин графа V на следующие подмножества:

- множество граничных вершин V_b (сепаратор);
- набор кластеров $\{V_1, V_2, \ldots, V_k\}$.

При этом должны выполняться условия

1) $V = V_b \cup V_1 \cup V_2 \cup \cdots \cup V_k$, причём $V_b, V_1, \ldots, V_k$ попарно не пересекаются;

2) для любых $i \neq j$ между вершинами кластеров V_i и V_j отсутствуют рёбра.

Пример декомпозиции вершин графа представлен на рис. 1.

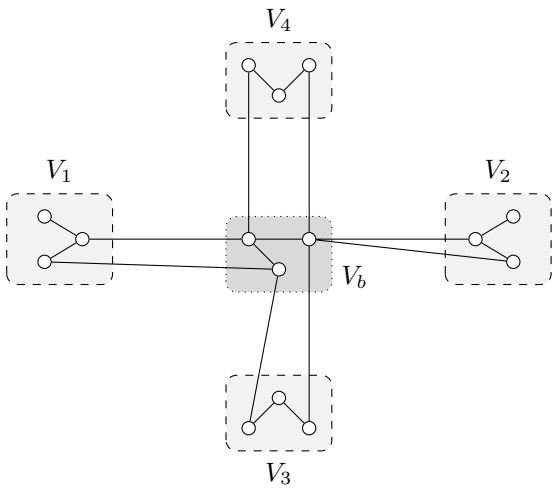


Рис. 1. Схематичное представление декомпозиции вершин графа

Если значения переменных x_{ik} зафиксированы для каждой граничной вершины (вершины, принадлежащей множеству граничных вершин V_b), то исходная задача распадается на набор независимых подзадач. В силу отсутствия рёбер между вершинами различных кластеров оптимальное решение для каждого кластера определяется исключительно цветами граничных вершин и может быть найдено независимо. Это позволяет

1) отсечь текущую ветвь в узле дерева поиска;

2) решить совокупность подзадач независимо;

3) передать полученное решение в основной алгоритм.

Однако на практике применение стандартных техник отсечения приводит к тому, что даже в случае разреженных графов мощность множества граничных вершин превышает максимальную глубину ветвления, определяемую как число вершин с зафиксированным цветом. Вследствие этого предложенный подход в своей исходной формулировке не обеспечивает снижения вычислительной трудоёмкости. Таким образом, необходимо модифицировать предложенную схему. Реализуем схему ветвления по вершинам с приоритетным выбором граничных вершин. Для текущей вершины i создаются подзадачи, соответствующие фиксации цвета ($x_{ik} = 1$, $k \in \{1, 2, 3\}$). В этом случае глубина узла в дереве поиска равна числу вершин, для которых цвет уже задан. Обозначим через d максимальную глубину дерева ветвления — ключевой параметр алгоритма. На глубине d выполняется следующая последовательность действий.

1. На основе вершин с зафиксированными цветами строится соответствующая частичная раскраска графа.

2. В множество подзадач добавляется новая задача, определяемая частичной раскраской, найденной на предыдущем шаге.

3. Текущая ветвь дерева ветвления исключается из дальнейшего рассмотрения.

Выполнение алгоритма предполагает обработку набора подзадач. Для этого используется выделенный набор вычислительных потоков (thread pool), в котором задачи решаются параллельно по мере поступления. Решение задач осуществляется по следующей схеме.

Шаг 1. Вычисление нижней оценки.

Шаг 1.1. Решается релаксация задачи целочисленного линейного программирования (ЦЛП), в которой переменные x_{ik}, соответствующие граничным вершинам, рассматриваются как непрерывные, при этом фиксируется часть переменных — те, для которых цвет задан в текущем частичном решении.

Шаг 1.2. Если полученная нижняя оценка позволяет отсечь текущую ветвь дерева поиска, то процесс завершается. Иначе осуществляется переход к следующему шагу.

Шаг 2. Поиск оптимального решения для текущей ветви.

Шаг 2.1. Решается задача ЦЛП, в которой цвета вершин, определённые в текущем частичном решении, фиксируются.

Шаг 2.2. Полученное решение сравнивается с наилучшим известным на данный момент решением.

Шаг 2.3. Если новое решение превосходит текущее лучшее, то оно передаётся в основной алгоритм для обновления рекорда.

В связи с использованием многопоточной обработки возникает потребность в синхронизации потоков. Однако детальное обсуждение методов синхронизации выходит за рамки настоящего исследования. Следует выделить существенный фактор, влияющий на производительность алгоритма, — частоту обновления рекорда. Поскольку данная операция выступает точкой синхронизации, её частое выполнение приводит к снижению степени параллелизма. Во избежание этого целесообразно инициализировать алгоритм решением, близким к оптимальному. Для нахождения начального решения рационально использовать метаэвристические методы. При реализации предложенного подхода был использован алгоритм имитации отжига. Ключевые преимущества алгоритма имитации отжига — высокая скорость работы и качество получаемых решений.

Эффективность предложенного метода существенно зависит от мощности множества граничных вершин: сокращение мощности этого множества позволяет уменьшить глубину ветвления d, тем самым обеспечивая выигрыш в производительности. Если не ограничивать число вершин в кластерах, то решение задачи декомпозиции вершин графа вырождается в разбиение, включающее множество граничных вершин и один кластер. Во избежание тривиального решения введём дополнительное ограничение на максимальный размер кластера. Для декомпозиции множества вершин предлагается комбинированный подход, сочетающий

- рандомизированный алгоритм;
- алгоритм имитации отжига.

Схема рандомизированного алгоритма выглядит следующим образом.

Шаг 1. Вершины перемешиваются в случайном порядке.

Шаг 2. Для каждой вершины v выполняется следующая проверка.

Шаг 2.1. Если v не смежна ни с одним из существующих кластеров, то

- если существует кластер, для которого не нарушается ограничение на размер, то вершина v добавляется в него;
- иначе создаётся новый кластер, состоящий из вершины v.

Шаг 2.2. Если v смежна ровно с одним кластером, то

- вершина v добавляется в кластер при соблюдении ограничения на его размер;
- иначе вершина v добавляется в множество граничных вершин.

Шаг 2.3. Если v смежна более чем с одним кластером, то вершина v добавляется в множество граничных вершин.

Алгоритм имитации отжига, применяемый для декомпозиции вершин графа, использует экспоненциальную функцию снижения температуры. Функция энергии определяется как мощность множества граничных вершин. На каждой итерации случайно выбранная вершина кластера добавляется в множество граничных вершин. После этого для соседей данной вершины, принадлежащих множеству граничных, проверяется возможность добавления в один из кластеров; при выполнении условий вершина добавляется в соответствующий кластер.

Осталось рассмотреть три ключевых компонента: стратегию обхода дерева поиска, правило выбора вершины ветвления и генерацию отсечений. В качестве стратегии обхода выбран поиск в глубину (depth-first search). Такой выбор обусловлен низким потреблением памяти, а также возможностью эффективного пересчёта релаксаций при переходе к дочерним узлам (за счёт использования «горячего» старта и других механизмов современных решателей). Эффективный метод выбора переменной ветвления даёт стратегия, основанная на псевдостоимостях (pseudo-costs [25]). Решатель SCIP вычисляет эти значения для каждой переменной и предоставляет программный интерфейс для доступа к ним. В качестве вершины ветвления выбирается граничная вершина v, цвет которой ещё не зафиксирован, обладающая максимальной суммарной псевдостоимостью. Суммарная псевдостоимость вершины v определяется как величина $\sum_{k=1}^{3} c(x_{vk})$, где $c(x_{vk})$ — псевдостоимость переменной x_{vk}. Для генерации отсечений используются неравенства типа «велосипедное колесо» [19], построенные на цикле длины 3, показавшие наибольшую эффективность в предварительном численном эксперименте.

4. Вычислительный эксперимент

Вычислительный эксперимент проведён на персональном компьютере с процессором Intel Core i9-13900HK и 32 ГБ оперативной памяти. Алгоритм реализован на языке C++ с использованием оптимизационного пакета SCIP версии 9.2.3. Оценка эффективности предложенного подхода проведена в сравнении с базовой конфигурацией решателя SCIP. Для корректности в обеих версиях использованы идентичные настройки: стратегия обхода в глубину и генерация отсечений типа «велосипедное колесо». В дальнейшем разработанный метод ветвей и отсечений будем называть SCIP-mod, а базовую конфигурацию — SCIP-std.

Для эксперимента сформированы четыре группы тестовых графов:

- выборка примеров из работы [24];
- набор компонент связности сетей мобильной связи;
- графы, полученные с помощью генератора [26], предоставленного организаторами конкурса MOTOR 2025 Challenge 5 [27];

Таблица 1

Графы из статьи [24]

n	m	SCIP-mod		SCIP-std		δ, %
		t, с	N	t, с	N	
50	306	64	20 829	132	12 153	51,5
		68	23 114	119	8 780	42,9
		74	20 114	95	7 313	22,1
		118	29 058	141	13 994	16,3
		183	31 873	234	38 438	21,8
70	232	24	15 309	59	7 682	59,3
	235	8	7 069	13	1 640	38,5
	245	20	10 309	57	4 752	64,9
	250	50	19 474	147	15 551	66
	254	91	20 884	145	23 051	37,2
80	346	2	1 166	3	351	33,3
	350	232	26 991	321	37 931	27,7
		508	66 392	538	95 395	5,6
		439	42 431	364	51 425	−20,6
		280	38 163	269	29 807	−4,1
93	593	56	9 020	32	2 301	−75
114	546	9	2,184	8	1 012	−12,5
Общее время работы		2 226 с		2 677 с		16,8

• unit-disk-графы.

Примеры графов, сгенерированных с помощью генератора, и unit-disk-графов доступны для скачивания по ссылке [28]. Unit-disk-графы генерировались следующим образом: в единичном квадрате $[0, 1] \times [0, 1]$ случайным образом размещались n точек, соответствующих вершинам графа. Две вершины соединялись ребром тогда и только тогда, когда расстояние между соответствующими точками не превышало порогового значения $2r$ (что эквивалентно пересечению кругов радиуса r с центрами в этих точках).

Результаты численного эксперимента для каждой из четырёх рассматриваемых групп тестовых примеров представлены в виде табл. 1–4, в которых приняты следующие обозначения:

• n — число вершин графа;

• m — число рёбер;

• t — время работы алгоритма в секундах;

• N — число узлов дерева поиска, просмотренных в ходе работы алгоритма;

Таблица 2

Компоненты связности сетей мобильной связи

n	m	SCIP-mod		SCIP-std		δ, %
		t, с	N	t, с	N	
53	293	53	14 780	87	15 610	39,1
56	406	33	4 680	36	4 753	8,3
70	600	336	31 457	441	26 784	23,8
		705	46 076	1 355	204 065	47,9
114	891	1 450	338 470	1 673	366 494	13,3
Общее время работы		2 577 с		3 592 с		28,3

• δ — относительное сокращение времени вычисления в процентах, рассчитываемое по формуле

$$\delta = \frac{t_S - t_M}{t_S} \cdot 100\%, \tag{1}$$

где t_S и t_M — время работы SCIP-std и SCIP-mod соответственно.

В последней строке каждой таблицы приведено суммарное время работы стандартной и модифицированной версий решателя для соответствующей группы примеров, а также итоговый относительный выигрыш по времени, вычисленный по формуле (1).

Анализ полученных результатов показывает, что выигрыш по времени составил 16,8% для первой группы, 28,3% для второй, 33,1% для третьей и 41,2% для четвёртой. Стоит отметить, что в первой группе встречались отдельные задачи, на которых модифицированная версия уступала стандартной. Во всех остальных случаях предложенный метод показал меньшее время работы.

Что касается размера дерева поиска, то сокращение числа просмотренных узлов было зафиксировано примерно для 30% примеров первой группы, 80% — второй, 28% — третьей и 60% — четвёртой групп.

Отдельного пояснения требует тот факт, что размер дерева поиска сократился не во всех тестовых примерах. Причина заключается в различных схемах ветвления. В стандартной настройке решатель использует классическое ветвление по дробной переменной x_{vc}. В предлагаемой модификации применяется ветвление по вершинам: для выбранной вершины v создаются три дочерние ветви, фиксирующие её цвет ($x_{v1} = 1$, $x_{v2} = 1$, $x_{v3} = 1$ соответственно). Данная схема ветвления не была интегрирована в базовую версию SCIP по следующим причинам.

1. Реализация пользовательского правила ветвления вносит дополнительные вычислительные накладные расходы.

Таблица 3

Графы, полученные генератором MOTOR 2025 Challenge 5

n	m	SCIP-mod		SCIP-std		δ, %
		t, с	N	t, с	N	
90	355	42	11 676	76	4 113	44,7
	361	163	24 990	251	21 390	35,1
	372	42	11 230	116	5 132	63,8
	374	111	25 073	226	15 456	50,9
	385	55	15 576	116	6 844	52,6
95	350	48	10 325	65	6 109	26,2
	354	48	9 652	95	6 528	49,5
	360	83	12 727	137	9 717	39,4
	365	87	16 545	306	38 650	71,6
	391	212	30 481	234	25 712	9,4
100	410	433	49 392	460	48 541	5,9
	411	218	31 424	358	33 417	39,1
	413	300	35 442	366	27 556	18
	432	199	28 003	295	19 632	32,6
	467	236	27 887	310	17 049	23,9
105	357	228	39 061	250	24 574	8,8
	374	120	21 851	120	12 771	0
	376	155	12 725	214	19 137	27,6
	378	440	41 482	556	51 183	20,9
	386	163	25 063	241	20 831	32,4
110	323	324	34 535	384	61 219	15,6
	354	150	22 749	189	19 435	20,6
	362	57	11 899	83	5 779	31,3
	417	244	27 790	440	41 916	44,6
	471	538	40 114	1 515	200 173	64,5
Общее время работы		4 696 с		7 019 с		33,1

2. Предварительный эксперимент показал, что в рамках стандартной настройки переход к ветвлению по вершинам на значительной части тестов приводит к снижению производительности по сравнению со встроенным механизмом.

Результаты экспериментов подтверждают, что предложенная техника уменьшения глубины ветвления обеспечивает существенное увеличение производительности.

Заключение

В работе предложен новый метод построения оптимального решения для задачи о минимальном 3-разбиении. Основу этого метода составляет

Таблица 4

Unit-disk-графы

n	m	SCIP-mod		SCIP-std		δ, %
		t, с	N	t, с	N	
90	296	145	54 362	254	91 569	42,9
	304	160	44 775	201	44 140	20,4
		270	103 814	410	82 672	34,1
	308	297	54 997	905	263 058	67,2
	312	68	29 836	132	28 477	48,5
95	324	779	144 914	1 025	419 258	24
	339	318	78 337	354	39 948	10,2
	345	177	52 559	251	51 534	29,5
	346	2 332	222 172	3 075	958 663	24,2
	366	346	44 692	578	63 006	40,1
100	358	2 699	477 727	4 422	1 431 610	39
	367	137	34 526	419	132 001	67,3
	374	539	82 168	1 274	387 737	57,7
	376	206	37 958	376	36 415	45,2
	383	2 516	263 350	5 000	2 250 000	49,7
Общее время работы		10 990 с		18 676 с		41,2

сокращение глубины дерева ветвления посредством разбиения вершин графа на множество граничных вершин и кластеры, между которыми отсутствуют рёбра, с последующим приоритетным ветвлением по граничным вершинам. Результаты проведённого численного эксперимента подтвердили высокую эффективность разработанного метода.

Финансирование работы

Исследование выполнено в рамках государственного задания Институт математики им. С. Л. Соболева (проект № FWNF–2026–0021). Дополнительных грантов на проведение или руководство этим исследованием получено не было.

Конфликт интересов

Авторы заявляют, что у них нет конфликта интересов.

Литература

1. **Karp R. M.** Reducibility among combinatorial problems // Complexity of computer computations. Proc. Symp. (New York, USA, March 20–22, 1972). New York: Plenum Press, 1972. P. 85–103. DOI: 10.1007/978-1-4684-2001-2_9.

2. **Borodin O. V.** Colorings of plane graphs: A survey // Discrete Math. 2013. V. 313, No. 4. P. 517–539. DOI: 10.1016/j.disc.2012.11.011.
3. **Husfeldt T.** Graph colouring algorithms // Topics in chromatic graph theory. Cambridge: Cambridge Univ. Press, 2015. P. 277–303. (Encycl. Math. Appl.). DOI: 10.1017/CBO9781139519793.016.
4. **Kostochka A. V., Yancey M.** On coloring of sparse graphs // Computer science — Theory and applications. Proc. 8th Int. Comput. Sci. Symp. in Russia (Yekaterinburg, Russia, June 25–29, 2013). Heidelberg: Springer, 2013. P. 224–234. (Lect. Notes Comput. Sci.; V. 7913). DOI: 10.1007/978-3-642-38536-0_20.
5. **Brélaz D.** New methods to color the vertices of a graph // Commun. ACM. 1979. V. 22, No. 4. P. 251–256. DOI: 10.1145/359094.359101.
6. **Mehrotra A., Trick M. A.** A column generation approach for graph coloring // INFORMS J. Comput. 1996. V. 8, No. 4. P. 344–354. DOI: 10.1287/ijoc.8.4.344.
7. **Galinier P., Hao J.-K.** Hybrid evolutionary algorithms for graph coloring // J. Comb. Optim. 1999. V. 3. P. 379–397. DOI: 10.1023/A:1009823419804.
8. **Moalic L., Gondran A.** Variations on memetic algorithms for graph coloring problems // J. Heuristics. 2018. V. 24, No. 4. P. 1–24. DOI: 10.1007/s10732-017-9354-9.
9. **Cranston D. W., Rabern L. W.** Coloring a graph with $\Delta - 1$ colors: Conjectures equivalent to the Borodin–Kostochka conjecture that appear weaker // Eur. J. Comb. 2015. V. 44. P. 23–42. DOI: 10.1016/j.ejc.2014.09.006.
10. **Choi I., Kierstead H. A., Rabern L. W.** The list version of the Borodin–Kostochka conjecture for graphs with large maximum degree // Discrete Math. 2023. V. 346, No. 11. Article ID 113300. 12 p. DOI: 10.1016/j.disc.2022.113300.
11. **Maus Y.** Distributed graph coloring made easy. Ithaca, NY, 2021. 22 p. (e-Print Archive / Cornell Univ.; arXiv:2105.05575). DOI: 10.48550/arXiv.2105.05575.
12. **Dailey D. P.** Uniqueness of colorability and colorability of planar 4-regular graphs are NP-complete // Discrete Math. 1980. V. 30, No. 3. P. 289–293. DOI: 10.1016/0012-365X(80)90236-8.
13. **Meijer L.** 3-Coloring in time $O(1.3217^n)$. Ithaca, NY, 2023. 17 p. (e-Print Archive / Cornell Univ.; arXiv:2302.13644). DOI: 10.48550/arXiv.2302.13644.
14. **Dvořák Z., Kawarabayashi K.-I., Thomas R.** Three-coloring triangle-free planar graphs in linear time // ACM Trans. Algorithms. 2011. V. 7, No. 4. Article ID 41. 14 p. DOI: 10.1145/2000807.2000809.
15. **Goemans M., Williamson D. P.** Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming // J. ACM. 1995. V. 42, No. 6. P. 1115–1145. DOI: 10.1145/227683.227684.
16. **Frieze A., Jerrum M.** Improved approximation algorithms for MAX-k-CUT and MAX BISECTION // Algorithmica. 1997. V. 18. P. 67–81. DOI: 10.1007/BF02523688.

17. De Klerk E., Pasechnik D. V., Warners J. P. On approximate graph colouring and MAX-k-CUT algorithms based on the ϑ-function // J. Comb. Optim. 2004. V. 8, No. 3. P. 267–294. DOI: 10.1023/B:JOCO.0000038911.67280.3f.

18. Margot F. Pruning by isomorphism in Branch-and-cut // Math. Program. Ser. A. 2002. V. 94. P. 71–90. DOI: 10.1007/s10107-002-0358-2.

19. Chopra S., Rao M. The partition problem // Math. Program. 1993. V. 59, No. 1. P. 87–115. DOI: 10.1007/BF01581239.

20. Eisenblätter A. The semidefinite relaxation of the k-partition polytope is strong // Integer programming and combinatorial optimization. Proc. 9th Int. IPCO Conf. (Cambridge, USA, May 27–29, 2002). Heidelberg: Springer, 2002. P. 273–290. (Lect. Notes Comput. Sci.; V. 2337). DOI: 10.1007/3-540-47867-1_20.

21. De Sousa R. Global optimization of the maximum K-cut problem: PhD Thesis. Montreal, 2018. 95 p.

22. Ghaddar B., Anjos M. F., Liers F. A branch-and-cut algorithm based on semidefinite programming for the minimum k-partition problem // Ann. Oper. Res. 2011. V. 188, No. 1. P. 155–174. DOI: 10.1007/s10479-008-0481-4.

23. Ales Z., Knippel A. The K-partitioning problem: Formulations and branch-and-cut // Networks. 2020. V. 76, No. 3. P. 323–349. DOI: 10.1002/net.21944.

24. Healy P., Jozefowiez N., Laroche P., Marchetti F., Martin S., Róka Z. A branch-and-cut algorithm for the connected max-k-cut problem // Eur. J. Oper. Res. 2024. V. 312, No. 1. P. 117–124. DOI: 10.1016/j.ejor.2023.06.015.

25. Benichou M., Gauthier J. M., Girodet P., Hentges G., Ribiere G., Vincent O. Experiments in mixed-integer linear programming // Math. Program. 1971. V. 1. P. 76–94. DOI: 10.1007/BF01584074.

26. Kovrizhnykh N. A. 3-Coloring: Graph generator for MOTOR 2025 Challenge 5. 2025. URL: github.com/sedefe/3-coloring (accessed: 6.08.2026).

27. MOTOR 2025 challenges. Novosibirsk: IM SO RAN, 2025. URL: old.math.nsc.ru/conference/motor/2025/challenges.html (accessed: 6.08.2026).

28. Dataset of generated and unit-disk graphs. 2025. URL: https://disk.yandex.ru/d/cbMBI1Rp0J535Q (accessed: 6.08.2026).

Жуков Георгий Алексеевич
Ерзин Адиль Ильясович
Плотников Роман Викторович

Статья поступила
17 декабря 2025 г.
После доработки —
15 февраля 2026 г.
Принята к публикации
23 марта 2026 г.

DISKRETNYI ANALIZ I ISSLEDOVANIE OPERATSII
/DISCRETE ANALYSIS AND OPERATIONS RESEARCH/
April–June 2026. Vol. 33, No. 2. P. 46–63

UDC 519.852+519.856+519.6 DOI: 10.33048/daio.2026.33.848

BRANCH-AND-CUT METHOD FOR THE WEIGHTED 3-COLORING PROBLEM

G. A. Zhukov[1, a], *A. I. Erzin*[1, 2, b], and *R. V. Plotnikov*[3, c]

[1] Novosibirsk State University,
2 Pirogov Street, 630090 Novosibirsk, Russia
[2] Sobolev Institute of Mathematics,
4 Acad. Koptyug Avenue, 630090 Novosibirsk, Russia
[3] Ledas Ltd.,
11/5 Nikolaev Street, 630090 Novosibirsk, Russia
E-mail: [a] g.zhukov@g.nsu.ru,
[b] adilerzin@math.nsc.ru, [c] rvplotnikov@gmail.com

Abstract. This study addresses the NP-hard problem of coloring the vertices of an edge-weighted undirected graph with 3 colors, aiming to minimize the total weight of edges connecting vertices of the same color. This problem is mathematically equivalent to the well-known Maximum k-Cut problem, where the objective is to maximize the weight of edges between vertices from different subsets. In this work, we present an exact algorithm for the case $k = 3$, implemented within the branch-and-cut framework. While most existing research focuses on classes of valid inequalities, the key contribution of this paper is a method for reducing the depth of the branching tree based on vertex decomposition integrated into the branch-and-cut scheme. Computational experiments demonstrate the high efficiency of the proposed algorithm. Tab. 4, illustr. 1, bibliogr. 28.

Keywords: 3-coloring, branch-and-cut, maximum k-cut.

References

1. **R. M. Karp,** Reducibility among combinatorial problems, in *Complexity of Computer Computations*, Proc. Symp. (New York, USA, March 20–22, 1972) (Plenum Press, New York, 1972), pp. 85–103, DOI: 10.1007/978-1-4684-2001-2_9.

English transl.: *Journal of Applied and Industrial Mathematics* **20** (2) (2026).

2. **O. V. Borodin,** Colorings of plane graphs: A survey, *Discrete Math.* **313** (4), 517–539 (2013), DOI: 10.1016/j.disc.2012.11.011.
3. **T. Husfeldt,** Graph colouring algorithms, in *Topics in Chromatic Graph Theory* (Cambridge Univ. Press, Cambridge, 2015), pp. 277–303 (Encycl. Math. Appl.), DOI: 10.1017/CBO9781139519793.016.
4. **A. V. Kostochka** and **M. Yancey,** On coloring of sparse graphs, in *Computer Science — Theory and Applications*, Proc. 8th Int. Comput. Sci. Symp. in Russia (Yekaterinburg, Russia, June 25–29, 2013) (Springer, Heidelberg, 2013), pp. 224–234 (Lect. Notes Comput. Sci., Vol. 7913), DOI: 10.1007/978-3-642-38536-0_20.
5. **D. Brélaz,** New methods to color the vertices of a graph, *Commun. ACM* **22** (4), 251–256 (1979), DOI: 10.1145/359094.359101.
6. **A. Mehrotra** and **M. A. Trick,** A column generation approach for graph coloring, *INFORMS J. Comput.* **8** (4), 344–354 (1996), DOI: 10.1287/ijoc.8.4.344.
7. **P. Galinier** and **J.-K. Hao,** Hybrid evolutionary algorithms for graph coloring, *J. Comb. Optim.* **3**, 379–397 (1999), DOI: 10.1023/A:1009823419804.
8. **L. Moalic** and **A. Gondran,** Variations on memetic algorithms for graph coloring problems, *J. Heuristics* **24** (4), 1–24 (2018), DOI: 10.1007/s10732-017-9354-9.
9. **D. W. Cranston** and **L. W. Rabern,** Coloring a graph with $\Delta - 1$ colors: Conjectures equivalent to the Borodin–Kostochka conjecture that appear weaker, *Eur. J. Comb.* **44**, 23–42 (2015), DOI: 10.1016/j.ejc.2014.09.006.
10. **I. Choi, H. A. Kierstead,** and **L. W. Rabern,** The list version of the Borodin–Kostochka conjecture for graphs with large maximum degree, *Discrete Math.* **346** (11), ID 113300 (2023), DOI: 10.1016/j.disc.2022.113300.
11. **Y. Maus,** Distributed graph coloring made easy (Ithaca, NY, 2021) (e-Print Archive / Cornell Univ., arXiv:2105.05575), DOI: 10.48550/arXiv.2105.05575.
12. **D. P. Dailey,** Uniqueness of colorability and colorability of planar 4-regular graphs are NP-complete, *Discrete Math.* **30** (3), 289–293 (1980), DOI: 10.1016/0012-365X(80)90236-8.
13. **L. Meijer,** 3-Coloring in time $O(1.3217^n)$ (Ithaca, NY, 2023) (e-Print Archive / Cornell Univ., arXiv:2302.13644), DOI: 10.48550/arXiv.2302.13644.
14. **Z. Dvořák, K.-I. Kawarabayashi,** and **R. Thomas,** Three-coloring triangle-free planar graphs in linear time, *ACM Trans. Algorithms* **7** (4), ID 41 (2011), DOI: 10.1145/2000807.2000809.
15. **M. Goemans** and **D. P. Williamson,** Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming, *J. ACM* **42** (6), 1115–1145 (1995), DOI: 10.1145/227683.227684.
16. **A. Frieze** and **M. Jerrum,** Improved approximation algorithms for MAX-k-CUT and MAX BISECTION, *Algorithmica* **18**, 67–81 (1997), DOI: 10.1007/BF02523688.

17. E. De Klerk, D. V. Pasechnik, and **J. P. Warners,** On approximate graph colouring and MAX-k-CUT algorithms based on the ϑ-function, *J. Comb. Optim.* **8** (3), 267–294 (2004), DOI: 10.1023/B:JOCO.0000038911.67280.3f.

18. F. Margot, Pruning by isomorphism in Branch-and-Cut, *Math. Program., Ser. A*, **94**, 71–90 (2002), DOI: 10.1007/s10107-002-0358-2.

19. S. Chopra and **M. Rao,** The partition problem, *Math. Program.* **59** (1), 87–115 (1993), DOI: 10.1007/BF01581239.

20. A. Eisenblätter, The semidefinite relaxation of the k-partition polytope is strong, in *Integer Programming and Combinatorial Optimization*, Proc. 9th Int. IPCO Conf. (Cambridge, USA, May 27–29, 2002) (Springer, Heidelberg, 2002), pp. 273–290 (Lect. Notes Comput. Sci., Vol. 2337), DOI: 10.1007/3-540-47867-1_20.

21. R. de Sousa, Global optimization of the maximum K-cut problem, *PhD Thesis* (Montreal, 2018).

22. B. Ghaddar, M. F. Anjos, and **F. Liers,** A branch-and-cut algorithm based on semidefinite programming for the minimum k-partition problem, *Ann. Oper. Res.* **188** (1), 155–174 (2011), DOI: 10.1007/s10479-008-0481-4.

23. Z. Ales and **A. Knippel,** The K-partitioning problem: Formulations and branch-and-cut, *Networks* **76** (3), 323–349 (2020), DOI: 10.1002/net.21944.

24. P. Healy, N. Jozefowiez, P. Laroche, F. Marchetti, S. Martin, and **Z. Róka,** A branch-and-cut algorithm for the connected max-k-cut problem, *Eur. J. Oper. Res.* **312** (1), 117–124 (2024), DOI: 10.1016/j.ejor.2023.06.015.

25. M. Benichou, J. M. Gauthier, P. Girodet, G. Hentges, G. Ribiere, and **O. Vincent,** Experiments in mixed-integer linear programming, *Math. Program.* **1**, 76–94 (1971), DOI: 10.1007/BF01584074.

26. N. A. Kovrizhnykh, 3-Coloring: Graph generator for MOTOR 2025 Challenge 5 (2025), URL: github.com/sedefe/3-coloring (accessed: 6.08.2026).

27. MOTOR 2025 Challenges (IM SO RAN, Novosibirsk, 2025), URL: old.math.nsc.ru/conference/motor/2025/challenges.html (accessed: 6.08.2026).

28. Dataset of generated and unit-disk graphs (2025), URL: https://disk.yandex.ru/d/cbMBI1Rp0J535Q (accessed: 6.08.2026).

Georgy A. Zhukov
Adil I. Erzin
Roman V. Plotnikov

Received December 17, 2025
Revised February 15, 2026
Accepted March 23, 2026

ДИСКРЕТНЫЙ АНАЛИЗ И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ
Апрель–июнь 2026. Т. 33, № 2. С. 64–81

УДК 519.176+519.8+519.7 DOI: 10.33048/daio.2026.33.859

ПОИСК СЕМЕЙСТВ СУБОПТИМАЛЬНЫХ ЦИРКУЛЯНТНЫХ СЕТЕЙ С ПОМОЩЬЮ БОЛЬШИХ ЯЗЫКОВЫХ МОДЕЛЕЙ

Э. А. Монахова[1, 2, a], *О. Г. Монахов*[2, b]

[1] Национальный исследовательский университет «Высшая школа экономики», ул. Мясницкая, 20, 101000 Москва, Россия

[2] Институт вычислительной математики и математической геофизики, пр. Акад. Лаврентьева, 6, 630090 Новосибирск, Россия

E-mail: [a] emilia@rav.sscc.ru, [b] monakhov@rav.sscc.ru

Аннотация. На основе анализа разработанного авторами датасета двумерных кольцевых циркулянтных сетей решается задача поиска семейств субоптимальных по диаметру циркулянтных сетей. Субоптимальные двумерные кольцевые циркулянты — это графы с минимально возможным диаметром для данного числа вершин, превышающим на единицу точную нижнюю границу диаметра циркулянтов. Представлена новая оригинальная визуализация на плоскости точек датасета, соответствующих субоптимальным циркулянтам. С помощью обнаруженной визуальной регулярности открыты последовательности аналитических описаний субоптимальных циркулянтов. Применение больших языковых моделей в качестве ИИ-ассистента при разработке программ позволило найти новые семейства субоптимальных графов, составляющие в общей сложности 75% субоптимальных точек датасета. Теоретически доказана масштабируемость найденных семейств графов при росте их диаметров, при доказательствах теорем использовалась большая языковая модель. Найденные аналитически задаваемые семейства циркулянтов представляют интерес как топологии сетей на кристалле и коммуникационных систем. Табл. 3, ил. 3, библиогр. 16.

Ключевые слова: кольцевая циркулянтная сеть, диаметр, датасет оптимальных циркулянтов, сеть на кристалле, большая языковая модель.

Введение

Циркулянтные графы степени четыре изучаются в теории графов и прикладных областях, включая их использование в виде топологий

© Э. А. Монахова, О. Г. Монахов, 2026

сетей связи для сетей на кристалле [1–7]. Циркулянтная сеть степени четыре представляет собой неориентированный граф $C(N; s_1, s_2)$, $1 \leqslant s_1 < s_2 < N/2$, в котором каждая вершина $i \in V = \{0, 1, \ldots, N-1\}$ смежна вершинам $(i \pm s_1) \bmod N$ и $(i \pm s_2) \bmod N$. Числа s_1, s_2 — образующие, а N — порядок графа. Граф $C(N; s_1, s_2)$ связен, если НОД$(N, s_1, s_2) = 1$. При $s_1 = 1$ граф $C(N; 1, s)$ называется двумерным (по числу образующих) кольцевым циркулянтным графом. В данной работе исследуется класс циркулянтных графов вида $C(N; 1, s)$. На рис. 1 изображён граф $C(10; 1, 4)$.

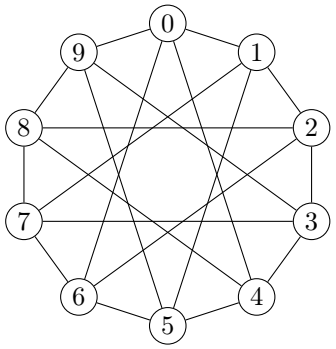


Рис. 1. Циркулянтный граф $C(10; 1, 4)$

Диаметр циркулянта $d = d(N; s_1, s_2)$ — длина максимального кратчайшего пути на множестве всевозможных пар вершин. Оптимальным называется граф $C(N; 1, s)$ с минимально возможным для данного N диаметром $d(N) = \min\limits_{s}\{d(N; 1, s)\}$. Известна точная нижняя граница диаметра двумерных циркулянтов $C(N; s_1, s_2)$ для любого $N > 4$ [3]:

$$d(N) \geqslant D(N) = \lceil(-1 + \sqrt{2N-1})/2\rceil.$$

Отметим, что диаметр как показатель топологии сети связи зависит от числа вершин графа, находящихся на определённом расстоянии (уровне одного расстояния) от выделенной вершины — например, вершины 0 в силу вершинной симметрии циркулянта. Пусть n_i означает число вершин, находящихся на уровне i от 0, $i = 1, \ldots, d$, где d — диаметр графа. Для двумерных циркулянтов максимально возможное число вершин на i-м уровне равно $4i$. Семейство оптимальных двумерных кольцевых циркулянтов $\{C(2d^2 + 2d + 1; 1, 2d + 1) \mid d \geqslant 1\}$ с полностью заполненными уровнями расстояний, включая диаметр, существует при любом диаметре [3].

В классе графов $C(N; 1, s)$ точная нижняя граница диаметра $D(N)$ достижима не для всех N. По гипотезе Д. Цвиели [8], подтверждённой

для всех $N \leqslant 8 \cdot 10^6$, минимально возможный диаметр графа $C(N; 1, s)$ для любого N не превышает $D(N)+1$. Графы с $d(N) = D(N)+1$ назовём субоптимальными, т. е. субоптимальны графы с минимально возможным диаметром для данного числа вершин, превышающим на единицу известную точную нижнюю границу диаметра циркулянтов. Субоптимальными назовём также семейства графов, все члены которых субоптимальны, и соответствующие им значения N.

В данной работе исследуется существование семейств субоптимальных сетей класса $C(N; 1, s)$. Авторами получены и представлены в открытом доступе база данных (датасет) описаний графов минимально возможного диаметра до $N \leqslant 50\,000$ вершин [9] и аналитически описываемые семейства оптимальных графов с $d(N) = D(N)$, найденные на основе анализа датасета [10]. Первое бесконечное семейство субоптимальных графов $C(N; 1, s)$ представлено в пионерской работе [11]. Это семейство содержит графы порядка $N = 2d^2+2d$ с образующей $s = 2d+1$ для любого неограниченно растущего диаметра $d > 1$. Также несколько подобных семейств получены в работах [8, 12–14].

В разд. 1 представлена оригинальная визуализация точек датасета, соответствующих параметрам субоптимальных циркулянтов. В разд. 2 приведён алгоритм поиска описаний семейств субоптимальных циркулянтов, основанный на темплейтах, отражающих закономерности полученной визуализации. Описано использование больших языковых моделей для разработки программ поиска и анализа. Приведены описания полученных семейств субоптимальных графов $C(N; 1, s)$ с порядками и образующими в виде полиномов от диаметра. В разд. 3 теоретически доказывается масштабируемость (бесконечность по диаметру) новых семейств субоптимальных графов.

