МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПРИКЛАДНАЯ ДИСКРЕТНАЯ МАТЕМАТИКА

Научный журнал

2024 № 64

Зарегистрирован в Федеральной службе по надзору
в сфере связи и массовых коммуникаций

Свидетельство о регистрации ПИ № ФС 77-33762 от 16 октября 2008 г.

Подписной индекс в объединённом каталоге «Пресса России» 38696

УЧРЕДИТЕЛЬ
Томский государственный университет

Адрес редакции и издателя: 634050, г. Томск, пр. Ленина, 36
E-mail: pank@mail.tsu.ru

В журнале публикуются результаты фундаментальных и прикладных научных исследований отечественных и зарубежных ученых, включая студентов и аспирантов, в области дискретной математики и её приложений в криптографии, компьютерной безопасности, кибернетике, информатике, программировании, теории надёжности, интеллектуальных системах.

Периодичность выхода журнала: 4 номера в год.

Редактор *Н. И. Шидловская*
Редактор-переводчик *Т. В. Бутузова*
Верстка *И. А. Панкратовой*

Подписано к печати 31.05.2024. Формат $60 \times 84\frac{1}{8}$. Усл. п. л. 15. Тираж 300 экз.
Заказ № 5935. Цена свободная. Дата выхода в свет 04.06.2024.

Отпечатано на оборудовании
Издательства Томского государственного университета
634050, г. Томск, пр. Ленина, 36
Тел.: 8(3822)53-15-28, 52-98-49

СОДЕРЖАНИЕ

CONTENTS

К 85-ЛЕТИЮ ГЕННАДИЯ ПЕТРОВИЧА АГИБАЛОВА (17.05.1939 — 16.12.2020)

Одна лишь просьба к поколеньям новым:
Когда наш луч сверкнёт из вечной тьмы,
Вы нас хорошим помяните словом —
Не самые плохие были мы.
Г. П. Агибалов

17 мая 2024 года исполнилось 85 лет со дня рождения Геннадия Петровича Агибалова — основателя журнала «Прикладная дискретная математика» (ПДМ), его первого и бессменного (2008–2020) главного редактора.

Журнал ПДМ основан в 2008 году и задумывался сначала как площадка для публикации трудов Сибирской научной школы-семинара «Компьютерная безопасность и криптография» — SIBECRYPT, однако почти сразу, с 2009 года, по совету Дмитрия Анатольевича Катунина, тогдашнего главного редактора журнала «Вестник ТГУ», Геннадий Петрович решил сделать ПДМ полноценным регулярным (ежеквартальным) научным журналом, с более широкой, чем компьютерная безопасность и криптография, тематикой. При этом он опирался на богатые традиции ведущей научной школы ТГУ — Школы прикладной дискретной математики, основанной ещё в 1959 году Аркадием Дмитриевичем Закревским. Труды конференции SIBECRYPT стали выходить отдельным приложением к журналу ПДМ, которое в 2012 году получило самостоятельный статус: теперь наряду с основным журналом издаётся «ПДМ. Приложение» (ежегодно).

Первые годы существования журнала ПДМ (до его вхождения в список ВАК и зарубежные базы цитирования) были трудными, не все учёные с радостью посылали свои статьи в новорожденный журнал; ПДМ работал, что называется, «с колёс» — с минимальной задержкой времени от поступления статьи до её публикации. Тем не менее с самого начала Геннадий Петрович задал высокую планку требований к статьям (папка «Отказ» в портфеле журнала была лишь чуть менее пухлая, чем папка

«Опубликованные»), он активно привлекал к написанию и рецензированию статей ведущих учёных России, Украины, Беларуси, и мы благодарны всем, кто помогал тогда и продолжает поддерживать журнал сейчас.

Вклад Геннадия Петровича в становление и признание журнала невозможно переоценить; это не только общее руководство: Геннадий Петрович сам прочитывал, рецензировал и правил статьи; переписывал (зачастую — полностью) английские аннотации. Когда журнал «горел» (чаще всего сентябрьский номер), срочно писал статью сам.

К публикациям в журнале всегда активно привлекались студенты и аспиранты; вот уж на кого Геннадий Петрович времени никогда не жалел — дотошно и терпеливо обсуждал, советовал и вычитывал статью. Никаких скидок на «неопытность» автора не делалось, студенческие статьи, как и прочие, проходили процедуру тщательного рецензирования. А вот публиковались они обычно без соавторства Геннадия Петровича, он за этим никогда не гнался, хотя порой его вклад в работу кратно превышал вклад ученика.

С 2021 года журнал живёт без Геннадия Петровича. Конечно, его очень не хватает. Вспоминая с бесконечной благодарностью замечательного Человека, учёного, наставника, мы постараемся, чтобы дело его жизни продолжалось и развивалось.

Редакционная коллегия

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ПРИКЛАДНОЙ ДИСКРЕТНОЙ МАТЕМАТИКИ

УДК 512.718 DOI 10.17223/20710410/64/1

КРИТЕРИЙ НЁТЕРОВОСТИ ПО УРАВНЕНИЯМ И СЛОЖНОСТЬ ПРОБЛЕМЫ РАЗРЕШИМОСТИ ДЛЯ СИСТЕМ УРАВНЕНИЙ НАД ЧАСТИЧНО УПОРЯДОЧЕННЫМИ МНОЖЕСТВАМИ

А. Ю. Никитин, И. Д. Кудык

Омский государственный университет им. Ф. М. Достоевского, г. Омск, Россия

E-mail: nikitinlexey@gmail.com, volume8091@mail.ru

Представлены результаты, касающиеся основной проблемы алгебраической геометрии над частично упорядоченными множествами с вычислительной точки зрения, а именно задачи разрешимости системы уравнений над частичным порядком. Задача разрешимости систем уравнений разрешима за полиномиальное время, если ориентированный граф, соответствующий частичному порядку, является приведённым интервальным орграфом, и является NP-полной, если основание ориентированного графа соответствующего частичного порядка является циклом длины не меньше 4. Получен также результат, характеризующий возможность перехода от бесконечных систем уравнений над частичным порядком к конечным системам. Алгебраические системы, обладающие указанным свойством, называются нётеровыми по уравнениям. Частично упорядоченное множество обладает свойством нётеровости по уравнениям тогда и только тогда, когда любые его верхние и нижние конусы с базой являются конечно определёнными.

Ключевые слова: *системы уравнений, вычислительная сложность, частично упорядоченное множество, нётеровость по уравнениям, конусы, разрешимость.*

CRITERION FOR EQUATIONAL NOETHERIANITY AND COMPLEXITY OF THE SOLVABILITY PROBLEM FOR SYSTEMS OF EQUATIONS OVER PARTIALLY ORDERED SETS

A. Yu. Nikitin, I. D. Kudyk

Dostoevsky Omsk State University, Omsk, Russia

Results are presented concerning the main problem of algebraic geometry over partially ordered sets from a computational point of view, namely, the solvability problem for systems of equations over a partial order. This problem is solvable in polynomial time if the directed graph corresponding to the partial order is a adjusted interval digraph, and is NP-complete if the base of the directed graph corresponding to the partial order is a cycle of length at least 4. We also present a result characterizing the possibility of transition from infinite systems of equations over partial orders to finite systems. Algebraic systems with this property are called equationally Noetherian. A partially ordered set is equationally Noetherian if and only if any of its upper and lower cones with base are finitely defined.

Keywords: *systems of equations, computational complexity, partially ordered set, poset, equationally Noetherian property, cones, solvability.*

Введение

Классическая алгебраическая геометрия изучает уравнения и системы уравнений над полями вещественных и комплексных чисел. Во второй половине XX века бурный рост переживает такое направление математики, как универсальная алгебраическая геометрия, которая изучает системы уравнений над произвольными алгебраическими системами. Общие закономерности для алгебраических систем выводятся на основе изучения различных алгебраических систем, таких, как свободные неабелевы группы, абелевы группы, разрешимые группы, метабелевы группы, полугруппы, решётки, графы и т. д. Данная работа посвящена вопросам алгебраической геометрии над частично упорядоченными множествами и представляет собой продолжение исследований алгебраической геометрии над алгебраическими системами без функциональных символов.

Системы уравнений бывают как конечные, так и бесконечные. Свойство нётеровости по уравнениям говорит о том, что можно из бесконечных систем уравнений выделить эквивалентные конечные подсистемы, оно характеризует «податливость» системы к изучению с точки зрения алгебраической геометрии. Если система обладает данным свойством, то из этого сразу же следует множество свойств, характеризующих конечность цепочек алгебраических конструкций над этой системой, таких, как убывающие цепочки алгебраических подмножеств, цепочки собственных эпиморфизмов координатных алгебр, обрыв возрастающих цепочек дизъюнктивных радикальных идеалов и т. д. [1]. Существуют алгебраические структуры, обладающие данным свойством как безусловно, так и при определённом условии (известны критерии нётеровости). В п. 1 данной работы приведены необходимые предварительные сведения. Пункт 2 посвящён формулировке критерия нётеровости по уравнениям для частично упорядоченных множеств (ЧУМ).

Для систем уравнений над алгебраическими системами существуют такие важные объекты, как координатная алгебра и радикал системы. Они определяют общее решение системы, если оно есть. Но перед тем, как решать систему уравнений и искать её общее решение, важно знать: разрешима ли эта система, имеет ли она решение в принципе? Этому вопросу посвящен п. 3, где показана связь ЧУМ с подклассом ориентированных графов и связь задачи разрешимости систем уравнений с подзадачей удовлетворения ограничений — задачей списочного гомоморфизма.

Использование частично упорядоченных множеств на практике широко распространено. С их помощью можно строить модели таких структур, как базы данных, потоки данных в сети, события во временных рядах и др., где требуется задание иерархии. Уравнения над частичными порядками задают некоторые подструктуры (подпорядки) в данных моделях. Поэтому изучение систем уравнений над частично упорядоченными множествами также важно и с практической точки зрения. Например, если задана модель компьютерных вычислений в качестве последовательно-параллельного частичного порядка, то через решение системы уравнений можно определять доступность одного вычисления для другого. Но по свойствам изначальной модели можно сразу понять: разрешимы ли системы за полиномиальное время или нет? И если нет, то модель вычислений можно менять так, чтобы системы решались эффективно по времени.

Теоремы 1, 5, 6 и леммы 1 и 2 доказаны А. Ю. Никитиным. Следствие 1 выведено И. Д. Кудыком.

1. Предварительные сведения

Дадим базовые определения теории решёток [2, 3] и универсальной алгебраической геометрии [1] и введём вспомогательные определения порождающих элементов конуса, конуса с базой, конечно порождённого и конечно определённого конуса, необходимые для формулировки результатов.

Частично упорядоченным множеством (*частичным порядком*) называется алгебраическая система $\mathcal{P} = \langle P; \{\leqslant^{(2)}, A\}\rangle$, где $\leqslant$ — предикатный символ отношения порядка и A — множество константных символов, на которой выполнены следующие три аксиомы:

1) $\forall p \in P\ p \leqslant p$ (рефлексивность);
2) $\forall p_1, p_2 \in P\ (p_1 \leqslant p_2 \wedge p_2 \leqslant p_1) \Rightarrow p_1 = p_2$ (антисимметричность);
3) $\forall p_1, p_2, p_3 \in P\ (p_1 \leqslant p_2 \wedge p_2 \leqslant p_3) \Rightarrow p_1 \leqslant p_3$ (транзитивность).

Язык (сигнатуру) частичных порядков с множеством константных символов A будем обозначать как L_A.

Для частично упорядоченного множества $\mathcal{P} = \langle P; \{\leqslant^{(2)}, A\}\rangle$ предикат частичного порядка $\leqslant^{(2)}$ задаёт соотношения для элементов носителя P. Множество таких соотношений обозначается $\leqslant^{\mathcal{P}}$. Если для элементов p_i, p_j частичного порядка $\mathcal{P}$ выражение $p_i \leqslant p_j$ выполнимо над $\mathcal{P}$, то это можно обозначить как $p_i \leqslant p_j \in \leqslant^{\mathcal{P}}$, или в терминах выполнимости как $\mathcal{P} \models p_i \leqslant p_j$.

Алгебраическая система называется *диофантовой*, если между носителем алгебраической системы и множеством её константных символов в языке существует взаимно однозначное соответствие.

Элементы x и y частичного порядка $\mathcal{P}$ называются *сравнимыми*, если в $\mathcal{P}$ либо $x \leqslant y$, либо $y \leqslant x$. В противном случае, элементы называются *несравнимыми*, это обозначается как $x \not\sim y$ [3, с. 16]. Далее по тексту в некоторых местах удобно будет писать не $a \leqslant b$, а $b \geqslant a$, что означает одно и то же.

Для любого множества элементов A частичного порядка $\mathcal{P}$ определены множества $A^{\uparrow} = \{x \in \mathcal{P} : \forall a \in A\ a \leqslant x\}$ и $A^{\downarrow} = \{x \in \mathcal{P} : \forall a \in A\ x \leqslant a\}$. Эти множества называются *верхним* и *нижним конусами* множества A (или просто верхнее и нижнее множества A) [2, с. 90]. Для одноэлементного множества $A = \{a\}$ будем обозначать $a^{\uparrow}$ и $a^{\downarrow}$ соответственно.

Множеством *порождающих элементов* конуса $A^{\uparrow}$ ($A^{\downarrow}$) называется множество элементов B частичного порядка $\mathcal{P}$, для которого верно $B^{\uparrow} = A^{\uparrow}$ ($B^{\downarrow} = A^{\downarrow}$). Конус $A^{\uparrow}$ ($A^{\downarrow}$) называется *конечно порождённым*, если существует конечное множество B порождающих этого конуса.

Пусть задано множество элементов частичного порядка A. *Конусом с базой* называется пара $(A, A^{\uparrow})$, которая состоит из *базы* A и верхнего конуса $A^{\uparrow}$, порождённого базой A. Аналогично определяется нижний конус с базой. Верхний конус с базой $(A, A^{\uparrow})$ называется *конечно определённым*, если существует $B \subseteq A$, такое, что $|B| < \infty$ и $B^{\uparrow} = A^{\uparrow}$.

Термом в языке L от переменных X называется выражение, определенное рекурсивно следующим образом:

1) любая переменная $x \in X$ есть терм;
2) любая константа языка L есть терм;

3) если $t_1, \ldots, t_n$ — термы и $F^{(n)}$ — функциональный символ языка L, то $F(t_1, \ldots, t_n)$ является термом.

Через $T_L(X)$ обозначим множество всех термов языка L от переменных X.

Атомарной формулой языка L от переменных X называется выражение, определенное следующим образом:

1) для любых $t_i, t_j \in T_L(X)$ выражение $t_i = t_j$ является атомарной формулой;
2) для любого предикатного символа $R^{(n)}$ языка L и для любых термов $t_1, \ldots, t_n \in \in T_L(X)$ выражение $R(t_1, \ldots, t_n)$ является атомарной формулой.

Множество атомарных формул в языке L от переменных X обозначается $At_L(X)$. В алгебраической геометрии *уравнением* в языке L от переменных X называется атомарная формула в этом языке от переменных X. Произвольное множество уравнений из $At_L(X)$ называется *системой уравнений*. Дадим определение уравнения для частичных порядков.