1. Визуализация точек датасета субоптимальных циркулянтов

Датасет [9] циркулянтов $C(N; 1, s)$ с $10 \leqslant N \leqslant 50\,000$ и $2 \leqslant d \leqslant 156$ включает весь набор образующих оптимальных (субоптимальных) графов, что позволило найти семейства оптимальных циркулянтных графов с параметрами, заданными полиномами от диаметра (см. [10]). В настоящей работе продолжен поиск и исследование наилучших графов класса $C(N; 1, s)$. Для этого из общего датасета точек, соответствующих оптимальным и субоптимальным графам, выделены все точки (N, d), далее обозначаемые как датасет Subopt, относящиеся к субоптимальным описаниям. Здесь $d = D(N) + 1$ — минимально возможный для данного N диаметр, полученный путём перебора всех образующих $2 \leqslant s < N/2$.

Затем разработана новая графическая визуализация точек датасета Subopt. Предложенный способ отображения точек (N, d) через параметр

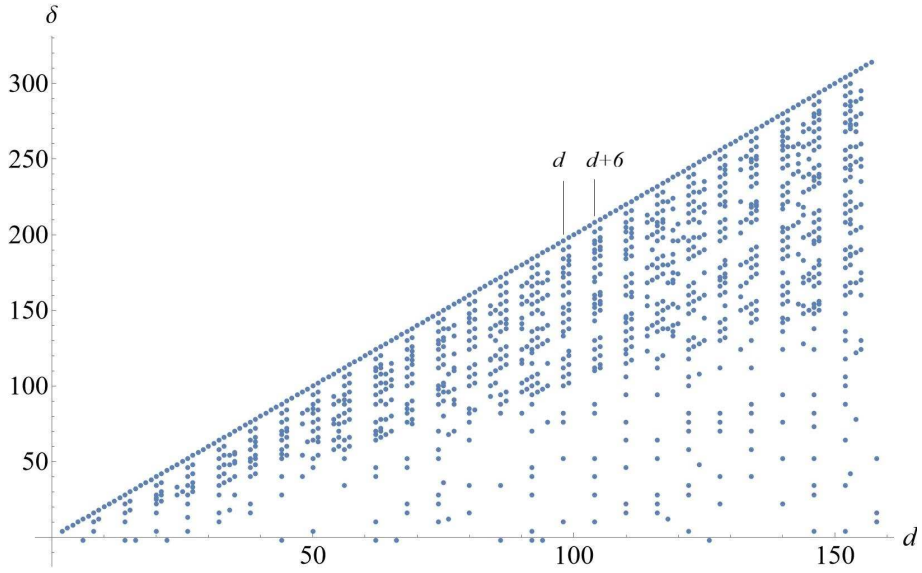


Рис. 2. Расположение порядков субоптимальных циркулянтов $C(N;1,s)$ в зависимости от диаметра для датасета Subopt

$\delta = N - 2d^2$ позволил выявить регулярности, которые неочевидны при стандартных представлениях. На рис. 2 показан фрагмент графической визуализации точек датасета Subopt, отражающих параметры субоптимальных описаний (N, d). Ось абсцисс соответствует диаметру $d = D(N)$, ось ординат — значениям, равным разнице $\delta = N - 2d^2$, $-2 \leqslant \delta \leqslant 2d + 1$, при заданном d, $2 \leqslant d \leqslant 156$. Таким образом, верхняя линия графика соответствует субоптимальным точкам с $N = 2d^2 + 2d$, а на оси абсцисс могли находиться значения $N = 2d^2$, если бы среди них присутствовали субоптимальные значения.

Как видно из представленного графика, точки датасета, соответствующие субоптимальным описаниям, все расположены выше линии $N = 2d^2 - 3$. Это можно объяснить тем, что при $N < 2d^2 - 2$ число вершин оптимального графа на диаметре d меньше $2d - 2$ и есть широкий выбор значений образующих для построения оптимальных графов. Тем самым анализ датасета циркулянтов класса $C(N;1,s)$, $10 \leqslant N \leqslant 50\,000$, позволяет сформулировать следующую гипотезу.

Пусть $d > 2$ — целое число. Тогда для любого графа $C(N;1,s)$ с $2d^2 - 2d + 1 < N < 2d^2 - 2$ существует такое $1 < s < N/2$, что $d(N) = D(N)$.

При приближении порядка графа к числу $N = 2d^2 + 2d$, когда уровень вершин, находящихся на диаметре d, уже близок к полному заполнению (значению $4d$), свобода в выборе оптимальных образующих

уменьшается и растёт число субоптимальных N. При $\delta > d$, что соответствует значениям $N > 2d^2 + d$, на графике можно увидеть регулярность в появлении субоптимальных точек N как по диаметру с периодом, равным шести, так и по вертикальному смещению двух соседних точек N, равному шести в большинстве случаев. Регулярность при появлении субоптимальных точек для N будем использовать далее при формировании вида темплейтов для порядков и образующих при поиске семейств субоптимальных циркулянтов.

2. Поиск семейств субоптимальных сетей

На первом этапе поиска семейств субоптимальных сетей на множестве графов датасета Subopt авторами разработан алгоритм автоматизированного поиска аналитически описываемых семейств с заданными темплейтами $N(d)$ и $s(d)$.

В качестве темплейтов для порядков графов семейств и их образующих рассмотрены следующие полиномы:

$$N = 2d^2 + 2d - B, \quad 0 \leqslant B < d,$$
$$s \in \{d^2 + ad + b,\ 2d + c\}, \quad -d < a, b, c < d,\ d = D(N).$$

Искомое семейство $\Psi_i(d)$ субоптимальных графов может существовать, начиная с некоторого начального диаметра $d = d_m$ в соответствующем диапазоне изменения диаметра d с периодом $1 \leqslant p \leqslant B$ существования членов семейства, как правило, равном делителю B:

$$\Psi_i(d) = \{C(N(d);\ 1, s(d)) \mid d = d_m + kp,\ k \geqslant 0\}.$$

Все точки датасета, начиная с некоторого значения $d_m > B$, тестировались на наличие общего вида образующих при заданных N отдельно, последовательно по возрастанию, по всем вычетам диаметров $0, 1, \ldots, p-1$. При этом осуществлялся полный перебор коэффициентов из заданных диапазонов их значений. Если в датасете существуют точки, соответствующие полученным при этом значениям порядков и образующих, то принимается гипотеза, что такое семейство с полученными коэффициентами и значениями порядка и образующих существует при заданных диапазонах диаметров. Количество найденных точек на этом этапе могло меняться от 4 и более.

Для валидации найденных семейств осуществлялся поиск точек датасета, соответствующих найденному описанию семейства при больших значениях диаметра, но ограниченного рамками датасета. Если после этого этапа подтверждалось существование семейства для всех проверенных точек, то решение о существовании данного семейства считалось положительным.

Таблица 1

Семейства субоптимальных циркулянтов $C(2d^2+2d-B;1,s)$ с чётным диаметром

$N(d)$	$s(d)$	d	p	d_m
$2d^2+2d$	$2d-1$, $\triangleright 2d+1$ $2d+3$ $4d\pm 1$	$d \geqslant d_m$	2	2
$2d^2+2d-6$	$d^2-2d-10$ d^2-2 d^2-6 $\triangleright 2d-3$, $2d+3$ $2d\pm 5$ $4d\pm 9$	$6k+2$	6	8
$\triangleright 2d^2+2d-12$	d^2-9	$6k+2$	6	14
$\triangleright 2d^2+2d-14$ $2d^2+2d-14$	d^2-10 $4d+9$	$14k+6$	14	20
$2d^2+2d-15$ $\triangleright 2d^2+2d-15$	$2d+5$ $2d-7$	$15k+5$, $15k+9$, k нечётное	15	20, 24
$2d^2+2d-18$	$d^2-2d-20$ d^2-6 $2d\pm 5$	$6k+2$	6	20
	$2d\pm 7$		18	
$2d^2+2d-20$	d^2-7	$20k+4$, $20k+14$	20	24, 34
$2d^2+2d-24$	d^2-9	$6k+2$	6	26
$2d^2+2d-26$	d^2-10	$26k+12$	26	38
$2d^2+2d-30$	$2d\pm 7$ $2d\pm 9$	$6k+2$, $30k+4$, $30k+24$	6 30	32, 34, 54

Для реализации описанного выше алгоритма полного перебора по заданным темплейтам на языке Python в качестве ИИ-ассистента использована большая языковая модель Gemini. Полученная с помощью неё программа проверена на корректность, доработана для различных диапазонов коэффициентов и запущена на компьютере с процессором AMD Ryzen 5 5500U и памятью 16 ГБ. Параметр B при поиске субоптимальных семейств изменялся в диапазоне $0 \leqslant B \leqslant 30$. Все семейства субоптимальных циркулянтов, найденные программой, приведены в табл. 1–3. Семейства, найденные в [11, 14], отмечены в таблицах знаком $\triangleright$.

Отметим следующие результаты поиска субоптимальных семейств.

• Для тех значений B, $0 \leqslant B \leqslant 30$, которые отсутствуют в табл. 1 и 2, реализованная программа поиска не нашла образующих по заданным темплейтам.

Таблица 2

Семейства субоптимальных циркулянтов $C(2d^2+2d-B;1,s)$ с нечётным диаметром

$N(d)$	$s(d)$	d	p	d_m
$2d^2+2d$	d^2 $2d-1$, $\triangleright 2d+1$ $2d+3$ $4d+3$ $4d+5$	$d \geqslant d_m$	2	3
$2d^2+2d-6$	d^2-2d+1 d^2-1 d^2-5 $\triangleright 2d-3$, $2d\pm 5$, $2d+3$	$6k+3$	6	9
$\triangleright 2d^2+2d-12$	d^2-4	$6k+3$	12	15
$\triangleright 2d^2+2d-14$	d^2-5	$14k+7$	14	21
$2d^2+2d-15$ $\triangleright 2d^2+2d-15$	$2d+5$ $2d-7$	$15k+5$, $15k+9$, k чётное	15	20, 24
$2d^2+2d-18$	d^2-13 $2d\pm 5$ $2d\pm 7$	$6k+3$	6	21
$2d^2+2d-20$	d^2-14	$20k+5$ $20k+15$	20	25 35
$2d^2+2d-24$	d^2-16	$6k+3$	6	27
$2d^2+2d-26$	d^2-17	$26k+13$	26	39
$2d^2+2d-30$	$2d\pm 7$ $2d\pm 9$	$6k+3$, $6k+5$	6	33, 35

• При поиске субоптимальных семейств из табл. 3 использована регулярность появления значений N со значениями B, кратными 6: $B=6m$, где $m=6$, 8, 9, 10, 13, 14, 16, 19, 21, для диаметров $d=6k+2$ и $d=6k+3$.

• Большая часть найденных в табл. 1–3 образующих графов субоптимальных семейств относится к виду $s=d^2-c$ или $s=2d\pm c$.

• При проведённом переборе на множестве образующих других видов найдено всего восемь семейств.

Все семейства субоптимальных циркулянтов, кроме отмеченных знаком $\triangleright$, являются новыми семействами. Все семейства из табл. 1–3 объединим в общее множество Finefams найденных семейств. На рис. 3 показано пересечение порядков найденных семейств из множества Finefams с точками датасета Subopt. Точки (N,d) найденных семейств обозначены жёлтым цветом. Точки найденных семейств составляют 75% от общего числа точек датасета Subopt. Ограничения при поиске субоптимальных

Таблица 3

Семейства субоптимальных циркулянтов $C(2d^2+2d-B;1,s)$, $B \equiv 0 \pmod 6$, $p=6$

$N(d)$	$s(d)$	d	d_m	$N(d)$	$s(d)$	d	d_m
$2d^2+2d-36$	d^2-23	$6k+2$	38	$2d^2+2d-36$	d^2-14	$6k+3$	39
$2d^2+2d-48$	$2d-9$ $2d+11$		50	$2d^2+2d-48$	$2d-9$ $2d+11$		51
$2d^2+2d-54$	d^2-22		56	$2d^2+2d-54$	d^2-33		57
$2d^2+2d-60$	d^2-25		62	$2d^2+2d-60$	d^2-36		63
$2d^2+2d-78$	d^2-46		80	$2d^2+2d-78$	d^2-33		81
$2d^2+2d-84$	d^2-49		86	$2d^2+2d-84$	d^2-36		87
$2d^2+2d-96$	$2d-13$ $2d+15$		98	$2d^2+2d-96$	$2d-13$ $2d+15$		99
$2d^2+2d-114$	d^2-50		116	$2d^2+2d-114$	d^2-65		117
$2d^2+2d-126$	$2d-15$ $2d+17$		128	$2d^2+2d-126$	$2d-15$ $2d+17$		129

семейств связаны с ограничением на диаметр ($d \leqslant 156$) графов датасета Subopt, поскольку при больших периодах существования ($p > 30$) членов семейств уменьшается число точек, доступных для тестирования при поиске. Кроме того, начальный диаметр d_m, с которого начинается каждое новое семейство, также может быть достаточно большим.

Авторами проверено существование найденных семейств при диаметрах больших, чем диаметры графов датасета Subopt. Для этапа валидации использовалась программа анализа структурных характеристик циркулянтных графов [15], которая по значениям порядка и образующих циркулянта выдаёт диаметр, среднее расстояние и дополнительно векторы распределения числа вершин графов по уровням от 1 до диаметра включительно. С её помощью проверено существование субоптимальных семейств из табл. 1–3 для всех допустимых значений диаметров до числа вершин $N = 10^6$. Полученное подтверждение их существования при таких больших числах говорит в пользу гипотезы, что найденные семейства бесконечны. В разд. 3 это свойство доказывается теоретически на примере ряда семейств.

3. Доказательства существования семейств при неограниченном росте диаметра графов

Для доказательства существования бесконечных семейств субоптимальных циркулянтов $C(N;1,s)$ будем использовать леммы 1 и 2, полученные в [14].

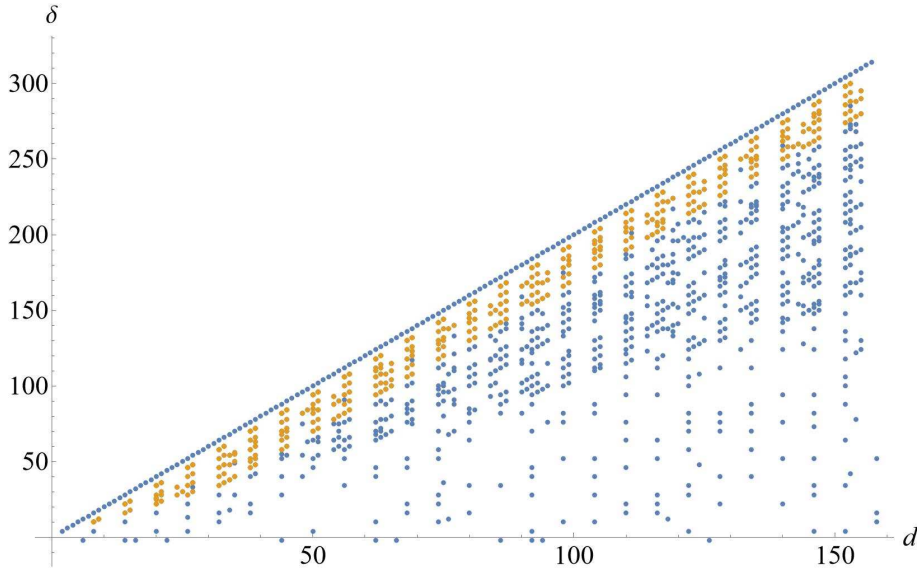


Рис. 3. Семейства субоптимальных циркулянтов $C(N;1,s)$ из табл. 1–3 (множество Finefams)

Лемма 1 [14, следствие 3.2]. *Пусть* $N = 2d^2 + 2d - B$, *где* $B \in \overline{1, d-1}$ *и* $2B+1$ *простое. Тогда оптимальный граф* $C(N;1,s)$ *существует, если и только если* НОД$(B,d) = 1$ *или* НОД$(B,d+1) = 1$.

Предварительные пояснения к лемме 2. На двумерной целочисленной решётке $\mathbb{Z}^2$ зафиксирована точка отсчёта $(0,0) \in \mathbb{Z}^2$. Для произвольной точки $(x,y) \in \mathbb{Z}^2$ определена функция отображения, порождённая укладкой циркулянтного графа $C(N;1,s)$ на плоскости $\mathbb{Z}^2$:

$$l(x,y) = (x + ys) \bmod N \in \mathbb{Z}_N.$$

Рассматривается множество всех нулей на плоскости, порождённых описанием $C(N;1,s)$: $X = \{(x,y) \in \mathbb{Z}^2 \mid l(x,y) = 0\}$. На множестве X (нулей) выбираются две нулевые точки укладки: нуль (u,v) в первом квадранте имеет минимум суммы $u+v$, нуль $(-a,b)$ во втором квадранте имеет минимум суммы $a+b$ (в случае равенства сумм выбирается максимум u и максимум b).

Лемма 2 [14, лемма 2.5]. *Пусть* $C(N;1,s)$ *— циркулянтный граф, где* $1 < s < N$, $s \neq N/2$, *а* $u,v,a,b \in \mathbb{Z}$ *— неотрицательные числа такие, что* (u,v) *и* $(-a,b)$ *— нулевые точки укладки графа* $C(N;1,s)$ *на плоскости,*

$$r_1 = \lfloor (u+v)/2 \rfloor, \quad r_2 = \lfloor (a+b)/2 \rfloor,$$
$$r_3 = \lfloor (|u-a| + v + b)/2 \rfloor, \quad r_4 = \lfloor (u + a + |v-b|)/2 \rfloor.$$

Если $N = av + bu$ *и* $b \geqslant a$, $u \geqslant v$, $u > a$, $v < b$ (*или* $b < a$, $u < v$, $u < a$, $v > b$), *то диаметр графа* $C(N; 1, s)$ *равен*

$$d(N;1,s) = \begin{cases} r_3 - 1, & \text{если } (u+a)(v+b) \\ & \quad \text{нечётно и } r_3 = r_4, \\ \max\{r_1, r_2, & \\ \quad \min\{r_3, r_4\}\} & \text{в остальных случаях.} \end{cases} \tag{1}$$

Ниже приводятся доказательства существования новых бесконечных семейств субоптимальных циркулянтов.

Теорема 1. *Если* $N = 2d^2 + 2d - 20$, *то*

$$\Psi_1 = \{C(N;1,s) \mid d = 20k + 4,\ 20k + 14,\ k \geqslant 1\},$$
$$\Psi_2 = \{C(N;1,s) \mid d = 20k + 5,\ 20k + 15,\ k \geqslant 1\}$$

— бесконечные субоптимальные семейства циркулянтов с образующими $s = d^2 - 7$, $d^2 - 14$ *соответственно.*

Доказательство. При $B = 20$ число $2B+1$ простое. В силу леммы 1 граф $C(N; 1, s)$ с числом вершин $N = 2d^2 + 2d - 20$ не может быть оптимальным. Действительно, при всех указанных диаметрах $\text{НОД}(20, d) \neq 1$ и $\text{НОД}(20, d + 1) \neq 1$. Остаётся показать, что соответствующие образующие формируют субоптимальный граф.

1. Для семейства Ψ_1 при $d = 20k+4$ и $d = 20k+14$ образующая равна $s = d^2 - 7$. В качестве координат двух нулей $(-a, b)$, (u, v), применимых как для $d = 20k + 4$, так и для $d = 20k + 14$, возьмём $a = d - 1$, $b = d + 3$, $u = 2d - 6$, $v = 2$. Подстановкой значений убедимся, что

$$av + bu = 2(d - 1) + (2d - 6)(d + 3) = 2d^2 + 2d - 20 = N.$$

Проверим, что указанные параметры задают нулевые точки при заданных d. Имеем

$$-a + bs = -d + 1 + (d + 3)(d^2 - 7) =$$
$$= d^3 + 3d^2 - 8d - 20 = (d + 2)N/2 \equiv 0 \pmod N,$$
$$u + vs = (2d - 6) + 2(d^2 - 7) = 2d^2 + 2d - 20 = N \equiv 0 \pmod N.$$

Согласно лемме 2 получаем $r_1 = d - 2$, $r_2 = d + 1$, $r_3 = d$, $r_4 = 2d - 3$ и $d(N; 1, s) = d + 1$. Таким образом, граф $C(N; 1, s)$ субоптимальный.

2. Для семейства Ψ_2 при $d = 20k + 5$, $d = 20k + 15$ образующая равна $s = d^2 - 14$. В качестве координат двух нулей $(-a, b)$, (u, v) возьмём $a = 4$, $b = 2d - 6$, $u = d + 2$, $v = d - 2$. Подстановкой проверим, что

$$av + bu = 4(d - 2) + (2d - 6)(d + 2) = 2d^2 + 2d - 20 = N.$$

Покажем, что указанные параметры задают нулевые точки при заданных d. Имеем

$$-a+bs=-4+(2d-6)(d^2-14)=$$
$$=2d^3-6d^2-28d+80=(d-4)N\equiv 0 \pmod N,$$
$$u+vs=d+2+(d-2)(d^2-14)=$$
$$=d^3-2d^2-13d+30=(d-3)N/2\equiv 0 \pmod N.$$

Согласно лемме 2 получаем $r_1=d$, $r_2=d-1$, $r_3=2d-5$, $r_4=d+1$ и $d(N;1,s)=d+1$, т. е. граф $C(N;1,s)$ субоптимальный. Теорема 1 доказана.

Также для тех семейств из табл. 1–3, для которых число $2B+1$ простое, а это семейства графов с $B=6$, 14, 15, 18, 20, 26, 30, 36, 48, 54, 78, 96, 114, можно искать значения параметров a,b,u,v, а затем применить, по мнению авторов, аналогичный метод для доказательства их бесконечности. Остаются значения $B=12,24,60,84,126$. Для $B=12$ доказательство бесконечности соответствующего семейства из табл. 2 и 3 получено в [14]. Для оставшихся B доказательства приводим ниже.

Теорема 2. *Если* $N=2d^2+2d-24$, *то*

$$\Psi_3=\{C(N;1,s)\mid d=6k+2,\ 6k+3,\ k\geqslant 4\}$$

— бесконечное субоптимальное семейство циркулянтов с образующей

$$s=\begin{cases} d^2-9 & \text{при } d=6k+2,\\ d^2-16 & \text{при } d=6k+3.\end{cases}$$

Доказательство. В указанных условиях $2B+1=49$ — составное число.

Пусть $d=6k+2$, $s=d^2-9$. Возьмём $a=d-3$, $b=d+3$, $u=2d-6$, $v=2$. Имеем

$$av+bu=(d-3)2+(d+3)(2d-6)=2d^2+2d-24=N.$$

Проверим, что $(-a,b)$, (u,v) — координаты двух нулей. Действительно, при $d=6k+2$ имеем

$$-a+bs=(d+3)(d^2-9)-(d-3)=$$
$$d^3+3d^2-10d-24=(d+2)N/2\equiv 0 \pmod N,$$
$$u+vs=2(d^2-9)+2d-6=2d^2+2d-24=N\equiv 0 \pmod N.$$

Согласно лемме 2 получаем $r_1=d-2$, $r_2=d$, $r_3=d+1$, $r_4=2d-4$ и $d(N;1,s)=d+1$, т. е. граф $C(N;1,s)$ субоптимальный.

При $d = 6k + 3$ возьмём $a = 0$, $b = 2d - 6$, $u = d + 4$, $v = d - 2$. Тогда

$$av + bu = (2d - 6)(d + 4) = 2d^2 + 2d - 24 = N.$$

Имеем

$$-a + bs = (2d - 6)(d^2 - 16) = \\ = 2(d - 3)(d + 4)(d - 4) = (d - 4)N \equiv 0 \pmod N,$$
$$u + vs = (d - 2)(d^2 - 16) + (d + 4) = (d - 3)N/2 \equiv 0 \pmod N.$$

Таким образом, $(-a, b)$ и (u, v) — координаты двух нулей.

По лемме 2 диаметр $d(N; 1, s) = \max\{d+1, d-3, \min\{2d-2, d\}\} = d+1$, т. е. граф $C(N; 1, s)$ субоптимальный. Теорема 2 доказана.

Теорема 3. *Если* $N = 2d^2 + 2d - 60$, *то*

$$\Psi_4 = \{C(N; 1, s) \mid d = 6k + 2,\ 6k + 3,\ k \geqslant 10\}$$

— бесконечное субоптимальное семейство циркулянтов с образующей

$$s = \begin{cases} d^2 - 25 & \text{при } d = 6k + 2, \\ d^2 - 36 & \text{при } d = 6k + 3. \end{cases}$$

Доказательство. В качестве значений координат двух нулей $(-a, b)$ и (u, v) возьмём $a = d - 5$, $b = d + 5$, $u = 2d - 10$, $v = 2$ при $d = 6k + 2$ или $a = 0$, $b = 2d - 10$, $u = d + 6$, $v = d - 4$ при $d = 6k + 3$.

Действительно, при $d = 6k + 2$

$$-a + bs = (d + 5)(d^2 - 25) - (d - 5) = (d + 4)N/2 \equiv 0 \pmod N,$$
$$u + vs = 2d - 10 + 2(d^2 - 25) = 2d^2 + 2d - 60 = N \equiv 0 \pmod N.$$

При $d = 6k + 3$

$$-a + bs = (2d - 10)(d^2 - 36) = (d - 6)N \equiv 0 \pmod N,$$
$$u + vs = (d + 6) + (d - 4)(d^2 - 36) = (d - 5)N/2 \equiv 0 \pmod N.$$

Проверяем, что при заданных диаметрах $av + bu = N$. Согласно лемме 2 в первом случае диаметр графа равен

$$d(N; 1, s) = \max\{d - 4, d, \min\{d + 1, 2d - 6\}\} = d + 1,$$

а во втором —

$$d(N; 1, s) = \max\{d + 1, d - 5, \min\{2d - 4, d\}\} = d + 1.$$

Таким образом, в обоих случаях имеем субоптимальные графы. Теорема 3 доказана.

Теорема 4. *Если* $N = 2d^2 + 2d - 84$, *то*

$$\Psi_5 = \{C(N; 1, s) \mid d = 6k + 2,\ 6k + 3,\ k \geqslant 14\}$$

— бесконечное субоптимальное семейство циркулянтов с образующей $s = d^2 - 49$ *при* $d = 6k + 2$ *и* $s = d^2 - 36$ *при* $d = 6k + 3$.

Доказательство. Пусть $d = 6k + 2$, $s = d^2 - 49$. Возьмём $a = 0$, $b = 2d - 12$, $u = d + 7$, $v = d - 5$. Имеем

$$av + bu = (2d - 12)(d + 7) = 2d^2 + 2d - 84 = N.$$

Проверим, что $(-a, b)$, (u, v) — координаты двух нулей:

$$-a + bs = (2d - 12)(d^2 - 49) = (d - 7)N \equiv 0 \pmod{N},$$

$$u + vs = d + 7 + (d - 5)(d^2 - 49) = (d/2 - 3)N \equiv 0 \pmod{N}.$$

Согласно лемме 2 диаметр равен $d(N; 1, s) = d + 1$, т. е. граф $C(N; 1, s)$ субоптимальный.

При $d = 6k + 3$ возьмём $a = d - 6$, $b = d + 6$, $u = 2d - 12$, $v = 2$. Тогда

$$av + bu = 2(d - 6) + (d + 6)(2d - 12) = 2d^2 + 2d - 84 = N.$$

Аналогично определяем, что $(-a, b)$, (u, v) — два соседних нуля, и согласно лемме 2 диаметр графа равен $d(N; 1, s) = d + 1$, т. е. граф $C(N; 1, s)$ субоптимальный. Теорема 4 доказана.

Теорема 5. *Если* $N = 2d^2 + 2d - 126$, *то*

$$\Psi_6 = \{C(N; 1, s) \mid d = 6k + 2,\ 6k + 3,\ k \geqslant 21\}$$

— субоптимальное семейство циркулянтов с образующей $s = 2d - 15$.

Доказательство бесконечности семейства Ψ_6 аналогично доказательствам теорем 2–4: достаточно взять $a = d - 9$, $b = d + 9$, $u = d - 6$, $v = d + 8$ при $d = 6k + 2$, $d = 6k + 3$ и выполнить соответствующие действия. Теорема 5 доказана.

При формировании видов функций a, b, u, v в доказательствах теорем 1–5 использована модель LLM Qwen3-Coder в качестве ИИ-помощника. Для этого на основе описанного нами на языке Wolfram Mathematica алгоритма определения значений a, b, u, v ИИ-модель создала программу на Python. С помощью данной программы ИИ-модель формировала для конкретного семейства числовые значения a, b, u, v по полученным на входе числовым значениям N и s для трёх последовательно наращиваемых диаметров. Затем мы определяли по полученным числовым значениям вид искомых линейных функций a, b, u, v в зависимости от диаметра и использовали их в доказательствах. Поскольку при больших значениях диаметров для теоремы 5 в программе требовались экстремально

большие массивы (до 60 миллионов элементов размера $(N/2)^2/2$), ИИ-модель предложила итерационный алгоритм для сокращения массива используемых данных, который был успешно ею реализован.

Отметим, что в [13, теорема 6.2] найдены достаточные условия субоптимальности циркулянта $C(N;1,s)$, состоящие из 15 соотношений между $N = 2d^2 + 2d - B$, $0 < B < 4d - 1$, и d. Используя эти условия, автор [13] показал, что субоптимальные графы существуют при всех значениях $N \leqslant 8 \cdot 10^6$. Однако эти условия не позволяют находить аналитические описания семейств субоптимальных графов. В [13, пример 6.3] рассмотрены значения $d = 25$, $B = 20$ и для $N = 1280$ при выполнении условий субоптимальности графа найдены образующие $s = 42, 138, 238, 444$. Рассмотренные параметры d, B, N соответствуют семейству Ψ_2 из теоремы 1, и нами была дополнительно найдена образующая $s = 611 = d^2 - 14$, которая позволяет сформировать аналитически задаваемое бесконечное семейство субоптимальных графов.

Возможность аналитического задания параметров оптимальных (субоптимальных) семейств циркулянтных графов открывает перспективу разработки и использования алгоритмов маршрутизации аналитического вида с приемлемыми затратами требуемых ресурсов по памяти для сетей на кристалле [7]. Таким образом, полученные в настоящей работе топологии семейств субоптимальных циркулянтных сетей с минимальной структурной задержкой при межузловых обменах обладают также возможностью эффективной организации маршрутизации для них, так как для найденных семейств сетей могут быть применены масштабируемые по числу узлов алгоритмы маршрутизации сложности $O(1)$ [7, 16].

Заключение

Проектирование оптимальных сетевых топологий представляет собой один из основных этапов при разработке сетей на кристалле. В данной работе продолжены исследования по генерации и анализу датасетов оптимальных кольцевых циркулянтных сетей степени четыре. На основе анализа большого массива данных исследовано решение задачи поиска двумерных кольцевых циркулянтных сетей с минимально возможным диаметром при заданном порядке графа. Существенно расширен поиск и исследование свойств субоптимальных циркулянтных сетей с диаметром, превышающим на единицу точную нижнюю границу диаметра.

Новый подход к визуализации точек датасета, разработка темплейтов для субоптимальных описаний графов, а также применение больших языковых моделей для написания программ темплейт-ориентированного перебора и участие моделей в подборе функций параметров укладки графов позволили открыть множество новых аналитически задаваемых семейств субоптимальных циркулянтных сетей.

Доказаны теоремы о существовании субоптимальных семейств при неограниченном росте диаметров графов. Вопросы полноты получения множеств семейств оптимальных и субоптимальных графов $C(N; 1, s)$ и исследования других методов анализа датасета, а также использования алгоритмов маршрутизации для них при разработке сетей на кристалле являются темой будущих исследований.

Финансирование работы

Исследование выполнено за счёт Российского научного фонда (проект № 25–11–00248, rscf.ru/en/project/25-11-00248). Дополнительных грантов на проведение или руководство этим исследованием получено не было.

Конфликт интересов

Авторы заявляют, что у них нет конфликта интересов.

Литература

1. **Bermond J.-C., Comellas F., Hsu D. F.** Distributed loop computer networks: A survey // J. Parallel Distrib. Comput. 1995. V. 24. P. 2–10.
2. **Hwang F. K.** A survey on multi-loop networks // Theor. Comput. Sci. 2003. V. 299, No. 1–3. P. 107–121.
3. **Monakhova E. A.** A survey on undirected circulant graphs // Discrete Math. Algorithms Appl. 2012. V. 4, No. 1. Article ID 1250002. 30 p.
4. **Lewis R. R.** Analysis and construction of extremal circulant and other Abelian Cayley graphs: PhD Thesis. London, 2021. 390 p.
5. **Huang X., Ramos A. F., Deng Y.** Optimal circulant graphs as low-latency network topologies // J. Supercomput. 2022. V. 78, No. 11. P. 13491–13510. DOI: 10.1007/s11227-022-04396-5.
6. **Liu H., Li X., Wang S.** Construction of dual optimal bidirectional double-loop networks for optimal routing // Mathematics. 2022. V. 10. Article ID 4016. 17 p.
7. **Монахов О. Г., Монахова Э. А.** Масштабируемый подход к кодизайну топологий и алгоритмов маршрутизации для семейств оптимальных циркулянтных сетей степени четыре // Дискрет. анализ и исслед. операций. 2025. Т. 32, № 2. С. 88–106.
8. **Tzvieli D.** Minimal diameter double-loop networks. 1. Large infinite optimal families // Networks. 1991. V. 21, No. 4. P. 387–415.
9. **Monakhova E. A., Monakhov O. G.** Double-loop-networks. Datasets of optimal double loop networks or optimal circulant graphs $(N; 1, s)$. 2024. github.com/mila0411/Double-loop-networks (accessed: 1.08.2026).
10. **Монахова Э. А., Монахов О. Г.** Анализ базы данных оптимальных двухконтурных кольцевых сетей // Прикл. дискрет. математика. 2024. № 64. С. 56–71.

11. Du D.-Z., Hsu D. F., Li Q., Xu J. A combinatorial problem related to distributed loop networks // Networks. 1990. V. 20. P. 173–180.
12. Bermond J.-C., Tzvieli D. Minimal diameter double-loop networks: Dense optimal families // Networks. 1991. V. 21. P. 1–9.
13. Tzvieli D. Double-loop interconnection networks with minimal transmission delay: PhD Thesis. Baton Rouge, LA, 1988. 156 p.
14. Chen B.-X., Meng J.-X., Xiao W.-J. Some new optimal and suboptimal infinite families of undirected double-loop networks // Discrete Math. Theor. Comput. Sci. 2006. V. 8. P. 299–312.
15. Монахов О. Г., Монахова Э. А. Программа вычисления характеристик регулярных графов с параметрическим описанием. Свид. ... № 2013619128. М.: Фед. сл. по интеллект. собств., патентам и товар. знакам, 2013.
16. Chen B.-X., Meng J.-X., Xiao W.-J. A constant time optimal routing algorithm for undirected double-loop networks // Mobile ad-hoc and sensor networks. Proc. 1st Int. Conf. (Wuhan, China, Dec. 13–15, 2005). Heidelberg: Springer, 2005. P. 308–316. (Lect. Notes Comput. Sci.; V. 3794).

Монахова Эмилия Анатольевна
Монахов Олег Геннадьевич

Статья поступила
18 марта 2026 г.
После доработки —
5 апреля 2026 г.
Принята к публикации
22 апреля 2026 г.

DISKRETNYI ANALIZ I ISSLEDOVANIE OPERATSII
/DISCRETE ANALYSIS AND OPERATIONS RESEARCH/
April–June 2026. Vol. 33, No. 2. P. 64–81

UDC 519.176+519.8+519.7 DOI: 10.33048/daio.2026.33.859

SEARCH FOR FAMILIES OF SUBOPTIMAL CIRCULANT NETWORKS BASED ON AI

E. A. Monakhova[1,2,a] and *O. G. Monakhov*[2,b]

[1] HSE University,
20 Myasnitskaya Street, 101000 Moscow, Russia
[2] Institute of Computational Mathematics and Mathematical Geophysics,
6 Acad. Lavrentyev Avenue, 630090 Novosibirsk, Russia
E-mail: [a] emilia@rav.sscc.ru, [b] monakhov@rav.sscc.ru

Abstract. Based on the analysis of a dataset of two-dimensional ring circulant networks, a solution to the problem of finding families of suboptimal circulant networks with suboptimal diameters is investigated. Suboptimal two-dimensional ring circulants are graphs with the minimum possible diameter for a given number of vertices, exceeding by one the known exact lower bound on the circulant diameter. A new original plane visualization of dataset points corresponding to suboptimal circulants is presented. Using the discovered visual regularity of suboptimal circulant description sequences and large language models as an AI assistant for program development, new families of suboptimal graphs are found, comprising a total of 75% of the suboptimal dataset points. The scalability of the discovered graph families with increasing diameters is theoretically proven, while a large language model is used in proving the theorems. The analytically defined families of two-dimensional ring circulants found are of interest as topologies of communication networks for networks on a chip and communication systems. Tab. 3, illustr. 3, bibliogr. 16.

Keywords: undirected double loop network, diameter, dataset of optimal circulants, network on a chip, LLM.

References

1. **J.-C. Bermond, F. Comellas,** and **D. F. Hsu,** Distributed loop computer networks: A survey, *J. Parallel Distrib. Comput.* **24**, 2–10 (1995).
2. **F. K. Hwang,** A survey on multi-loop networks, *Theor. Comput. Sci.* **299** (1–3), 107–121 (2003).