Пусть $X_n = \{x_1, \ldots, x_n\}$ — множество переменных и задан частичный порядок $\mathcal{P} = \langle P; \{\leqslant, A\}\rangle$. *Уравнением* в языке L_A частичного порядка $\mathcal{P}$ от переменных X_n называется одно из следующих выражений:

1) $a_i = a_j$, где $a_i, a_j \in A$;
2) $a_i \leqslant a_j$ (или $a_j \geqslant a_i$), где $a_i, a_j \in A$;
3) $x_i = a_j$, где $x_i \in X_n$, $a_j \in A$;
4) $x_i = x_j$, где $x_i, x_j \in X_n$;
5) $a_i \leqslant x_j$ (или $x_j \geqslant a_i$), где $x_j \in X_n$, $a_i \in A$;
6) $x_i \leqslant a_j$ (или $a_j \geqslant x_i$), где $x_i \in X_n$, $a_j \in A$;
7) $x_i \leqslant x_j$ (или $x_j \geqslant x_i$), где $x_i, x_j \in X_n$.

Для системы уравнений $S(X)$ над частичным порядком $\mathcal{P}$ в языке L_A от переменных X договоримся обозначать множество уравнений типа $a_i = a_j$, где $a_i, a_j \in A$, в системе уравнений $S(X)$ через $S_{a=a}$. Аналогично обозначаются остальные множества типов уравнений:

$$S_{a=a}, \quad S_{a\leqslant a}, \quad S_{x=a}, \quad S_{x=x}, \quad S_{a\leqslant x}, \quad S_{x\leqslant a}, \quad S_{x\leqslant x}. \tag{1}$$

Множество $A^n = \{(a_1, \ldots, a_n) : a_i \in A\}$ называется *аффинным n-мерным пространством* над алгебраической системой $\mathcal{A}$, а его элементы — точками.

Точка $p = (a_1, \ldots, a_n) \in A^n$ называется *корнем* (*решением*) уравнения $s \in At_L(X)$, если $\mathcal{A} \vDash s(a_1, \ldots, a_n)$. Точка p является решением системы уравнений $S \subseteq At_L(X)$, если она является решением для каждого уравнения из системы S.

Пусть S — система уравнений языка L_A от переменных X. Множество всех решений системы S в пространстве A^n обозначается через $V_{\mathcal{A}}(S)$ (или $V(S)$ для краткости) и определяется как $V_{\mathcal{A}}(S) = \{(a_1, \ldots, a_n) \in A^n : \forall s \in S\, \mathcal{A} \vDash s(a_1, \ldots, a_n)\}$.

Система уравнений S называется *несовместной* над $\mathcal{A}$, если $V_{\mathcal{A}}(S) = \varnothing$; иначе она называется *совместной*. Две системы уравнений S_1 и S_2 называются *эквивалентными* над $\mathcal{A}$ (обозначается $S_1 \sim_{\mathcal{A}} S_2$), если $V_{\mathcal{A}}(S_1) = V_{\mathcal{A}}(S_2)$.

2. Критерий нётеровости по уравнениям

Сформулируем и докажем критерии свойств нётеровости по уравнениям и слабой нётеровости по уравнениям для частичных порядков.

Алгебраическая система $\mathcal{A}$ называется *слабо нётеровой по уравнениям*, если для любого конечного множества X и любой системы уравнений $S \subseteq At_L(X)$ существует

такая конечная система S_0, что $V_{\mathcal{A}}(S) = V_{\mathcal{A}}(S_0)$. Если при этом $S_0 \subseteq S$, то алгебраическая система $\mathcal{A}$ называется *нётеровой по уравнениям.*

Теорема 1. Частичный порядок $\mathcal{P}$ в языке L_A в диофантовом случае обладает свойством нётеровости по уравнениям тогда и только тогда, когда для любого подмножества B элементов частичного порядка $\mathcal{P}$ верхний и нижний конусы с базой B являются конечно определёнными.

Доказательство. Пусть для частичного порядка $\mathcal{P}$ любые конусы с базой являются конечно определёнными и задана система уравнений $S(X_n)$ над $\mathcal{P}$ в языке L_A от n переменных. Предположим, что эта система содержит все семь типов уравнений (1) и уравнений каждого типа бесконечное число.

Без ограничения общности считаем, что все различные константы в системе $S(X_n)$ имеют разные интерпретации на $\mathcal{P}$. Это условие не является обязательным, но упрощает дальнейшие строгие рассуждения и не влияет на суть доказательства.

Требуется доказать, что из системы $S(X_n)$ можно выбрать конечную подсистему $S'(X_n)$, эквивалентную $S(X_n)$. Множество решений системы $S(X_n)$ можно определить следующим образом: $V(S) = V(S_{a=a}) \cap V(S_{a\leqslant a}) \cap V(S_{x=a}) \cap V(S_{x=x}) \cap V(S_{x\leqslant a}) \cap \cap V(S_{a\leqslant x}) \cap V(S_{x\leqslant x})$. Покажем, что из подсистем каждого типа уравнений можно выбрать конечную подсистему, эквивалентную изначальной.

Сначала рассмотрим подсистему $S_{a=a} \subset S(X_n)$. Если это множество уравнений совместно над $\mathcal{P}$, то оно никак не влияет на множество решений системы уравнений $S(X_n)$ и определяется подсистема $S'_{a=a} = \varnothing$. Если $S_{a=a}$ несовместна над $\mathcal{P}$, то существует уравнение $a_i = a_j \in S_{a=a}$, которое не выполнено над частичным порядком $\mathcal{P}$, и это уравнение эквивалентно подсистеме $S_{a=a}$. Аналогичные рассуждения применимы к множеству уравнений $S_{a\leqslant a} \subset S(X_n)$.

Ввиду конечности множества переменных можно выбрать конечные подмножества уравнений $S'_{x=x}$ и $S'_{x\leqslant x}$, эквивалентных подсистемам $S_{x=x}$ и $S_{x\leqslant x}$ соответственно. Отметим, что системы $S_{x=x}$ и $S_{x\leqslant x}$ не могут быть несовместными.

Если множество уравнений $S_{x=a} \subset S(X_n)$ совместно, то, опять ввиду конечности множества переменных, существует конечная подсистема $S'_{x=a} \subset S_{x=a}$, эквивалентная $S_{x=a}$. Иначе в $S_{x=a}$ существует такая пара уравнений $x_i = a_j$, $x_i = a_k$, что $a_j \neq a_k$. Эта пара уравнений является эквивалентной $S_{x=a}$ подсистемой.

Пусть выделено подмножество уравнений $S_{x_i\leqslant a} \subset S_{x\leqslant a}$, в котором все уравнения зависят от переменной x_i. Это множество можно представить следующим образом: $S_{x_i\leqslant a} = \{x_i \leqslant a_j : j \in J,\ |J| = \infty\}$. Из этого представления видно, что $V(S_{x_i\leqslant a}) = A_i^{\downarrow}$, где $A_i = \{a_j : j \in J\}$. Пользуясь свойством, что любые конусы с базой частичного порядка $\mathcal{P}$ конечно определённые, можно выбрать такое конечное подмножество $J' \subset J$, что $A'_i = \{a_j : j \in J',\ |J'| < \infty\}$ и $(A'_i)^{\downarrow} = A_i^{\downarrow}$. Это означает, что можно выбрать конечную подсистему $S'_{x_i\leqslant a} \sim S_{x_i\leqslant a}$.

Применяя описанную процедуру для всех переменных из $S_{x\leqslant a}$, можно выделить конечную подсистему $S'_{x\leqslant a} \sim S_{x\leqslant a}$. Аналогичными рассуждениями можно вывести существование конечной подсистемы $S'_{a\leqslant x} \sim S_{a\leqslant x}$.

Таким образом, построены конечные подсистемы $S'_{a=a}$, $S'_{a\leqslant a}$, $S'_{x=a}$, $S'_{x=x}$, $S'_{a\leqslant x}$, $S'_{x\leqslant a}$ и $S'_{x\leqslant x}$, эквивалентные подсистемам $S_{a=a}$, $S_{a\leqslant a}$, $S_{x=a}$, $S_{x=x}$, $S_{a\leqslant x}$, $S_{x\leqslant a}$ и $S_{x\leqslant x}$ системы $S(X_n)$ соответственно. Это означает, что $V(S) = V(S'_{a=a}) \cap V(S'_{a\leqslant a}) \cap V(S'_{x=a}) \cap \cap V(S'_{x=x}) \cap V(S'_{x\leqslant a}) \cap V(S'_{a\leqslant x}) \cap V(S'_{x\leqslant x})$.

Теперь пусть частичный порядок $\mathcal{P}$ нётеров по уравнениям. Рассмотрим произвольное бесконечное множество элементов A частичного порядка $\mathcal{P}$. У него есть

нижний конус $A^{\downarrow}$. Составим систему уравнений $S(x) = \{x \leqslant a_i : a_i \in A\}$ над $\mathcal{P}$. Ввиду того, что $\mathcal{P}$ нётеров по уравнениям, можно выбрать конечную подсистему $S'(x) = \{x \leqslant a_i : a_i \in A',\ |A'| < \infty\} \sim S(x)$. Это означает, что $A^{\downarrow} = A'^{\downarrow}$. Тем самым доказана конечная определённость нижнего конуса с базой для любого множества элементов A частичного порядка $\mathcal{P}$. Аналогично доказывается конечная определённость верхних конусов с базой. ■

Следует отметить, что если конус с базой B не является конечно определённым, то это не означает, что $B^{\uparrow}$ или $B^{\downarrow}$ не являются конечно порождёнными. Пример: пусть задан частичный порядок $\mathcal{P} = \mathbb{Z} \cup \{q\}$, где $\mathbb{Z}$ — линейный порядок на множестве целых чисел и q — единичный элемент, который не сравним ни с одним элементом из $\mathbb{Z}$. Фрагмент графа данного частичного порядка приведён на рис. 1.

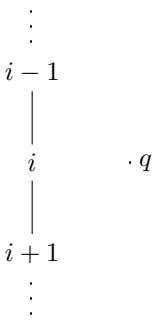


Рис. 1. Частичный порядок $\mathcal{P} = \mathbb{Z} \cup \{q\}$

Легко видеть, что $\mathbb{Z}^{\downarrow} = \varnothing$; для любого конечного подмножества $B \subset \mathbb{Z}$ верно $B^{\downarrow} \neq \varnothing$. Но если взять множество $C = \{0, q\}$, то $C^{\downarrow} = \varnothing = \mathbb{Z}^{\downarrow}$. Это означает, что $\mathbb{Z}^{\downarrow}$ является конечно порождённым.

Из этого примера и критерия нётеровости по уравнениям получаем следующее

Следствие 1. Частичный порядок $\mathcal{P}$ в языке L_A обладает свойством слабой нётеровости по уравнениям тогда и только тогда, когда для любого подмножества A элементов частичного порядка $\mathcal{P}$ конусы $B^{\uparrow}$ и $B^{\downarrow}$ являются конечно порождёнными.

3. Сложность проблемы разрешимости систем уравнений над частичными порядками

Сформулируем задачу разрешимости систем уравнений над частичными порядками и необходимые условия её полиномиальной разрешимости и NP-полноты. Для этого введём ряд определений [4] и переформулируем их с алгебраической точки зрения.

Граф $G = G(V)$ с множеством вершин V есть некоторое семейство сочетаний или пар вида $E = (a, b)$, $a, b \in V$, указывающее, какие вершины считаются соединёнными [4, с. 11].

Если рассматривать граф как алгебраическую систему, то можно сказать, что граф $G = \langle V; E\rangle$ — это алгебраическая система с множеством вершин V в качестве носителя и бинарным предикатным отношением E в языке, задающим множество рёбер в графе [1, с. 21]. Пара вершин $u, v \in V$, для которой выполнено $E(u, v)$, называется *ребром* графа G. Далее ребро будем обозначать (u, v). Если порядок вершин в ребре не существенен, то есть $(u, v) = (v, u)$, то такое ребро называется неориентированным или звеном. Если порядок существенен, то ребро называется ориентированным или *дугой*. Для дуги (u, v) вершина u является исходящей, а вершина v — входящей. Граф

является неориентированным, если каждое его ребро неориентированное, и ориентированным, если все его ребра ориентированы. Если ребро графа имеет своё начало и конец в одной и той же вершине, то такое ребро называется *петлёй*.

Ориентированному графу G можно поставить в соответствие неориентированный граф $U(G)$, полученный из G путём замены дуг на неориентированные рёбра.

Далее будем использовать графы специального вида, которые имеют ключевую роль в работе [5]. Пусть задано множество интервалов $\mathcal{I} = \{I_1, \ldots, I_n\}$ на вещественной прямой $\mathbb{R}$. Неориентированный граф $G = \langle V; E\rangle$ называется *интервальным*, если его множеству вершин $V = \{v_1, \ldots, v_n\}$ соответствует множество интервалов $\mathcal{I}$ и $(v_i, v_j) \in E(G)$ тогда и только тогда, когда $I_{v_i} \cap I_{v_j} \neq \varnothing$. Пусть теперь задана пара множеств интервалов $\{I_1, \ldots, I_n\}$ и $\{J_1, \ldots, J_n\}$. Ориентированный граф $G = \langle V; E\rangle$ называется *интервальным*, если каждой вершине v соответствует пара интервалов (I_v, J_v) и $(u, v) \in E(G)$ тогда и только тогда, когда $I_u \cap J_v \neq \varnothing$. Ориентированный интервальный граф называется *приведённым*, если для каждой вершины v интервалы I_v и J_v имеют общую левую точку. На рис. 2 представлен пример приведённого интервального ориентированного графа и его интервальная форма.

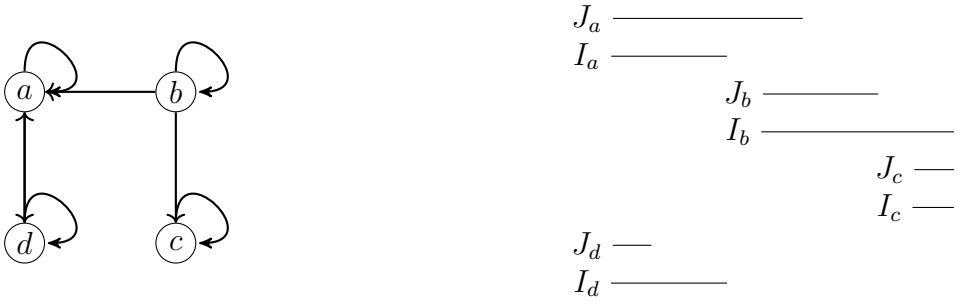


Рис. 2. Приведённый интервальный ориентированный граф и его интервальная форма

Пусть даны два графа H и G. *Гомоморфизмом* графа H в граф G называется такое отображение $\varphi : V(H) \to V(G)$, что для всех $u, v \in V(H)$ если $(u, v) \in E(H)$, то $(\varphi(u), \varphi(v)) \in E(G)$.

Известна связь между частичными порядками и графами [4]. Пусть задан частичный порядок $\mathcal{P}$, ему соответствует граф $\Gamma_{\mathcal{P}}$, вершинам которого соответствуют элементы частичного порядка, и дуга (p_i, p_j) есть в графе $\Gamma_{\mathcal{P}}$, если и только если $p_j \leqslant p_i$. Граф $\Gamma_{\mathcal{P}}$ является рефлексивным, транзитивным, антисимметричным и ациклическим, если не рассматривать петли. Назовём граф, построенный по частичному порядку, *p-графом*.

Пусть дан диофантов частичный порядок $\mathcal{P}$ в языке L_A. Задача разрешимости системы уравнений $S(X_n)$ над частичным порядком $\mathcal{P}$ формулируется следующим образом: совместна ли система уравнений $S(X_n)$ над частичным порядком $\mathcal{P}$ в языке L_A? Будем обозначать данную задачу как $\mathrm{Cons}(\mathcal{P})$.