English transl.: *Journal of Applied and Industrial Mathematics* **20** (2) (2026).

3. **E. A. Monakhova,** A survey on undirected circulant graphs, *Discrete Math. Algorithms Appl.* **4** (1), ID 1250002 (2012).
4. **R. R. Lewis,** Analysis and construction of extremal circulant and other Abelian Cayley graphs, *PhD Thesis* (London, 2021).
5. **X. Huang, A. F. Ramos,** and **Y. Deng,** Optimal circulant graphs as low-latency network topologies, *J. Supercomput.* **78** (11), 13491–13510 (2022), DOI: 10.1007/s11227-022-04396-5.
6. **H. Liu, X. Li,** and **S. Wang,** Construction of dual optimal bidirectional double-loop networks for optimal routing, *Mathematics* **10**, ID 4016 (2022).
7. **O. G. Monakhov** and **E. A. Monakhova,** A scalable approach to co-design of topologies and routing algorithms for families of optimal degree-four circulant networks, *Diskretn. Anal. Issled. Oper.* **32** (2), 88–106 (2025) [Russian] [*J. Appl. Ind. Math.* **19** (2), 291–301 (2025), DOI: 10.1134/S1990478925020085].
8. **D. Tzvieli,** Minimal diameter double-loop networks. 1. Large infinite optimal families, *Networks* **21** (4), 387–415 (1991).
9. **E. A. Monakhova** and **O. G. Monakhov,** Double-loop-networks. Datasets of optimal double loop networks or optimal circulant graphs $(N; 1, s)$. 2024. github.com/mila0411/Double-loop-networks (accessed: 1.08.2026).
10. **E. A. Monakhova** and **O. G. Monakhov,** Database analysis of optimal double-loop networks, *Prikl. Diskretn. Mat.*, No. 64, 56–71 (2024) [Russian].
11. **D.-Z. Du, D. F. Hsu, Q. Li,** and **J. Xu,** A combinatorial problem related to distributed loop networks, *Networks* **20**, 173–180 (1990).
12. **J.-C. Bermond** and **D. Tzvieli,** Minimal diameter double-loop networks: Dense optimal families, *Networks* **21**, 1–9 (1991).
13. **D. Tzvieli,** Double-loop interconnection networks with minimal transmission delay, *PhD Thesis* (Baton Rouge, LA, 1988).
14. **B.-X. Chen, J.-X. Meng,** and **W.-J. Xiao,** Some new optimal and suboptimal infinite families of undirected double-loop networks, *Discrete Math. Theor. Comput. Sci.* **8**, 299–312 (2006).
15. **O. G. Monakhov** and **E. A. Monakhova,** A program for calculating characteristics of regular graphs with a parametric description. Certif. ... No. 2013619128 (Fed. Serv. Intellect. Prop. Pat. Trademarks, Moscow, 2013) [Russian].
16. **B.-X. Chen, J.-X. Meng,** and **W.-J. Xiao,** A constant time optimal routing algorithm for undirected double-loop networks, in *Mobile ad-hoc and sensor networks*, Proc. 1st Int. Conf. (Wuhan, China, Dec. 13–15, 2005) (Springer, Heidelberg, 2005), pp. 308–316 (Lect. Notes Comput. Sci., Vol. 3794).

Emiliya A. Monakhova
Oleg G. Monakhov

Received March 18, 2026
Revised April 5, 2026
Accepted April 22, 2026

ДИСКРЕТНЫЙ АНАЛИЗ И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ
Апрель–июнь 2026. Т. 33, № 2. С. 82–96

УДК 519.712 DOI: 10.33048/daio.2026.33.843

О ВЫЧИСЛИТЕЛЬНОЙ СЛОЖНОСТИ НАХОЖДЕНИЯ ХРЕБТОВ БУЛЕВЫХ ФОРМУЛ

В. Ю. Попов

Уральский федеральный университет,
ул. Ленина, 51, 620083 Екатеринбург, Россия
E-mail: popovvvv@gmail.com

Аннотация. Хребты являются важными элементами решений ряда комбинаторных задач. В частности, хребты рассматривались для задачи коммивояжёра и различных вариантов задачи выполнимости. В данной статье рассматриваются хребты для задачи выполнимости. Хребты булевых формул представляют собой подмножества переменных, имеющие единственное допустимое значение. Существует ряд различных применений хребтов. Хребты широко используются для решения задачи выполнимости с помощью эвристических алгоритмов. В данной статье рассматривается вычислительная сложность задачи поиска хребта, а также задачи нахождения допустимых значений для хребтов. Библиогр. 38.

Ключевые слова: вычислительная сложность, задача выполнимости, NP-трудность, хребты.

Введение

Исследования пространств решений комбинаторных задач физическими методами выявили для некоторых задач феномен, называемый замораживанием переменных. В частности, наличие данного феномена обнаружено для задачи коммивояжёра в исследованиях, представленных в работе [1]. Замораживание переменных наблюдается и для задачи выполнимости [2–4].

Замораживание переменных проявляется в том, что некоторые части хороших наборов значений переменных остаются практически неизменными почти для всех хороших наборов значений переменных. Замораживание переменных указывает на закономерную неравномерность в пространстве решений: наличие областей повышенной плотности и значительных разреженных областей. Следует отметить, что многие современные криптографические алгоритмы в значительной степени опираются на разреженные или плотные данные (см., например, [5–7]). Это

© В. Ю. Попов, 2026

естественным образом ведёт к разработке методов криптоанализа, которые направлены на поиск специфических закономерностей, связанных с разреженными и плотными данными [8–10]. В силу этого феномен замораживания переменных представляет существенный интерес как для криптографии, так и для криптоанализа.

Выявлено наличие тесной связи замораживания переменных с другим важным направлением теоретического исследования задачи выполнимости — фазовым переходом и пороговой гипотезой [11]. Явление фазового перехода указывает на то, что трудность задачи зависит не столько от размера исходных данных, сколько от некоторых их структурных особенностей. В последние годы исследования в области фазового перехода получили существенное развитие (см., например, [12]). В частности, пороговая гипотеза получила подтверждение как для задачи выполнимости [13, 14], так и в общем случае [15, 16].

Исследования в области фазового перехода показывают, что средние вычислительные затраты на поиск решения для задачи выполнимости растут линейно с увеличением числа переменных булевой формулы [17]. Однако для небольшой доли булевых формул наблюдается экспоненциальный рост затрат [17], что обусловлено присутствием замораживания переменных в булевых формулах [18]. Проявления феномена замораживания переменных для различных комбинаторных задач были формализованы на основе понятия хребта [18, 19].

От способности решателя учитывать структурные особенности булевых формул в значительной мере зависит его производительность [20]. Хребты рассматриваются как одна из основных структурных особенностей [21, 22]. Многие современные решатели используют поиск хребтов [23–25]. Экспериментально установленная зависимость между наличием хребта и трудностью анализа булевой формулы указывает на естественный путь к повышению производительности решателей: необходимо как можно раньше обнаружить хребет и элиминировать его из формулы, присваивая конкретные значения переменным. Данная работа посвящена вычислительной сложности соответствующих задач.

1. Основные определения

Как обычно, для произвольного множества M через $|M|$ будем обозначать его мощность, а для произвольной булевой формулы f через $V(f)$ — множество всех её переменных. Произвольный набор значений переменных из подмножества S множества $V(f)$ задаётся некоторым отображением $A\colon S \to \{0,1\}$. Будем обозначать через $F[A(S)]$ булеву формулу, полученную из формулы f подстановкой вместо каждого вхождения переменной x её значения $A(x)$ для всех $x \in S$. Следуя [26], для булевой

формулы f множество $S \subseteq V(f)$ будем называть хребтом, если существует единственное отображение A такое, что формула $F[A(S)]$ выполнима при условии выполнимости формулы f.

2. Нахождение хребтов

Понятия и определения, которые не приводятся в настоящей статье непосредственно, можно найти в книге [27]. В [28] показано, что если P $\neq$ NP, то не существует аппроксимационной процедуры, которая бы гарантировала нахождение достаточно большого хребта за полиномиальное время. Для произвольного множества булевых формул $\mathcal{H}$ и для любого действительного числа β, $0 < \beta < 1$, через $L(\mathcal{H}, \beta)$ обозначим язык, состоящий из всех булевых формул h, удовлетворяющих следующим условиям: $h \in \mathcal{H}$ и существует хребет $S \subseteq V(h)$ такой, что $|S| \geqslant \beta|V(h)|$. В [29] установлено, что для любого действительного числа β, $0 < \beta < 1$, существует генерируемое полиномиальным алгоритмом множество булевых формул $\mathcal{B}$ такое, что язык $L(\mathcal{B}, \beta)$ NP-полный. Результаты исследований [28, 29] указывают на то, что нахождение хребта может быть не менее трудной задачей, чем решение задачи выполнимости. Следующее утверждение показывает, что нахождение хребта может быть значительно труднее, чем решение задачи выполнимости.

Теорема 1. *Для любого $\beta \in (0, 1)$ существует генерируемое полиномиальным алгоритмом множество булевых формул $\mathcal{F}$ такое, что язык $L(\mathcal{F}, \beta)$* DP*-полный.*

Доказательство. Для произвольной булевой формулы p обозначим через p^* булеву формулу, полученную из p заменой каждого вхождения переменной x на $\neg x$ для всех переменных $x \in V(p)$. Зафиксируем произвольное $\beta \in (0, 1)$. Пусть $\mathcal{F}$ — множество всевозможных булевых формул вида

$$\Big(f \vee \bigwedge_{x \in W} x\Big) \wedge (g \vee g^*), \tag{1}$$

удовлетворяющих следующим трём условиям:

1) булевы формулы f и g представлены в конъюнктивной нормальной форме, причём каждая дизъюнкция содержит по три литерала;

2) множества переменных $V(f)$, $V(g)$ и W попарно не пересекаются;

3) выполняются неравенства

$$|W| \geqslant \beta(|V(f)| + |V(g)| + |W|), \tag{2}$$

$$|V(f)| < \beta(|V(f)| + |V(g)| + |W|). \tag{3}$$

Легко понять, что множество булевых формул $\mathcal{F}$ рекурсивно. В частности, элементы множества $\mathcal{F}$ можно сгенерировать по их номерам. Заметим, что размер элементов множества $\mathcal{F}$ экспоненциально больше размера их естественных номеров в бинарной кодировке, поэтому для установления полиномиального соответствия между размерами входа и выхода при генерации естественно использовать стандартный в таких случаях приём, основанный на представлении номеров в унарной кодировке (см., например, [30] для случая уникальной задачи остановки). Легко понять, что по номерам, представленным в унарном виде, элементы множества $\mathcal{F}$ можно генерировать за полиномиальное время.

Доказательство DP-полноты языка $L(\mathcal{F}, \beta)$ построим на обосновании трудности распознавания принадлежности формул множества $\mathcal{F}$ языку $L(\mathcal{F}, \beta)$.

Непосредственно из определения хребта вытекает, что формула вида (1) имеет хребет лишь при условии, что формула $g \vee g^*$ выполняется. Выполнимость формулы $g \vee g^*$ равносильна выполнимости по крайней мере одной из формул g и g^*. Из определения формулы g^* следует, что $g[A(V(g))] = 1$ тогда и только тогда, когда $g^*[A^*(V(g))] = 1$, где $A(y) \neq A^*(y)$ для всех $y \in V(g)$. Следовательно, выполнимость формулы $g \vee g^*$ равносильна выполнимости формулы g. Кроме того, ни одна из переменных множества $V(g)$ не принадлежит хребту.

Допустим, что формула f выполнима. Легко понять, что если формула f выполнима, то ни одна из переменных множества W не принадлежит хребту: либо формула (1) не будет выполнимой, либо формула (1) выполняется для любых значений переменных из множества W, поэтому если формула f выполнима, то для любого хребта S справедливо неравенство

$$|S| \leqslant |V(f)|. \tag{4}$$

Из определения языка $L(\mathcal{H}, \beta)$ и системы неравенств (3) и (4) очевидным образом вытекает, что формула (1) не принадлежит языку $L(\mathcal{F}, \beta)$. Таким образом, чтобы формула (1) принадлежала языку $L(\mathcal{F}, \beta)$, необходимо, чтобы формула f не была выполнимой.

Рассмотрим следующую задачу.

Задача SAT-UNSAT (Выполнимость-невыполнимость). *Даны булевы формулы f и g, каждая из которых представлена в конъюнктивной нормальной форме, причём каждая дизъюнкция содержит по три литерала. Верно ли, что формула g выполнима, а формула f выполнимой не является?*

Хорошо известно, что задача SAT-UNSAT DP-полна (см., например, [27, теорема 17.1]). Легко понять, что если формула g выполнима, а формула f выполнимой не является, то формула (1) выполняется лишь

при условии, что все переменные множества W принимают значение 1, т. е. множество W является хребтом. Учитывая неравенство (2), формула (1) принадлежит языку $L(\mathcal{F}, \beta)$ тогда и только тогда, когда формула g выполнима, а формула f выполнимой не является. Легко понять, что для произвольных формул f и g, выбирая достаточно большое множество W, можно построить формулу вида (1) из множества $\mathcal{F}$. Отсюда и из DP-полноты задачи SAT-UNSAT очевидным образом вытекает DP-полнота языка $L(\mathcal{F}, \beta)$. Теорема 1 доказана.

Заметим, что обнаружение хребта позволяет существенно упростить анализ исходной формулы. Однако даже нахождение большого хребта не обеспечивает того, что полученная формула станет лёгкой для дальнейшего анализа, по крайней мере потому, что она тоже может содержать хребет. Тем самым представляет интерес исследование максимальных хребтов, т. е. хребтов S таких, что для любой переменной $x \notin S$ множество $S \cup \{x\}$ хребтом не является. Соответствующая задача в явном виде может быть сформулирована следующим образом.

Задача MB (Проверка хребта на максимальность). *Даны булева формула f и хребет S формулы f. Верно ли, что хребет S максимален?*

Следующее утверждение показывает, что проверка найденного хребта на максимальность является вычислительно трудной задачей.

Теорема 2. *Задача* MB co-NP-*трудна.*

Доказательство. Для произвольной булевой формулы g в конъюнктивной нормальной форме рассмотрим булеву формулу

$$\bigwedge_{x \in W} x \wedge ((g \vee g^*) \vee z), \tag{5}$$

предполагая, что z — переменная такая, что $z \notin W$, $z \notin V(g)$, а множества W и $V(g)$ не имеют общих элементов. Легко понять, что для любой формулы g формула (5) выполнима. Следовательно, множество W является хребтом. Кроме того, ни одна переменная множества $V(g)$ в хребет не входит. Следовательно, максимальным хребтом является либо множество W, либо $W \cup \{z\}$. Очевидно, что $W \cup \{z\}$ — максимальный хребет тогда и только тогда, когда булева формула g невыполнима. Отсюда в силу co-NP-полноты задачи SAT-COMPLEMENT (см., например, [27, п. 10.1]) вытекает co-NP-трудность проверки хребта на максимальность. Теорема 2 доказана.

Хотя теоремы 1 и 2 мы доказываем конструктивно, т. е. строим соответствующие множества булевых формул в явном виде, результаты этих

утверждений относятся к количественной оценке хребтов: существует ли достаточно большой хребет и можно ли известный хребет увеличить. Результаты работы [31] показывают, что, даже зная хребет, найти значения его переменных — вычислительно трудная задача. Количественные оценки ещё менее информативны. В частности, даже зная мощность максимального хребта, мы, вообще говоря, не имеем эффективного способа нахождения хребта.

Теорема 3. *Если* $\mathrm{P} \neq \mathrm{NP} \cap \text{co-NP}$, *то для любого натурального* k *существует множество булевых формул* $\mathcal{F}$ *такое, что каждая формула* $f \in \mathcal{F}$ *имеет максимальный хребет мощности* k, *но не существует полиномиального алгоритма, позволяющего по произвольной формуле* $f \in \mathcal{F}$ *найти её хребет.*

Доказательство. При доказательстве теоремы будем пользоваться подходом, предложенным в [32] и адаптированным для исследования хребтов в [33]. Предположим, что $\mathrm{P} \neq \mathrm{NP} \cap \text{co-NP}$. Тогда существует язык L в некотором алфавите Σ такой, что $L \in (\mathrm{NP} \cap \text{co-NP}) \setminus \mathrm{P}$. Поскольку $L \in \text{co-NP}$, язык $\Sigma^* \setminus L$, где Σ^* — множество всевозможных слов в алфавите Σ, принадлежит классу NP. Зафиксируем некоторое рекурсивное множество

$$\{\mathfrak{N}_i \mid i \in \mathbb{N}\} \tag{6}$$

недетерминированных машин Тьюринга, распознающих всевозможные языки из класса NP за полиномиальное время. Обычно (см., например, [34–36]) в качестве множества (6) используются недетерминированные машины Тьюринга с часами, предложенные в работе [37]. Поскольку $L \in \mathrm{NP}$ и $\Sigma^* \setminus L \in \mathrm{NP}$, в множестве (6) найдутся машины $\mathfrak{N}_s$ и $\mathfrak{N}_t$, распознающие языки L и $\Sigma^* \setminus L$ соответственно. Используя конструкцию Кука [38], можем получить множество булевых формул

$$\{f(\mathfrak{N}_i, w) \mid i \in \mathbb{N},\ w \in \Sigma^*\} \tag{7}$$

такое, что формула $f(\mathfrak{N}_i, w)$ выполнима тогда и только тогда, когда слово w принадлежит языку, распознаваемому машиной $\mathfrak{N}_i$. Пусть $\mathcal{F}$ — подмножество множества (7), состоящее из формул вида

$$\Big(\bigwedge_{x \in U} x \vee f(\mathfrak{N}_s, w) \vee f^*(\mathfrak{N}_s, w)\Big) \wedge \Big(\bigwedge_{y \in W} y \vee f(\mathfrak{N}_t, w) \vee f^*(\mathfrak{N}_t, w)\Big) \tag{8}$$

для всех $w \in \Sigma^*$, где множества U, W, $V(f(\mathfrak{N}_s, w))$, $V(f(\mathfrak{N}_t, w))$ попарно не пересекаются, $|U| = |W| = k$.

Заметим, что формулы $f(\mathfrak{N}_s, w)$ и $f(\mathfrak{N}_t, w)$ выполнимы тогда и только тогда, когда выполнимы формулы $f^*(\mathfrak{N}_s, w)$ и $f^*(\mathfrak{N}_t, w)$ соответственно, поэтому переменные из множеств $V(f(\mathfrak{N}_s, w))$ и $V(f(\mathfrak{N}_t, w))$ не принадлежат хребту. Поскольку машина $\mathfrak{N}_s$ распознаёт язык L, а машина $\mathfrak{N}_t$ —

дополнение языка L, формула $f(\mathfrak{N}_s, w)$ выполнима тогда и только тогда, когда формула $f(\mathfrak{N}_t, w)$ выполнимой не является.

Допустим, что формула $f(\mathfrak{N}_s, w)$ выполнима. Тогда переменные множества U хребту не принадлежат, а множество W — максимальный хребет формулы (8). Если формула $f(\mathfrak{N}_t, w)$ выполнима, то переменные множества W хребту не принадлежат, а множество U является максимальным хребтом формулы (8). Таким образом, если $w \in L$, то множество W является максимальным хребтом формулы (8), а если $w \notin L$, то множество U является максимальным хребтом формулы (8). Поскольку $L \notin \mathrm{P}$, не существует полиномиального алгоритма, позволяющего по произвольной формуле $f \in \mathcal{F}$ найти её хребет. Теорема 3 доказана.

3. Нахождение значений хребтов

При анализе булевой формулы f нахождение хребта S само по себе позволяет получить важную информацию о структуре формулы f и пространстве решений. Однако для упрощения дальнейшего анализа формулы f необходимо нахождение значения хребта S, т. е. конкретных значений переменных из множества S, обеспечивающих выполнимость формулы f, что само по себе является вычислительно трудной задачей (см. [31]). Следующее утверждение является аналогом результатов работы [31] для случая максимальных хребтов.

Теорема 4. *Если* $\mathrm{P} \neq \mathrm{NP} \cap \text{co-NP}$, *то для любого натурального* k *существует множество булевых формул* $\mathcal{F}$ *такое, что для каждой формулы* $f \in \mathcal{F}$ *известен максимальный хребет мощности* k, *но не существует полиномиального алгоритма, позволяющего по произвольной формуле* $f \in \mathcal{F}$ *найти значения хребта.*

Доказательство осуществим модификацией рассуждений, проведённых при доказательстве теоремы 3. Рассмотрим множество $\mathcal{F}$, предполагая, что оно является подмножеством множества (7), состоящим из формул вида

$$\Big(\bigwedge_{x\in U} x \vee f(\mathfrak{N}_s, w) \vee f^*(\mathfrak{N}_s, w)\Big) \wedge \Big(\bigwedge_{x\in U} \neg x \vee f(\mathfrak{N}_t, w) \vee f^*(\mathfrak{N}_t, w)\Big) \qquad (9)$$

для всех $w \in \Sigma^*$, где множества U, $V(f(\mathfrak{N}_s, w))$, $V(f(\mathfrak{N}_t, w))$ попарно не пересекаются, $|U| = k$. Легко понять, что множество U является максимальным хребтом формулы (9). Если $w \in L$, то формула (9) выполняется, когда все переменные множества U принимают значение 0. Если $w \notin L$, то формула (9) выполняется, когда все переменные множества U принимают значение 1. Поскольку $L \notin \mathrm{P}$, не существует полиномиального алгоритма, позволяющего по произвольной формуле $f \in \mathcal{F}$ найти значения хребта. Теорема 4 доказана.

Заключение

В данной статье рассмотрена вычислительная сложность задач, связанных с использованием хребтов при анализе булевых формул: существование хребта, нахождение хребта и нахождение значений хребта. Ранее уже было известно, что исследование булевых формул на основе хребтов может быть не менее трудной задачей, чем нахождение набора, на котором формула выполняется [28, 29, 33]. Мы показали, что подход с использованием хребтов может быть значительно более трудным, чем задача выполнимости.

Финансирование работы

Исследование выполнено за счёт бюджета Уральского федерального университета. Дополнительных грантов на проведение или руководство этим исследованием получено не было.

Конфликт интересов

Автор заявляет, что у него нет конфликта интересов.

Литература

1. **Kirkpatrick S., Toulouse G.** Configuration space analysis of travelling salesman problems // J. Phys. 1985. V. 45, No. 8. P. 1277–1292. DOI: 10.1051/jphys:019850046080127700.
2. **Monasson R., Zecchina R.** Entropy of the K-satisfiability problem // Phys. Rev. Lett. 1996. V. 76, No. 21. P. 3881–3885. DOI: 10.1103/PhysRevLett.76.3881.
3. **Monasson R., Zecchina R.** Statistical mechanics of the random K-satisfiability model // Phys. Rev. E. 1997. V. 56, No. 2. P. 1357–1370. DOI: 10.1103/PhysRevE.56.1357.
4. **Monasson R., Zecchina R., Kirkpatrick S., Selman B., Troyansky L.** 2+p-SAT: Relation of typical-case complexity to the nature of the phase transition // Random Struct. Algorithms. 1999. V. 15, No. 3–4. P. 414–435. DOI: 10.1002/(SICI)1098-2418(199910/12)15:3/4<414::AID-RSA10>3.0.CO;2-G.
5. **Bohra D. D., Banerjee D. S., Sanadhya S.** Sparse-aware NTT: Accelerating lattice-based cryptography on FPGAs // 2025 IEEE Comput. Soc. Annu. Symp. Very Large Scale Integration (Kalamata, Greece, July 6–9, 2025). Piscataway: IEEE, 2025. Article ID 11130302. 6 p. DOI: 10.1109/ISVLSI65124.2025.11130302.
6. **Dao Q., Jain A.** Lossy cryptography from code-based assumptions dense-sparse LPN: A new subexponentially hard LPN variant in SZK // J. Cryptol. 2025. V. 38, No. 32. P. 1–46. DOI: 10.1007/s00145-025-09553-6.

7. **He P., Tu Y., Bao T., Koç Ç. Ç., Xie J.** HSPA: High-throughput sparse polynomial multiplication for code-based post-quantum cryptography // ACM Trans. Embed. Comput. Syst. 2024. V. 24, No. 1. Article ID 16. 24 p. DOI: 10.1145/3703837.
8. **Campa L., Roy A.** Gröbner basis cryptanalysis of Anemoi // Advances in cryptology — EUROCRYPT 2025. Proc. 44th Annu. Int. Conf. Theory and Applications of Cryptographic Techniques (Madrid, Spain, May 4–8, 2025). Pt. 1. Cham: Springer, 2025. P. 303–332. (Lect. Notes Comput. Sci.; V. 15601). DOI: 10.1007/978-3-031-91107-1_11.
9. **Dinur I., Keller N., Klein O.** Fine-grained cryptanalysis: Tight conditional bounds for dense k-SUM and k-XOR // J. ACM. 2024. V. 71, No. 3. Article ID 23. 41 p. DOI: 10.1145/3653014.
10. **Jain A., Lin H., Saha S.** A systematic study of sparse LWE // Advances in cryptology — CRYPTO 2024. Proc. 44th Annu. Int. Cryptology Conf. (Santa Barbara, USA, Aug. 18–22, 2024). Pt. 3. Cham: Springer, 2024. P. 210–245. (Lect. Notes Comput. Sci.; V. 14922). DOI: 10.1007/978-3-031-68382-4_7.
11. **Bollobás B., Borgs C., Chayes J. T., Kim J. H., Wilson D. B.** The scaling window of the 2-SAT transition // Random Struct. Algorithms. 2001. V. 18, No. 3. P. 201–256. DOI: 10.1002/rsa.1006.
12. **Perkins W.** Searching for (sharp) thresholds in random structures: Where are we now? // Bull. Amer. Math. Soc. 2025. V. 62, No. 1. P. 113–143. DOI: 10.1090/bull/1857.
13. **Ding J., Sly A., Sun N.** Satisfiability threshold for random regular NAE-SAT // Commun. Math. Phys. 2016. V. 341. P. 435–489. DOI: 10.1007/s00220-015-2492-8.
14. **Ding J., Sly A., Sun N.** Proof of the satisfiability conjecture for large k // Ann. Math. 2022. V. 196, No. 1. P. 1–388. DOI: 10.4007/annals.2022.196.1.1.
15. **Park J., Pham H. T.** A proof of the Kahn–Kalai conjecture // J. Amer. Math. Soc. 2024. V. 37. P. 235–243. DOI: 10.1090/jams/1028.
16. **Park B., Vondrák J.** A simple proof of the nonuniform Kahn–Kalai conjecture // SIAM J. Discrete Math. 2024. V. 38, No. 3. P. 2060–2073. DOI: 10.1137/23M1587075.
17. **Selman B., Kirkpatrick S.** Critical behavior in the computational cost of satisfiability testing // Artif. Intell. 1996. V. 81, No. 1–2. P. 273–295. DOI: 10.1016/0004-3702(95)00056-9.
18. **Monasson R., Zecchina R., Kirkpatrick S., Selman B., Troyansky L.** Determining computational complexity from characteristic "phase transitions" // Nature. 1999. V. 400. P. 133–137. DOI: 10.1038/22055.
19. **Schneider J., Froschhammer C., Morgenstern I., Husslein T., Singer J. M.** Searching for backbones — An efficient parallel algorithm for the traveling salesman problem // Comput. Phys. Commun. 1996. V. 96, No. 2–3. P. 173–188. DOI: 10.1016/0010-4655(96)00062-8.

20. Alyahya T. N., Menai M. E. B., Mathkour H. On the structure of the Boolean satisfiability problem: A survey // ACM Comput. Surv. 2022. V. 55, No. 3. Article ID 46. 34 p. DOI: 10.1145/3491210.

21. Dubois O., Dequen G. A backbone-search heuristic for efficient solving of hard 3-SAT formulae // Proc. 17th Int. Jt. Conf. Artificial Intelligence (Seattle, USA, Aug. 4–10, 2001). V. 1. San Francisco: Morgan Kaufmann Publ., 2001. P. 248–253.

22. Menaï M. E. B., Batouche M. A backbone-based co-evolutionary heuristic for partial MAX-SAT // Artificial evolution. Rev. Sel. Pap. 7th Int. Conf. (Lille, France, Oct. 26–28, 2005). Heidelberg: Springer, 2006. P. 155–166. (Lect. Notes Comput. Sci.; V. 3871). DOI: 10.1007/11740698_14.

23. Biere A., Faller T., Fazekas K., Fleury M., Froleyks N., Pollitt F. CaDiCaL 2.0 // Computer aided verification. Proc. 36th Int. Conf. (Montreal, Canada, July 24–27, 2024). Pt. 1. Cham: Springer, 2024. P. 133–152. (Lect. Notes Comput. Sci.; V. 14681). DOI: 10.1007/978-3-031-65627-9_7.

24. Froleyks N., Yu E., Biere A. BIG backbones // Proc. 23rd Conf. Formal Methods in Computer-Aided Design (Ames, USA, Oct. 23–27, 2023). Wien: TU Wien Acad. Press, 2023. P. 162–167. DOI: 10.34727/2023/isbn.978-3-85448-060-0_24.

25. Schreiber D., Rigi-Luperti N, Biere A. Streamlining distributed SAT solver design // Proc. 28th Int. Conf. Theory and Applications of Satisfiability Testing (Glasgow, UK, Aug. 12–15, 2025). Wadern: Leibniz-Zentrum Inform., 2025. P. 27:1–27:23. (Leibniz Int. Proc. Inform.; V. 341). DOI: 10.4230/LIPIcs.SAT.2025.27.

26. Williams R., Gomes C. P., Selman B. Backdoors to typical case complexity // Proc. 18th Int. Jt. Conf. Artificial Intelligence (Acapulco, Mexico, Aug. 9–15, 2003). San Francisco: Morgan Kaufmann Publ., 2003. P. 1173–1178.

27. Papadimitriou C. H. Computational complexity. Reading, MA: Addison-Wesley Publ., 1994. 524 p.

28. Kilby P., Slaney J., Thiebaux S., Walsh T. Backbones and backdoors in satisfiability // Proc. 20th Nat. Conf. Artificial Intelligence (Pittsburgh, USA, July 9–13, 2005). Washington: AAAI Press, 2005. P. 1368–1373.

29. Hemaspaandra L. A., Narváez D. E. Existence versus exploitation: The opacity of backdoors and backbones // Prog. Artif. Intell. 2021. V. 10. P. 297–308. DOI: 10.1007/s13748-021-00234-6.

30. Blass A., Gurevich Y. On the unique satisfiability problem // Inf. Control. 1982. V. 55, No. 1–3. P. 80–88. DOI: 10.1016/S0019-9958(82)90439-9.

31. Hemaspaandra L. A., Narváez D. E. The opacity of backbones // Proc. 31st AAAI Conf. Artificial Intelligence (San Francisco, USA, Feb. 4–9, 2017). Washington: AAAI Press, 2017. P. 3900–3906.

32. Borodin A. B., Demers A. J. Some comments on functional self-reducibility and the NP hierarchy: Tech. rep. Ithaca, NY: Cornell Univ., 1976. 22 p.

33. Hemaspaandra L. A., Narváez D. E. The opacity of backbones // Inf. Comput. 2021. V. 281. Article ID 104772. 10 p. DOI: 10.1016/j.ic.2021.104772.

34. Landweber L. H., Lipton R. J., Robertson E. L. On the structure of sets in NP and other complexity classes // Theor. Comput. Sci. 1981. V. 15, No. 2. P. 181–200. DOI: 10.1016/0304-3975(81)90069-4.

35. Hartmanis J., Immerman N. On complete problems for NP∩CoNP // Automata, languages and programming. Proc. 12th Int. Colloq. (Nafplion, Greece, July 15–19, 1985). Heidelberg: Springer, 1985. P. 250–259. (Lect. Notes Comput. Sci.; V. 194). DOI: 10.1007/BFb0015750.

36. Hartmanis J., Hemachandra L. A. Complexity classes without machines: On complete languages for UP // Theor. Comput. Sci. 1988. V. 58, No. 1–3. P. 129–142. DOI: 10.1016/0304-3975(88)90022-9.

37. Baker T., Gill J., Solovay R. Relativizations of the P =? NP question // SIAM J. Comput. 1975. V. 4, No. 4. P. 431–442. DOI: 10.1137/0204037.

38. Cook S. A. The complexity of theorem-proving procedures // Proc. 3rd Annu. ACM Symp. Theory of Computing (Shaker Heights, USA, May, 3–5, 1971). New York: ACM, 1971. P. 151–158. DOI: 10.1145/800157.805047.

Попов Владимир Юрьевич

Статья поступила
5 ноября 2025 г.
После доработки —
11 февраля 2026 г.
Принята к публикации
23 марта 2026 г.

DISKRETNYI ANALIZ I ISSLEDOVANIE OPERATSII
/DISCRETE ANALYSIS AND OPERATIONS RESEARCH/
April–June 2026. Vol. 33, No. 2. P. 82–96

UDC 519.712 DOI: 10.33048/daio.2026.33.843

ON COMPUTATIONAL COMPLEXITY OF FINDING BACKBONES OF BOOLEAN FORMULAS

V. Yu. Popov

Ural Federal University,
51 Lenin Street, 620083 Yekaterinburg, Russia
E-mail: popovvvv@gmail.com

Abstract. Backbones are important parts of solutions for a number of combinatorial problems. In particular, backbones have been considered for the traveling salesman problem and various modifications of the satisfiability problem. In this paper, we consider backbones for the latter. The backbones of Boolean formulas are subsets of variables with unique proper assignment. There is a number of different applications of backbones. They are extensively used to solve the satisfiability problem with heuristic algorithms. In this paper, we consider the computational complexity of the problem of finding backbones and of the problem of finding proper assignments for backbones. Bibliogr. 38.

Keywords: computational complexity, satisfiability problem, NP-hardness, backbones.

References

1. **S. Kirkpatrick** and **G. Toulouse,** Configuration space analysis of travelling salesman problems, *J. Phys.* **45** (8), 1277–1292 (1985), DOI: 10.1051/jphys:019850046080127700.
2. **R. Monasson** and **R. Zecchina,** Entropy of the K-satisfiability problem, *Phys. Rev. Lett.* **76** (21), 3881–3885 (1996), DOI: 10.1103/PhysRevLett.76.3881.
3. **R. Monasson** and **R. Zecchina,** Statistical mechanics of the random K-satisfiability model, *Phys. Rev. E.* **56** (2), 1357–1370 (1997), DOI: 10.1103/PhysRevE.56.1357.
4. **R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman,** and **L. Troyansky,** 2+p-SAT: Relation of typical-case complexity to the nature of the phase transition, *Random Struct. Algorithms* **15** (3–4), 414–435 (1999), DOI: 10.1002/(SICI)1098-2418(199910/12)15:3/4<414::AID-RSA10>3.0.CO;2-G.

English transl.: *Journal of Applied and Industrial Mathematics* **20** (2) (2026).

5. **D. D. Bohra, D. S. Banerjee,** and **S. Sanadhya,** Sparse-aware NTT: Accelerating lattice-based cryptography on FPGAs, in *2025 IEEE Comput. Soc. Annu. Symp. Very Large Scale Integration* (Kalamata, Greece, July 6–9, 2025) (IEEE, Piscataway, 2025), ID 11130302, DOI: 10.1109/ISVLSI65124.2025.11130302.
6. **Q. Dao** and **A. Jain,** Lossy cryptography from code-based assumptions dense-sparse LPN: A new subexponentially hard LPN variant in SZK, *J. Cryptol.* **38** (32), 1–46 (2025), DOI: 10.1007/s00145-025-09553-6.
7. **P. He, Y. Tu, T. Bao, Ç. Ç. Koç** and **J. Xie,** HSPA: High-throughput sparse polynomial multiplication for code-based post-quantum cryptography, *ACM Trans. Embed. Comput. Syst.* **24** (1), ID 16 (2024), DOI: 10.1145/3703837.
8. **L. Campa** and **A. Roy,** Gröbner basis cryptanalysis of Anemoi, in *Advances in Cryptology — EUROCRYPT 2025*, Proc. 44th Annu. Int. Conf. Theory and Applications of Cryptographic Techniques (Madrid, Spain, May 4–8, 2025), Pt. 1 (Springer, Cham, 2025), pp. 303–332 (Lect. Notes Comput. Sci., Vol. 15601), DOI: 10.1007/978-3-031-91107-1_11.
9. **I. Dinur, N. Keller,** and **O. Klein,** Fine-grained cryptanalysis: Tight conditional bounds for dense k-SUM and k-XOR, *J. ACM* **71** (3), ID 23 (2024), DOI: 10.1145/3653014.
10. **A. Jain, H. Lin,** and **S. Saha,** A systematic study of sparse LWE, in *Advances in Cryptology — CRYPTO 2024*, Proc. 44th Annu. Int. Cryptology Conf. (Santa Barbara, USA, Aug. 18–22, 2024), Pt. 3 (Springer, Cham, 2024), pp. 210–245 (Lect. Notes Comput. Sci., Vol. 14922), DOI: 10.1007/978-3-031-68382-4_7.
11. **B. Bollobás, C. Borgs, J. T. Chayes, J. H. Kim,** and **D. B. Wilson,** The scaling window of the 2-SAT transition, *Random Struct. Algorithms* **18** (3), 201–256 (2001), DOI: 10.1002/rsa.1006.
12. **W. Perkins,** Searching for (sharp) thresholds in random structures: Where are we now?, *Bull. Amer. Math. Soc.* **62** (1), 113–143 (2025), DOI: 10.1090/bull/1857.
13. **J. Ding, A. Sly,** and **N. Sun,** Satisfiability threshold for random regular NAE-SAT, *Commun. Math. Phys.* **341**, 435–489 (2016), DOI: 10.1007/s00220-015-2492-8.
14. **J. Ding, A. Sly,** and **N. Sun,** Proof of the satisfiability conjecture for large k, *Ann. Math.* **196** (1), 1–388 (2022), DOI: 10.4007/annals.2022.196.1.1.
15. **J. Park** and **H. T. Pham,** A proof of the Kahn–Kalai conjecture, *J. Amer. Math. Soc.* **37**, 235–243 (2024), DOI: 10.1090/jams/1028.
16. **B. Park** and **J. Vondrák,** A simple proof of the nonuniform Kahn–Kalai conjecture, *SIAM J. Discrete Math.* **38** (3), 2060–2073 (2024), DOI: 10.1137/23M1587075.
17. **B. Selman** and **S. Kirkpatrick,** Critical behavior in the computational cost of satisfiability testing, *Artif. Intell.* **81** (1–2), 273–295 (1996), DOI: 10.1016/0004-3702(95)00056-9.

18. **R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman,** and **L. Troyansky,** Determining computational complexity from characteristic "phase transitions", *Nature* **400**, 133–137 (1999), DOI: 10.1038/22055.
19. **J. Schneider, C. Froschhammer, I. Morgenstern, T. Husslein,** and **J. M. Singer,** Searching for backbones — An efficient parallel algorithm for the traveling salesman problem, *Comput. Phys. Commun.* **96** (2–3), 173–188 (1996), DOI: 10.1016/0010-4655(96)00062-8.
20. **T. N. Alyahya, M. E. B. Menai,** and **H. Mathkour,** On the structure of the Boolean satisfiability problem: A survey, *ACM Comput. Surv.* **55** (3), ID 46 (2022), DOI: 10.1145/3491210.
21. **O. Dubois** and **G. Dequen,** A backbone-search heuristic for efficient solving of hard 3-SAT formulae, in *Proc. 17th Int. Jt. Conf. Artificial Intelligence* (Seattle, USA, Aug. 4–10, 2001), Vol. 1 (Morgan Kaufmann Publ., San Francisco, 2001), pp. 248–253.
22. **M. E. B. Menaï** and **M. Batouche,** A backbone-based co-evolutionary heuristic for partial MAX-SAT, in *Artificial Evolution*, Rev. Sel. Pap. 7th Int. Conf. (Lille, France, Oct. 26–28, 2005) (Springer, Heidelberg, 2006), pp. 155–166 (Lect. Notes Comput. Sci., Vol. 3871), DOI: 10.1007/11740698_14.
23. **A. Biere, T. Faller, K. Fazekas, M. Fleury, N. Froleyks,** and **F. Pollitt,** CaDiCaL 2.0, in *Computer Aided Verification*, Proc. 36th Int. Conf. (Montreal, Canada, July 24–27, 2024), Pt. 1 (Springer, Cham, 2024), pp. 133–152 (Lect. Notes Comput. Sci., Vol. 14681), DOI: 10.1007/978-3-031-65627-9_7.
24. **N. Froleyks, E. Yu,** and **A. Biere,** BIG backbones, in *Proc. 23rd Conf. Formal Methods in Computer-Aided Design* (Ames, USA, Oct. 23–27, 2023) (TU Wien Acad. Press, Wien, 2023), pp. 162–167, DOI: 10.34727/2023/isbn.978-3-85448-060-0_24.
25. **D. Schreiber, N Rigi-Luperti,** and **A. Biere,** Streamlining distributed SAT solver design, in *Proc. 28th Int. Conf. Theory and Applications of Satisfiability Testing* (Glasgow, UK, Aug. 12–15, 2025) (Leibniz-Zentrum Inform., Wadern, 2025), pp. 27:1–27:23 (Leibniz Int. Proc. Inform., Vol. 341), DOI: 10.4230/LIPIcs.SAT.2025.27.
26. **R. Williams, C. P. Gomes,** and **B. Selman,** Backdoors to typical case complexity, in *Proc. 18th Int. Jt. Conf. Artificial Intelligence* (Acapulco, Mexico, Aug. 9–15, 2003) (Morgan Kaufmann Publ., San Francisco, 2003), pp. 1173–1178.
27. **C. H. Papadimitriou,** *Computational Complexity* (Addison-Wesley Publ., Reading, MA, 1994).
28. **P. Kilby, J. Slaney, S. Thiebaux,** and **T. Walsh,** Backbones and backdoors in satisfiability, in *Proc. 20th Nat. Conf. Artificial Intelligence* (Pittsburgh, USA, July 9–13, 2005) (AAAI Press, Washington, 2005), pp. 1368–1373.
29. **L. A. Hemaspaandra** and **D. E. Narváez,** Existence versus exploitation: The opacity of backdoors and backbones, *Prog. Artif. Intell.* **10**, 297–308 (2021), DOI: 10.1007/s13748-021-00234-6.

30. A. Blass and **Y. Gurevich,** On the unique satisfiability problem, *Inf. Control.* **55** (1–3), 80–88 (1982), DOI: 10.1016/S0019-9958(82)90439-9.

31. L. A. Hemaspaandra and **D. E. Narváez,** The opacity of backbones, in *Proc. 31st AAAI Conf. Artificial Intelligence* (San Francisco, USA, Feb. 4–9, 2017) (AAAI Press, Washington, 2017), pp. 3900–3906.

32. A. B. Borodin and **A. J. Demers,** Some comments on functional self-reducibility and the NP hierarchy, *Tech. Rep.* (Cornell Univ., Ithaca, NY, 1976).

33. L. A. Hemaspaandra and **D. E. Narváez,** The opacity of backbones, *Inf. Comput.* **281**, ID 104772 (2021), DOI: 10.1016/j.ic.2021.104772.

34. L. H. Landweber, R. J. Lipton, and **E. L. Robertson,** On the structure of sets in NP and other complexity classes, *Theor. Comput. Sci.* **15** (2), 181–200 (1981), DOI: 10.1016/0304-3975(81)90069-4.

35. J. Hartmanis and **N. Immerman,** On complete problems for NP ∩ CoNP, in *Automata, Languages and Programming*, Proc. 12th Int. Colloq. (Nafplion, Greece, July 15–19, 1985) (Springer, Heidelberg, 1985), pp. 250–259 (Lect. Notes Comput. Sci., Vol. 194), DOI: 10.1007/BFb0015750.

36. J. Hartmanis and **L. A. Hemachandra,** Complexity classes without machines: On complete languages for UP, *Theor. Comput. Sci.* **58** (1–3), 129–142 (1988), DOI: 10.1016/0304-3975(88)90022-9.

37. T. Baker, J. Gill, and **R. Solovay,** Relativizations of the P =? NP question, *SIAM J. Comput.* **4** (4), 431–442 (1975), DOI: 10.1137/0204037.

38. S. A. Cook, The complexity of theorem-proving procedures, in *Proc. 3rd Annu. ACM Symp. Theory of Computing* (Shaker Heights, USA, May, 3–5, 1971) (ACM, New York, 1971), pp. 151–158, DOI: 10.1145/800157.805047.

Vladimir Yu. Popov

Received November 5, 2025
Revised February 11, 2026
Accepted March 23, 2026

ДИСКРЕТНЫЙ АНАЛИЗ И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ
Апрель–июнь 2026. Т. 33, № 2. С. 97–114

УДК 519.17 DOI: 10.33048/daio.2026.33.858

ЧЕТЫРЁХМЕРНЫЙ БУЛЕВ КУБ НЕ 3-ПРЕДСТАВИМ

А. В. Пяткин[1, a], *А. И. Сафарова*[2, b]

[1] Институт математики им. С. Л. Соболева,
пр. Акад. Коптюга, 4, 630090 Новосибирск, Россия
[2] Новосибирский гос. университет,
ул. Пирогова, 2, 630090 Новосибирск, Россия
E-mail: [a] artem@math.nsc.ru, [b] a.safarova1@g.nsu.ru

Аннотация. Граф называется 3-представимым, если существует 3-униформное слово, буквами которого являются вершины графа, причём две вершины смежны тогда и только тогда, когда соответствующие буквы чередуются в слове. В работе представлено компьютерное доказательство того, что четырёхмерный куб не 3-представим. Ил. 1, библиогр. 11.

Ключевые слова: представимый в виде слов граф, число представимости, четырёхмерный куб.

Введение

Пусть задан неориентируемый граф $G = (V, E)$ без петель и кратных рёбер. Говорим, что слово w над алфавитом V *представляет* граф G, если любые две вершины u и v смежны тогда и только тогда, когда соответствующие буквы чередуются в слове w (т. е. индуцированное буквами u и v подслово не содержит ни uu, ни vv). Граф называется *представимым в виде слова* (или просто *представимым*), если существует слово, которое его представляет. Это понятие введено в работе [1], и с тех пор по этой тематике опубликованы несколько десятков статей, включая обзор [2], а также книга [3].

В [1], в частности, доказано, что любой представимый граф можно представить k-униформным словом для некоторого k, т. е. таким словом, в котором каждая буква встречается ровно k раз. В этом случае говорим, что граф G *k-представим*, а минимальное такое k называется *числом представимости* графа G и обозначается через $\mathcal{R}(G)$. Очевидно, что класс 1-представимых графов состоит только из полных графов. Известно, что класс 2-представимых графов совпадает с классом графов пересечений хорд окружности [4]. При $k \geqslant 3$ отсутствует какая-либо

© А. В. Пяткин, А. И. Сафарова, 2026

характеризация k-представимых графов. Для $k = 3$ известно несколько классов 3-представимых графов: призмы [5], прямоугольные и цилиндрические решётки [6] и все представимые сплит-графы [7].

Граф называется *k-мерным кубом*, если его вершинами являются все k-мерные булевы векторы, и две вершины смежны тогда и только тогда, когда они различаются ровно в одной координате. Обозначим k-мерный куб через Q_k. В [8] доказано, что Q_k k-представим и высказана гипотеза, что $\mathcal{R}(Q_k) = k$. При $k \leqslant 2$ это очевидно, а при $k = 3$ следует из того, что граф, получающийся при удалении одной вершины 3-мерного куба, не является графом пересечения хорд [4]. В работе [9] эта гипотеза опровергнута с помощью асимптотической оценки

$$\mathcal{R}(Q_k) = O\Big(\frac{\log k}{\log\log k}\Big)$$

при достаточно большом k. Однако остался открытым вопрос определения минимального k, для которого k-мерный куб является $(k-1)$-представимым графом.

Отметим, что при определении числа представимости k для некоторого графа или класса графов наибольшую трудность представляет именно доказательство нижней оценки числа представимости, т. е. того факта, что он не $(k-1)$-представим. Имеется немного работ, где получены аналитические доказательства нижних оценок для числа представимости тех или иных графов. Так, в [10] доказана точная формула $\mathcal{R}(H_{n,n}) = \lceil n/2 \rceil$ при $n \geqslant 5$, где $H_{n,n}$ — граф-корона, который получается из полного двудольного графа удалением совершенного паросочетания. В [11] получены нижние оценки $\mathcal{R}(Q_k) \geqslant \Omega(\log\log k)$ для k-мерного куба и $\mathcal{R}(G) \geqslant n/2 - o(n)$ для почти любого двудольного графа G, имеющего по n вершин в каждой доле.

Основным результатом статьи является

Теорема 1. *Граф 4-мерного куба Q_4 не 3-представим.*

Доказательство этой теоремы, основанное на нетривиальном компьютерном переборе, представлено в следующем разделе.

1. Основной результат

Сначала введём некоторые обозначения, которые будут использоваться далее. Очевидно, что любое слово w представляет однозначно определённый граф, обозначаемый через $G(w)$. Пусть x — буква k-униформного слова w. Тогда через x_i, $i = 1, \ldots, k$ обозначим i-ю копию буквы x при чтении слова w слева направо. Запись $x_i < y_j$ означает, что i-я копия буквы x встречается в слове w раньше (левее), чем j-я копия буквы y.

Тогда условие чередования букв x и y в слове w можно записать в следующем виде:

$$(x_i < y_i < x_{i+1} \text{ для всех } i) \quad \text{или} \quad (y_i < x_i < y_{i+1} \text{ для всех } i).$$

Для произвольного подмножества букв $A \subset V$ через $w|_A$ обозначим подслово w, порождённое буквами из A. Иначе говоря, слово $w|_A$ получается из w вычёркиванием всех букв из $V \setminus A$.

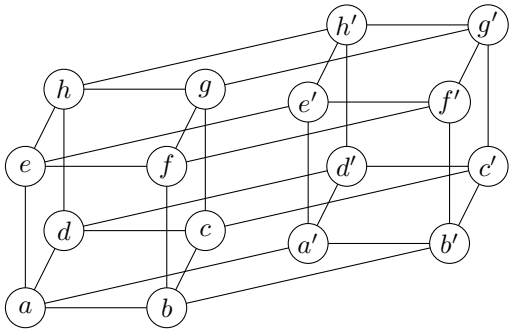


Рис. 1. Четырёхмерный куб Q_4

Обозначим так, как показано на рис. 1, вершины четырёхмерного куба Q_4. Для вершины $x \in V(Q_4)$ обозначим через $N(x)$ множество её соседей и положим $N[x] = x \cup N(x)$. Заметим, что если $y \in N(x)$, то между любыми x_i и x_{i+1} должна встречаться ровно одна копия буквы y. Докажем простое, но полезное

Утверждение 1. *Пусть w — k-униформное слово, содержащее букву x, и $w_i = x_i \dots x_{i+1}$ — подслово слова w, заключённое между вхождениями x_i и x_{i+1}, $i = 1, \dots, k-1$. В этом случае вершины x и y смежны в графе $G(w)$ тогда и только тогда, когда y присутствует в каждом из слов w_i ровно один раз.*

Доказательство. Если буква y встречается в w_i более одного раза, то подслово $w|_{\{x,y\}}$ содержит yy. Если буква y не встречается в w_i, то $w|_{\{x,y\}}$ содержит xx. В обоих случаях x и y несмежны. Если каждое w_i содержит по одной копии буквы y, то оставшаяся k-я копия y в слове w должна находиться перед x_1 или после x_k. Тогда $w|_{\{x,y\}} = (yx)^k$ или $w|_{\{x,y\}} = (xy)^k$ соответственно, а значит, вершины x и y смежны в $G(w)$. Утверждение 1 доказано.

Заметим, что буквы x и y чередуются в слове w тогда и только тогда, когда они чередуются в его подслове $x_1 \dots x_k$.

1.1. Поиск в глубину. Наша задача — показать, что не существует 3-униформного слова w над алфавитом $\{a, b, \dots, h, a', \dots, h'\}$, представляющего Q_4. Отметим, что всего имеется

$$\frac{48!}{6^{16}} \approx 4 \cdot 10^{48}$$

3-униформных слов длины 48, что делает полный их перебор совершенно нереальным (под полным перебором понимается алгоритм поиска в глубину, при котором на каждом шаге в слово добавляется одна из букв алфавита, которая ранее встречалась в слове менее трёх раз). Однако есть несколько ограничений и соображений, позволяющих исключить некоторые слова и варианты ветвления как очевидно не подходящие или необязательные для рассмотрения.

1. Поскольку при циклическом сдвиге (а также при развороте задом наперёд) 3-униформного слова представляемый им граф не изменяется, можно считать, что искомое слово начинается с a.

2. Из определения смежности следует, что если в слове появилась x_i, то x_{i+1} не может появиться раньше, чем встретятся все буквы из $N(x)$.

3. Если в строящемся слове w' появилась буква x_3, то по утверждению 1 можно вычислить множество N всех соседей буквы x в графе $G(w)$ для любого слова w, префиксом которого является w'. Если $N \neq N(x)$, то очевидно, что дальнейшее ветвление можно не рассматривать.

Выполнение п. 2 гарантирует, что каждая вершина x останется смежной со всеми своими соседями, а п. 3 — что у неё не появится лишних соседей.

Поиск 3-униформного слова, представляющего граф Q_4, начнём следующим алгоритмом поиска в глубину с учётом ограничений 1–3. В алгоритме используются такие обозначения: word — текущее строящееся слово, $\mathrm{cnt}[x]$ — число неиспользованных в слове word копий буквы x, $\mathrm{forbid}[x]$ — список букв, которые должны появиться в слове word согласно ограничению 2, прежде чем можно будет добавить букву x, wordList — искомый список слов, представляющих граф G.

Алгоритм 1. Поиск в глубину с ограничениями

Вход: граф G, заданный в виде множества вершин V (оно же является алфавитом искомого слова) и множеств соседей $N_G(x)$ для каждой вершины x, число представимости k.

Выход: список найденных слов wordList.

Шаг 1. Положить

$$\mathrm{word} = a, \quad \mathrm{cnt}[a] = k - 1, \quad \mathrm{forbid}[a] = N_G(a), \quad \mathrm{wordList} = \varnothing.$$

Для каждой буквы $x \neq a$ положить $\mathrm{cnt}[x] = k$, $\mathrm{forbid}[x] = \varnothing$.

Шаг 2. Запустить процедуру search(word, cnt, forbid).

Процедура search(word, cnt, forbid)

Шаг 1. Если $|\text{word}| = k|V|$, то добавить word в wordList и перейти на шаг 7.

Шаг 2. Для каждой буквы $x \in V$ проверить условия $\text{forbid}[x] = \varnothing$ и $\text{cnt}[x] > 0$. Если выполняются, то перейти на шаг 3, иначе — на шаг 8.

Шаг 3. Положить

$$\text{word} = \text{word} + x, \quad \text{cnt}[x] = \text{cnt}[x] - 1, \quad \text{forbid}[x] = N_G(x).$$

Для каждой буквы $y \in N_G(x)$ положить $\text{forbid}[y] = \text{forbid}[y] - x$.

Шаг 4. Если $\text{cnt}[x] > 0$, то перейти на шаг 6, иначе — на шаг 5.

Шаг 5. Для каждой буквы $y \in V \setminus N_G[x]$ проверить, чередуются ли буквы x и y в текущем слове word. Если ответ «да» хотя бы для одной буквы y, то перейти на шаг 7, иначе — на шаг 6.

Шаг 6. Запустить процедуру search(word, cnt, forbid).

Шаг 7. Удалить последнюю букву x из word. Положить $\text{cnt}[x] = \text{cnt}[x] + 1$, $\text{forbid}[x] = \varnothing$. Для всех $y \in N_G(x)$, удовлетворяющих условию $\text{cnt}[y] < k$, положить $\text{forbid}[y] = \text{forbid}[y] + x$.

Шаг 8. Перейти к следующей букве (из цикла на шаге 2).

Отметим, что выполнение ограничения 3 обеспечивается на шаге 5 рекурсивной процедуры поиска. Это гарантирует корректность алгоритма 1, поскольку для каждого слова w, помещённого в список на шаге 1, множество соседей каждой вершины x в графе $G(w)$ совпадает с $N_G(x)$.

При попытке применить алгоритм 1 к графу Q_4 выяснилось, что за 10 часов работы компьютера меняется лишь 16-й символ префикса искомого слова. Поскольку число префиксов длины 15, удовлетворяющих условиям 1–3, можно грубо оценить снизу как $15! \approx 10^{12}$, очевидно, что алгоритм 1 не сможет доказать теорему 1 за приемлемое время. Тем не менее, с помощью этого алгоритма удалось получить списки всех слов, начинающихся с буквы a, которые представляют графы Q_2 и Q_3. Для Q_2 список состоит из 36 слов длины 12, а для Q_3 — из 82 944 слов длины 24. Эти списки сыграли важную роль в получении компьютерного доказательства теоремы 1, приведённого в п. 1.3.

1.2. Эквивалентность и скрещивание слов. Пусть заданы слова w_1 и w_2 над непересекающимися алфавитами A и B соответственно. Говорим, что слово w над алфавитом $A \cup B$ получено *скрещиванием* слов w_1 и w_2, если $w|_A = w_1$ и $w|_B = w_2$. Смысл этого понятия заключается в следующем. Пусть множество вершин графа $G(w)$, представленного словом w, разбито на непересекающиеся подмножества A и B.

Тогда очевидно, что индуцированные подграфы $G[A]$ и $G[B]$ представлены словами $w_1 = w|_A$ и $w_2 = w|_B$ соответственно, а само слово w получено скрещиванием слов w_1 и w_2. Это порождает следующую идею доказательства теоремы 1: попробовать угадать слова, представляющие некоторые подграфы A и B, а затем построить все их возможные скрещивания. Однако, для реализации этой идеи следует существенно сократить число слов, рассматриваемых для скрещивания.

Разворотом слова $w = x_1x_2\dots x_n$ называется слово $x_1x_nx_{n-1}\dots x_2$, получающееся из w с помощью инверсии и циклического сдвига на одну позицию вправо. Назовём два k-униформных слова *тривиально эквивалентными*, если одно из них переводится в другое с помощью операций разворота и/или циклического сдвига. Очевидно, что два тривиально эквивалентных слова представляют один и тот же граф, а среди таких слов, начинающихся с буквы a, каждый класс эквивалентности состоит из $2k$ слов. Биекция $F\colon V \to V$ называется *автоморфизмом* графа G, если она сохраняет смежность и несмежность вершин, т. е. $xy \in E$ тогда и только тогда, когда $F(x)F(y) \in E$. Пусть два слова $w_1 = x_1x_2\dots x_n$ и w_2 над алфавитом V начинаются с буквы a и представляют граф $G = (V, E)$. Говорим, что они *эквивалентны*, если существует такой автоморфизм F графа G, что слово $F(x_1)F(x_2)\dots F(x_n)$ тривиально эквивалентно слову w_2. Поскольку автоморфизм — это просто переименование вершин графа, два эквивалентных слова представляют один и тот же граф. Для иллюстрации вышеприведённой концепции докажем аналитически

Утверждение 2. *Существует* 36 3*-униформных слов, начинающихся с буквы* a *и представляющих граф* Q_2 (4*-цикл*)*, среди которых ровно два неэквивалентных.*

Доказательство. Сначала докажем вторую часть утверждения 2. Пусть 3-униформное слово $w = aI_1aI_2aI_3$ представляет 4-цикл $abcd$. Тогда каждая из букв b и d встречается в каждом из подслов I_j ровно один раз, а буква c встречается в каком-то из I_j более одного раза. Ясно, что никакое из I_j не может содержать три копии c, так как тогда она не будет смежна с b и d. С помощью операций циклического сдвига и/или разворота можно добиться того, что интервалы I_1, I_2 и I_3 содержат 0, 1 и 2 копии буквы c соответственно. Можно считать, что в I_1 буквы b и d идут в порядке bd (иначе применим автоморфизм, меняющий в цикле вершины b и d местами). Тогда $w = abdac\{b,d\}ac\{b,d\}c$. Поскольку b и d несмежны, хотя бы в одном из множеств $\{b,d\}$ буквы должны идти в порядке db. В итоге имеем три возможных неэквивалентных слова:

$$w_1 = abdacbdacdbc, \quad w_2 = abdacdbacbdc, \quad w_3 = abdacdbacdbc.$$

Заметим, однако, что слово w_3 эквивалентно w_1: применим автоморфизм, меняющий в цикле вершины a и c местами, и прочитаем полученное слово задом наперёд. Для доказательства неэквивалентности слов w_1 и w_2 воспользуемся следующим приёмом. Назовём *следом* буквы x в слове w интервал между двумя её копиями, который содержит только двух её соседей. Нетрудно видеть, что каждая из букв имеет ровно один след в каждом из слов w_1 и w_2 (следы обозначены скобками):

$$w_1 = (abd)acbd(ac)(dbc), \quad w_2 = a(bdac)db(acbd)c.$$

При этом в слове w_1 все 4 следа идут подряд, тогда как в слове w_2 они разбиты на две отдельные группы по 2 следа. Поскольку операции сдвига, разворота и автоморфизма никак не влияют на взаимное расположение следов, слова w_1 и w_2 неэквивалентны.

Для доказательства первой части утверждения 2 найдём мощности классов эквивалентности слов w_1 и w_2. Заметим, что существует ровно 8 автоморфизмов графа Q_2 (4 способа выбрать букву a и 2 способа выбрать её соседей). Каждый из автоморфизмов порождает 6 тривиально эквивалентных слов, начинающихся с буквы a и представляющих Q_2, что даёт верхнюю оценку 48 на размер класса эквивалентности, но при этом некоторые автоморфизмы могут перевести слово в тривиально эквивалентное ему самому. Ясно, что каждый такой автоморфизм сокращает размер класса эквивалентности вдвое. Для слова w_1 таким автоморфизмом является $F(a) = d$, $F(b) = c$, $F(c) = b$, $F(d) = a$ (при прочтении полученного слова задом наперёд начиная с 10-й буквы получится w_1). У слова w_2 имеется два таких автоморфизма. Первый из них тот же, что и для w_1 (только читать слово задом наперёд надо с 6-й буквы), а второй $F(a) = c$, $F(b) = d$, $F(c) = a$, $F(d) = b$ (слово читается задом наперёд). Непосредственной проверкой можно убедиться, что остальные автоморфизмы не переводят эти слова в тривиально эквивалентные себе. Таким образом, имеется 24 слова, эквивалентных w_1, и 12 слов, эквивалентных w_2, всего 36 3-униформных слов, начинающихся с буквы a и представляющих Q_2. Утверждение 2 доказано.

Для получения аналогичного утверждения относительно графа Q_3 пришлось применить компьютерный метод и действовать в обратном порядке: сначала с помощью алгоритма 1 получен список fullList из 82 944 3-униформных слов, начинающихся с буквы a и представляющих Q_3, а потом к нему применён следующий алгоритм, последовательно удаляющий из списка эквивалентные слова и выдающий на выходе список неэквивалентных слов unList. Для автоморфизма F графа $G = (V, E)$ и слова word над алфавитом V через $F(\text{word})$ обозначается слово, полученное заменой каждой буквы x слова word на $F(x)$.

Алгоритм 2. Получение списка неэквивалентных слов

Вход: список fullList k-униформных слов, начинающихся с буквы a и представляющих граф G. Список автоморфизмов $\operatorname{Aut} G$ графа G.

Выход: список попарно неэквивалентных слов unList.

Шаг 1. Положить unList $= \varnothing$.

Шаг 2. Выбрать текущее слово word из списка fullList и добавить его в unList.

Шаг 3. Для каждого автоморфизма $F \in \operatorname{Aut} G$ найти все $2k$ слов, тривиально эквивалентных $F(\text{word})$, и удалить их из списка fullList.

Шаг 4. Если fullList $\neq \varnothing$, то перейти на шаг 2.

Для построения списка автоморфизмов $\operatorname{Aut} G$ куба Q_3 используется следующая процедура. Сначала выбирается образ $x = F(a)$ буквы a (8 вариантов). Затем рассматриваются все возможные варианты назначения образов букв b, d и e среди соседей $N(x)$ буквы x в кубе Q_3 (6 вариантов). Тогда образы букв c, f и h определяются однозначно как попарные общие соседи букв $F(b), F(d)$ и $F(e)$, а последняя буква будет образом g. Всего получается 48 автоморфизмов.

Применяя алгоритм 1 к кубу Q_3 и подавая найденный им список слов в качестве входа алгоритма 2, получаем машинное доказательство следующего утверждения.

Утверждение 3. *Существует* 82 944 3*-униформных слов, начинающихся с буквы* a *и представляющих граф* Q_3*, среди которых ровно* 308 *попарно неэквивалентных.*

Список попарно неэквивалентных слов из утверждения 3 приводится в прил. А.

Далее выпишем алгоритм скрещивания двух слов, играющий ключевую роль в доказательстве теоремы 1. Он работает очень схоже с алгоритмом 1, но на каждом шаге выбор добавляемой буквы ограничен лишь двумя вариантами, а именно, текущими буквами в скрещиваемых словах (их позиции указаны переменными p_1 и p_2). Остальные переменные (cnt, forbid, word и wordList) определены так же, как и в алгоритме 1.

Алгоритм 3. Скрещивание двух слов

Вход: два k-униформных слова w_1 и w_2 над алфавитами A и B, $A \cap B = \varnothing$, причём w_1 начинается с буквы $a \in A$. Граф G с множеством вершин $V = A \cup B$, чьи подграфы $G[A]$ и $G[B]$ представлены словами w_1 и w_2 соответственно.

Выход: список найденных скрещиваний wordList.

Шаг 1. Положить

$$\text{word} = a, \quad \text{cnt}[a] = k-1, \quad \text{forbid}[a] = N_G(a), \quad \text{wordList} = \varnothing.$$

Для каждой буквы $x \neq a$ положить $\text{cnt}[x] = k$, $\text{forbid}[x] = \varnothing$. Установить счётчики $p_1 = 2$ и $p_2 = 1$.

Шаг 2. Запустить процедуру mix(word, cnt, forbid, p_1, p_2).

Процедура mix(word, cnt, forbid, p_1, p_2)

Шаг 1. Если $|\text{word}| = k|V|$, то добавить word в wordList и перейти на шаг 8.

Шаг 2. Задать множество $\text{next} = \{w_1(p_1), w_2(p_2)\}$.

Шаг 3. Для каждой буквы $x \in \text{next}$ проверить условия $\text{forbid}[x] = \varnothing$ и $\text{cnt}[x] > 0$. Если выполняются, то перейти на шаг 4, иначе — на шаг 9.

Шаг 4. Положить

$$\text{word} = \text{word} + x, \quad \text{cnt}[x] = \text{cnt}[x] - 1, \quad \text{forbid}[x] = N_G(x), \quad p_i = p_i + 1,$$

где $i = 1$, если $x \in A$, и $i = 2$, если $x \in B$. Для каждой буквы $y \in N_G(x)$ положить $\text{forbid}[y] = \text{forbid}[y] - x$.

Шаг 5. Если $\text{cnt}[x] > 0$, то перейти на шаг 7, иначе — на шаг 6.

Шаг 6. Для каждой буквы $y \in V \setminus N_G[x]$ проверить, чередуются ли буквы x и y в текущем слове word. Если ответ «да» хотя бы для одной буквы y, то перейти на шаг 8, иначе — на шаг 7.

Шаг 7. Запустить процедуру mix(word, cnt, forbid, p_1, p_2).

Шаг 8. Удалить последнюю букву x из word. Положить $\text{cnt}[x] = \text{cnt}[x] + 1$, $\text{forbid}[x] = \varnothing$, $p_i = p_i - 1$. Для всех $y \in N_G(x)$, удовлетворяющих условию $\text{cnt}[y] < k$, положить $\text{forbid}[y] = \text{forbid}[y] + x$.

Шаг 9. Перейти к следующей букве (из цикла на шаге 3).

Отметим, что алгоритм 3 работает гораздо быстрее алгоритма 1, поскольку на каждом шаге ветвления имеется не более двух вариантов, тогда как в алгоритме 1 их число может доходить до 15. Также оказалось, что скрещивание длинного слова с коротким происходит значительно быстрее, чем двух слов средней длины. Это предопределило технику поиска гипотетического слова w, представляющего Q_4, описанную в следующем пункте.

1.3. Компьютерное доказательство теоремы 1. С помощью алгоритма 3 выполнены последовательно следующие шаги.

1. Построены все возможные скрещивания 308 неэквивалентных слов из утверждения 3 (см. прил. А) со всеми 144 словами, представляющими цикл $a'b'c'd'$ (см. прил. Б). В результате получился список list_1 из 12 590 580 слов на букву a, представляющих $Q_4 \setminus \{e', f', g', h'\}$.

2. Построены все возможные скрещивания слов из списка list_1 со списком из двух возможных слов $\{e'f'e'f'e'f',\ f'e'f'e'f'e'\}$, представляющих ребро $e'f'$. В результате получился список list_2 из 142 597 981 слов на букву a, представляющих $Q_4 \setminus \{g', h'\}$.

3. Построены все возможные скрещивания слов из списка list_2 с единственным словом $h'h'h'$, представляющих одновершинный подграф h'. В результате получился список list_3 из 9 673 917 слов на букву a, представляющих $Q_4 \setminus \{g'\}$.

4. Наконец, построены все возможные скрещивания слов из списка list_3 с единственным словом $g'g'g'$, представляющим одновершинный подграф g'. Итоговый список оказался пустым.

Отсюда следует, что граф Q_4 не 3-представим. Действительно, предположим, что найдётся 3-униформное слово w, начинающееся с буквы a, которое его представляет. Пусть $A = \{a, b, \ldots, h\}$. Тогда подслово $w_1 = w|_A$ представляет подграф Q_3. По утверждению 3 найдётся такой автоморфизм F графа Q_3, что $F(w_1)$ тривиально эквивалентен одному из 308 слов, приведённых в прил. А. Заметим, что автоморфизм F подграфа Q_3 однозначно расширяется до автоморфизма F' графа Q_4, поскольку каждая вершина из A имеет ровно одного соседа в $\{a', b', \ldots, h'\}$, и наоборот. Тогда слово $F'(w)$ можно с помощью операций циклического сдвига и/или разворота превратить в такое слово w', начинающееся на a, что подслово $w'|_A$ содержится в списке из прил. А. Пусть $B = \{a', b', c', d'\}$. Тогда подслово $w'|_B$, представляющее 4-цикл $a'b'c'd'$, должно присутствовать в списке из прил. Б. Значит, подслово $w'|_{A\cup B}$ лежит в списке list_1. Аналогично подслово $w'|_{V\setminus\{g',h'\}}$ должно лежать в списке list_2, а подслово $w'_1 = w'|_{V\setminus\{g'\}}$ — в списке list_3, но тогда слово w' должно получаться скрещиванием w'_1 со словом $g'g'g'$, а таких слов нет. Теорема 1 доказана.

Компьютерное доказательство теоремы 1 заняло порядка 300 часов машинного времени. Авторы располагают списками $\mathrm{list}_1, \mathrm{list}_2, \mathrm{list}_3$, общий объём которых составляет свыше 7,2 ГБ.

Заключение

В работе исследована задача определения числа представимости четырёхмерного куба. С помощью нетривиального компьютерного поиска показано, что данное число равно 4. Проблема поиска полностью аналитического доказательства этого факта остаётся открытой. Также открыта более общая проблема, сформулированная в [9]: определить минимальное число k, для которого k-мерный куб будет $(k-1)$-представимым графом.

Финансирование работы

Исследование выполнено в рамках государственного задания Института математики им. С. Л. Соболева (проект № FWNF–2026–0021). Дополнительных грантов на проведение или руководство этим исследованием получено не было.

Конфликт интересов

Авторы заявляют, что у них нет конфликта интересов.

Литература

1. **Kitaev S. V., Pyatkin A. V.** On representable graphs // J. Autom. Lang. Comb. 2008. V. 13, No. 1. P. 45–54.
2. **Китаев С. В., Пяткин А. В.** Графы, представимые в виде слов. Обзор результатов // Дискрет. анализ и исслед. операций. 2018. Т. 25, № 2. С. 19–53.
3. **Kitaev S. V., Lozin V. V.** Words and graphs. Cham: Springer, 2015. 264 p.
4. **Bouchet A.** Circle graph obstructions // J. Comb. Theory, Ser. B. 1994. V. 60, No. 1. P. 107–144.
5. **Kitaev S. V.** On graphs with representation number 3 // J. Autom. Lang. Comb. 2013. V. 18, No. 2. P. 97–112.
6. **Alshammari N. S., Kitaev S. V., Pyatkin A. V.** On the representation number of grid graphs and cylindric grid graphs. Ithaca, NY, 2025. 10 p. (e-Print Archive / Cornell Univ.; arXiv:2507.16469). DOI: `10.48550/arXiv.2507.16469`.
7. **Dwary T., Mozhui K., Krishna K. V.** Representation number of word-representable split graphs. Ithaca, NY, 2025. 14 p. (e-Print Archive / Cornell Univ.; arXiv:2502.00872). DOI: `10.48550/arXiv.2502.00872`.
8. **Broere B., Zantema H.** The k-dimensional cube is k-representable // J. Autom. Lang. Comb. 2019. V. 24, No. 1. P. 3–12.
9. **Hefty Z., Horn P., Muir C., Owens A.** Word-representable graphs: Orientations, posets, and bounds // Electron. J. Comb. 2024. V. 31, No. 4. Article ID P4.2. 26 p.
10. **Glen M., Kitaev S. V., Pyatkin A. V.** On the representation number of a crown graph // Discrete Appl. Math. 2018. V. 244. P. 89–93.
11. **Hefty Z., Horn P., Muir C., Owens A.** Word-representation numbers of graphs: Bottlenecks and bounds // J. Comb. Theory, Ser. A. 2026. V. 223. Article ID 106215. 25 p.