Задача разрешимости системы уравнений над частичным порядком тесно связана с задачей *списочного гомоморфизма* (list homomorphism problem) [5–8]. Некоторые авторы называют её задачей *списка H-раскраски* (list H-coloring problem). Она формулируется следующим образом: пусть заданы два графа G и H и для каждой вершины v графа G задано множество вершин $L(v) \subseteq V(H)$ графа H. Существует ли такой гомоморфизм $\varphi : G \to H$, что для каждой вершины $v \in V(G)$ вершина $\varphi(v)$ принадлежит заданному списку вершин $L(v)$? В задаче граф H фиксирован, а граф G и множества

вершин $\{L(v_i) : L(v_i) \subseteq V(H),\ v_i \in V(G)\}$ являются входными параметрами. Задача списочного гомоморфизма для графа H обозначается L-HOM(H) [5].

В случае, когда для любой вершины $v \in V(G)$ список $L(v)$ равен всему множеству $V(H)$, задача L-HOM(H) является обычной задачей о гомоморфизме графов. В данной работе рассмотрена также задача RET(H), которая является задачей L-HOM(H), где для каждой вершины v входного графа G список $L(v)$ содержит либо одну вершину u графа H (для каждой v вершина u своя), либо равен множеству $V(H)$.

У задачи L-HOM(H) есть иерархия сложности, поэтому нужно показать, какое место в этой иерархии занимает задача Cons($\mathcal{P}$). Приведём известные результаты.

Теорема 2 (T. Feder, P. Hell, and J. Hunag [5, Теорема 2.4, с. 6]). Пусть H — рефлексивный орграф, у которого $U(H)$ является циклом длины больше 3. Тогда задача RET(H) является NP-полной.

Теорема 3 (T. Feder, P. Hell, and J. Hunag [5, Следствие 3.8, с. 11]). Если H — рефлексивный приведённый интервальный орграф, то L-HOM(H) разрешима за полиномиальное время.

Поскольку RET(H) является подзадачей L-HOM(H), то видно, что L-HOM(H) не всегда полиномиально разрешима. Более того, существует условие, при котором задача L-HOM(H) является NP-полной.

Теорема 4 (P. Hell and A. Rafiey [7, Теорема 3.2, с. 6]). Пусть H — ориентированный граф. Если H содержит направленную астероидную тройку, то задача L-HOM(H) является NP-полной.

Направленная астероидная тройка — это достаточно сложный объект, определённый для ориентированных графов. Его полное определение дано в [7].

По данным результатам можно понять, что задача RET(H) тоже может быть полиномиально разрешима, если H удовлетворяет некоторым условиям, например если H — рефлексивное колесо, т. е. орграф, полученный присоединением вершины, которая доминирует над всеми остальными [5, с. 7].

Далее показано сведение задачи Cons($\mathcal{P}$) к задаче L-HOM(H).

Лемма 1. Пусть задан диофантов частичный порядок $\mathcal{P}$ в языке L_A ($|A| = m$) и этому частичному порядку соответствует p-граф H. Задача Cons($\mathcal{P}$) сводится к задаче L-HOM(H) за полиномиальное время.

Доказательство. Для диофантового случая частичного порядка $\mathcal{P}$ в языке L_A определим множество уравнений $\overline{S}_{a \leqslant a}$ всех отношений порядка между элементами носителя $\mathcal{P}$. Для доказательства леммы опишем алгоритм 1. Графы представляются матрицами смежности, уравнения кодируются тройками: первый аргумент, второй аргумент, предикат.

Необходимо показать, что решениям системы уравнений $S(X_n)$ над $\mathcal{P}$ взаимно однозначно соответствуют гомоморфизмы из графа G в граф H, удовлетворяющие спискам L. Пусть для решения $p_i = (p_{i_1}, \ldots, p_{i_n})$ задано отображение

$$\varphi_i(x) = \begin{cases} h_{a_j}, & x = g_{a_j},\ g_{a_j} \in G_A, \\ h_{i_k}, & x = g_{i_k},\ k \in \{1, \ldots, n\},\ g_{i_k} \notin G_A. \end{cases}$$

Сначала нужно показать, что φ_i — это гомоморфизм, то есть

$$\forall g_r, g_s \in V(G)\big((g_r, g_s) \in E(G) \Rightarrow (\varphi_i(g_r), \varphi_i(g_s)) \in E(H)\big).$$

Так как константные символы $S(X_n)$ преобразуются по шагу 4 алгоритма 1 в вершины G_A и все их соотношения добавлены в систему на шаге 2, то можно выделить подграф $W(G_A, E)$ графа G, который изоморфен p-графу H, построенному на шаге 3 алгоритма. Поэтому

$$\forall g_{a_r}, g_{a_s} \in G_A\big((g_{a_r}, g_{a_s}) \in E(G) \Rightarrow (\varphi_i(g_{a_r}), \varphi_i(g_{a_s})) \in E(H)\big),$$

что обеспечивается построением φ_i.

Алгоритм 1. Сведение входа задачи Cons($\mathcal{P}$) к L-HOM(H)

Вход: система уравнений $S(X_n)$ над частичным порядком $\mathcal{P}$ в языке L_A (вход для задачи Cons($\mathcal{P}$)).

Выход: граф G со списками вершин L (вход для задачи L-HOM(H)).

1: Система уравнений $S(X_n)$ разбивается на семь непересекающихся подсистем уравнений (1). При этом уравнение $t_1 = t_2$ над $\mathcal{P}$, где t_1 и t_2 — термы, эквивалентно паре неравенств $t_1 \leqslant t_2$ и $t_2 \leqslant t_1$. Поэтому система $S(X_n)$ заменяется на эквивалентную ей систему уравнений $S'(X_n)$, содержащую только уравнения типов $S_{a\leqslant a}$, $S_{x\leqslant a}$, $S_{a\leqslant x}$ и $S_{x\leqslant x}$.

2: Если подсистема $S_{a\leqslant a}$ в $S'(X_n)$ совместна, то её можно исключить из рассмотрения, потому что она не влияет на разрешимость системы $S'(X_n)$. Если $S_{a\leqslant a}$ в $S'(X_n)$ несовместна, то: 1) система $S(X_n)$ несовместна; 2) в системе $S_{a\leqslant a}$ существует уравнение $a_i \leqslant a_j$, которое не верно над $\mathcal{P}$. Вводится система $S''(X_n) = S'(X_n) \cup \overline{S}_{a\leqslant a}$, эквивалентная системе $S(X_n)$.

3: По частичному порядку $\mathcal{P}$ строится p-граф H. Каждому элементу носителя $p_i \in \mathcal{P}$ ставится в соответствие вершина графа $h_i \in V(H)$. Если для $p_i, p_j \in \mathcal{P}$ верно $\mathcal{P} \vDash p_j \leqslant p_i$, то $(h_i, h_j) \in E(H)$. Так как рассматривается диофантов случай, то константные символы $a_i \in A$ также переходят в элемент h_i, как и p_i.

4: По системе $S''(X_n)$ строится граф G: вершинам графа G соответствуют переменные X_n и константы языка A; дуга (t_i, t_j) присутствует в графе G, если уравнение $t_j \leqslant t_i$ содержится в системе $S'(X_n)$, $t_i, t_j \in \{X_n \cup A\}$. Для вершин графа G, соответствующих константным символам A, вводится обозначение $G_A = \{g_{a_1}, \ldots, g_{a_m}\}$.

5: По подсистеме $S_{x\leqslant a} \cup S_{a\leqslant x}$ строятся списки вершин графа H. Для каждой переменной x_i в подсистеме выделяются те уравнения, которые зависят только от этой переменной: $S_{x_i\leqslant a} \cup S_{a\leqslant x_i}$. Далее из системы $S_{x_i\leqslant a}$ выбираются все константы (их множество обозначим $\overline{A}_{x_i}$). Аналогично выбирается множество $\underline{A}_{x_i}$ констант из $S_{a\leqslant x_i}$. Определяется множество констант $L(x_i) = \overline{A}^{\downarrow}_{x_i} \cap \underline{A}^{\uparrow}_{x_i}$. Если подсистема $S_{x_i\leqslant a}$ пустая, то $L(x_i) = A$; для всех констант $L(a_i) = \{a_i\}$.