Приложение A. Попарно неэквивалентные 3-униформные слова, начинающиеся на a и представляющие Q_3

1) $abdceafbhedagchdfebgfhcg$
2) $abdceafbhedagchdfebghfcg$
3) $abdceafbhedagchdfegbfhcg$
4) $abdceafbhedagchdfegbhfcg$
5) $abdceafbhedagchdfeghbfcg$
6) $abdceafbhedagchdfgebfhcg$

7) $abdceafbhedagchdfgebhfcg$
8) $abdceafbhedagchdfgehbfcg$
9) $abdceafbhedaghcdfebgfchg$
10) $abdceafbhedaghcdfebgfhcg$
11) $abdceafbhedaghcdfebghfcg$
12) $abdceafbhedaghcdfegbfchg$
13) $abdceafbhedaghcdfegbfhcg$
14) $abdceafbhedaghcdfegbhfcg$
15) $abdceafbhedaghcdfeghbfcg$
16) $abdceafbhedaghcdfgebfchg$
17) $abdceafbhedaghcdfgebfhcg$
18) $abdceafbhedaghcdfgebhfcg$
19) $abdceafbhedaghcdfgehbfcg$
20) $abdceafbhedgachdfebgfhcg$
21) $abdceafbhedgachdfebghfcg$
22) $abdceafbhedgachdfegbfhcg$
23) $abdceafbhedgachdfegbhfcg$
24) $abdceafbhedgachdfeghbfcg$
25) $abdceafbhedgachdfgebfhcg$
26) $abdceafbhedgachdfgebhfcg$
27) $abdceafbhedgachdfgehbfcg$
28) $abdceafbhedgahcdfebgfchg$
29) $abdceafbhedgahcdfebgfhcg$
30) $abdceafbhedgahcdfebghfcg$
31) $abdceafbhedgahcdfegbfchg$
32) $abdceafbhedgahcdfegbfhcg$
33) $abdceafbhedgahcdfegbhfcg$
34) $abdceafbhedgahcdfeghbfcg$
35) $abdceafbhedgahcdfgebfchg$
36) $abdceafbhedgahcdfgebfhcg$
37) $abdceafbhedgahcdfgebhfcg$
38) $abdceafbhedgahcdfgehbfcg$
39) $abdceafbhedghacdfebgfchg$
40) $abdceafbhedghacdfebgfhcg$
41) $abdceafbhedghacdfebghfcg$
42) $abdceafbhedghacdfegbfchg$
43) $abdceafbhedghacdfegbfhcg$
44) $abdceafbhedghacdfegbhfcg$
45) $abdceafbhedghacdfeghbfcg$
46) $abdceafbhedghacdfgebfchg$
47) $abdceafbhedghacdfgebfhcg$
48) $abdceafbhedghacdfgebhfcg$
49) $abdceafbhedghacdfgehbfcg$
50) $abdceafbhegdachdfebgfhcg$
51) $abdceafbhegdachdfebghfcg$
52) $abdceafbhegdachdfegbfhcg$
53) $abdceafbhegdachdfegbhfcg$
54) $abdceafbhegdachdfeghbfcg$
55) $abdceafbhegdachdfgebfhcg$
56) $abdceafbhegdachdfgebhfcg$
57) $abdceafbhegdachdfgehbfcg$
58) $abdceafbhegdahcdfebgfchg$
59) $abdceafbhegdahcdfebgfhcg$
60) $abdceafbhegdahcdfebghfcg$
61) $abdceafbhegdahcdfegbfchg$
62) $abdceafbhegdahcdfegbfhcg$
63) $abdceafbhegdahcdfegbhfcg$
64) $abdceafbhegdahcdfeghbfcg$
65) $abdceafbhegdahcdfgebfchg$
66) $abdceafbhegdahcdfgebfhcg$
67) $abdceafbhegdahcdfgebhfcg$
68) $abdceafbhegdahcdfgehbfcg$
69) $abdceafbhegdhacdfebgfchg$
70) $abdceafbhegdhacdfebgfhcg$
71) $abdceafbhegdhacdfebghfcg$
72) $abdceafbhegdhacdfegbfchg$
73) $abdceafbhegdhacdfegbfhcg$
74) $abdceafbhegdhacdfegbhfcg$
75) $abdceafbhegdhacdfeghbfcg$
76) $abdceafbhegdhacdfgebfchg$
77) $abdceafbhegdhacdfgebfhcg$
78) $abdceafbhegdhacdfgebhfcg$
79) $abdceafbhegdhacdfgehbfcg$
80) $abdceafbhegfdachdebghcfg$
81) $abdceafbhegfdachdebghfcg$
82) $abdceafbhegfdachdegbhcfg$
83) $abdceafbhegfdachdegbhfcg$
84) $abdceafbhegfdachdeghbfcg$
85) $abdceafbhegfdachdgebhcfg$
86) $abdceafbhegfdachdgebhfcg$
87) $abdceafbhegfdachdgehbfcg$
88) $abdceafbhegfdachedbghcfg$
89) $abdceafbhegfdachedbghfcg$
90) $abdceafbhegfdachedgbhcfg$

91) $abdceafbhegfdachedgbhfcg$
92) $abdceafbhegfdachedghbfcg$
93) $abdceafbhegfdahcdebgchfg$
94) $abdceafbhegfdahcdebghcfg$
95) $abdceafbhegfdahcdebghfcg$
96) $abdceafbhegfdahcdegbchfg$
97) $abdceafbhegfdahcdegbhcfg$
98) $abdceafbhegfdahcdegbhfcg$
99) $abdceafbhegfdahcdeghbfcg$
100) $abdceafbhegfdahcdgebchfg$
101) $abdceafbhegfdahcdgebhcfg$
102) $abdceafbhegfdahcdgebhfcg$
103) $abdceafbhegfdahcdgehbfcg$
104) $abdceafbhegfdahcedbgchfg$
105) $abdceafbhegfdahcedbghcfg$
106) $abdceafbhegfdahcedbghfcg$
107) $abdceafbhegfdahcedgbchfg$
108) $abdceafbhegfdahcedgbhcfg$
109) $abdceafbhegfdahcedgbhfcg$
110) $abdceafbhegfdahcedghbfcg$
111) $abdceafbhegfdahecdbgchfg$
112) $abdceafbhegfdahecdbghcfg$
113) $abdceafbhegfdahecdbghfcg$
114) $abdceafbhegfdahecdgbchfg$
115) $abdceafbhegfdahecdgbhcfg$
116) $abdceafbhegfdahecdgbhfcg$
117) $abdceafbhegfdahecdghbfcg$
118) $abdceafbhegfdhacdebgchfg$
119) $abdceafbhegfdhacdebghcfg$
120) $abdceafbhegfdhacdebghfcg$
121) $abdceafbhegfdhacdegbchfg$
122) $abdceafbhegfdhacdegbhcfg$
123) $abdceafbhegfdhacdegbhfcg$
124) $abdceafbhegfdhacdeghbfcg$
125) $abdceafbhegfdhacdgebchfg$
126) $abdceafbhegfdhacdgebhcfg$
127) $abdceafbhegfdhacdgebhfcg$
128) $abdceafbhegfdhacdgehbfcg$
129) $abdceafbhegfdhacedbgchfg$
130) $abdceafbhegfdhacedbghcfg$
131) $abdceafbhegfdhacedbghfcg$
132) $abdceafbhegfdhacedgbchfg$
133) $abdceafbhegfdhacedgbhcfg$
134) $abdceafbhegfdhacedgbhfcg$
135) $abdceafbhegfdhacedghbfcg$
136) $abdceafbhgedachdfebgfhcg$
137) $abdceafbhgedachdfebghfcg$
138) $abdceafbhgedachdfegbfhcg$
139) $abdceafbhgedachdfegbhfcg$
140) $abdceafbhgedachdfeghbfcg$
141) $abdceafbhgedahcdfebgfchg$
142) $abdceafbhgedahcdfebgfhcg$
143) $abdceafbhgedahcdfebghfcg$
144) $abdceafbhgedahcdfegbfchg$
145) $abdceafbhgedahcdfegbfhcg$
146) $abdceafbhgedahcdfegbhfcg$
147) $abdceafbhgedahcdfeghbfcg$
148) $abdceafbhgedhacdfebgfchg$
149) $abdceafbhgedhacdfebgfhcg$
150) $abdceafbhgedhacdfebghfcg$
151) $abdceafbhgedhacdfegbfchg$
152) $abdceafbhgedhacdfegbfhcg$
153) $abdceafbhgedhacdfegbhfcg$
154) $abdceafbhgedhacdfeghbfcg$
155) $abdceafbhgefdachdebghcfg$
156) $abdceafbhgefdachdebghfcg$
157) $abdceafbhgefdachdegbhcfg$
158) $abdceafbhgefdachdegbhfcg$
159) $abdceafbhgefdachdeghbfcg$
160) $abdceafbhgefdachedbghcfg$
161) $abdceafbhgefdachedbghfcg$
162) $abdceafbhgefdachedgbhcfg$
163) $abdceafbhgefdachedgbhfcg$
164) $abdceafbhgefdachedghbfcg$
165) $abdceafbhgefdahcdebghcfg$
166) $abdceafbhgefdahcdebghfcg$
167) $abdceafbhgefdahcdegbhcfg$
168) $abdceafbhgefdahcdegbhfcg$
169) $abdceafbhgefdahcdeghbfcg$
170) $abdceafbhgefdahcedbghfcg$
171) $abdceafbhgefdahcedgbhcfg$
172) $abdceafbhgefdahcedgbhfcg$
173) $abdceafbhgefdahcedghbfcg$
174) $abdceafbhgefdhacdebghfcg$

175) $abdceafbhgefdhacdegbhfcg$
176) $abdceafbhgefdhacdeghbfcg$
177) $abdceafbhgefdhacedgbhfcg$
178) $abdceafbhgefdhacedghbfcg$
179) $abdceafbhgefdhaecdghbfcg$
180) $abdceafhbedaghcdfeghbfcg$
181) $abdceafhbedgachdfeghbfcg$
182) $abdceafhbedgahcdfeghbfcg$
183) $abdceafhbedghacdfeghbfcg$
184) $abdceafhbegdachdfeghbfcg$
185) $abdceafhbegdahcdfeghbfcg$
186) $abdceafhbegdhacdfeghbfcg$
187) $abdceafhbegfdachdeghbfcg$
188) $abdceafhbegfdachdgehbfcg$
189) $abdceafhbegfdachedghbfcg$
190) $abdceafhbegfdahcdeghbfcg$
191) $abdceafhbegfdahcdgehbfcg$
192) $abdceafhbegfdahcedghbfcg$
193) $abdceafhbegfdhacdeghbfcg$
194) $abdceafhbegfdhacdgehbfcg$
195) $abdceafhbegfdhacedghbfcg$
196) $abdceafhbgedachdfeghbfcg$
197) $abdceafhbgedahcdfeghbfcg$
198) $abdceafhbgedhacdfeghbfcg$
199) $abdceafhbgefdachdeghbfcg$
200) $abdceafhbgefdachedghbfcg$
201) $abdceafhbgefdahcdeghbfcg$
202) $abdceafhbgefdhacdeghbfcg$
203) $abdceafhbgefdhacedghbfcg$
204) $abdceafhdeghbacfbegfdchg$
205) $abdceafhdeghbacfbgefdchg$
206) $abdceafhdeghbacfebgfdchg$
207) $abdceafhdeghbafcbegfdchg$
208) $abdceafhdeghbafcbgefdchg$
209) $abdceafhdeghbafcebgfdchg$
210) $abdceafhdeghbafecbgfdchg$
211) $abdceafhdeghbfacbegfdchg$
212) $abdceafhdeghbfacbgefdchg$
213) $abdceafhdeghbfacebgfdchg$
214) $abdceafhdgehbacfbegfdchg$
215) $abdceafhdgehbacfebgfdchg$
216) $abdceafhdgehbafcebgfdchg$
217) $abdceafhdgehbafecbgfdchg$
218) $abdceafhdgehbfacbegfdchg$
219) $abdceafhdgehbfacebgfdchg$
220) $abdceafhebdgahcdfeghbfcg$
221) $abdceafhebdghacdfeghbfcg$
222) $abdceafhebgdachdfeghbfcg$
223) $abdceafhebgdahcdfeghbfcg$
224) $abdceafhebgdhacdfeghbfcg$
225) $abdceafhebgfdachdeghbfcg$
226) $abdceafhebgfdachdgehbfcg$
227) $abdceafhebgfdachedghbfcg$
228) $abdceafhebgfdahcdeghbfcg$
229) $abdceafhebgfdahcdgehbfcg$
230) $abdceafhebgfdhacdeghbfcg$
231) $abdceafhebgfdhacdgehbfcg$
232) $abdceafhedbaghcdfeghbfcg$
233) $abdceafhedbgahcdfeghbfcg$
234) $abdceafhedbghacdfeghbfcg$
235) $abdceafhedghbacdfbegfchg$
236) $abdceafhedghbacdfbegfhcg$
237) $abdceafhedghbacdfbeghfcg$
238) $abdceafhedghbacdfbgefchg$
239) $abdceafhedghbacdfbgefhcg$
240) $abdceafhedghbacdfbgehfcg$
241) $abdceafhedghbacdfebgfchg$
242) $abdceafhedghbacdfebgfhcg$
243) $abdceafhedghbacdfebghfcg$
244) $abdceafhedghbacfbegfdchg$
245) $abdceafhedghbacfbgefdchg$
246) $abdceafhedghbacfebgfdchg$
247) $abdceafhedghbafcbgefdchg$
248) $abdceafhedghbafcebgfdchg$
249) $abdceafhedghbafecbgfdchg$
250) $abdceafhedghbfacbgefdchg$
251) $abdceafhedghbfacebgfdchg$
252) $abdceafhegdhbacdfbegfhcg$
253) $abdceafhegdhbacdfbeghfcg$
254) $abdceafhegdhbacdfbgefchg$
255) $abdceafhegdhbacdfbgehfcg$
256) $abdceafhegdhbacdfebgfchg$
257) $abdceafhgedhbacdfbeghfcg$
258) $abdceafhgedhbacdfebgfchg$

259) *abdceafhgedhbacdfebghfcg*
260) *abdceahdfbeagcfhegdhbfcg*
261) *abdceahdfbeagcfhgedhbfcg*
262) *abdceahdfbeagfhcegdhbfcg*
263) *abdceahdfbeagfhcgedhbfcg*
264) *abdceahdfbeagfhecgdhbfcg*
265) *abdceahdfbegacfhegdhbfcg*
266) *abdceahdfbegacfhgedhbfcg*
267) *abdceahdfbegafchegdhbfcg*
268) *abdceahdfbegafchgedhbfcg*
269) *abdceahdfbegafhcegdhbfcg*
270) *abdceahdfbegafhcgedhbfcg*
271) *abdceahdfbegfachegdhbfcg*
272) *abdceahdfbegfachgedhbfcg*
273) *abdceahdfbegfahcgedhbfcg*
274) *abdceahdfbegfhacgedhbfcg*
275) *abdceahdfegbacfbhedghfcg*
276) *abdceahdfegbacfbhegdhfcg*
277) *abdceahdfegbacfbhgedhfcg*
278) *abdceahdfegbafcbhedghcfg*
279) *abdceahdfegbafcbhedghfcg*
280) *abdceahdfegbafcbhegdhcfg*
281) *abdceahdfegbafcbhegdhfcg*
282) *abdceahdfegbafcbhgedhcfg*
283) *abdceahdfegbfacbhedghcfg*
284) *abdceahdfegbfacbhedghfcg*
285) *abdceahdfegbfacbhegdhcfg*
286) *abdceahdfegbfacbhgedhcfg*
287) *abdceahdfgebafcbhedghcfg*
288) *abdceahdfgebafcbhegdhcfg*
289) *abdceahdfgebfacbhedghcfg*
290) *abdceahfdeghbacfbgefdchg*
291) *abdceahfdeghbacfebgfdchg*
292) *abdceahfdeghbafcbgefdchg*
293) *abdceahfdeghbafcebgfdchg*
294) *abdceahfdeghbfacbgefdchg*
295) *abdceahfdeghbfacebgfdchg*
296) *abdceahfdgehbacfebgfdchg*
297) *abdceahfdgehbfacebgfdchg*
298) *abdceahfedghbacdfbgefchg*
299) *abdceahfedghbacdfebgfchg*
300) *abdceahfedghbacfbgefdchg*
301) *abdceahfedghbacfebgfdchg*
302) *abdceahfedghbafcbgefdchg*
303) *abdceahfedghbfacbgefdchg*
304) *abdceahfgedhbacdfebgfchg*
305) *abdcefabhgedahcdfebgfchg*
306) *abdcefabhgedahcdfegbfchg*
307) *abdcefabhgedhacdfebgfchg*
308) *abdcefahedghbacdfebgfchg*

Приложение Б. Все 3-униформные слова, представляющие 4-цикл

1) *abdacbdacdbc*
2) *abdacbdcadbc*
3) *abdacdbacbdc*
4) *abdacdbacdbc*
5) *abdacdbcabdc*
6) *abdacdbcadbc*
7) *abdcabdacdbc*
8) *abdcadbacbdc*
9) *abdcadbacdbc*
10) *adbacbdacbdc*
11) *adbacbdacdbc*
12) *adbacbdcabdc*
13) *adbacbdcadbc*
14) *adbacdbacbdc*
15) *adbacdbcabdc*
16) *adbcabdacbdc*
17) *adbcabdacdbc*
18) *adbcadbacbdc*
19) *acbdacbdcadb*
20) *acbdacdbcabd*
21) *acbdacdbcadb*
22) *acbdcabdacdb*
23) *acbdcabdcadb*
24) *acbdcadbacbd*
25) *acbdcadbacdb*
26) *acbdcadbcabd*
27) *acbdcadbcadb*
28) *acdbacbdcabd*
29) *acdbacbdcadb*
30) *acdbacdbcabd*
31) *acdbcabdacbd*
32) *acdbcabdacdb*
33) *acdbcabdcabd*
34) *acdbcabdcadb*
35) *acdbcadbacbd*
36) *acdbcadbcabd*
37) *bacbdacbdcad*
38) *bacbdacdbcad*
39) *bacbdcabdacd*

40) *bacbdcabdcad*
41) *bacbdcadbacd*
42) *bacbdcadbcad*
43) *bacdbacbdcad*
44) *bacdbcabdacd*
45) *bacdbcabdcad*
46) *bdacbdacdbca*
47) *bdacbdcadbac*
48) *bdacbdcadbca*
49) *bdacdbacbdca*
50) *bdacdbacdbca*
51) *bdacdbcabdac*
52) *bdacdbcabdca*
53) *bdacdbcadbac*
54) *bdacdbcadbca*
55) *bdcabdacdbac*
56) *bdcabdacdbca*
57) *bdcabdcadbac*
58) *bdcadbacbdac*
59) *bdcadbacbdca*
60) *bdcadbacdbac*
61) *bdcadbacdbca*
62) *bdcadbcabdac*
63) *bdcadbcadbac*
64) *bcabdacbdacd*
65) *bcabdacbdcad*
66) *bcabdacdbacd*
67) *bcabdacdbcad*
68) *bcabdcabdacd*
69) *bcabdcadbacd*
70) *bcadbacbdacd*
71) *bcadbacbdcad*
72) *bcadbcabdacd*
73) *dacbdacdbcab*
74) *dacbdcadbacb*
75) *dacbdcadbcab*
76) *dacdbacbdcab*
77) *dacdbacdbcab*
78) *dacdbcabdacb*
79) *dacdbcabdcab*
80) *dacdbcadbacb*
81) *dacdbcadbcab*
82) *dbacbdacbdca*
83) *dbacbdacdbca*
84) *dbacbdcabdac*
85) *dbacbdcabdca*
86) *dbacbdcadbac*
87) *dbacbdcadbca*
88) *dbacdbacbdca*
89) *dbacdbcabdac*
90) *dbacdbcabdca*
91) *dbcabdacbdac*
92) *dbcabdacbdca*
93) *dbcabdacdbac*
94) *dbcabdacdbca*
95) *dbcabdcabdac*
96) *dbcabdcadbac*
97) *dbcadbacbdac*
98) *dbcadbacbdca*
99) *dbcadbcabdac*
100) *dcabdacdbacb*
101) *dcabdacdbcab*
102) *dcabdcadbacb*
103) *dcadbacbdacb*
104) *dcadbacbdcab*
105) *dcadbacdbacb*
106) *dcadbacdbcab*
107) *dcadbcabdacb*
108) *dcadbcadbacb*
109) *cabdacbdacdb*
110) *cabdacbdcadb*
111) *cabdacdbacbd*
112) *cabdacdbacdb*
113) *cabdacdbcabd*
114) *cabdacdbcadb*
115) *cabdcabdacdb*
116) *cabdcadbacbd*
117) *cabdcadbacdb*
118) *cadbacbdacbd*
119) *cadbacbdacdb*
120) *cadbacbdcabd*
121) *cadbacbdcadb*
122) *cadbacdbacbd*
123) *cadbacdbcabd*
124) *cadbcabdacbd*
125) *cadbcabdacdb*
126) *cadbcadbacbd*
127) *cbdacbdcadba*
128) *cbdacdbcabda*
129) *cbdacdbcadba*
130) *cbdcabdacdba*
131) *cbdcabdcadba*
132) *cbdcadbacbda*
133) *cbdcadbacdba*
134) *cbdcadbcabda*
135) *cbdcadbcadba*
136) *cdbacbdcabda*
137) *cdbacbdcadba*
138) *cdbacdbcabda*
139) *cdbcabdacbda*
140) *cdbcabdacdba*
141) *cdbcabdcabda*
142) *cdbcabdcadba*
143) *cdbcadbacbda*
144) *cdbcadbcabda*

Пяткин Артём Валерьевич
Сафарова Алиса Исламовна

Статья поступила
11 марта 2026 г.
После доработки —
10 апреля 2026 г.
Принята к публикации
22 апреля 2026 г.

DISKRETNYI ANALIZ I ISSLEDOVANIE OPERATSII
/DISCRETE ANALYSIS AND OPERATIONS RESEARCH/
April–June 2026. Vol. 33, No. 2. P. 97–114

UDC 519.17 DOI: 10.33048/daio.2026.33.858

THE FOUR-DIMENSIONAL BOOLEAN CUBE IS NOT 3-REPRESENTABLE

A. V. Pyatkin[1, a] and *A. I. Safarova*[2, b]

[1] Sobolev Institute of Mathematics,
4 Koptyug Avenue, 630090 Novosibirsk, Russia
[2] Novosibirsk State University,
2 Pirogov Street, 630090 Novosibirsk, Russia
E-mail: [a] artem@math.nsc.ru, [b] a.safarova1@g.nsu.ru

Abstract. A graph is 3-representable if there exists a 3-uniform word whose letters are the vertices of the graph, and two vertices are adjacent if and only if the corresponding letters alternate in the word. In this paper we provide a computer-aided proof of the fact that the 4-dimensional cube is not 3-representable. Illustr. 1, bibliogr. 11.

Keywords: word-representable graph, representation number, four-dimensional cube.

References

1. **S. V. Kitaev** and **A. V. Pyatkin,** On representable graphs, *J. Autom. Lang. Comb.* **13** (1), 45–54 (2008).
2. **S. V. Kitaev** and **A. V. Pyatkin,** Word-representable graphs: A survey, *Diskretn. Anal. Issled. Oper.* **25** (2), 19–53 (2018) [Russian] [*J. Appl. Ind. Math.* **12** (2), 278–296 (2018)].
3. **S. V. Kitaev** and **V. V. Lozin,** *Words and Graphs* (Springer, Cham, 2015).
4. **A. Bouchet,** Circle graph obstructions, *J. Comb. Theory, Ser. B*, **60** (1), 107–144 (1994).
5. **S. V. Kitaev,** On graphs with representation number 3, *J. Autom. Lang. Comb.* **18** (2), 97–112 (2013).
6. **N. S. Alshammari, S. V. Kitaev,** and **A. V. Pyatkin,** On the representation number of grid graphs and cylindric grid graphs (Ithaca, NY, 2025) (e-Print Archive / Cornell Univ., arXiv:2507.16469), DOI: 10.48550/arXiv.2507.16469.

English transl.: *Journal of Applied and Industrial Mathematics* **20** (2) (2026).

7. **T. Dwary, K. Mozhui,** and **K. V. Krishna,** Representation number of word-representable split graphs (Ithaca, NY, 2025) (e-Print Archive / Cornell Univ., arXiv:2502.00872), DOI: 10.48550/arXiv.2502.00872.
8. **B. Broere** and **H. Zantema,** The k-dimensional cube is k-representable, *J. Autom. Lang. Comb.* **24** (1), 3–12 (2019).
9. **Z. Hefty, P. Horn, C. Muir,** and **A. Owens,** Word-representable graphs: Orientations, posets, and bounds, *Electron. J. Comb.* **31** (4), ID P4.2 (2024).
10. **M. Glen, S. V. Kitaev,** and **A. V. Pyatkin,** On the representation number of a crown graph, *Discrete Appl. Math.* **244**, 89–93 (2018).
11. **Z. Hefty, P. Horn, C. Muir,** and **A. Owens,** Word-representation numbers of graphs: Bottlenecks and bounds, *J. Comb. Theory, Ser. A*, **223**, ID 106215 (2026).

Artem V. Pyatkin
Alisa I. Safarova

Received March 11, 2026
Revised April 10, 2026
Accepted April 22, 2026

ДИСКРЕТНЫЙ АНАЛИЗ И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ
Апрель–июнь 2026. Т. 33, № 2. С. 115–135

УДК 519.8 DOI: 10.33048/daio.2026.33.863

УСТОЙЧИВОСТЬ ВЕРШИННЫХ ПОКРЫТИЙ В ЗАДАЧЕ О ВЕЧНОМ ВЕРШИННОМ ПОКРЫТИИ

С. Ю. Утюпин

Институт математики им. С. Л. Соболева,
пр. Акад. Коптюга, 4, 630090 Новосибирск, Россия
E-mail: stepan.utyupin@gmail.com

Аннотация. Задача о вечном вершинном покрытии является вариантом задачи о вершинном покрытии графа и может рассматриваться как динамическая игра двух сторон (атакующего и защитника) с бесконечным числом шагов. Для построения стратегии защитника задача представляется в виде динамической игры, на каждом шаге которой взаимодействие противоборствующих сторон формализуется как двухуровневая задача математического программирования. В данной работе предложена процедура, позволяющая ответить на вопрос, является ли заданное вершинное покрытие вечным. Идея процедуры заключается в рекурсивной проверке устойчивости вершинных покрытий, полученных в результате решения задач защитника и построения дополнительного графа покрытий и переходов. Табл. 2, ил. 7, библиогр. 12.

Ключевые слова: вечное вершинное покрытие, защита рёбер графа, алгоритм проверки устойчивости, динамическая игра Штакельберга.

Введение

В задаче о вершинном покрытии задан граф $G = (V, E)$, целое положительное число k и требуется ответить на вопрос, существует ли подмножество $C \subseteq V$, называемое вершинным покрытием, размера не более k, содержащее по меньшей мере одну вершину каждого ребра из E. Данная задача широко исследована и является классической. Однако существует множество различных динамических её вариантов. Такие задачи — вечной защиты рёбер или вершин графа — появились в 1990-х гг. как формализация стратегии императора Константина по защите Римской империи [1]. В них рёбра или вершины графа защищаются от бесконечной последовательности атак. Защита осуществляется с помощью

© С. Ю. Утюпин, 2026

охранников, размещаемых в вершинах графа и способных перемещаться по его рёбрам после атаки. Говорят, что вершина (ребро) графа защищённое, если охранник находится в вершине или соседней (инцидентной ребру) вершине.

В настоящей работе исследуем вариант задачи, где объектами нападения являются рёбра графа. Такой вариант динамической задачи о вершинном покрытии, называемый задачей о вечном вершинном покрытии (eternal vertex cover problem), предложен в [2]. Эту задачу можно интерпретировать как бесконечную игру двух сторон — атакующего и защитника. Первоначально защитник размещает в вершинах заданного графа охранников так, что вершины, в которых размещены охранники, образуют вершинное покрытие. Далее на каждом шаге атакующий выбирает для атаки любое ребро, в одной из вершин которого нет охранника. В ответ защитник перемещает охранника из вершины атакованного ребра в другую вершину этого ребра. При этом защитник имеет право из любой вершины переместить охранника вдоль любого ребра в соседнюю вершину, если в ней не было охранника, либо охранник был перемещён из этой вершины. Также в одну вершину не могут быть перемещены несколько охранников, при этом один и тот же охранник не может быть перемещён более одного раза на одном шаге. Если реконструированное размещение охранников в вершинах образует вершинное покрытие, то начинается следующий шаг с той же последовательностью действий. Если же с помощью указанной процедуры не удаётся построить вершинное покрытие, т. е. хотя бы одно ребро окажется незащищённым, то атакующий побеждает и игра завершается. Победителем становится защитник, если он может защититься от бесконечной последовательности атак. Задача нахождения минимального числа охранников EVCN (eternal vertex cover number), образующих вершинное покрытие, для которого в рассматриваемой игре может быть построена выигрышная стратегия защитника, есть задача о вечном вершинном покрытии.

Ранее показано, что эта задача относится к классу NP-трудных [3], а также остаётся NP-трудной для двудольных графов [4]. Кроме того, задача о вечном вершинном покрытии широко исследована с точки зрения теории графов: многие работы либо предлагают алгоритмы для конкретных семейств графов, либо приводят оценки для общих или частных случаев. Так, предлагается квадратичный по времени алгоритм вычисления EVCN для хордальных графов [5], а для двусвязных хордальных графов предложен полиномиальный алгоритм [6]. Также представлены линейный алгоритм для кактусового графа [7], полиномиальные алгоритмы для цепных графов и кографов [8], а также линейный алгоритм для расщеплённых графов [8]. Для графов, построенных путём замены каждого ребра дерева произвольным элементарным двудольным графом

(или произвольной кликой), показано, что вычисление EVCN возможно за полиномиальное время [9]. Кроме того, доказано существование полиномиального алгоритма для двудольных графов [4]. Помимо этого показано, что для задачи о вечном вершинном покрытии существует 2-приближённый полиномиальный по времени алгоритм [3]. Предложена нижняя оценка для EVCN произвольного графа через мощность вершинного покрытия минимального размера, содержащего все точки сочленения данного графа [5]. Проведено исследование о связи числа EVCN с минимальным числом охранников, необходимых для бесконечной защиты вершин графа (задача о вечном доминировании) [10].

Задача проверки, является ли заданное вершинное покрытие вечным, несмотря на внешнюю близость с оригинальной постановкой, по-видимому, более сложна. В литературе на данный момент не обнаружено рассмотренной задачи в такой постановке. В данной работе рассматриваем ситуацию, когда задан граф, на структуру которого не наложено никаких ограничений, а также дано некоторое вершинное покрытие без каких-либо предположений о нём. Рассматривается вопрос, какой игрок выигрывает, если игра начнётся с заданного вершинного покрытия. Ответ на этот вопрос требует анализа всех бесконечных последовательностей атак.

На практике достаточно рассматривать последовательности некоторой достаточно большой длины из-за конечности множества возможных состояний игры, однако стоит ожидать, что эта длина будет неизвестной. Для построения стратегии защитника рассматриваемая задача представляется в виде динамической игры Штакельберга. На каждом шаге этой игры взаимодействие противоборствующих сторон формализуется в виде задачи двухуровневого математического программирования. Решение задачи верхнего уровня (задачи атакующего) определяет ребро, атака на которое приводит к максимальному числу незащищённых рёбер после реконструкции защитником размещения охранников. Решение задачи нижнего уровня (задачи защитника) при заданном решении задачи атакующего изменяет размещение охранников с целью минимизировать число незащищённых рёбер. В [11, 12] представлены формулировка данной задачи в виде двухуровневой модели целочисленно-линейного программирования и алгоритм, который отвечает на вопрос, существует ли выигрышная стратегия для защитника, позволяющая защитить заданное вершинное покрытие за заданное конечное число шагов. В данной работе предложена процедура, позволяющая ответить на вопрос, является ли заданное вершинное покрытие вечным, а также проведён анализ сложности задачи, рассматриваемой на одном шаге игры.

1. Формулировка задачи, рассматриваемой на одном шаге игры

Формулировка двухуровневой задачи математического программирования, рассматриваемой на одном шаге игры защитника и атакующего, представлена в [12]. Пусть $G = (I, J)$ — ориентированный граф без петель с множеством вершин I и дуг J. Для произвольного $j \in J$ через $i(j)$ и $k(j)$ обозначим начальную и конечную вершины дуги j соответственно, и будем называть эти вершины соседними. Отметим, что рассмотрение ориентированного графа не уменьшает общности двухуровневой задачи, рассматриваемой на каждом шаге исследуемой игры. В самом деле, в неориентированном графе каждое ребро можно заменить двумя противоположно ориентированными параллельными дугами. Будем считать, что перемещение охранника по дуге происходит от начальной вершины дуги к конечной.

Вершинным покрытием в графе G считаем всякое подмножество множества I, содержащее начальную или конечную вершину любой дуги из множества J.

Для записи задачи используются следующие обозначения:

- $x_{i0} \in \{0, 1\}$ — заданные величины, определяющие начальное для рассматриваемого шага вершинное покрытие: величина x_{i0} равна единице, если в вершине $i \in I$ размещён охранник, и нулю в противном случае;
- $x_i \in \{0, 1\}$ — переменные, задающие размещение охранников по завершении шага: величина x_i равна единице, если в вершине $i \in I$ размещён охранник, и нулю в противном случае;
- $y_j \in \{0, 1\}$ — переменные, обозначающие атакованную на шаге дугу: величина y_j равна единице, если дуга $j \in J$ атакована, и нулю в противном случае;
- $u_j \in \{0, 1\}$ — переменные, обозначающие дуги, которые по завершении шага оказались незащищёнными: величина u_j равна единице, если в начальной и конечной вершине дуги $j \in J$ не размещён охранник, и нулю в противном случае;
- $v_j \in \{0, 1\}$ — переменные, обозначающие перемещение охранников по дугам $j \in J$ на шаге: величина v_j равна единице, если охранник перемещается по дуге $j \in J$ из вершины $i(j)$ в вершину $k(j)$, и нулю в противном случае.

С использованием введённых обозначений модель, формализующая взаимодействие сторон на шаге рассматриваемой игры, записывается как следующая задача двухуровневого математического программирования:

$$\max_{(y_j)} \sum_{j \in J} u_j^*, \tag{1}$$

$$\sum_{j\in J} y_j \leqslant 1, \tag{2}$$

$$x_{i(j)0} \geqslant y_j, \quad j \in J, \tag{3}$$

$$x_{i(j)0} + x_{k(j)0} + y_j \leqslant 2, \quad j \in J, \tag{4}$$

$$y_j \in \{0,1\}, \quad j \in J, \tag{5}$$

где $(x_j^*), (v_j^*), (u_j^*)$ — оптимальное решение задачи

$$\min_{(x_i),(v_j),(u_j)} \sum_{j\in J} u_j, \tag{6}$$

$$v_j \geqslant y_j, \quad j \in J, \tag{7}$$

$$x_{i0} \geqslant \sum_{j\colon i(j)=i} v_j, \quad i \in I, \tag{8}$$

$$1 - x_{i0} \geqslant \sum_{j\colon k(j)=i} v_j - \sum_{j\colon i(j)=i} v_j, \quad i \in I, \tag{9}$$

$$x_i = x_{i0} + \sum_{j\colon k(j)=i} v_j - \sum_{j\colon i(j)=i} v_j, \quad i \in I, \tag{10}$$

$$x_{i(j)} + x_{k(j)} + u_j \geqslant 1, \quad j \in J, \tag{11}$$

$$x_i, u_j, v_j \in \{0,1\}, \quad i \in I,\ j \in J. \tag{12}$$

Данная модель, как и всякая двухуровневая задача, включает задачу верхнего уровня (1)–(5) (задачу атакующего) и задачу нижнего уровня (6)–(12) (задачу защитника).

Целевые функции (1) и (6) обеих задач показывают, сколько дуг осталось без защиты после реконструкции вершинного покрытия на рассматриваемом шаге. Неравенство (2) означает, что на рассматриваемом шаге атакованной может быть только одна дуга, а из (3) и (4) следует, что атакованной может быть только дуга, у которой охранник размещён в начальной вершине и нет охранника в конечной вершине. Ограничения (7)–(9) задачи нижнего уровня формализуют правила возможных перемещений охранников вдоль соответствующих дуг. Условие (7) означает, что если дуга атакована, то охранник перемещается в конечную вершину дуги. Из (8) следует, что если в рассматриваемом на этом шаге вершинном покрытии в вершине i размещён охранник, то его перемещение возможно не более чем по одной дуге, исходящей из этой вершины, а если в вершине i охранника нет, то перемещение охранников по дугам, исходящим из вершины i, невозможно. Аналогично из (9) с учётом условия (8) получаем, что если в рассматриваемом на этом шаге вершинном покрытии в вершине i нет охранника, то $\sum\limits_{j\colon k(j)=i} v_j \leqslant 1$. Это означает,

что если охранник не размещён в вершине i, то не более чем по одной дуге, входящей в вершину i, может быть перемещён охранник. Если же в вершине i размещён охранник, то из неравенств (8) и (9) получаем

$$1 \geqslant \sum_{j\colon i(j)=i} v_j \geqslant \sum_{j\colon k(j)=i} v_j.$$

Это значит, что по одной из дуг, входящих в вершину i, может быть перемещён охранник, но только в случае, когда охранник, размещённый в вершине i, также перемещается. Равенство (10) определяет новое размещение охранников в результате их перемещения на шаге. Неравенство (11) заставляет переменную u_j принимать ненулевое значение, когда в начальной и конечной вершинах дуги j нет охранников.

Допустимым решением двухуровневой задачи на рассматриваемом шаге игры назовём пару $((y_j),(x_i,v_j,u_j))$, где (y_j) — допустимое решение задачи верхнего уровня, а (x_i,v_j,u_j) — оптимальное решение задачи нижнего уровня. Допустимое решение будет оптимальным, если значение целевой функции на этом решении не меньше, чем на других допустимых решениях. Вычисление оптимального решения позволяет проверить текущее вершинное покрытие на устойчивость к атаке. Если оптимальное значение целевой функции равно нулю, то ответ положительный. Если же это значение больше нуля, то существует допустимое решение задачи верхнего уровня (атакуемая дуга), для которого не существует реконструкции вершинного покрытия, приводящей к новому вершинному покрытию.

Как следует из сказанного выше, возможность восстановления (реконструкции) вершинного покрытия после атаки определяется значениями целевых функций задач верхнего и нижнего уровней в двухуровневой модели (1)–(12), рассматриваемой на каждом шаге исследуемой игры.

Вершинное покрытие $C_0 = (x_{i0})$ будем называть *устойчивым к атаке на дугу* $j_0 \in J$, если оптимальное значение целевой функции задачи защитника равно нулю при допустимом решении (y_j), $y_{j_0} = 1$, задачи атакующего.

2. Вычислительная сложность задачи, рассматриваемой на одном шаге игры

Нетрудно видеть, что задачу проверки устойчивости к атаке на ребро можно переформулировать следующим образом. Пусть заданы граф $G = (I, J)$ с множеством вершин I и рёбер J и множество охранников $C \subset I$, размещённых в некоторых вершинах графа, образующее вершинное покрытие. Задано подвергающееся атаке ребро, на одном из концов которого нет охранника. Существует ли вершинное покрытие, которое

может быть получено из размещения охранников C так, что каждый из охранников либо перемещён в соседнюю вершину, либо остался на месте, при условии, что по атакованному ребру перемещён охранник? Назовём данную задачу *переформулированной задачей одного шага.*

Теорема 1. *Переформулированная задача одного шага* NP*-полна.*

Доказательство. Нетрудно видеть, что данная задача принадлежит классу NP.

Покажем, что она также NP-трудна. Сведём задачу о вершинном покрытии к рассматриваемой задаче. Даны граф G и натуральное число k. К графу G добавим $n-k$ вершин u_i, $i \in \overline{1, n-k}$, каждую из которых соединим со всеми вершинами G. Также добавляем вершины s и t, соединив их ребром, а также соединив s с каждой вершиной из U. Обозначим полученный граф через G'. Разместим $n+1$ охранников во всех вершинах графа G, а также в вершине s. Нетрудно заметить, что множество охранников образует вершинное покрытие в графе G'. Выбираем для атаки ребро между вершинами s и t. Тогда G имеет вершинное покрытие размера не более k тогда и только тогда, когда для G', указанного начального размещения охранников и выбранного ребра ответ в рассматриваемой задаче «да».

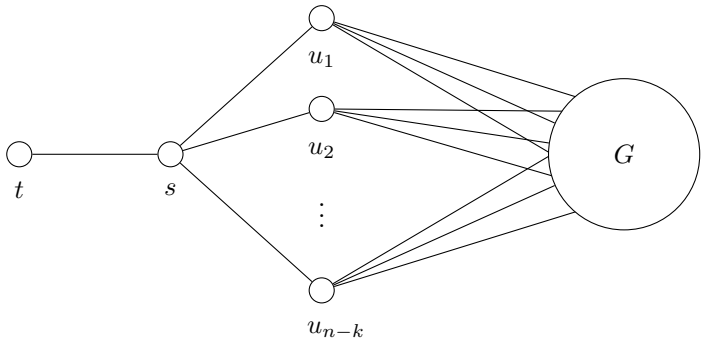


Рис. 1. Граф G'

Предположим, что C — некоторое вершинное покрытие графа G размера k. Перемещаем охранника из s в t. Тогда в G' остаётся переместить охранников из $n-k$ вершин графа G, не входящих в C, в вершины из U. Очевидно, что данное перемещение возможно осуществить таким образом, что каждый охранник перемещается в соседнюю вершину, так как u_i соединена с каждой вершиной в G. Нетрудно видеть, что новое размещение является вершинным покрытием в G'.

Пусть в G' существует новое размещение C', удовлетворяющее требованиям рассматриваемой задачи. Так как по условиям задачи охранник перемещён вдоль атакованного ребра из s в t, а новое размещение является вершинным покрытием, во всех вершинах из U должны находиться охранники, иначе ребро (u_i, s) для некоторого $u_i \in U$ не будет защищено. Таким образом, в вершинах подграфа G графа G' находится $n + 1 - (n - k) - 1 = k$ охранников. Поскольку C' — вершинное покрытие, эти k охранников находятся в вершинах, образующих вершинное покрытие размера k в подграфе G графа G'. Теорема 1 доказана.

3. Устойчивость вершинного покрытия в игре с бесконечным числом шагов

Пусть задан неориентированный граф $G = (V, E)$.

Для данного графа можно построить эквивалентный ориентированный граф $G = (I, J)$, где $I = V$, а J построено таким образом, что для каждого ребра $(v_1, v_2) \in E$ существует пара противоположно направленных дуг $j_{v_1v_2}, j_{v_2v_1} \in J$, выходящих из v_1 в v_2 и наоборот соответственно.

Вершинное покрытие $C_0 = (x_{i0})$ назовём *устойчивым к атаке на ребро* $(v_1, v_2) \in E$, если оптимальное значение целевой функции задачи защитника равно нулю при допустимом решении (y_j), $y_{j_{v_1v_2}} + y_{j_{v_2v_1}} = 1$, задачи атакующего.

Вершинное покрытие назовём 1-*устойчивым*, если оптимальное значение целевой функции задачи атакующего равно нулю, иначе это покрытие *неустойчивое*. Таким образом, процедура проверки 1-устойчивости заданного вершинного покрытия C_0 сводится к проверке устойчивости этого покрытия к атаке всякого допустимого для атаки ребра $e \in E$.

Рассмотрим игру, состоящую из $L > 1$ шагов, с начальным вершинным покрытием C_0. Пусть на шагах $1, 2, \ldots, L-1$ оптимальное значение целевой функции задачи атакующего равно нулю, а на шаге L больше нуля. Это означает, что в результате $L - 1$ шагов получено вершинное покрытие такое, что после атаки некоторого ребра успешная реконструкция этого вершинного покрытия невозможна. Заметим, что отсюда не следует, что при начальном покрытии C_0 у защитника нет выигрышной стратегии. Действительно, полученное к шагу L вершинное покрытие есть результат последовательных изменений вершинных покрытий, построенных на предыдущих шагах. Каждое такое изменение определяется оптимальным решением задачи защитника, которое может быть не единственным. Следовательно, при соответствующем выборе оптимальных решений на шагах, предшествующих шагу L, может быть получено вершинное покрытие, которое будет устойчивым к атаке любого ребра на шаге L.

Вершинное покрытие назовём *L-устойчивым*, если оно 1-устойчивое и для любого допустимого решения задачи атакующего существует оптимальное решение задачи защитника, дающее $(L-1)$-устойчивое вершинное покрытие.

Вершинное покрытие будем называть *вечно устойчивым*, или для краткости *вечным*, если оно L-устойчиво при любом L.

Для рассматриваемого графа и некоторого вершинного покрытия C_0 рассмотрим множество $\widehat{C}$ всех покрытий той же мощности, что и C_0. Так как множество $\widehat{C}$ конечно, можем занумеровать все покрытия числами $0, \ldots, |\widehat{C}|$. Обозначим множество номеров через $\widehat{N}$. Тогда C_q будет означать вершинное покрытие из $\widehat{C}$ с номером $q \in \widehat{N}$.

Для пары вершинных покрытий $C_q, C_s \in \widehat{C}$ рассмотрим множество P решений задачи защитника таких, что для каждого решения $\rho \in P$ вектор $\left(x^\rho_{i0}\right)$ определяется покрытием C_q, а $\left(x^\rho_i\right)$ — покрытием C_s. Для каждого такого решения определим множество

$$E^\rho_{qs} = \left\{(v_1, v_2) \in E \mid v^\rho_{j_{v_1 v_2}} + v^\rho_{j_{v_2 v_1}} = 1\right\}.$$

Пусть $E_{qs} = \bigcup\limits_{\rho \in P} E^\rho_{qs}$ — множество всех рёбер, по которым перемещаются охранники хотя бы при некоторой реконструкции покрытия C_q в покрытие C_s. Нетрудно заметить, что $E_{qs} = E_{sq}$. Действительно, если возможно переместить охранников из покрытия C_q по некоторым рёбрам так, что новое размещение является покрытием C_s, то возможно их переместить по тем же рёбрам в обратном направлении.

Для графа G определим взвешенный *граф реконструкций* $H = (N, R)$ следующим образом: $N \subseteq \widehat{N}$, а ребро (q, s), $q, s \in N$, имеет вес E_{qs} и принадлежит R тогда и только тогда, когда $E_{qs} \neq \varnothing$.

Граф H может содержать петли. Рассмотрим вершинное покрытие C_q и рёбра из E такие, что обе вершины каждого из этих рёбер содержатся в C_q. По условию задачи такие рёбра не могут быть подвергнуты атаке, поскольку они защищены с обеих сторон. Таким образом, петля (q, q) в графе H имеет вес E_{qq}, и множество E_{qq} состоит из рёбер, которые не могут быть атакованы.

Вершину $q \in N$ будем называть *полной*, если $\bigcup\limits_{s \in N} E_{qs} = E$. Граф реконструкций $H = (N, R)$ будем называть *завершённым*, если любая вершина $q \in N$ полная.

Тогда для проверки того, является ли данное вершинное покрытие вечным, можно воспользоваться следующей леммой.

Лемма 1. *Если граф реконструкций $H = (N, R)$ завершённый, то для любого $q \in N$ вершинное покрытие C_q вечное.*

Доказательство. Предположим, что для некоторого $s \in N$ вершинное покрытие C_s не вечное. Тогда существует конечная последовательность атак $C_s \xrightarrow{e_{k_0}} C_{s_1} \xrightarrow{e_{k_1}} \cdots \xrightarrow{e_{k_{L-1}}} C_{s_L} \xrightarrow{e_{k_L}}$, где $s_i \in N$ для любого $i \in \overline{1, L}$, а при атаке на ребро e_{k_L} для покрытия C_{s_L} не существует решения соответствующей задачи защитника. В то же время, так как $s_L \in N$ и граф реконструкций по условию завершённый, s_L является полной вершиной, а следовательно, найдётся вершина $p \in N$ такая, что $e_{k_L} \in E_{s_L p}$. Вершинное покрытие C_p соответствует решению задачи защитника при атаке на ребро e_{k_L} покрытия C_{s_L}; противоречие. Лемма 1 доказана.

Лемма 2. *Если вершинное покрытие C_q, $q \in \widehat{N}$, графа G вечное, то для G существует завершённый граф реконструкций $H = (N, R)$, при этом $q \in N$.*

Доказательство. Рассмотрим $\widehat{H} = (\widehat{N}, \widehat{R})$ такой, что множество $\widehat{R}$ содержит в себе все возможные рёбра между всеми вершинами из $\widehat{N}$. Рассмотрим его компоненту связности $H = (N, R)$, $q \in N$. Покажем, что H завершённый.

Допустим, что H не завершённый. Тогда найдётся вершина $s_L \in N$ такая, что C_{s_L} — неполная вершина. Это означает, что существует ребро $e' \in E$ такое, что $e' \notin \bigcup\limits_p E_{s_L p}$, где $p \in N$. Таким образом, при атаке на ребро e' для покрытия C_{s_L} не существует решения соответствующей задачи защитника, которое бы соответствовало некоторому покрытию, номер которого содержался бы в N. Однако, так как по условию $\widehat{R}$ содержит в себе все возможные рёбра между всеми вершинами из $\widehat{N}$, R также содержит в себе все возможные рёбра между всеми вершинами из N. Отсюда следует, что рассматриваемая задача защитника не имеет решения.

Так как H связный, существует путь между q и s_L, а значит, существует последовательность атак $C_q \xrightarrow{e_{k_0}} C_{s_1} \xrightarrow{e_{k_1}} \cdots \xrightarrow{e_{k_{L-1}}} C_{s_L} \xrightarrow{e'}$, где $s_i \in N$ для любого $i \in \overline{0, L}$, приводящая к победе атакующего. Значит, вершинное покрытие C_q не будет вечным; противоречие. Лемма 2 доказана.

Таким образом, если для заданного графа удастся построить завершённый граф реконструкций такой, что проверяемое вершинное покрытие будет одной из вершин графа реконструкций, то данное вершинное покрытие будет вечным.

Опишем процедуру построения завершённого графа реконструкций $H = (N, R)$, которая состоит из определённого числа однотипных итераций. На каждой итерации проверяемое вершинное покрытие C_q, $q \in N$, исследуется на устойчивость к атакам на допустимые для атаки рёбра. Также имеется список T номеров запрещённых покрытий, которые

не могут быть включены в граф реконструкций, а также список A номеров вершинных покрытий, проверка которых начата на предыдущих итерациях, но не завершена. Элементы списка A упорядочены по порядку вхождения в список.

На первой итерации рассматривается граф реконструкций, состоящий из одной вершины, соответствующей исследуемому вершинному покрытию C_0. Список A состоит из единственного элемента — номера вершинного покрытия C_0, а список T пуст.

На каждой итерации, если $A \neq \varnothing$, рассматривается вершинное покрытие C_q, где номер $q \in N$ — последний в списке A. При этом номер q удаляется из A. Рассматриваемое вершинное покрытие C_q проверяется на устойчивость к атакам на рёбра из множества E_q^*, состоящее из допустимых для атаки рёбер графа G за исключением рёбер, входящих в множества E_{qs} для $s \in N$.

Такая проверка состоит из определённого числа однотипных шагов, на каждом из которых рассматривается ребро $(v_1, v_2) \in E_q^*$. Для (v_1, v_2) рассматривается задача защитника при условии $x_{i0} = 1$, если $i \in C_q$, $y_{j_{v_1 v_2}} + y_{j_{v_2 v_1}} = 1$, и дополнительных ограничениях, задаваемых вершинными покрытиями из списка T и отсекающих решения, приводящие к запрещённым покрытиям. Если $s \in T$ и $x'_i = 1$ для $i \in C_s$, то соответствующее ограничение имеет вид

$$\sum_{i \in I} x'_i x_i \leqslant \sum_{i \in I} x'_i - 1.$$

Пусть для ребра (v_1, v_2) оптимальное значение целевой функции задачи защитника больше нуля, а следовательно, успешная реконструкция вершинного покрытия C_q невозможна. Если C_q — исследуемое вершинное покрытие C_0, то процедура завершается с отрицательным результатом. Иначе номер q добавляется в список T, вершина с номером q удаляется из графа H, все соседние с ней вершины добавляются в A, и начинается следующая итерация.

Пусть при атаке на рассматриваемое ребро (v_1, v_2) вершинное покрытие C_q оказывается устойчивым и получено вершинное покрытие C_s.

Если $s \notin N$, то в H добавляется вершина с номером s, соответствующая вершинному покрытию C_s, и ребро (q, s) с весом E_{qs}. Множество E_{qs} включает рёбра графа G, по которым перемещаются охранники при реконструкции вершинного покрытия C_q в вершинное покрытие C_s. Кроме того, в граф H добавляется петля (s, s) с весом E_{ss}, включающим ребра, запрещённые для атаки вершинного покрытия C_s. Далее ребро (v_1, v_2) удаляется из множества E_q^* и, если $E_{qs} \neq \varnothing$, вершина q возвращается в список A. После этого вершина s добавляется в список A, и начинается следующая итерация.

Если полученное вершинное покрытие C_s такое, что $s \in N$, то вес E_{qs} ребра (q, s) графа H пополняется рёбрами графа G, по которым перемещаются охранники при реконструкции вершинного покрытия C_q в C_s. Далее ребро (v_1, v_2) удаляется из множества E_q^*. Если $E_q^* \neq \varnothing$, то начинается следующий шаг исследуемой итерации, иначе начинается следующая итерация.

Процедура завершается с положительным результатом тогда, когда на очередной итерации список A пуст, а все шаги по проверке текущего вершинного покрытия завершены с положительным результатом.

Если процедура завершится с положительным результатом, то для данного графа удалось построить завершённый граф реконструкций, одной из вершин которого является C_0. Иначе все покрытия, получаемые при реконструкции в ответ на атаку на некоторое ребро вершинного покрытия C_0, содержатся в списке T, что означает существование некоторой последовательности атак, приводящей к победе атакующего.

3.1. Показательный пример. Приведём пример работы алгоритма построения графа реконструкций $H = (N, R)$ для данного вершинного покрытия C_0 графа G. На рис. 2 представлен граф G, для которого задано вершинное покрытие $C_0 = \{a, b, c\}$.

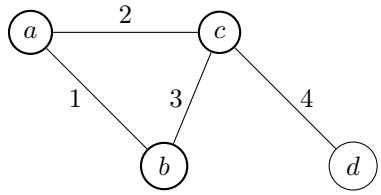


Рис. 2. Исходный граф G и покрытие C_0

На итерации 1 алгоритма граф H имеет единственную вершину, соответствующую вершинному покрытию $C_0 = \{a, b, c\}$. Эта вершина имеет петлю $(0, 0)$ с весом $E_{0,0} = \{1, 2, 3\}$, список $A = \{0\}$. Начальный вид графа H представлен на рис. 3.

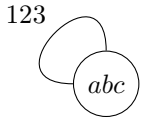


Рис. 3. Начальное состояние графа реконструкций

Сначала рассматривается вершина с номером $q = 0$ и соответствующее вершинное покрытие $C_0 = \{a, b, c\}$ проверяется на устойчивость

к атакам на рёбра из множества $E_0^* = \{4\}$. При этом номер этого вершинного покрытия удаляется из списка A.

На шаге 1 итерации 1 рассматривается задача защитника для покрытия C_0 при атаке на ребро H. Результатом решения этой задачи является вершинное покрытие $C_1 = \{a, b, d\}$, а при реконструкции C_0 в C_1 охранник перемещается только по ребру 4.

Поскольку $1 \notin N$, в множество N вершин графа H добавляется номер вершинного покрытия $C_1 = \{a, b, d\}$, а в множество R рёбер графа H — ребро $(0, 1)$ с весом $E_{0,1} = \{4\}$. Кроме того, в множество R добавляется петля $(1, 1)$ с весом $E_{1,1} = \{1\}$. После этого номер вершинного покрытия $C_1 = \{a, b, d\}$ добавляется в список A.

После шага 1 список J_0 пуст, и итерация 1 завершается. Полученный в результате итерации 1 граф H представлен на рис. 4.

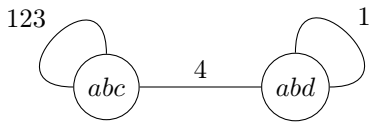


Рис. 4. Результат итерации 1

На итерации 2 имеем $A = \{1\}$, поэтому рассматриваем вершину с номером $q = 1$, соответствующую вершинному покрытию $C_1 = \{a, b, d\}$, для которого $E_1^* = \{2, 3\}$, и удаляем её номер из списка A.

На шаге 1 итерации 2 решается задача защитника для покрытия $C_1 = \{a, b, d\}$ при атаке на ребро 2. Одним из возможных решений задачи является вершинное покрытие $C_2 = \{a, c, d\}$, при построении которого охранники перемещались по рёбрам 1 и 2.

Поскольку $2 \notin N$, в множество N вершин графа H добавляется номер вершинного покрытия $C_2 = \{a, c, d\}$, а в множество R рёбер графа H — ребро $(1, 2)$ с весом $E_{1,2} = \{1, 2\}$. Кроме того, в множество R добавляется петля $(2, 2)$ с весом $E_{2,2} = \{2, 4\}$. После этого номер вершинного покрытия $C_2 = \{a, c, d\}$ добавляется в список A. Полученный в результате шага 1 итерации 2 граф H представлен на рис. 5.

На шаге 2 итерации 2 исследуется устойчивость вершинного покрытия $C_1 = \{a, b, d\}$ при атаке на ребро 3. Одним из возможных решений задачи защитника является вершинное покрытие $C_2 = \{a, c, d\}$, при построении которого охранник перемещался по ребру 3.

Поскольку $2 \in N$, множество вершин графа H не увеличивается, новые рёбра не появляются, но вес ребра $(1, 2)$ становится равным $E_{1,2} = \{1, 2, 3\}$.

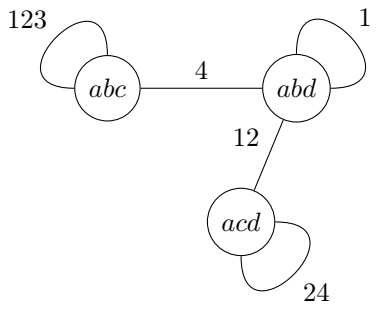


Рис. 5. Результат шага 1 итерации 2

После шага 2 список J_0 пуст, и итерация 2 завершается. Полученный в результате итерации 2 граф H представлен на рис. 6.

На итерации 3 $A = \{2\}$, поэтому исследуем вершинное покрытие с номером $q = 2$. Исключаем номер 2 из списка A, а поскольку $E_2^* = \varnothing$, завершаем итерацию. Так как $A = \varnothing$, завершаем алгоритм. Полученный граф реконструкций H, изображённый на рис. 6, полный, а одна из его вершин является номером исследуемого вершинного покрытия C_0.

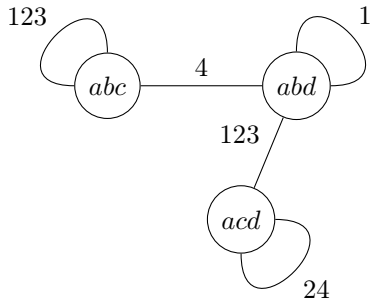


Рис. 6. Результат итерации 2

4. Вычислительные эксперименты

В экспериментах исследованы вычислительные возможности предложенного алгоритма проверки устойчивости вершинных покрытий, а также влияние на его производительность некоторых используемых процедур. Расчёты проведены на компьютере с четырёхъядерным процессором

3,3 ГГц и 32 ГБ оперативной памяти. Алгоритмы написаны на языке программирования C# , а для поиска оптимальных решений возникающих задач целочисленного линейного программирования использовался решатель Gurobi.

В экспериментах использованы случайно генерируемые связные графы. Поскольку нахождение минимального покрытия — задача труднорешаемая, в экспериментах используются покрытия, у которых ни одно из собственных подмножеств не остаётся покрытием. Алгоритмы генерации графов и вершинных покрытий приведены в [12].

4.1. Устойчивость вершинного покрытия в игре с бесконечным числом шагов. В ходе экспериментов, описанных в данном пункте, сгенерировано несколько тысяч покрытий для различных графов с 4, 8, 16 и 32 вершинами способом, описанным в предыдущих пунктах.

Структура графа реконструкций. В данном разделе представлены результаты вычислительного эксперимента, позволяющего изучить зависимость структуры получаемых графов реконструкций от размерности исходного графа. В табл. 1 показана зависимость числа вершин и рёбер в получаемом графе реконструкций от числа вершин n исходного графа.

Таблица 1

Зависимость числа вершин и рёбер в графе H от числа вершин n исходного графа

n	hv_0	hv_1	he_1
4	1,43	2,89	1,15
8	2,26	8,1	19
16	8,02	41,9	266
32	64,1	624	9954

В ходе эксперимента для каждого числа вершин были рассчитаны величины hv — среднее число вершин графа реконструкций, полученного по окончании работы алгоритма, и he — среднее число рёбер в графе реконструкций. Результаты были разделены для покрытий, которые оказались вечными (индекс 1), и для покрытий, для которых нашлась последовательность атак, приводящая к победе атакующего (индекс 0). Величина he_0 не представлена в табл. 1, так как в случае, когда для проверяемого покрытия удалось найти последовательность атак, приводящую к победе атакующего, результатом работы алгоритма является граф реконструкций, в котором отсутствуют рёбра.

Из табл. 1 видно, что уже для графов с 32 вершинами графы реконструкций имеют несколько сотен вершин. В случае, если проверяемое вершинное покрытие вечное, граф реконструкций будет иметь несколько тысяч рёбер. Таким образом, уже для графов средней размерности доказательство вечности покрытий имеет высокую сложность по памяти, так как требуется хранить всю структуру графа реконструкций.

Доля вечных покрытий. *Заполненностью* графа G с n вершинами и m рёбрами назовём величину

$$f = \frac{2(m-n+1)}{(n-1)(n-2)}.$$

Заметим, что $f = 0$ для деревьев и $f = 1$ для полного графа.

Приведём результаты вычислительного эксперимента, демонстрирующего один из вариантов применения алгоритма по проверке вечной устойчивости вершинного покрытия. Его цель состоит в исследовании зависимости доли вечных неприводимых вершинных покрытий от заполненности графа. Случайность генерируемых графов и недетерминированность полученных неприводимых вершинных покрытий позволяет утверждать, что полученные результаты достаточно общие при достаточном размере выборки.

На рис. 7 представлена зависимость доли вечных неприводимых покрытий от заполненности графа для графов с 8, 16 и 32 вершинами.

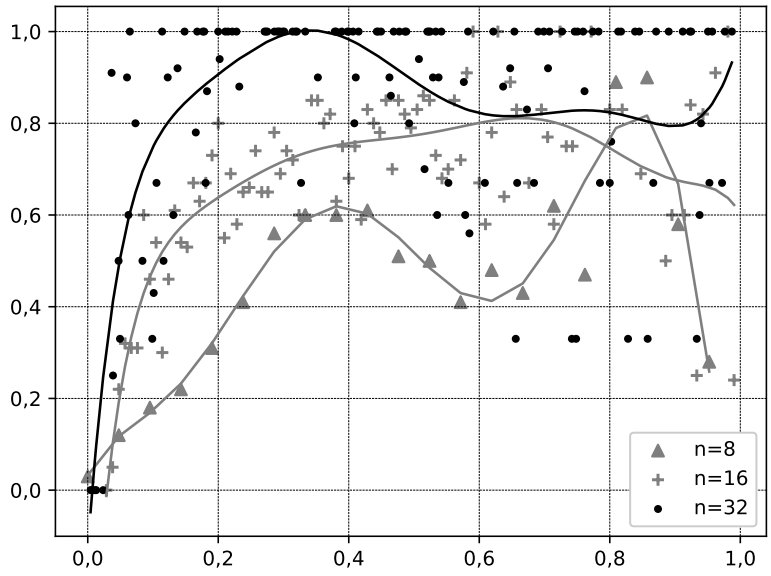


Рис. 7. Зависимость доли вечных покрытий от заполненности графа

В ходе эксперимента для каждого числа вершин построена своя зависимость. Для каждого рассматриваемого графа рассчитано значение его заполненности, затем среди покрытий, соответствующих графам с одинаковым числом заполненности, рассчитана доля вечных. Кроме того, для большей наглядности для каждой зависимости было построено приближение полиномом 8-й степени.

Из рис. 7 видно, что чем выше число вершин в графе, тем в среднем бóльшая часть неприводимых вершинных покрытий является вечными. Можно также сделать вывод, что увеличение числа заполненности графа чаще влечёт увеличение доли вечных покрытий.

Время работы алгоритма. Проведённые вычислительные эксперименты позволяют сделать некоторые выводы о времени работы алгоритма в зависимости от числа вершин графа.

Таблица 2

Зависимость времени работы алгоритма и числа запусков оптимизации модели задачи защитника от числа вершин n

n	t_0, с	t_1, с	mc_0	mc_1
4	0,01	0,01	2	4
8	0,06	0,14	6	31
16	1,93	2,89	56	427
32	466	277	644	16 133

В табл. 2 представлена зависимость времени работы алгоритма при проверке вечности вершинных покрытий графов соответствующей размерности. В ходе эксперимента для каждого числа вершин рассчитаны величины t — времени работы алгоритма в секундах и mc — число запусков оптимизации модели задачи защитника. Результаты выписаны отдельно для покрытий, которые оказались вечными (индекс 1), и для покрытий, для которых нашлась последовательность атак, приводящая к победе атакующего (индекс 0).

Как видим, время работы алгоритма практически не отличается для обоих классов вершинных покрытий, т. е. доказательство вечности вершинного покрытия занимает почти столько же времени, сколько поиск последовательности атак, которая приводит к победе атакующего. Также можно видеть, что время работы алгоритма уже для графов с 32 вершинами равно примерно 5 мин.

При этом число запусков оптимизации модели задачи защитника для покрытий, оказавшихся вечными, значительно больше, чем для покрытий, для которых нашлась последовательность атак, приводящая к победе атакующего. Это можно объяснить тем, что время, затрачиваемое

на поиск оптимального решения задачи защитника с дополнительными ограничениями, возрастает с ростом числа запрещённых покрытий.

Заключение

В работе предложена процедура, позволяющая ответить на вопрос, является ли заданное вершинное покрытие вечным. В то же время, эта процедура позволяет находить сколь угодно длинные приводящие к победе атакующего последовательности атак, если они существуют.

Идея процедуры заключается в рекурсивной проверке устойчивости вершинных покрытий, полученных в результате решения задач защитника. Для построения стратегии защитника задача о вечном вершинном покрытии представляется как динамическая игра Штакельберга, на каждом шаге которой взаимодействие противоборствующих сторон формализуется как двухуровневая задача математического программирования. Предложенная процедура протестирована на различных случайно сгенерированных графах и вершинных покрытиях. Вычислительные эксперименты продемонстрировали способность данной процедуры находить решения за адекватное время для графов средней размерности.

В работе представлена реализация построения графа реконструкций вершинных покрытий на основе рекурсивного решения задач защитника. При данном подходе не гарантируется, что данный граф получится минимального размера, тем самым существует пространство для ускорения данной процедуры.

Финансирование работы

Исследование выполнено в рамках государственного задания Института математики им. С. Л. Соболева (проект № FWNF–2026–0021). Дополнительных грантов на проведение или руководство этим исследованием получено не было.

Конфликт интересов

Автор заявляет, что у него нет конфликта интересов.

Литература

1. **Klostermeyer W. F., Mynhardt C. M.** Protecting a graph with mobile guards // Appl. Anal. Discrete Math. 2016. V. 10, No. 1. P. 1–29.
2. **Klostermeyer W. F., Mynhardt C. M.** Edge protection in graphs // Australas. J. Comb. 2009. V. 45. P. 235–250.
3. **Fomin F. V., Gaspers S., Golovach P. A., Kratsch D., Saurabh S.** Parameterized algorithm for eternal vertex cover // Inf. Process. Lett. 2010. V. 110, No. 16. P. 702–706.

4. **Babu J., Misra N., Nanoti S. G.** Eternal vertex cover in bipartite graphs // Computer science — Theory and applications. Proc. 17th Int. Comp. Sci. Symp. in Russia (St. Petersburg, Russia, June 29 – July 1, 2022). Cham: Springer, 2022. P. 64–76. (Lect. Notes Comput. Sci.; V. 13296).
5. **Babu J., Prabhakaran V.** A new lower bound for the eternal vertex cover number of graphs // J. Comb. Optim. 2021. V. 44, No. 4. P. 2482–2498.
6. **Babu J., Chandran L. S., Francis M., Prabhakaran V., Rajendraprasad D., Warrier N. J.** On graphs whose eternal vertex cover number and vertex cover number coincide // Discrete Appl. Math. 2022. V. 319. P. 171–182.
7. **Babu J., Prabhakaran V., Sharma A.** A substructure based lower bound for eternal vertex cover number // Theor. Comput. Sci. 2021. V. 890. P. 87–104.
8. **Paul K., Pandey A.** Some algorithmic results for eternal vertex cover problem in graphs // WALCOM: Algorithms and computation. Proc. 17th Int. Conf. and Workshops (Hsinchu, Taiwan, Mar. 22–24, 2023). Cham: Springer, 2023. P. 242–253. (Lect. Notes Comput. Sci.; V. 13973).
9. **Araki H., Fujito T., Inoue S.** On the eternal vertex cover numbers of generalized trees // IEICE Trans. Fundam. Electron. Commun. Comput. Sci. 2015. V. E98-A, No. 6. P. 1153–1160.
10. **Klostermeyer W. F., Mynhardt C. M.** Graphs with equal eternal vertex cover and eternal domination numbers // Discrete Math. 2011. V. 311, No. 14. P. 1371–1379.
11. **Beresnev V. L., Melnikov A. A., Utyupin S. Yu.** Representation of the eternal vertex cover problem as a dynamic Stackelberg game // Optimization and applications. Rev. Sel. Pap. 14th Int. Conf. OPTIMA 2023 (Petrovac, Montenegro, Sept. 18–22, 2023). Cham: Springer, 2023. P. 3–13. (Lect. Notes Comput. Sci.; V. 14395).
12. **Береснев В. Л., Мельников А. А., Утюпин С. Ю.** Устойчивость вершинных покрытий в игре с конечным числом шагов // Дискрет. анализ и исслед. операций. 2024. Т. 31, № 2. С. 28–45.

Утюпин Степан Юрьевич

Статья поступила
8 апреля 2026 г.
После доработки —
11 мая 2026 г.
Принята к публикации
22 мая 2026 г.

DISKRETNYI ANALIZ I ISSLEDOVANIE OPERATSII
/DISCRETE ANALYSIS AND OPERATIONS RESEARCH/
April–June 2026. Vol. 33, No. 2. P. 115–135

UDC 519.8 DOI: 10.33048/daio.2026.33.863

STABILITY OF VERTEX COVERS IN THE ETERNAL VERTEX COVER PROBLEM

S. Yu. Utyupin

Sobolev Institute of Mathematics,
4 Acad. Koptyug Avenue, 630090 Novosibirsk Russia
E-mail: stepan.utyupin@gmail.com

Abstract. The eternal vertex cover problem is a modification of the graph vertex cover problem and can be viewed as a dynamic two-player (attacker and defender) game with an infinite number of moves. To construct defender's strategy, the problem is represented as a dynamic game, where at each step the interaction between the opposing players is formalized as a two-level mathematical programming problem. This study proposes a procedure that allows us to determine whether a given vertex cover is eternal or not. The idea of the procedure is based on recursive verification of stability of vertex covers obtained by solving defender's problems and constructing an additional graph of covers and transitions. Tab. 2, illustr. 7, bibliogr. 12.

Keywords: eternal vertex cover, graph edge protection, stability check algorithm, dynamic Stackelberg game.

References

1. **W. F. Klostermeyer** and **C. M. Mynhardt,** Protecting a graph with mobile guards, *Appl. Anal. Discrete Math.* **10** (1), 1–29 (2016).
2. **W. F. Klostermeyer** and **C. M. Mynhardt,** Edge protection in graphs, *Australas. J. Comb.* **45**, 235–250 (2009).
3. **F. V. Fomin, S. Gaspers, P. A. Golovach, D. Kratsch,** and **S. Saurabh,** Parameterized algorithm for eternal vertex cover, *Inf. Process. Lett.* **110** (16), 702–706 (2010).
4. **J. Babu, N. Misra,** and **S. G. Nanoti,** Eternal vertex cover in bipartite graphs, in *Computer Science — Theory and Applications*, Proc. 17th Int. Comp. Sci. Symp. in Russia (St. Petersburg, Russia, June 29 – July 1, 2022) (Springer, Cham, 2022), pp. 64–76 (Lect. Notes Comput. Sci., Vol. 13296).

English transl.: *Journal of Applied and Industrial Mathematics* **20** (2) (2026).

5. **J. Babu** and **V. Prabhakaran,** A new lower bound for the eternal vertex cover number of graphs, *J. Comb. Optim.* **44** (4), 2482–2498 (2021).
6. **J. Babu, L. S. Chandran, M. Francis, V. Prabhakaran, D. Rajendraprasad,** and **N. J. Warrier,** On graphs whose eternal vertex cover number and vertex cover number coincide, *Discrete Appl. Math.* **319**, 171–182 (2022).
7. **J. Babu, V. Prabhakaran,** and **A. Sharma,** A substructure based lower bound for eternal vertex cover number, *Theor. Comput. Sci.* **890**, 87–104 (2021).
8. **K. Paul** and **A. Pandey,** Some algorithmic results for eternal vertex cover problem in graphs, in *WALCOM: Algorithms and Computation*, Proc. 17th Int. Conf. and Workshops (Hsinchu, Taiwan, Mar. 22–24, 2023) (Springer, Cham, 2023), pp. 242–253 (Lect. Notes Comput. Sci., Vol. 13973).
9. **H. Araki, T. Fujito,** and **S. Inoue,** On the eternal vertex cover numbers of generalized trees, *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **E98-A** (6), 1153–1160 (2015).
10. **W. F. Klostermeyer** and **C. M. Mynhardt,** Graphs with equal eternal vertex cover and eternal domination numbers, *Discrete Math.* **311** (14), 1371–1379 (2011).
11. **V. L. Beresnev, A. A. Melnikov,** and **S. Yu. Utyupin,** Representation of the eternal vertex cover problem as a dynamic Stackelberg game, in *Optimization and Applications*, Rev. Sel. Pap. 14th Int. Conf. OPTIMA 2023 (Petrovac, Montenegro, Sept. 18–22, 2023) (Springer, Cham, 2023), pp. 3–13 (Lect. Notes Comput. Sci., Vol. 14395).
12. **V. L. Beresnev, A. A. Melnikov,** and **S. Yu. Utyupin,** Stability of vertex covers in a game with finitely many steps, *Diskretn. Anal. Issled. Oper.* **31** (2), 28–45 (2024) [Russian] [*J. Appl. Ind. Math.* **18** (2), 206–215 (2024)].