6: Для списков элементов из шага 5 определяются списки вершин: для каждого $x \in X_n$ по шагу 4 определена вершина $v \in V(G)$ и списку $L(x) = \{a_{i_1}, \ldots, a_{i_r}\}$ соответствует список $L(v) = \{h_{i_1}, \ldots, h_{i_r}\} \subseteq V(H)$, где вершина h_{i_k} соответствует константе a_{i_k} из шага 3. Константным символам $a_i \in A$ соответствуют вершины $g_i \in G_A$ по шагу 4 и $L(g_i) = h_i$, где h_i соответствует константе a_i по шагу 3.

Далее пусть зафиксирован элемент $g_{i_k} \in V(G) \setminus G_A$ для решения $p_i = (p_{i_1}, \ldots, p_{i_n})$ системы уравнений $S(X_n)$ и выбран произвольный элемент $g_l \in A$. Если $(g_{i_k}, g_l) \in E(G)$, то по шагу 4 алгоритма 1 вершина g_l соответствует элементу a_l, а вершина g_{i_k} — переменной x_k. Если $(g_{i_k}, g_l) \in E(G)$, то $x_k \leqslant a_l \in S(X_n)$. Для решения p_i переменной x_k соответствует элемент p_{i_k} и, следовательно, $\mathcal{P} \vDash p_{i_k} \leqslant p_l$. Из шага 3 алгоритма

следует, что $(h_{i_k}, h_l) \in E(H)$. Но $\varphi_i(g_l) = h_l$ и $\varphi_i(g_{i_k}) = h_{i_k}$. Эти рассуждения верны для всех элементов из $V(G) \setminus G_A$. Тем самым показано, что

$$\forall g_r \in \{V(G) \setminus G_A\} \forall g_{a_s} \in G_A \big((g_r, g_{a_s}) \in E(G) \to (\varphi_i(g_r), \varphi_i(g_{a_s})) \in E(H)\big).$$

Аналогичными рассуждениями доказывается, что

$$\forall g_{a_r} \in G_A \forall g_s \in \{V(G) \setminus G_A\} \big((g_{a_r r}, g_s) \in E(G) \to (\varphi_i(g_{a_r}), \varphi_i(g_s)) \in E(H)\big);$$
$$\forall g_r, g_s \in \{V(G) \setminus G_A\} \big((g_r, g_{a_s}) \in E(G) \to (\varphi_i(g_r), \varphi_i(g_s)) \in E(H)\big).$$

Таким образом, отображение $\varphi_i : G \to H$ является гомоморфизмом.

Теперь нужно показать, что данный гомоморфизм удовлетворяет условиям списков $L(g)$ для графа G.

По шагам 5 и 6 видно, что $L(g_i) = h_i$ для всех $g_i \in G_A$. Так как $\varphi_i(g_i) = h_i$ для всех $g_i \in G_A$, то для всех элементов из G_A гомоморфизм φ_i удовлетворяет условиям списков.

Пусть $g_{i_k} \in \{V(G) \setminus G_A\}$ и $\varphi_i(g_{i_k}) = h_{i_k}$. Для него по шагам 5 и 6 определён список вершин $L(g_{i_k}) \subseteq V(H)$. В шаге 4 вершина g_{i_k} сопоставлена переменной x_k, которой соответствует список элементов $L(x_k)$ (шаг 5). Так как $p_i = (p_{i_1}, \ldots, p_{i_n})$ — решение $S(X_n)$, то $a_{i_k} \in L(x_k)$. По шагу 6 это означает, что $h_{i_k} = \varphi_i(g_{i_k}) \in L(g_{i_k})$, т. е. φ_i удовлетворяет условиям списков вершин.

Таким образом, каждое решение системы уравнений $p_i = (p_{i_1}, \ldots, p_{i_n})$ определяет гомоморфизм φ_i, удовлетворяющий попаданию в списки вершин.

Для взаимно однозначного соответствия осталось показать, что других гомоморфизмов, удовлетворяющих спискам, не существует.

Пусть есть гомоморфизм $\psi : G \to H$, который удовлетворяет спискам вершин, полученным по алгоритму, но не соответствует ни одному из решений. Так как ψ удовлетворяет всем спискам, то он удовлетворяет и всем спискам для $g \in G_A$, то есть сохраняется изоморфность H и подграфа G. Если система уравнений несовместна из-за уравнений в $S_{a \leqslant a}$, то гомоморфизм $G \to H$ нельзя составить так, чтобы он удовлетворял спискам для G_A.

Пусть выделен некоторый элемент $g_i \in V(G) \setminus G_A$ и $\psi(g_i) = h_j$. Из алгоритма 1 видно, что переменная x_i соответствует вершине g_i, а элемент и константа (p_j и a_j) частичного порядка $\mathcal{P}$ — вершине h_j. Так как $h_j \in L(g_i)$, то $a_j \in L(x_i)$ по шагам 5 и 6 алгоритма. Аналогичную процедуру можно проделать по остальным элементам из $V(G) \setminus G_A$ и получить значения $\overline{a} = (a_{\alpha_1}, \ldots, a_{\alpha_n})$ для всех переменных X_n.

В системе уравнений четыре вида подсистем: $S_{a \leqslant a}$, $S_{x \leqslant a}$, $S_{a \leqslant x}$ и $S_{x \leqslant x}$. Как уже отмечалось, $S_{a \leqslant a}$ непротиворечива, иначе ψ не гомоморфизм. Подсистемы $S_{x \leqslant a}$ и $S_{a \leqslant x}$ не могут быть противоречивыми при подстановке значений из $\overline{a}$, так как они удовлетворяют всем спискам вершин, которые составляются по данным уравнениям на шаге 5. Значит, при подстановке значений $\overline{a}$ в систему $S_{x \leqslant x}$ получаются неверные над $\mathcal{P}$ выражения. Пусть таким выражением будет $a_k \leqslant a_l$, соответствующее уравнению $x_i \leqslant x_j$, т. е. $\mathcal{P} \not\models p_k \leqslant p_l$. Вершины h_k, h_l являются образами вершин g_i, g_j, соответствующих переменным x_i, x_j. По алгоритму 1 переменные x_i, x_j преобразуются в вершины $g_i, g_j \in \{V(G) \setminus G_A\}$ и между ними есть дуга $(g_j, g_i) \in E(G)$. Так как ψ — гомоморфизм, то $(\psi(g_j), \psi(g_i)) = (h_l, h_k) \in E(H)$. Но так как $\mathcal{P} \not\models p_k \leqslant p_l$, то дуги (h_l, h_k) в H быть не может. Противоречие. Тем самым доказано, что решения системы уравнений взаимно однозначно определяют гомоморфизмы между графами, удовлетворяющие спискам.