Stepan Yu. Utyupin

Received April 8, 2026
Revised May 11, 2026
Accepted May 22, 2026

ДИСКРЕТНЫЙ АНАЛИЗ И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ
Апрель–июнь 2026. Т. 33, № 2. С. 136–157

УДК 519.17 DOI: 10.33048/daio.2026.33.856

ЛОКАЛЬНАЯ НЕЛИНЕЙНОСТЬ СТЕПЕНЕЙ ПРЕОБРАЗОВАНИЯ

В. М. Фомичёв[1,2]

[1] Российский университет дружбы народов им. Патриса Лумумбы, ул. Миклухо-Маклая, 6, 117198 Москва, Россия

[2] Институт проблем информатики ФИЦ «Информатика и управление», ул. Вавилова, 44, корп. 2, 119333 Москва, Россия

E-mail: fomichev.2016@yandex.ru

Аннотация. В соответствии с шенноновской концепцией о перемешивающих свойствах преобразований информации развивается ранее предложенный автором матрично-графовый подход к оценке характеристик нелинейности произведения преобразований векторного пространства. Подход позволяет оценить локальные нелинейные свойства произведения преобразований с помощью троичных матриц и помеченных орграфов, однозначно соответствующих преобразованиям. Для троичных матриц и для помеченных орграфов с использованием длин путей определены свойство локальной 2-примитивности, оценки локальных 2-экспонентов и, в частности, точные значения. Получены критерии локальной 2-примитивности. Результаты для матриц и помеченных орграфов применяются для оценки характеристик произведения преобразований векторного пространства. Рассмотрены примеры. Библиогр. 18.

Ключевые слова: троичная матрица, локально примитивная матрица, 2-примитивная матрица, 2-экспонент матрицы, 2-примитивный помеченный орграф.

Введение

Преобразование данных в симметричных криптографических системах защиты информации часто строится на основе произведения функций векторных пространств, имеющих приемлемую сложность программной и/или аппаратной реализации. Задачи криптографического анализа произведения функций, как правило, сложны в силу аналитической сложности связей между элементами входных и выходных данных. Вместе с тем показал продуктивность матрично-графовый подход (МГП) к оценке некоторых характеристик произведения функций, в частности,

© В. М. Фомичёв, 2026

к оценке множеств существенных и нелинейных переменных координатных функций степеней преобразований векторных пространств [1–3].

МГП основывается на комбинаторных свойствах мультипликативных полугрупп неотрицательных матриц и ориентированных графов. Первая постановка задачи в этой области относится к 1912 г. (Фробениус [4]): была поставлена задача распознавания наличия положительной матрицы в циклической полугруппе, порождённой заданной неотрицательной матрицей. При наличии порождающая матрица была названа примитивной, а её наименьшая степень, у которой все элементы положительные, спустя полвека названа экспонентом примитивной матрицы [5]. Задача Фробениуса к середине XX — началу XXI вв. инициировала несколько связанных с МГП основополагающих результатов в области неотрицательных матриц и графов [6–18]. Таковыми являются критерии конечности экспонентов и их количественные оценки. Ряд зарубежных результатов касались локальных характеристик, однако, лишь для класса примитивных матриц и графов.

При дальнейшем развитии теоретических основ МГП обнаружилась связь с подходом Шеннона к перемешивающим свойствам криптографических преобразований, что существенно расширило область приложений МГП в криптографии и стимулировало обобщение понятий примитивности и распространение экспонента на несильно связные орграфы. Продвижение в изучении экспонентов матриц и графов, в частности, раскрывающие детализацию подхода Шеннона к перемешивающим свойствам криптографических преобразований, достигнуто в основном после 2014 г. Результаты представлены в статьях и отражены в учебной литературе.

Для множества существенных и нелинейных переменных произведения преобразований основные результаты даны в [1, 2]. Исследование локальных характеристик матриц и орграфов, связанных с произведением преобразований, представлено в [3] и относится к множеству существенных переменных. Данная работа дополняет понятийную базу оценки локальных характеристик нелинейности преобразований векторных пространств с помощью оценки локальных свойств троичных матриц и помеченных орграфов. Получены новые конструктивные критерии локальной примитивности и 2-примитивности орграфов.

Разработанный подход в части локальных характеристик нелинейности произведения преобразований применим вкупе с вычислительными экспериментами для анализа криптографических систем, построенных по итеративному принципу, например, симметричных блочных шифров и алгоритмов контроля целостности данных.

1. Основы матрично-графового подхода

Положительное свойство систем защиты, построенных по итеративному принципу, — существенная нелинейная зависимость от большого числа булевых входных переменных всех координатных функций произведения криптографических преобразований двоичных векторных пространств. Для произведения преобразований это свойство интерпретируется с помощью МГП как положительность большого числа элементов соответствующих перемешивающих матриц или наличие определённых путей в соответствующих перемешивающих орграфах. Применение МГП к степеням преобразований помогает в решении следующих задач:

- при определении существенных переменных — в распознавании примитивности или локальной примитивности двоичных матриц и ориентированных графов, а также вычислении их экспонентов или локальных экспонентов;
- при определении нелинейных переменных — в распознавании 2-примитивности или локальной 2-примитивности троичных матриц и помеченных орграфов, а также вычислении их 2-экспонентов или локальных 2-экспонентов.

Данные ниже определения дополняют введённую ранее терминологию [1–3] для изучения локально примитивных двоичных матриц и орграфов и локально 2-примитивных троичных матриц и помеченных орграфов.

2. Локальная 2-примитивность троичных матриц

В статье используются следующие обозначения:

- $\mathbb{N}$ — множество натуральных чисел;
- $n, m, d \in \mathbb{N}$; $L \subseteq \mathbb{N}$;
- $L^{\lceil n \rceil} = \{l \in L \mid l \leqslant n\}$;
- $L^{\lfloor m \rfloor} = \{l \in L \mid m < l\}$;
- $L^{(m,n]} = L^{\lfloor m \rfloor} \cap L^{\lceil n \rceil} = \{l \in L \mid m < l \leqslant n\}$;
- $n + L = \{n + l \mid l \in L\}$;
- $nL = \{nl \mid l \in L\}$;
- $\mathbb{Z}$ — множество целых чисел;
- $\mathbb{Z}_m$ — кольцо вычетов по модулю m;
- $m \mid n$ — число m делит n без остатка;
- $n \equiv_d m$ — число n сравнимо с m по модулю d;
- $\{L \equiv_d\} = \{l \bmod d \mid l \in L\}$ — множество целых неотрицательных вычетов;
- $G = \{0, 1, 2\}$ — коммутативная полугруппа по умножению, при этом $0g = 0$ для всех $g \in G$ и $gh = \max\{g, h\}$ для ненулевых $g, h \in G$;
- $M_n^{[G]}$ — множество квадратных троичных матриц порядка n над G.

Отметим следующие свойства множества $M_n^{[G]}$:

1) $M_n^{[G]}$ — мультипликативный моноид: если $A, M \in M_n^{[G]}$, $A = (a_{ij})$, $M = (m_{ij})$, то $(AM)_{ij} = \max\limits_{1 \leqslant p \leqslant n} a_{ip} m_{pj}$, при этом умножение выполняется в G;

2) моноид $M_n^{[G]}$ частично упорядочен: $A \leqslant M$ тогда и только тогда, когда $a_{ij} \leqslant m_{ij}, 1 \leqslant i, j \leqslant n$.

Приведём известное определение $2_{(i,j)}$-примитивной матрицы. Пусть $M \in M_n^{[G]}$ и $M^t = \left(m_{ij}^{(t)}\right)$, $t \geqslant 1$. Матрицу M называют $2_{(i,j)}$-*примитивной*, $1 \leqslant i, j \leqslant n$, если существует $\gamma \in \mathbb{N}$ такое, что $m_{ij}^{(t)} = 2$ при любом $t \geqslant \gamma$; наименьшее такое γ называется $2_{(i,j)}$-*экспонентом* матрицы M и обозначается через $\gamma_{2_{(i,j)}}$. Если матрица M не $2_{(i,j)}$-примитивная, то полагаем $\gamma_{2_{(i,j)}} = \infty$.

Матрица M 2-примитивная, если она $2_{(i,j)}$-примитивная для любых $i, j \in \overline{1,n}$, что символически записываем в виде $M^\gamma = (2)$. Для 2-экспонента матрицы M, обозначаемого через γ_2, выполнено свойство

$$\gamma_2 = \exp^{[2]} M = \max_{1 \leqslant i,j \leqslant n} \gamma_{2_{(i,j)}}.$$

Обобщим 2-примитивность, требуя $2_{(i,j)}$-примитивность лишь для собственного подмножества всех пар (i,j), $1 \leqslant i, j \leqslant n$. Пусть $B = (b_{ij})$ — ненулевая двоичная матрица (маска) порядка n, $B * M = (b_{ij} m_{ij})$, умножение $b_{ij} m_{ij}$ выполнено в G, т. е.

$$b_{ij} m_{ij} = \begin{cases} 0, & \text{если } b_{ij} = 0, \\ m_{ij}, & \text{если } b_{ij} = 1. \end{cases}$$

Матрицу $M \in M_n^{[G]}$ назовём

а) 2_B-*полной*, если из равенства $b_{ij} = 1$ следует $m_{ij} = 2$, $1 \leqslant i, j \leqslant n$, символически запишем это свойство так: $M = (2_B)$;

б) 2_B-*примитивной*, если из равенства $b_{ij} = 1$ следует, что M $2_{(i,j)}$-примитивная; 2_B-экспонент матрицы M, обозначаемый через $\exp^{[2_B]} M$ (или кратко через γ_{2_B}), равен

$$\gamma_{2_B} = \max_{(i,j)\colon\, b_{ij}=1} \gamma_{2_{(i,j)}}.$$

Из определений вытекают следующие свойства 2_B-примитивной матрицы $M \in M_n^{[G]}$:

1) M 2_B-примитивная тогда и только тогда, когда она $2_{(i,j)}$-примитивная при всех (i,j) таких, что $b_{ij} = 1$;

2) если M 2-примитивная, то она 2_B-примитивная при любой маске B и $\exp^{[2]} M \geqslant \exp^{[2_B]} M$;

3) *антимонотонность по матрицам*: если $A \leqslant M$ и матрица A 2_B-примитивная, то M тоже 2_B-примитивная и $\exp^{[2_B]} A \geqslant \exp^{[2_B]} M$;

4) *монотонность по маскам*: если $\varnothing \neq E \subseteq B$, то 2_B-примитивная матрица M будет 2_E-примитивной и $\exp^{[2_E]} M \leqslant \exp^{[2_B]} M$;

5) *ненаследственность*: из равенства $M^t = (2_B)$, $t \in \mathbb{N}$, вообще говоря, не следует, что $M^{t+1} = (2_B)$ (при этом заметим, что равенство $M^t = (2)$ влечёт $M^{t+1} = (2)$ для любого $t \in \mathbb{N}$).

В связи со свойством 5 дадим достаточные условия 2_B-полноты всех матриц $M^t, M^{t+1}, \ldots$. Обозначим i-ю строку и j-й столбец матрицы A через $A_{i\to}$ и $A_{j\downarrow}$ соответственно, а их скалярное произведение — через $(A_{i\to}, A_{j\downarrow})$.

Теорема 1. *Если для любых* $1 \leqslant i, j \leqslant n$ *при* $b_{ij} = 1$ *имеет место* $\max\{(m_{i\to}, b_{j\downarrow}), (b_{i\to}, m_{j\downarrow})\} > 0$, *то из равенства* $M^t = (2_B)$, $t \geqslant 1$, *следует, что* $M^{t+1} = (2_B)$ *и* $\gamma_{2_B} = \min\{t \mid M^t = (2_B)\}$.

Доказательство. Из равенств $M^{t+1} = MM^t = M^tM$ получаем

$$m_{ij}^{(t+1)} = \max_{1\leqslant r\leqslant n} m_{ir} m_{rj}^{(t)}, \quad m_{ij}^{(t+1)} = \max_{1\leqslant s\leqslant n} m_{is}^{(t)} m_{sj}.$$

Если $(m_{i\to}, b_{j\downarrow}) > 0$, то $m_{ir} = b_{r,j} > 0$ при некотором $r \in \overline{1,n}$, значит, $m_{ir} m_{rj}^{(t)} = 2m_{ir} = 2$ в силу 2_B-полноты матрицы M^t.

Если $(b_{i\to}, m_{j\downarrow}) > 0$, то $b_{is} = m_{sj} > 0$ при некотором $s \in \overline{1,n}$, значит, $m_{is}^{(t)} m_{sj} = 2m_{sj} = 2$ в силу 2_B-полноты матрицы M^t.

По условию при $b_{ij} = 1$ выполняется хотя бы одно из неравенств $(m_{i\to}, b_{j\downarrow}) > 0$, $(b_{i\to}, m_{j\downarrow}) > 0$. Значит, в обоих случаях $m_{ij}^{(t+1)} = 2$, т. е. $M^{t+1} = (2_B)$. Теорема 1 доказана.

3. Свойства наборов натуральных чисел

Пусть $L = \{l_1, \ldots, l_m\} \subset \mathbb{N}$ — бесповторная выборка. Обозначим через $\langle L\rangle$ порождённую множеством L аддитивную полугруппу, состоящую из множества всех линейных комбинаций (л. к.) чисел множества L; здесь и далее коэффициенты л. к. суть неотрицательные целые; $\langle\varnothing\rangle = \varnothing$.

При НОД $L = 1$ *числом Фробениуса* $F(L)$ называют наибольшее целое число $l \notin \langle L\rangle$; $F(1) = -1$; если $l > F(L)$, то $l \in \langle L\rangle$.

Известна формула $F(L)$ для двух аргументов: $F(l_1, l_2) = l_1l_2 - l_1 - l_2$. При $m > 2$ общей полиномиальной формулы для $F(L)$ не существует [2].

Если НОД $L = d_L \geqslant 1$, то $\frac{L}{d_L} = \left\{\frac{l_1}{d_L}, \ldots, \frac{l_m}{d_L}\right\}$ и НОД $\frac{L}{d_L} = 1$. Обобщая число Фробениуса, положим $\Phi(L) = d_L F\left(\frac{L}{d_L}\right)$, т. е. $\Phi(L)$ — наибольшее кратное d_L целое, не принадлежащее $\langle L\rangle$.

Из определений вытекают следующие свойства:

1) $\Phi(L) = F(L)$ при $d_L = 1$;

2) если $L \neq \varnothing$, то число $\Phi(L)$ существует;
3) $\Phi(l) = -l$ при $l \in \mathbb{N}$, а при $m = 2$ верно $\Phi(l_1, l_2) = \frac{l_1 l_2}{d_L} - l_1 - l_2$;
4) если $l > \Phi(L)$ и l кратно d_L, то $l \in \langle L \rangle$;
5) если $L \supseteq L'$, то $\Phi(L) \leqslant \Phi(L')$.

Элемент $l \in \langle L \cup H \rangle$, где $\varnothing \neq H = \{h_1, \ldots, h_r\} \subset \mathbb{N}$, $r > 0$, представляется (вообще говоря, неоднозначно) как л. к. вида

$$l = \alpha_1 l_1 + \cdots + \alpha_m l_m + \beta_1 h_1 + \cdots + \beta_r h_r,$$

где коэффициенты $\alpha_1, \ldots, \alpha_m, \beta_1, \ldots, \beta_r$ целые неотрицательные. Набор коэффициентов $(\alpha_1, \ldots, \alpha_m, \beta_1, \ldots, \beta_r)$ назовём *представлением числа l в базе* $\{L, H\}$. Множество представлений числа l в базе $\{L, H\}$ обозначим через $\Pi_{\{L,H\}}(l)$.

Назовём *H-весом числа l в базе* $\{L, H\}$ (обозначим $p_H(l)$) наибольшее число ненулевых коэффициентов в наборе $(\beta_1, \ldots, \beta_r)$ среди всех представлений из $\Pi_{\{L,H\}}(l)$, откуда $0 \leqslant p_H(l) \leqslant r$ для любого $l \in \langle L \cup H \rangle$.

Пусть $\langle L \cup [H]^+ \rangle = \{l \in \langle L \cup H \rangle \mid p_H(l) > 0\}$. Имеет место покрытие

$$\langle L \cup [H]^+ \rangle = \bigcup_{s=1}^{r} (h_s + \langle L \cup H \rangle). \tag{1}$$

Лемма 1. $\langle L \cup [H]^+ \rangle \supseteq (\min\limits_{1 \leqslant s \leqslant r} h_s + \Phi(L \cup H) + d_{L \cup H}\mathbb{N})$.

Доказательство следует из (1) и свойства полугруппы $\langle L \cup H \rangle$, так как все числа множества $h_s + \Phi(L \cup H) + d_{L \cup H}\mathbb{N}$ кратны $d_{L \cup H}$, $1 \leqslant s \leqslant r$. Лемма 1 доказана.

Пример 1. Пусть $L = \{15, 45\}$, $H = \{9, 21, 27\}$. Тогда $d_{L \cup H} = 3$, $\Phi(L \cup H) = 12$ и $\langle L \cup H \rangle = \langle 9, 15, 21, 27, 45 \rangle = \{0, 9, 15, 21\} \cup (21 + 3N) = \{0, 9\} \cup (12 + 3N)$.

Имеется единственное представление числа 24: $\Pi_{\{L,H\}}(24) = \{9 + 15\}$, $p_H(24) = 1$. Имеются 5 представлений числа 54:

$\Pi_{\{L,H\}}(54) = \{6 \cdot 9,\ 3 \cdot 9 + 27,\ 2 \cdot 9 + 15 + 21,\ 9 + 3 \cdot 15,\ 2 \cdot 27\}$, $p_H(54) = 2$. В соответствии с (1) имеем покрытие

$$\langle L \cup [H]^+ \rangle = (9 + \langle L \cup H \rangle) \cup (21 + \langle L \cup H \rangle) \cup (27 + \langle L \cup H \rangle).$$

Пусть $d, l \in \mathbb{N}$ и $\varphi \in \{-d\} \cup d\mathbb{N}$, т. е. φ кратно d. Бесповторную выборку $S(\varphi, d, l) = (l + B) \cup (l + \varphi + d\mathbb{N}) \subseteq \mathbb{N}$ назовём *(φ, d, l)-выборкой*, если $B \subseteq \{d, \ldots, \varphi - d\} \subseteq d\mathbb{N}$ при $\varphi > d$ и $B = \varnothing$ при $\varphi \in \{-d, d\}$. Заметим, что $l + \varphi \notin S(\varphi, d, l)$ при $\varphi > d$ и $b \equiv_d l$ для любого $b \in S(\varphi, d, l)$.

Множество выборок $\{S(\varphi_i, d_i, l_i)\}_{i=1}^{k}$ назовём *полным*, если имеет место включение $\bigcup\limits_{i=1}^{k} S(\varphi_i, d_i, l_i) \supseteq \mathbb{N}^{\lfloor q \rfloor}$ при некотором $q \in \mathbb{N}$. Например, полными будут выборка $S(\varphi, 1, l)$ при любых φ и l (тогда $q = l + \varphi$),

а также множество выборок $\{S(\varphi,d,l)\}_{l=1}^{d}$ при $\varphi>0$ и при любом d (тогда $q=d+\varphi$).

Положим $S=\bigcup\limits_{i=1}^{k}S(\varphi_i,d_i,l_i)$, $h=\max\limits_{1\leqslant i\leqslant k}\{\varphi_i+l_i\}$.

Теорема 2. *Положим* $D=\mathrm{НОК}\{d_1,\ldots,d_k\}$. *Множество выборок* $\{S(\varphi_i,d_i,l_i)\}_{i=1}^{k}$ *полное тогда и только тогда, когда полна система вычетов* $\{S^{(h,h+D]}\equiv_D\}$, *при этом*

$$\max\mathbb{N}\setminus S=\max\bigcap_{i=1}^{k}(\mathbb{N}^{\lceil\varphi_i+l_i\rceil}\setminus S(\varphi_i,d_i,l_i)^{\lceil\varphi_i+l_i\rceil}).$$

Вычисление $\max\mathbb{N}\setminus S$ *по множеству выборок* $\{S(\varphi_i,d_i,l_i)\}_{i=1}^{k}$ *требует менее* $k\lceil\log_2 k\rceil+\lceil\log_2 h\rceil\sum\limits_{i=1}^{k}\varphi_i$ *сравнений целых неотрицательных чисел.*

Доказательство. В силу того, что $d_i\mid D$, $i=1,\ldots,k$, выполнено $S(\varphi_i,d_i,l_i)\supseteq l_i+\varphi_i+D\mathbb{N}$. Следовательно, $S\supseteq h+D\mathbb{N}$. Отсюда равны множества вычетов $\{S^{(h+(s-1)D,h+sD]}\equiv_D\}=\{S^{(h,h+D]}\equiv_D\}$, $s\in\mathbb{N}$. Значит, полнота данного множества выборок равносильна полноте системы вычетов $\{S^{(h,h+D]}\equiv_D\}$.

В силу свойств операций объединения и пересечения множеств имеем

$$\mathbb{N}\setminus S=\mathbb{N}\setminus\bigcup_{i=1}^{k}S(\varphi_i,d_i,l_i)=\bigcap_{i=1}^{k}(\mathbb{N}\setminus S(\varphi_i,d_i,l_i)).$$

Заметим, что $\mathbb{N}\setminus S(\varphi_i,d_i,l_i)=\mathbb{N}^{\lceil\varphi_i+l_i\rceil}\setminus S(\varphi_i,d_i,l_i)^{\lceil\varphi_i+l_i\rceil}$, так как если целое $l>\varphi_i+l_i$ и $d_i\mid l-l_i$, то $l\in S(\varphi_i,d_i,l_i)$ по определению выборки. В силу полноты множества вычетов $\{S^{(h,h+D]}\equiv_D\}$ для любого целого $l>h$ в множестве $\{S(\varphi_i,d_i,l_i)\}_{i=1}^{k}$ найдётся содержащая число l выборка.

Множество $S^{(h,h+D]}$ состоит из чисел l, где $h<l\leqslant h+D$, содержащихся хотя бы в одной выборке множества $\{S(\varphi_i,d_i,l_i)\}_{i=1}^{k}$. Следовательно,

$$S^{(h,h+D]}=\bigcup_{i=1}^{k}\{\varphi_i+l_i+td_i\in(h,h+D]\mid t\in\mathbb{N}\}.$$

Оценим сложность вычисления числа $\gamma=\max\bigcap\limits_{i=1}^{k}(\mathbb{N}\setminus S(\varphi_i,d_i,l_i))$, упорядочив выборки по правилу $\varphi_t+l_t\geqslant\varphi_{t+1}+l_{t+1}$, $t=1,\ldots,k-1$.

Пусть $L_0=\mathbb{N}^{\lceil\varphi_1+l_1\rceil}$ — упорядоченный массив чисел.

Найдём γ за k шагов. На шаге i все числа $l\in S(\varphi_i,d_i,l_i)^{\lceil\varphi_i+l_i\rceil}$ удаляем из массива L_{i-1}. Полагаем $L_i=L_{i-1}\setminus S(\varphi_i,d_i,l_i)^{\lceil\varphi_i+l_i\rceil}$, $i=1,\ldots,k$. Наибольшее число массива L_k равно γ.

Для оценки вычислительной сложности алгоритма заметим, что

- $\varphi_1 + l_1 = h$;
- $\left|S(\varphi_i, d_i, l_i)^{\lceil \varphi_i + l_i \rceil}\right| < \varphi_i$ в силу определения (φ, d, l)-выборки;
- так как $|L_i| < h$, $i = 1, \ldots, k$, для поиска любого числа в упорядоченном массиве L_i требуется менее $\lceil \log_2 h \rceil$ сравнений чисел.

Предварительное упорядочение выборок требует не более $k\lceil \log_2 k \rceil$ сравнений. Тогда сложность алгоритма меньше $k\lceil \log_2 k \rceil + \lceil \log_2 h \rceil \sum\limits_{i=1}^{k} \varphi_i$. Теорема 2 доказана.

4. Усложнение простых путей орграфа

Пусть Γ_n — класс орграфов с множеством вершин $\overline{1,n}$.

Введём следующие обозначения для орграфа $\Gamma \in \Gamma_n$:

- $P(i,j)$ — множество всех простых путей из i в j, $1 \leqslant i, j \leqslant n$;
- l_w (или l_C) — длина пути w (или контура C), равная числу дуг, составляющих путь (или контур);
- $L(W) = \{l_w \mid w \in W\}$, где W — множество путей;
- $w \parallel w'$ — конкатенация (присоединение) путей w и w', где конечная вершина пути w совпадает с начальной вершиной пути w'.

Контур, проходящий через вершину пути w, назовём *w-инцидентным*. Контур C назовём *w-достижимым*, если он w-инцидентный или существует путь из w в некоторую вершину контура C. Путь w назовём *достижимым* из контура C, если этот контур w-инцидентный или существует путь из C в некоторую вершину пути w. При выполнении обоих условий путь и контур называются *взаимно достижимыми*.

В сильно связном орграфе любые путь и контур взаимно достижимы.

Пусть $P(i,j) \neq \varnothing$. Введём следующие обозначения:

- $\widetilde{\Gamma}_n^{(i,j)}$ — класс орграфов из Γ_n, в которых для любого пути $w \in P(i,j)$ каждый w-достижимый контур w-инцидентный;
- $\widehat{C}_w$ — множество всех w-инцидентных контуров, $w \in P(i,j)$;
- $C^{[k]}(v)$ — пройденный k-кратно из начальной вершины v контур C, $k \geqslant 0$;

Индексом (*$\widehat{C}_w$-индексом*) пути w назовём число $d_w = \text{НОД}(L(\widehat{C}_w))$, $d_w = 0$, если $\widehat{C}_w = \varnothing$. Подмножество $\widehat{C}_w'$ множества $\widehat{C}_w$ назовём *d_w-минимальным*, если $d(\widehat{C}_w') = d_w$ и из включения $\widehat{C}_w'' \subset \widehat{C}_w'$ следует, что $d(\widehat{C}_w'') > d_w$.

Без ограничения общности далее считаем, что множество $\widehat{C}_w$ d_w-минимально и $\widehat{C}_w = \{C_1, \ldots, C_m\}$, где $d_w \geqslant 1$ и $m \geqslant 1$.

На множестве путей орграфа Γ определим бинарное отношение «усложнения», которое обозначим через $\lesssim$. Если $w = (v_0, \ldots, v_t)$ — некоторый

путь и v_r — вершина w-инцидентного контура C, $0 \leqslant r \leqslant t$, то $C^{[k]}(v_r)$-*усложнением* пути w назовём его усложнение с помощью k-кратного обхода контура C, $k \geqslant 0$:

$$w(C^{[k]}(v_r)) = \begin{cases} C^{[k]}(v_0) \parallel w, & \text{если } r = 0, \\ (v_0, \ldots, v_r) \parallel C^{[k]}(v_r) \parallel (v_r, \ldots, v_t), & \text{если } 0 < r < t, \\ w \parallel C^{[k]}(v_t), & \text{если } r = t. \end{cases}$$

Путь $C^{[0]}(v_r)$ пустой. В соответствии с определением конкатенации путей

$$l_{w\parallel w'} = l_w + l_{w'}, \quad l_{w(C^{[k]}(v_r))} = l_w + kl_C.$$

Усложнение пути w с помощью контуров из $\widehat{C}_w$ назовём $\widehat{C}_w$-*усложнением*. Если контур C_s обходится k_s-кратно, $s = 1, \ldots, m$, то $C^{[k]}(v_r)$-усложнение пути w имеет длину $l_w + \sum\limits_{s=1}^{m} k_s l_s$. Множество вариантов кратности обхода контуров счётное, и w имеет счётное множество усложнений, если $\widehat{C}_w \neq \varnothing$. Если $\widehat{C}_w$-усложнение пути w усложнить некоторым контуром, не инцидентным пути w, то получим усложнение, отличное от $\widehat{C}_w$-усложнений. Такие усложнения пути w назовём w-*свободными*.

Введём следующие обозначения:

- $P^{(0)}$ — класс простых путей $w \in P(i,j)$ индекса $d_w = 0$;
- $P^{(+)}$ — класс простых путей $w \in P(i,j)$ индекса $d_w > 0$;
- U_w — класс всех усложнений простого пути w положительного индекса, как w-свободных, так и $\widehat{C}_w$-усложнений;
- $\widehat{U}_w$ — класс всех $\widehat{C}_w$-усложнений простого пути w положительного индекса; при $\widehat{C}_w = \varnothing$ имеем $\widehat{U}_w = \{w\}$, $L(\widehat{U}_w) = \{l_w\}$;
- $\widehat{U} = \bigcup\limits_{w \in P^{(+)}} \widehat{U}_w$;
- $U = \bigcup\limits_{w \in P^{(+)}} U_w$;
- $\Phi_w = \Phi(L(\widehat{C}_w))$;
- $\Phi^{(i,j)} = \max\limits_{w \in P^{(+)}} \Phi_w$;
- $h^{(i,j)} = \max\limits_{w \in P^{(+)}} (l_w + \Phi_w)$;
- $D = \text{НОК}\{d_w \mid w \in P^{(+)}\}$.

Теорема 3 (о длинах усложнений пути). *В орграфе* $\Gamma \in \Gamma_n$ *множество* $L(\widehat{U}_w)$ *длин всех* $\widehat{C}_w$*-усложнений пути* $w \in P^{(+)}$ *есть* (Φ_w, d_w, l_w)*-выборка:*

$$L(\widehat{U}_w) = l_w + \langle l_1, \ldots, l_m \rangle \supseteq l_w + \Phi_w + d_w\mathbb{N}.$$

Доказательство. Если при $\widehat{C}_w$-усложнении простого пути w контур C_s обходится k_s-кратно, $s = 1, \ldots, m$, то длина усложнения равна

$l_w + k_1 l_1 + \cdots + k_m l_m$. Тогда $L(\widehat{U}_w) = l_w + \langle l_1, \ldots, l_m \rangle$, так как допустима любая кратность обхода контуров из $\widehat{C}_w$.

По свойству числа Ф полугруппа $\langle l_1, \ldots, l_m \rangle$ содержит все числа, кратные d_w и превышающие Φ_w. Значит, $L(\widehat{U}_w) \supseteq l_w + \Phi_w + d_w\mathbb{N}$. Теорема 3 доказана.

Следствие 1. *Для любого $l \in L(\widehat{U}_w)$ имеет место сравнение $l \equiv_{d_w} l_w$.*

Доказательство. Число Φ_w и любое число из $\langle l_1, \ldots, l_m \rangle$ кратно d_w, так как $d_w \mid l_s$, $s = 1, \ldots, m$. Тогда $l \equiv_{d_w} l_w$ в силу теоремы 3. Следствие 1 доказано.

5. О локальной примитивности орграфа

В силу имеющейся биекции $M_n \leftrightarrow \Gamma_n$, где M_n — множество двоичных матриц порядка n, локальные примитивности матриц и орграфов равносильны. Используем более удобную графовую модель.

С помощью множества длин путей получим для орграфа $\Gamma \in \Gamma_n$ критерий (i,j)-примитивности и оценку (i,j)-экспонента. Покажем, что для орграфов $\Gamma \in \widetilde{\Gamma}_n^{(i,j)}$ данная оценка совпадает с точным значением.

Напомним, что Γ называется (i,j)-*примитивным*, $1 \leqslant i, j \leqslant n$, если найдётся число $\gamma \in \mathbb{N}$ такое, что для любого $t \geqslant \gamma$ в Γ имеется путь длины t из i в j (или дуга (i,j) в Γ^t); наименьшее такое γ называют (i,j)-*экспонентом* орграфа Γ (обозначается $\exp^{(i,j)}\Gamma$ или кратко $\gamma_{(i,j)}$).

По определению в (i,j)-примитивном орграфе имеются простой путь $w \in P^{(+)}$ и некоторый w-инцидентный контур. Иначе все простые пути имеют индекс 0 и число их усложнений конечно в силу тривиальности усложнений, что противоречит (i,j)-примитивности орграфа.

Теорема 4 (критерий (i,j)-примитивности орграфа). *Граф $\Gamma \in \Gamma_n$ (i,j)-примитивный тогда и только тогда, когда множество $\mathbb{N} \setminus L(U)$ конечное и $\gamma_{(i,j)} \leqslant 1 + \varepsilon_{(i,j)}$, где*

$$\varepsilon_{(i,j)} = \max \bigcap_{w \in P^{(+)}} (\mathbb{N}^{\lceil l_w + \Phi_w \rceil} \setminus L(\widehat{U}_w)^{\lceil l_w + \Phi_w \rceil}) \setminus L(P^{(0)}).$$

Если $\Gamma \in \widetilde{\Gamma}_n^{(i,j)}$, то Γ (i,j)-примитивный тогда и только тогда, когда полна система вычетов $\{L(\widehat{U})^{(h^{(i,j)}, h^{(i,j)}+D]} \equiv_D\}$ и $\gamma_{(i,j)} = 1 + \varepsilon_{(i,j)}$.

Вычисление $\varepsilon_{(i,j)}$ требует менее

$$|P^{(+)}| \lceil \log_2 |P^{(+)}| \rceil + \Big(\sum_{w \in P^+} \Phi_w + |P^{(0)}| \Big) \lceil \log_2 h^{(i,j)} \rceil$$

сравнений целых чисел, не больших $\max\{n, \Phi^{(i,j)}\}$.

Доказательство. В соответствии с определением орграф $\Gamma \in \Gamma_n$ (i,j)-примитивный с (i,j)-экспонентом $\gamma_{(i,j)}$ тогда и только тогда, когда $L(U) \supseteq \mathbb{N}^{\lfloor \gamma_{(i,j)}-1 \rfloor}$ и $(\gamma_{(i,j)}-1) \notin L(U)$. Значит, $L(U)$ содержит все числа, не меньшие $\gamma_{(i,j)}$, и $\gamma_{(i,j)} = 1 + \max \mathbb{N} \setminus L(U)$.

Так как $\widehat{U}_w \subseteq U_w$, $w \in P^{(+)}$, то $L(\widehat{U}_w) \subseteq L(U_w)$, отсюда $\Phi(L(\widehat{U}_w)) \geqslant \Phi(L(U_w))$ в силу свойства 5 чисел Φ. Следовательно, $\max\limits_{w\in P^{(+)}} \Phi(L(\widehat{U}_w)) \geqslant \max\limits_{w\in P^{(+)}} \Phi(L(U_w))$, значит, $\max \mathbb{N} \setminus L(\widehat{U}) \geqslant \max \mathbb{N} \setminus L(U) = \gamma_{(i,j)} - 1$.

По теореме 3 множество $L(\widehat{U}_w)$ представляет собой (Φ_w, d_w, l_w)-выборку. Объединение этих выборок для всех путей $w \in P^{(+)}$ есть $L(\widehat{U})$. По теореме 2 $L(\widehat{U}) = \bigcup\limits_{w\in P^{(+)}} L(\widehat{U}_w)$ полное тогда и только тогда, когда полна система вычетов $\{L(\widehat{U})^{(h^{(i,j)},h^{(i,j)}+D]} \equiv_D\}$.

В соответствии с теоремой 2 и с учётом путей нулевого индекса

$$\max \mathbb{N} \setminus L(\widehat{U}) = \max \bigcap_{w\in P^{(+)}} (\mathbb{N}^{\lceil l_w+\Phi_w \rceil} \setminus L(\widehat{U}_w)^{\lceil l_w+\Phi_w \rceil}) \setminus L(P^{(0)}) = \varepsilon_{(i,j)}.$$

Если $\Gamma \in \widetilde{\Gamma}_n^{(i,j)}$, то $L(U) = L(\widehat{U})$ и $1 + \varepsilon_{(i,j)}$ есть точное значение.

Количество сравнений целых чисел для вычисления $\varepsilon_{(i,j)}$ ограничено упорядочиванием множества $P^{(+)}$ и поиском чисел из $\mathbb{N}\setminus L(\widehat{U})$ в упорядоченном массиве мощности, меньшей $\Phi^{(i,j)}$. Сравниваемые числа не превышают n (количества разных длин простых путей), и числа усложнений длины не выше $l_w + \Phi_w$, которое не больше Φ_w для всех путей $w \in P^{(+)}$. Теорема 4 доказана.

Замечание 1. Множество $\{L(\widehat{U})^{(h^{(i,j)},h^{(i,j)}+D]} \equiv_D\}$ состоит из длин l всех $\widehat{C}_w$-усложнений путей $w \in P^{(+)}$, где $h^{(i,j)} < l \leqslant h^{(i,j)} + D$. Следовательно,

$$\{L(\widehat{U})^{(h^{(i,j)},h^{(i,j)}+D]} \equiv_D\} = \bigcup_{w\in P^{(+)}} \{l_w+\Phi_w+sd_w \in (h^{(i,j)},\, h^{(i,j)}+D] \mid s \in \mathbb{N}\}.$$

6. Об усложнениях путей помеченного орграфа

Введём следующие обозначения:

- $\Gamma_{n,G}$ — множество орграфов с множествами вершин $\overline{1,n}$ и меток дуг G;
- $\widetilde{\Gamma}_{n,G}^{(i,j)}$ — множество орграфов из $\widetilde{\Gamma}_n^{(i,j)}$ с множеством меток дуг G;
- $\theta\colon M_n^{[G]} \leftrightarrow \Gamma_n^{[G]}$ — биекция, при которой матрице $M = (m_{ij})$ порядка n над G соответствует орграф $\Gamma = \theta(M)$ с матрицей меток M.

В $\Gamma_{n,G}$ дуга (i,j) с меткой m_{ij} задана тройкой (i,m_{ij},j). Пути $w=(y_0,\dots,y_t)$ длины t соответствует слово $(m_1,\dots,m_t)$, где m_s — метка дуги (y_{s-1},y_s), $s=1,\dots,t$. Произведение $m=m_1\dots m_t$ (умножение в G) назовём *меткой* пути w:

$$m=\begin{cases}\max\limits_{1\leqslant s\leqslant t} m_s, & \text{если } m_s>0 \text{ при } s\in\{1,\dots,t\},\\ 0, & \text{если } m_s=0 \text{ для некоторого } s\in\{1,\dots,t\}.\end{cases}$$

Путь (контур) с меткой $g\in\{1,2\}$ назовём *g-путём* (g-контуром).

Если в орграфе Γ имеются пути $w_1,\dots,w_r$ из i в j длины t с метками $m_1,\dots,m_r$ соответственно, то метка дуги (i,j) в орграфе Γ^t есть произведение $m_1\cdots m_r$.

Пусть орграф $\Gamma'\in\Gamma_{n,G}$ имеет матрицу меток (μ_{ij}). Обозначим через V_{ij} множество вершин v таких, что (i,v) — дуга в Γ и (v,j) — дуга в Γ'. Тогда в орграфе $\Gamma\Gamma'$ при $V_{ij}=\varnothing$ метка дуги (i,j) есть 0, а при $V_{ij}\neq\varnothing$ имеется дуга (i,b_{ij},j) с меткой $b_{ij}=\max\limits_{v\in V_{ij}} m_{iv}\mu_{vj}$. Отсюда следует

Теорема 5 (о метках дуг). *Пусть орграф $\Gamma\in\Gamma_{n,G}$ имеет матрицу меток $M=(m_{ij})$. Тогда метка $m_{ij}^{(t)}$ дуги (i,j) в орграфе Γ^t равна наибольшей среди меток всех путей длины $t\geqslant 1$ из вершины i в вершину j орграфа Γ:*

$$m_{ij}^{(t)}=\begin{cases}0, & \textit{если } j \textit{ недостижима из } i \textit{ за } t \textit{ шагов в } \Gamma,\\ 1, & \textit{если каждый путь из } i \textit{ в } j \textit{ длины } t \textit{ в } \Gamma \textit{ есть 1-путь},\\ 2, & \textit{если имеется 2-путь из } i \textit{ в } j \textit{ длины } t \textit{ в } \Gamma.\end{cases}$$

Для $w\in P^{(+)}$ выполнено разбиение $\widehat{C}_w=\widehat{C}_w^1\cup\widehat{C}_w^2$ (пустые блоки допустимы), где $\widehat{C}_w^g$ — множество всех w-инцидентных g-контуров, $g\in\{1,2\}$.

Назовём $\widehat{C}_w^{[g]}$-*усложнением* пути w его $\widehat{C}_w$-усложнение с меткой g.

В орграфе $\Gamma\in\widetilde{\Gamma}_{n,G}^{(i,j)}$ опишем длины всех $\widehat{C}_w^{[g]}$-усложнений простого пути w с учётом меток путей и контуров. Пусть при $g\in\{1,2\}$

- $P^{(g,0)}$ — множество простых g-путей из i в j с нулевым индексом;
- $P^{(g,+)}$ — множество простых g-путей из i в j с положительным индексом;
- $\widehat{U}_w^{(g)}$ — множество всех $\widehat{C}_w^{[g]}$-усложнений пути w;
- $\widehat{U}^{(g)}=\bigcup\limits_{w\in P^{(+)}}\widehat{U}_w^{(g)}$;
- $\lambda_w=\min L\big(\widehat{C}_w^2\big)$ для пути $w\in P^{(1,+)}$;
- $\delta_w=l_w+\Phi_w+\lambda_w$ для пути $w\in P^{(1,+)}$;
- $\delta^{(i,j)}=\max\limits_{w\in P^{(+)}}\delta_w$.

Теорема 6 (о длинах $\widehat{C}_w^{[g]}$-усложнений). *Если $w \in P^{(+)}$, то*

$$L(\widehat{U}_w^{(1)}) = \begin{cases} l_w + \langle L(\widehat{C}_w^1)\rangle, & \text{если } w \in P^{(1,+)}, \\ \varnothing, & \text{если } w \in P^{(2,+)}, \end{cases}$$

$$L(\widehat{U}_w^{(2)}) = \begin{cases} \varnothing, & \text{если } w \in P^{(1,+)},\ \widehat{C}_w^2 = \varnothing, \\ l_w + \big(L(\widehat{C}_w^1) \cup \big[L(\widehat{C}_w^2)\big]^+\big), & \text{если } w \in P^{(1,+)},\ \widehat{C}_w^2 \neq \varnothing, \\ l_w + \langle L(\widehat{C}_w)\rangle, & \text{если } w \in P^{(2,+)}. \end{cases}$$

Доказательство. В силу теоремы 5 усложнение h-контуром g-пути w есть gh-путь. Тогда 1-усложнение возможно лишь 1-контурами для 1-пути w, и сумма длин усложняющих контуров есть число полугруппы $\langle L(\widehat{C}_w^1)\rangle$.

Для 2-пути любые контуры дают 2-усложнения, для 1-пути w лишь набор контуров, содержащий 2-контур, даёт 2-усложнение, тогда сумма длин усложняющих контуров есть число множества $L(\widehat{C}_w^1) \cup \big[L(\widehat{C}_w^2)\big]^+$. Формулы для $L(\widehat{U}_w^{(g)})$ доказаны, $g \in \{1, 2\}$. Теорема 6 доказана.

Следствие 2. *Для путей $w \in P^{(+)}$ верны включения*

$$L(\widehat{U}_w^{(2)}) \supseteq \begin{cases} \delta_w + d_w\mathbb{N}, & \text{если } w \in P^{(1,+)},\ \widehat{C}_w^2 \neq \varnothing, \\ l_w + \Phi_w + d_w\mathbb{N}, & \text{если } w \in P^{(2,+)},\ \widehat{C}_w \neq \varnothing. \end{cases}$$

Доказательство вытекает из теоремы 6 и леммы 1. Следствие 2 доказано.

7. О локальной 2-примитивности помеченного орграфа

Пусть $\Gamma \in \Gamma_{n,G}$ и $M = (m_{ij})$ — матрица меток. Обозначим через Γ_B орграф с матрицей меток $B * M$, где B — двоичная маска порядка n для орграфа Γ. Орграфы Γ и Γ_B назовём 2_B-*полными*, если $B * M = (2_B)$. Орграф Γ называется $2_{(i,j)}$-*примитивным*, $1 \leqslant i, j \leqslant n$, если существует число $\gamma \in \mathbb{N}$ такое, что для любого $t \geqslant \gamma$ в Γ имеется 2-путь длины t из i в j (или в Γ^t имеется дуга (i, j) с меткой 2); наименьшее такое γ обозначим $\exp^{[2_{(i,j)}]}\Gamma$ (кратко $\gamma_{2_{(i,j)}}$) и назовём $2_{(i,j)}$-*экспонентом* орграфа Γ. Орграф Γ называется 2_B-*примитивным*, если из равенства $b_{ij} = 1$ следует $2_{(i,j)}$-примитивность. Величину $\exp^{[2_B]}\Gamma$, кратко γ_{2_B}, назовём 2_B-*экспонентом*: $\gamma_{2_B} = \max\limits_{(i,j)\colon b_{ij}=1} \gamma_{2_{(i,j)}}$.

Свойства 2_B-примитивного n-вершинного орграфа Γ:

1) если $\Gamma = \theta(M)$, то $\exp^{[2_B]}\Gamma = \exp^{[2_B]} M$;

2) если орграф $2_{(i,j)}$-примитивный, то $P^{(1,+)} \cup P^{(2,+)} \neq \varnothing$;

3) если орграф 2-примитивный, то при любой маске B он 2_B-примитивный и $\exp^{[2]}\Gamma \geqslant \exp^{[2_B]}\Gamma$;

4) 2_B-примитивный орграф при некоторой маске B не обязательно связный или сильно связный;

5) *монотонность по маскам*: если $A \leqslant B$, то 2_B-примитивный орграф Γ является 2_A-примитивным и $\exp^{[2_A]}\Gamma \leqslant \exp^{[2_B]}\Gamma$.

Множество $L(U^{(2)})$ длин 2-усложнений простых путей из i в j вполне определяет $2_{(i,j)}$-примитивность помеченного орграфа и его $2_{(i,j)}$-экспонент.

Пусть

- $\Lambda^{(1)} = \bigcap\limits_{w\in P^{(1,+)}} \big(\mathbb{N}^{\lceil\delta_w\rceil} \setminus L\big(U_w^{(2)}\big)\big)^{\lceil\delta_w\rceil}$;
- $\Lambda^{(2)} = \bigcap\limits_{w\in P^{(2,+)}} \big(\mathbb{N}^{\lceil l_w+\Phi_w\rceil} \setminus L\big(U_w^{(2)}\big)\big)^{\lceil l_w+\Phi_w\rceil}$.

Теорема 7 (критерий $2_{(i,j)}$-примитивности). *Пусть для $\Gamma \in \Gamma_{n,G}$ имеем $P^{(+)} \neq \varnothing$. Орграф Γ $2_{(i,j)}$-примитивный тогда и только тогда, когда система вычетов $\{L(U^{(2)})^{(\sigma^{(i,j)},\,\sigma^{(i,j)}+D]} \equiv_D\}$ полна, где*

$$\sigma^{(i,j)} = \begin{cases} \delta^{(i,j)}, & \text{если } P^{(2,+)} = \varnothing, \\ h^{(i,j)} & \text{иначе.} \end{cases}$$

При этом для $2_{(i,j)}$-экспонента верна оценка

$$\gamma_{2_{(i,j)}} \leqslant 1 + \begin{cases} \max\Lambda^{(1)} \setminus L(P^{(0)}), & \text{если } P^{(2,+)} = \varnothing, \\ \max\Lambda^{(2)} \setminus L(P^{(0)}), & \text{если } P^{(1,+)} = \varnothing, \\ \max(\Lambda^{(1)} \cap \Lambda^{(2)}) \setminus L(P^{(0)}) & \text{иначе.} \end{cases} \qquad (2)$$

В частности, если $\Gamma \in \widetilde{\Gamma}_{n,G}^{(i,j)}$, то в (2) имеет место равенство.

Вычисление оценок требует менее

$$|P^{(+)}|\lceil\log_2|P^{(+)}|\rceil + \Big(\sum_{w\in P^+}\Phi_w + |P^{(0)}|\Big)\lceil\log_2\sigma^{(i,j)}\rceil$$

сравнений целых чисел, не больших $\max\{n,\, \Phi^{(i,j)}\}$.

Доказательство. Если орграф Γ $2_{(i,j)}$-примитивный с $2_{(i,j)}$-экспонентом, равным $\gamma_{2_{(i,j)}}$, то $L(U^{(2)})$ не содержит $\gamma_{2_{(i,j)}} - 1$ и содержит все целые числа, не меньшие $\gamma_{2_{(i,j)}}$. Значит, $L(U^{(2)})$ содержит все числа, не меньшие $\gamma_{2_{(i,j)}}$, и $\gamma_{2_{(i,j)}} = 1 + \max\mathbb{N} \setminus L(U^{(2)})$.

Пусть $\Gamma \in \widetilde{\Gamma}_{n,G}^{(i,j)}$. В соответствии с теоремой 3 возможны 3 случая.

1) $P^{(2,+)} = \varnothing$. Тогда $L(U_w)$ есть $(\Phi_w + \lambda_w, d_w, l_w)$-выборка. Полнота объединения $L(U^{(2)})$ выборок для путей $w \in P^{(1,+)}$ по теореме 2 равносильна полноте системы $\{L(U^{(2)})^{(\delta^{(i,j)},\,\delta^{(i,j)}+D]} \equiv_D\}$, откуда с учётом путей нулевого индекса $\gamma_{2_{(i,j)}} - 1 = \max\Lambda^{(1)} \setminus L(P^{(0)})$.

2) $P^{(1,+)} = \varnothing$. Тогда $L(U_w)$ есть (Φ_w, d_w, l_w)-выборка. Полнота объединения $L(U^{(2)})$ выборок для путей $w \in P^{(2,+)}$ по теореме 2 равносильна полноте системы $\{L(U)^{(h^{(i,j)}, h^{(i,j)}+D]} \equiv_D\}$, откуда с учётом $P^{(0)}$ имеем

$$\gamma_{2_{(i,j)}} - 1 = \max \Lambda^{(2)} \setminus L(P^{(0)}).$$

3) $P^{(1,+)} \neq \varnothing$ и $P^{(2,+)} \neq \varnothing$. Тогда $L(U_w)$ есть (Φ_w, d_w, l_w)-выборка. Полнота объединения $L(U^{(2)})$ выборок для путей $w \in P^{(+)}$ по теореме 3 равносильна полноте системы вычетов $\{L(U)^{(h^{(i,j)}, h^{(i,j)}+D]} \equiv_D\}$, откуда с учётом $P^{(0)}$

$$\gamma_{2_{(i,j)}} - 1 = \max(\Lambda^{(1)} \cap \Lambda^{(2)}) \setminus L(P^{(0)}).$$

Если $\Gamma \in \Gamma_{n,G}$, то множество $L(U^{(2)})$ может быть шире, чем для $\Gamma \in \widetilde{\Gamma}_{n,G}^{(i,j)}$, т. е. возможно снижение значения $\gamma_{2_{(i,j)}}$ по сравнению с оценкой.

Алгоритм вычисления оценок $\gamma_{2_{(i,j)}}$ состоит в упорядочивании множества $P^{(+)}$ (что требует не более $|P^{(+)}|\lceil \log_2 |P^{(+)}| \rceil$ сравнений чисел) и поиске чисел из $\mathbb{N} \setminus L(\widehat{U})$ в упорядоченном массиве мощности, меньшей $\Phi^{(i,j)}$. Сравниваемые числа не превышают n (количества разных длин простых путей), а числа усложнений длины не больше значения $l_w + \Phi_w$, ограниченного числом Φ_w для каждого пути $w \in P^{(+)}$. Во всех случаях в соответствии с теоремой 3 и с учётом путей из $P^{(0)}$ оценка сложности вычисления $\gamma_{2_{(i,j)}}$ имеет нужный вид.

При оценке вычислительной сложности алгоритма учтено, что

1) $l_{w_1} + \Phi_{w_1} + \lambda_{w_1} = \delta^{(i,j)}$ в силу упорядоченности путей;

2) $\left|L\big(U_{w_t}^{(2)}\big)^{\lceil \delta^{(i,j)} \rceil}\right| < \Phi_{w_t}$;

3) $|L^{(0)}| \leqslant |P^{(2,0)}|$, так как усложнения пути индекса 0 тривиальны;

4) в силу неравенств $|L_t| \leqslant \sigma^{(i,j)}$, $t = 0, \ldots, r$, для поиска числа в упорядоченном массиве L_t требуется не более $\lceil \log_2 \sigma^{(i,j)} \rceil$ сравнений чисел.

Теорема 7 доказана.

Пример 2. Для 5-вершинного орграфа Γ, заданного списком помеченных дуг (1,1,2), (1,1,3), (1,1,4), (2,1,3), (2,2,5), (3,1,4), (4,1,2), распознаем его $2_{(1,5)}$-примитивность, при положительном результате найдём $\exp^{[2_{(1,5)}]} \Gamma$.

Орграф Γ принадлежит $\widetilde{\Gamma}_{5,G}^{(1,5)}$ и содержит 1-контур $C = (2,3,4)$ длины $l = 3$, инцидентный простым 2-путям $(1,2,5)$, $(1,4,2,5)$ и $(1,3,4,2,5)$, $P^{(1,+)} = \varnothing$.

Характеристики путей из 1 в 5: $d = D = 3$, $\Phi(L(C)) = \Phi(3) = -3$, $h^{(1,5)} = \max\{2-3, 3-3, 4-3\} = 1$, $L(U^{(2)}) = \{2,3,4\}$. По теореме 10

$$\{L(U^{(2)})^{(h^{(1,5)},\, h^{(1,5)}+D]} \equiv_D\} = L(U^{(2)})^{(1,4]} = \{2,3,4\},$$

т. е. множество вычетов полное: $\{\{2,3,4\} \equiv_3\} = \{0,1,2\}$. Тогда по теореме 7 орграф $2_{(1,5)}$-примитивный и $\gamma_{2_{(1,5)}} = 2$.

Пример 3. Для 8-вершинного орграфа Γ, заданного списком помеченных дуг (1,1,2), (1,1,3), (2,1,3), (3,1,4), (3,1,5), (4,1,2), (5,2,6), (6,1,7), (6,2,8), (7,1,3), распознаем $2_{(1,8)}$-примитивность, и при наличии свойства найдём $\exp^{[2_{(1,8)}]}\Gamma$.

Орграф Γ принадлежит $\widetilde{\Gamma}_{8,G}^{(1,8)}$ и содержит 1-контур $C_1 = (2,3,4)$ длины $l_1 = 3$ и 2-контур $C_2 = (3,5,6,7)$ длины $l_2 = 4$. Тогда $\widehat{C} = \{C_1, C_2\}$ и через контуры проходят 2 простых пути из 1 в 8: 2-путь $(1,3,5,6,8)$ длины 4 и 2-путь $(1,2,3,5,6,8)$ длины 5; C_1 и C_2 инцидентны обоим путям, $P^{(1,+)} = \varnothing$.

Характеристики путей из 1 в 8: $d_1 = d_2 = D = 1$, $\Phi(L(\overline{C})) = \Phi(3,4) = 5$, $h = \max\{4{+}5, 5{+}5\} = 10$, $L(U^{(2)}) = \{4,5\}$. Множество вычетов полное:

$$L(U^{(2)})^{(h(1,8)+\delta(1,8),\, h(1,8)+\delta(1,8)+D]} = L(U^{(2)})^{(10,11]} = \{11\},$$

$\{\{11\} \equiv_1\} = \{0\}$. Значит, $2_{(1,8)}$-примитивность выполнена, и $\gamma_{2_{(1,8)}} = 10$ по теореме 7.

8. О нелинейности преобразований векторного пространства

Сложные криптографические функции часто построены с помощью произведения функций, относительно несложно реализуемых аппаратно и/или программно. Например, в симметричных блочных шифрах шифрующая подстановка есть произведение нескольких однотипных раундовых функций. Для оценки некоторых характеристик нелинейности произведения функций применим МГП, использующий троичные матрицы и помеченные орграфы.

Следуя ранее введённой терминологии, дадим ряд понятий.

Булеву функцию назовём *вполне нелинейной* (в. н. ф.), если она зависит нелинейно от каждого переменного. Заметим, что класс в. н. ф. содержит не только все максимально нелинейные б. ф., но и ряд равновероятных б. ф., перспективных с точки зрения криптографических приложений.

Для преобразования $g\colon V_n \to V_n$ его координатные б. ф. обозначим через $g_1(x_1,\ldots,x_n),\ldots,g_n(x_1,\ldots,x_n)$, а координатные б. ф. степени g^t, $t \geqslant 1$, — через $g_1^t(x_1,\ldots,x_n),\ldots,g_n^t(x_1,\ldots,x_n)$.

Преобразование g называется

а) *вполне нелинейным* (в. н. п.), если все его координатные б. ф. суть в. н. ф.;

б) *B-нелинейным*, если из равенства $b_{ij} = 1$ следует, что координатная б. ф. $g_j(x_1,\ldots,x_n)$ зависит нелинейно от x_i;

в) 2-*перфективным*, если в циклической полугруппе $\langle g\rangle$ содержится в. н. п.; наименьшее $t \in \mathbb{N}$, при котором g^t — в. н. п., назовём *показателем* 2-*перфективности* преобразования g и обозначим через $\mathrm{prf}^{[2]}\, g$;

г) 2_B-*перфективным*, если в циклической полугруппе $\langle g\rangle$ содержится B-нелинейное преобразование; наименьшее $t \in \mathbb{N}$, при котором g^t B-нелинейно, назовём *показателем* 2_B-*перфективности* преобразования g, обозначим через $\mathrm{prf}^{[2_B]}\, g$.

Преобразованию g пространства V_n однозначно соответствуют

а) троичная матрица $M_\theta(g) = (m_{ij}^g)$ порядка n над G, называемая *матрицей нелинейности* преобразования g, где

$$m_{ij}^g = \begin{cases} 0, & \text{если } g_j \text{ зависит от } x_i \text{ фиктивно,} \\ 1, & \text{если } g_j \text{ зависит от } x_i \text{ линейно,} \\ 2, & \text{если } g_j \text{ зависит от } x_i \text{ нелинейно,} \end{cases}$$

$1 \leqslant i, j \leqslant n$; положим $M_{\theta,B}(g) = B * M_\theta(g)$;

б) помеченный орграф $\Gamma_\theta(g)$ с множеством вершин $\overline{1,n}$, называемый *графом нелинейности* преобразования g, где дуга (i,j) помечена числом m_{ij}^g, $1 \leqslant i, j \leqslant n$, нулевая метка кодирует отсутствие дуги в графе; через $\Gamma_{\theta,B}(g)$ обозначим B-часть орграфа $\Gamma_\theta(g)$.

Матрица $M_\theta(g)$ есть матрица меток орграфа $\Gamma_\theta(g)$.

Обозначим через $\widehat{G} = \{g^{(1)}, \ldots, g^{(p)}\}$ множество преобразований пространства V_n, $p \geqslant 1$. Множество $\widehat{G}$ называется

а) 2-*перфективным*, если полугруппа $\langle\widehat{G}\rangle$ содержит в. н. п.; наименьшая длина слова в алфавите $\widehat{G}$, задающего в. н. п., называется *показателем* 2-перфективности множества $\widehat{G}$, обозначается $\mathrm{prf}^{[2]}\, \widehat{G}$;

б) 2_B-*перфективным*, если полугруппа $\langle\widehat{G}\rangle$ содержит B-нелинейное преобразование; наименьшая длина слова в алфавите $\widehat{G}$, задающего B-нелинейное преобразование, называется *показателем* 2_B-перфективности множества $\widehat{G}$, обозначается $\mathrm{prf}^{[2_B]}\, \widehat{G}$.

Множеству $\widehat{G}$ преобразований соответствуют множество матриц нелинейности $\widehat{M}_\theta(\widehat{G}) = \{M_\theta(g^{(1)}), \ldots, M_\theta(g^{(p)})\}$ и множество орграфов нелинейности $\widehat{\Gamma}_\theta(\widehat{G}) = \{\Gamma_\theta(g^{(1)}), \ldots, \Gamma_\theta(g^{(p)})\}$.

Теорема 8 (основная теорема МГП [3]). *Для произвольных преобразований $g^{(1)}, \ldots, g^{(p)}$ пространства V_n имеем*

$$M_\theta(g^{(1)} \ldots g^{(p)}) \leqslant M_\theta(g^{(1)}) \ldots M_\theta(g^{(p)}), \quad p \geqslant 1;$$

в частности, $M_\theta(g^t) \leqslant (M_\theta(g))^t$ для преобразования g при $t > 1$.

Отношение превосходства верно и для $M_{\theta,B}(g)$ при любой маске B.

Следствие 3. *Для 2-перфективного множества* $\widehat{G}$ *имеем*

$$\operatorname{prf}^{[2]}\widehat{G} \geqslant \exp^{[2]}\widehat{M}_\theta(\widehat{G}).$$

В частности, $\operatorname{prf}^{[2]} g \geqslant \exp^{[2]} M_\theta(g)$ для 2-перфективного преобразования g. Отсюда $\operatorname{prf}^{[2_B]}\widehat{G} \geqslant \exp^{[2_B]}\widehat{M}_\theta(\widehat{G})$ для любой матрицы B, в частности,

$$\operatorname{prf}^{[2_B]} g \geqslant \exp^{[2_B]} M_\theta(g).$$

Свойства 2_B-перфективных преобразований при любой маске B.

1) Свойства 2-перфективности и 2_B-перфективности не являются монотонными и наследственными: если преобразование g^t 2-перфективно (2_B-перфективно), $t \geqslant 1$, то не обязательно g^{t+k} или g^{kt} 2-перфективно (2_B-перфективно), $k \geqslant 1$.

2) Если порождённая небиективным 2_B-перфективным преобразованием g циклическая полугруппа $\langle g\rangle$ определена соотношением $g^d = g^{d+n}$, где d — циклическая глубина, а n — период преобразования g, то начиная со степени $t \geqslant d+n$ преобразования g^t повторяются, т. е. $g^{t-d} = g^t$. Это означает, что справедливы оценки

$$\exp^{[2_B]} M_\theta(g) \leqslant \operatorname{prf}^{[2_B]} g < d+n.$$

3) Если подстановка g 2_B-перфективна, то

$$\exp^{[2_B]} M_\theta(g) \leqslant \operatorname{prf}^{[2_B]} g < \operatorname{ord} g.$$

Пример 4. При $B = \{(1,2),(1,4),(2,3)\}$ оценим $\operatorname{prf}^{[2_B]} g$ для подстановки g регистра левого сдвига длины 4 с обратной связью $f = x_1 \oplus x_3x_4$. Составим матрицу нелинейности $M_\theta(g) = M$ и найдём $\exp^{[2_B]} M_\theta(g)$:

$$M = \begin{pmatrix} 0&0&0&1\\ 1&0&0&0\\ 0&1&0&2\\ 0&0&1&2 \end{pmatrix}, \quad M^2 = \begin{pmatrix} 0&0&1&2\\ 0&0&0&1\\ 1&0&2&2\\ 0&1&2&2 \end{pmatrix},$$

$$M^3 = \begin{pmatrix} 0&1&2&2\\ 0&0&1&2\\ 0&2&2&2\\ 1&2&2&2 \end{pmatrix}, \quad M^4 = \begin{pmatrix} 1&2&2&2\\ 0&1&2&2\\ 2&2&2&2\\ 2&2&2&2 \end{pmatrix},$$

$$M^5 = \begin{pmatrix} 2&2&2&2\\ 1&2&2&2\\ 2&2&2&2\\ 2&2&2&2 \end{pmatrix}, \quad M^6 = \begin{pmatrix} 2&2&2&2\\ 2&2&2&2\\ 2&2&2&2\\ 2&2&2&2 \end{pmatrix}.$$

Тогда в силу следствия 3 имеем

$$\operatorname{prf}^{[2_B]} g \geqslant \exp^{[2_B]} M = 4.$$

Вычислим АНФ координатных функций для степеней подстановки g. Используя закон рекурсии регистра сдвига, найдём его выходные функции $f_i(x_1, x_2, x_3, x_4)$, где g^i задана координатными функциями $f_{i+1}(x_1, x_2, x_3, x_4)$, $f_{i+2}(x_1, x_2, x_3, x_4)$, $f_{i+3}(x_1, x_2, x_3, x_4)$, $f_{i+4}(x_1, x_2, x_3, x_4)$, $i \geqslant 1$:

$$f_i(x_1, x_2, x_3, x_4) = x_i, \quad i = 1, 2, 3, 4,$$
$$f_5(x_1, x_2, x_3, x_4) = x_1 \oplus x_3x_4,$$
$$f_6(x_1, x_2, x_3, x_4) = x_2 \oplus x_1x_4 \oplus x_3x_4,$$
$$f_7(x_1, x_2, x_3, x_4) = x_3 \oplus x_1x_2 \oplus x_1x_4 \oplus x_3x_4 \oplus x_2x_3x_4,$$
$$f_8(x_1, x_2, x_3, x_4) = x_4 \oplus x_1x_2 \oplus x_1x_4 \oplus x_2x_3 \oplus x_3x_4 \oplus x_1x_3x_4 \oplus x_2x_3x_4.$$

Отсюда подстановки g, g^2, g^3 не B-нелинейные. Подстановка g^4 B-нелинейная, так как имеются нелинейные зависимости: f_6 и f_8 от x_1, f_7 от x_2. Следовательно, подстановка g 2_B-перфективна, $\mathrm{prf}^{[2_B]}\, g = 4$.

Финансирование работы

Исследование выполнено за счёт бюджетов организаций, указанных автором на первой странице статьи. Дополнительных грантов на проведение или руководство этим исследованием получено не было.

Конфликт интересов

Автор заявляет, что у него нет конфликта интересов.

Литература

1. **Fomichev V. M., Koreneva A. M.** Encryption performance and security of certain wide block ciphers // J. Comput. Virol. Hacking Tech. 2020. V. 16. P. 197–216.
2. **Фомичёв В. М., Авезова Я. А., Коренева А. М., Кяжин С. Н.** Примитивность и локальная примитивность орграфов и неотрицательных матриц // Дискрет. анализ и исслед. операций. 2018. Т. 25, № 3. С. 95–125.
3. **Фомичёв В. М.** Оценка характеристик нелинейности итеративных преобразований векторного пространства // Дискрет. анализ и исслед. операций. 2020. Т. 27, № 4. С. 131–151.
4. **Frobenius G.** Über Matrizen aus nicht negativen Elementen // Berl. Ber. 1912. S. 456–477. [German].
5. **Dulmage A. L., Mendelsohn N. S.** Gaps in the exponent set of primitive matrices // Ill. J. Math. 1964. V. 8, No. 4. P. 642–656.
6. **Berger T. P., Francq J., Minier M., Thomas G.** Extended generalized Feistel networks using matrix representation to propose a new lightweight block cipher: Lilliput // IEEE Trans. Comput. 2016. V. 65, No. 7. P. 2074–2089.

7. **Berger T., Minier M., Thomas G.** Extended generalized Feistel networks using matrix representation // Selected areas in cryptography — SAC 2013. Proc. 20th Int. Conf. (Burnaby, Canada, Aug. 14–16, 2013). Heidelberg: Springer, 2014. P. 289–305. (Lect. Notes Comput. Sci.; V. 8282).
8. **Brualdi R. A., Liu B.** Generalized exponents of primitive directed graphs // J. Graph Theory. 1990. V. 14, No. 4. P. 483–499.
9. **Huang Y., Liu B.** Generalized r-exponents of primitive digraphs // Taiwan. J. Math. 2011. V. 15, No. 5. P. 1999–2012.
10. **Liu B.** Generalized exponents of Boolean matrices // Lin. Algebra Appl. 2003. V. 373. P. 169–182.
11. **Miao Z., Zhang K.** The local exponent sets of primitive digraphs // Lin. Algebra Appl. 2000. V. 307. P. 15–33.
12. **Shen J., Neufeld S.** Local exponents of primitive digraphs // Lin. Algebra Appl. 1998. V. 268. P. 117–129.
13. **Suzaki T., Minematsu K.** Improving the generalized Feistel // Fast software encryption. Proc. 17th Int. Workshop (Seoul, Korea, Feb. 7–10, 2010). Heidelberg: Springer, 2010. P. 19–39. (Lect. Notes Comput. Sci.; V. 6147).
14. **Wielandt H.** Unzerlegbare, nicht negative Matrizen // Math. Z. 1950. Bd. 52. S. 642–648. [German].
15. **Perkins P.** A theorem on regular graphs // Pac. J. Math. 1961. V. 2. P. 1529–1533.
16. **Dulmage A. L., Mendelsohn N. S.** The exponent of a primitive matrix // Can. Math. Bull. 1962. V. 5, No. 3 P. 241–244.
17. **Neufeld S. W.** A diameter bound on the exponent of a primitive directed graph // Lin. Algebra Appl. 1996. V. 245. P. 27–47.
18. **Nyberg K.** Generalized Feistel networks // Advances in cryptology — ASIACRYPT'96. Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur. (Kyongju, Korea, Nov. 3–7, 1996). Heidelberg: Springer, 1996. P. 91–104. (Lect. Notes Comput. Sci.; V. 1163).

Владимир Михайлович Фомичёв

Статья поступила
4 марта 2026 г.
После доработки —
10 марта 2026 г.
Принята к публикации
23 марта 2026 г.

DISKRETNYI ANALIZ I ISSLEDOVANIE OPERATSII
/DISCRETE ANALYSIS AND OPERATIONS RESEARCH/
April–June 2026. Vol. 33, No. 2. P. 136–157

UDC 519.17 DOI: 10.33048/daio.2026.33.856

LOCAL NONLINEARITY OF TRANSFORMATION ITERATIONS

V. M. Fomichev[1,2]

[1] RUDN University,
6 Miklukho-Maklay Street, 117198 Moscow, Russia
[2] Institute of Informatics Problems of FRC CSC RAS,
44 Bld. 2 Vavilov Street, 119333 Moscow, Russia
E-mail: fomichev.2016@yandex.ru

Abstract. In accordance with Shannon's concept of the mixing properties of information transformations, a previously proposed matrix-graph approach is developed for assessing the characteristics of nonlinearity in the product of transformations in vector spaces. This approach makes it possible to evaluate local nonlinear properties of the product of transformations using ternary matrices and labeled digraphs that correspond unambiguously to the transformations. For ternary matrices and labeled digraphs, the property of local 2-primitivity and assessments of local 2-exponents, including exact values, is defined using path lengths. Criteria for local 2-primitivity is established. The results for matrices and labeled digraphs are applied to evaluate the characteristics of the product of transformations in vector spaces. Examples are considered. Bibliogr. 18.

Keywords: ternary matrix, locally primitive matrix, 2-primitive matrix, 2-exponent of a matrix, 2-primitive labeled orgraph.

References

1. **V. M. Fomichev** and **A. M. Koreneva,** Encryption performance and security of certain wide block ciphers, *J. Comput. Virol. Hacking Tech.* **16**, 197–216 (2020).
2. **V. M. Fomichev, Ya. A. Avezova, A. M. Koreneva,** and **S. N. Kyazhin,** Primitivity and local primitivity of digraphs and nonnegative matrices, *Diskretn. Anal. Issled. Oper.* **25** (3), 95–125 (2018) [Russian] [*J. Appl. Ind. Math.* **12** (3), 453–469 (2018)].

English transl.: *Journal of Applied and Industrial Mathematics* **20** (2) (2026).

3. **V. M. Fomichev,** Estimating nonlinearity characteristics for iterative transformations of a vector space, *Diskretn. Anal. Issled. Oper.* **27** (4), 131–151 (2020) [Russian] [*J. Appl. Ind. Math.* **14** (4), 610–622 (2020)].
4. **G. Frobenius,** Über Matrizen aus nicht negativen Elementen, *Berl. Ber.*, 456–477 (1912) [German].
5. **A. L. Dulmage** and **N. S. Mendelsohn,** Gaps in the exponent set of primitive matrices, *Ill. J. Math.* **8** (4), 642–656 (1964).
6. **T. P. Berger, J. Francq, M. Minier,** and **G. Thomas,** Extended generalized Feistel networks using matrix representation to propose a new lightweight block cipher: Lilliput, *IEEE Trans. Comput.* **65** (7), 2074–2089 (2016).
7. **T. Berger, M. Minier,** and **G. Thomas,** Extended generalized Feistel networks using matrix representation, in *Selected Areas in Cryptography — SAC 2013*, Proc. 20th Int. Conf. (Burnaby, Canada, Aug. 14–16, 2013) (Springer, Heidelberg, 2014), pp. 289–305 (Lect. Notes Comput. Sci., Vol. 8282).
8. **R. A. Brualdi** and **B. Liu,** Generalized exponents of primitive directed graphs, *J. Graph Theory* **14** (4), 483–499 (1990).
9. **Y. Huang** and **B. Liu,** Generalized r-exponents of primitive digraphs, *Taiwan. J. Math.* **15** (5), 1999–2012 (2011).
10. **B. Liu,** Generalized exponents of Boolean matrices, *Lin. Algebra Appl.* **373**, 169–182 (2003).
11. **Z. Miao** and **K. Zhang,** The local exponent sets of primitive digraphs, *Lin. Algebra Appl.* **307**, 15–33 (2000).
12. **J. Shen** and **S. Neufeld,** Local exponents of primitive digraphs, *Lin. Algebra Appl.* **268**, 117–129 (1998).
13. **T. Suzaki** and **K. Minematsu,** Improving the generalized Feistel, in *Fast Software Encryption*, Proc. 17th Int. Workshop (Seoul, Korea, Feb. 7–10, 2010) (Springer, Heidelberg, 2010), pp. 19–39 (Lect. Notes Comput. Sci., Vol. 6147).
14. **H. Wielandt,** Unzerlegbare, nicht negative Matrizen, *Math. Z.* **52**, 642–648 (1950). [German].
15. **P. Perkins,** A theorem on regular graphs, *Pac. J. Math.* **2**, 1529–1533 (1961).
16. **A. L. Dulmage** and **N. S. Mendelsohn,** The exponent of a primitive matrix, *Can. Math. Bull.* **5** (3), 241–244 (1962).
17. **S. W. Neufeld,** A diameter bound on the exponent of a primitive directed graph, *Lin. Algebra Appl.* **245**, 27–47 (1996).
18. **K. Nyberg,** Generalized Feistel networks, in *Advances in Cryptology — ASIACRYPT'96*, Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur. (Kyongju, Korea, Nov. 3–7, 1996) (Springer, Heidelberg, 1996), pp. 91–104 (Lect. Notes Comput. Sci., Vol. 1163).

Vlaimir M. Fomichev

Received March 4, 2026
Revised March 10, 2026
Accepted March 23, 2026

ДИСКРЕТНЫЙ АНАЛИЗ
И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ

2026 г. Том 33, № 2

Зав. редакцией Е. В. Горкунов

Журнал подготовлен с использованием макропакета LaTeX 2_ε.
The present publication has been typeset using LaTeX 2_ε.

Журнал зарегистрирован в Федеральной службе по надзору
в сфере связи, информационных технологий и массовых коммуникаций.
Свидетельство о регистрации ЭЛ № ФС77-85978 от 26.09.2023 г.
Размещение в сети Интернет: `math-sobolev.ru`.

Дата размещения в сети Интернет 25.09.2026 г.
Формат $70 \times 100\,1/16$. Усл. печ. л. 12,8. Объём 1,33 МБ.

Издательство Института математики,
пр. Академика Коптюга, 4, 630090 Новосибирск, Россия