Алгоритм сведения задач является полиномиальным по числу элементов частичного порядка (m) и количеству переменных в системе уравнений (n). Первый шаг алгоритма — замена подсистем $S_{a=a}$, $S_{x=a}$ и $S_{x=x}$ на $S_{a\leqslant a}$, $S_{x\leqslant a}$, $S_{a\leqslant x}$ и $S_{x\leqslant x}$ — увеличивает количество уравнений не более чем вдвое. Следующий шаг — проверка совместности подсистемы $S_{a\leqslant a}$; проверяется совместность не более m^2 уравнений. Построение p-графа H по частичному порядку $\mathcal{P}$ представляет собой копирование матрицы смежности. Для получения графа G строится матрица смежности с $(n+m)$ вершинами и не более чем $(n+m)^2$ дугами. Наконец, построение списков вершин — это поиск пересечений конусов. Для элемента частичного порядка a построение $a^{\uparrow}$ и $a^{\downarrow}$ происходит путём сравнения элемента a со всеми элементами частичного порядка. Поэтому трудоёмкость данной операции оценивается как $O(m)$. Построение $A^{\uparrow}_{x_i}$ произвольного подмножества элементов частичного порядка проходит так, что сначала строится $a^{\uparrow}$ для любого элемента $a \in A_{x_i}$. Далее остальные элементы из A_{x_i} сравниваются с $a^{\uparrow}$. Сложность этой процедуры можно оценить как $O(m^2)$. Необходимо построить n таких множеств. Следовательно, алгоритм сведения задачи $\mathrm{Cons}(\mathcal{P})$ к задаче L-HOM(H) является полиномиальным по числу элементов частичного порядка и числу переменных в системе уравнений и имеет трудоёмкость $O(n^2+nm^2+m^2)$. ■

Следует отметить, что задачу L-HOM(H) нельзя свести к задаче $\mathrm{Cons}(\mathcal{P})$: задача L-HOM$(H)$ шире, чем задача $\mathrm{Cons}(\mathcal{P})$. Например, на рис. 3 показана диаграмма Хассе частичного порядка $\mathcal{M}$. Рассмотрим все возможные конусы:

1) $\varnothing^{\downarrow} = \varnothing^{\uparrow} = a_1^{\downarrow} = a_3^{\uparrow} = \{a_1, a_2, a_3\}$;
2) $a_2^{\uparrow} = \{a_2, a_3\}^{\uparrow} = \{a_1, a_2\}$;
3) $a_2^{\downarrow} = \{a_1, a_2\}^{\downarrow} = \{a_2, a_3\}$;
4) $\{a_1, a_2\}^{\uparrow} = \{a_1, a_3\}^{\uparrow} = \{a_1, a_2, a_3\}^{\uparrow} = \{a_1\}$;
5) $\{a_2, a_3\}^{\downarrow} = \{a_1, a_3\}^{\downarrow} = \{a_1, a_2, a_3\}^{\downarrow} = \{a_3\}$.

Если задать список $L(v) = \{a_1, a_3\}$, то видно, что никакое пересечение из всевозможных конусов не даст множество элементов $\{a_1, a_3\}$. А так как списки элементов формируются по системам уравнений $S_{x\leqslant a}$ и $S_{a\leqslant x}$, то и ни по какой системе уравнений нельзя составить указанный список элементов.

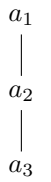


Рис. 3. Частичный порядок $\mathcal{M}$

Теорема 5. Пусть $\mathcal{P}$ — частичный порядок и H — соответствующий этому частичному порядку p-граф. Тогда если H — приведённый интервальный орграф, то задача $\mathrm{Cons}(\mathcal{P})$ разрешима за полиномиальное время.

Доказательство. По лемме 1 задача $\mathrm{Cons}(P)$ полиномиально сводится к задаче L-HOM(H), где H — рефлексивный ацикличный транзитивный ориентированный граф. Рефлексивные ацикличные транзитивные орграфы являются подклассом рефлексивных орграфов. Поэтому по теореме 3 задача L-HOM(H) разрешима за полиномиальное время, если H — приведённый интервальный орграф. ■

Можно отметить случай, в котором частичный порядок $\mathcal{P}$ задан в языке L без констант. Тогда $\mathrm{Cons}(\mathcal{P})$ решается тривиально. Любая система уравнений $S(X_n)$ над $\mathcal{P}$ состоит из уравнений двух типов: $S_{x=x}$ и $S_{x\leqslant x}$. Следовательно, решениями данной системы будут точки вида $P_i = (p_i, \ldots, p_i)$, $p_i \in \mathcal{P}$.

Что касается задачи $\mathrm{RET}(H)$, то её можно свести к задаче $\mathrm{Cons}(\mathcal{P})$ для p-графов H.

Лемма 2. Пусть задан p-граф H и ему соответствует частичный порядок $\mathcal{P}$ в языке L_A. Задача $\mathrm{RET}(H)$ сводится к задаче $\mathrm{Cons}(\mathcal{P})$ за полиномиальное время.

Доказательство. Для сведения построим по графам G, H и спискам $L(v)$ систему уравнений $S(X_n)$ и частичный порядок $\mathcal{P}$.

Дано: p-граф G и списки вершин $L(v) \in V(H)$, $v \in V(G)$ (вход для задачи $\mathrm{RET}(H)$). Нужно построить систему уравнений $S(X_n)$ над частичным порядком $\mathcal{P}$ в языке L_A (вход для задачи $\mathrm{Cons}(\mathcal{P})$).

По p-графу H получим частичный порядок $\mathcal{P}$ стандартным образом.

Пусть в графе G всего n вершин. По графу G строится система уравнений $S_{x\leqslant x}$ следующим образом: вершинам графа G ставятся в соответствие переменные X_n; если в графе G присутствует дуга (g_i, g_j), то в систему добавляется уравнение $x_i \geqslant x_j$. По спискам $L(v)$ строится множество уравнений $S_{x=a}$: если для вершины g_i графа G список $L(g_i) = \{h_j\}$, то в систему добавляется уравнение $x_i = a_j$, где x_i соответствует вершине g_i, а константа a_j — вершине h_j. Если $L(g_i) = V(H)$, то никакие уравнения в систему не добавляются. Система $S(X_n)$ представляет собой объединение $S_{x\leqslant x} \cup S_{x=a}$.

Аналогично сведению задачи $\mathrm{Cons}(\mathcal{P})$ к задаче $\text{L-HOM}(H)$ в лемме 1 можно показать соответствие гомоморфизмов между графом G со списками L и графом H и решениями системы $S(X_n)$ над $\mathcal{P}$ в языке L_A.

Полиномиальность сведения (по числу вершин графов $|V(G)| = n$ и $|V(H)| = m$) заключается в том, что построение частичного порядка $\mathcal{P}$ по графу H является копированием матрицы смежности и имеет трудоёмкость $O(m)$. Построение системы уравнений $S(X_n)$ не превосходит по сложности $O(n^2)$, так как построение $S_{x\leqslant x}$ оценивается порядком числа дуг графа G, а построение системы $S_{x=a}$ — количеством вершин графа G (это множество уравнений определяется по спискам $L(v)$). Итого, трудоёмкость сведения оценивается как $O(n^2 + m)$. ■

Теорема 6. Пусть задан конечный частичный порядок $\mathcal{P}$. Если для соответствующего ему p-графа H граф $U(H)$ является циклом длины больше 3, то задача $\mathrm{Cons}(\mathcal{P})$ является NP-полной.

Доказательство. По лемме 2 задача $\mathrm{Cons}(\mathcal{P})$ полиномиально сводится к задаче $\mathrm{RET}(H)$, где H — рефлексивный ациклический транзитивный ориентированный граф. Рефлексивные ациклические транзитивные орграфы являются подклассом рефлексивных орграфов. Поэтому по теореме 2 задача $\text{L-HOM}(H)$ является NP-полной, если $U(H)$ является циклом длины больше 3. ■

Такие частичные порядки существуют. Один из них представлен на рис. 4.

Таким образом, показано, что для задачи $\mathrm{Cons}(\mathcal{P})$ область значений переменных определяется пересечением конусов в частичном порядке $\mathcal{P}$, в отличие от произвольного случая задания списков в задаче $\text{L-HOM}(H)$ и дуальностью задания списков в задаче $\mathrm{RET}(H)$. Задача $\mathrm{Cons}(\mathcal{P})$ может быть как полиномиально разрешимой, так и NP-полной, в зависимости от частичного порядка $\mathcal{P}$. Необходимые условия для удовлетворения данных свойств получены в теоремах 5 и 6.

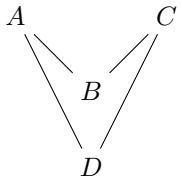


Рис. 4. Частичный порядок $\mathcal{P}$, у которого граф $U(H)$ является циклом длины 4

Заключение

Сформулирован критерий нётеровости по уравнениям для частично упорядоченных множеств и даны необходимые условия для полиномиальной разрешимости и NP-полноты задачи разрешимости системы уравнений над частично упорядоченными множествами.

ЛИТЕРАТУРА

1. *Даниярова Э. Ю., Мясников А. Г., Ремесленников В. Н.* Алгебраическая геометрия над алгебраическими системами. Новосибирск: Изд-во СО РАН, 2016. 243 с.
2. *Гуров С. И.* Булевы алгебры, упорядоченные множества, решетки: Определения, свойства, примеры. М.: Либроком, 2013. 221 с.
3. *Гретцер Г.* Общая теория решеток. М.: Мир, 1982. 456 с.
4. *Оре О.* Теория графов. М.: Наука, 1980. 380 с.
5. *Feder T., Hell P., Hunag J., and Rafiey A.* Interval graphs, adjusted interval digraphs, and reflexive list homomorphism // Discr. Appl. Math. 2012. V. 160. Iss. 6. P. 697–707.
6. *Feder T., Hell P., and Hunag J.* List Homomorphisms and Retractions to Reflexive Digraphs. `http://theory.stanford.edu/~tomas/ref.pdf`. 2007. 26 p.
7. *Hell P. and Rafiey A.* The Dichotomy of List Homomorphisms for Digraphs. `http://arxiv.org/abs/1004.2908`. 2010.
8. *Feder T. and Hell P.* List homomorphisms to reflexive graphs // J. Combinatorial Theory. 1998. V. 72. No. 2. P. 236–250.

REFERENCES

1. *Daniyarova E. Yu., Myasnikov A. G., and Remeslennikov V. N.* Algebraicheskaya geometriya nad algebraicheskimi sistemami [Algebraic Geometry over Algebraic Systems]. Novosibirsk, SB RAS Publ., 2016. 243 p. (in Russian)
2. *Gurov S. I.* Bulevy algebry, uporyadochennye mnozhestva, reshetki: Opredeleniya, svoystva, primery [Boolean Algebras, Ordered Sets, Lattices: Definitions, Properties, Examples]. Moscow, Librocom Publ., 2013. 221 p. (in Russian)
3. *Grätzer G.* General Lattice Theory. Basel, Birkhäuser Verlag, 1978.
4. *Ore O.* Theory of Graphs. AMS, 1965. 270 p.
5. *Feder T., Hell P., Hunag J., and Rafiey A.* Interval graphs, adjusted interval digraphs, and reflexive list homomorphism. Discr. Appl. Math., 2012, vol. 160, iss. 6, pp. 697–707.
6. *Feder T., Hell P., and Hunag J.* List Homomorphisms and Retractions to Reflexive Digraphs. `http://theory.stanford.edu/~tomas/ref.pdf`, 2007. 26 p.
7. *Hell P. and Rafiey A.* The Dichotomy of List Homomorphisms for Digraphs. `http://arxiv.org/abs/1004.2908`, 2010
8. *Feder T. and Hell P.* List homomorphisms to reflexive graphs. J. Combinatorial Theory, 1998, vol. 72, no. 2, pp. 236–250.

УДК 512.532 DOI 10.17223/20710410/64/2

СОПОСТАВЛЕНИЕ СВОЙСТВ ВНЕШНЕПЛАНАРНОСТИ И ОБОБЩЁННОЙ ВНЕШНЕПЛАНАРНОСТИ ГРАФОВ КЭЛИ ПЛАНАРНЫХ ПОЛУГРУПП

Д. В. Соломатин

Омский государственный педагогический университет, г. Омск, Россия

E-mail: solomatin_dv@omgpu.ru

Найдены две бесконечные серии полугрупп, свойство внешнепланарности графов Кэли в которых эквивалентно свойству обобщённой внешнепланарности их графов Кэли, но не эквивалентно свойству планарности, и одна бесконечная серия полугрупп, свойство обобщённой внешнепланарности графов Кэли которых эквивалентно свойству планарности их графов Кэли, но не эквивалентно внешнепланарности. Доказано, что граф Кэли конечной полугруппы не изоморфен ни одному из запрещённых подграфов Седлачека, взятых с любой ориентацией и раскраской рёбер, по характеристическому свойству обобщённой внешнепланарности.

Ключевые слова: *графы Чартрэнда — Харари, графы Седлачека, полугруппы с планарными графами Кэли.*

COMPARISON OF OUTERPLANARITY AND GENERALIZED OUTERPLANARITY PROPERTIES FOR CAYLEY GRAPHS OF PLANAR SEMIGROUPS

D. V. Solomatin

Omsk State Pedagogical University, Omsk, Russia

We have found two infinite series of semigroups whose Cayley graphs have an outerplanarity property equivalent to the generalized outerplanarity property of their Cayley graphs, but not equivalent to the planarity property, and one infinite series of semigroups whose Cayley graphs have a generalized outerplanarity property equivalent to the planarity property of their Cayley graphs, but not equivalent to outerplanarity. It is proved that the Cayley graph of a finite semigroup is not isomorphic to any of the forbidden Sedláček's subgraphs by the characteristic property of generalized outer planarity with any orientation and edge coloring.

Keywords: *Chartrand — Harari graphs, Sedláček graphs, semigroups with planar Cayley graphs.*

Введение

Внешнепланарные графы, как связные графы, имеющие не менее трёх вершин, которые можно вложить в плоскость так, чтобы все вершины лежали во внешней грани, введены Г. Чартрэндом и Ф. Харари [1] при решении вопросов планарности графов, образованных совершенными паросочетаниями, связывающими две копии одного графа.

Дальнейшее обобщение внешнепланарных графов развивалось в разных направлениях, исторически первым было предложенное в [2] И. Седлачеком рассмотрение таких планарных графов, каждое ребро которых принадлежит внешней грани хотя бы одной из своих вершин.

В цифровую эпоху внешнепланарные графы находят применение в информатике, особенно в областях, связанных с сетевыми структурами и химической информатикой, а именно: внешнепланарные графы используют для описания отношений в сетевых структурах, так как они представляют собой подмножество плоских и круговых графов; они могут применяться для моделирования связности, сбора данных, маршрутизации, мобильности, энергоэффективности, управления топологией, анализа трафика, поиска кратчайших путей и балансировки нагрузки. Приложения в хемоинформатике ещё шире, так как многие химические соединения можно описать с помощью внешнепланарных графов. Например, в [3] описан полиномиальный алгоритм, который вычисляет максимальный общий подграф между двумя внешнепланарными графами, он оказался полезен для решения задач прогнозирования в обозначенной сфере. Способность внешнепланарных графов представлять сложные отношения визуально-интуитивным образом делает их ценным инструментом в различных областях дискретной математики. Планарные графы, в свою очередь, находят приложения при проектировании микросхем. Известно также применение частично коммутативных полугрупп при анализе возможности распараллеливания алгоритмов и программ.

В обзоре [4] анализируются вопросы внешнепланарности графов Кэли полугрупп, среди прочего — в классах частично коммутативных полугрупп, конечных свободных коммутативных полугрупп, моноидов и полугрупп с нулём, связанные с возможностью их обобщения. В разных классах полугрупп нередко возникают такие ситуации, что полугруппа допускает обобщённый внешнепланарный, но не допускает внешнепланарный граф Кэли [5]. Очевидны примеры полугрупп, в которых свойство планарности графов Кэли эквивалентно свойству внешнепланарности и свойству обобщённой внешнепланарности, такими среди прочих являются полугруппы с нулевым умножением. В работе приведены несколько классов полугрупп, для которых свойство графа Кэли быть обобщённым внешнепланарным эквивалентно лишь его внешнепланарности либо лишь его планарности.

1. Предварительные сведения

Приведём определения ключевых понятий. *Графом Кэли полугруппы* S относительно множества образующих её элементов X называем ориентированный мультиграф $\mathrm{Cay}(S, X) = (S, \{(a, x, b) : a, b \in S, x \in X, ax = b\})$ с помеченными дугами. Дуга (a, x, b) начинается в вершине $a \in S$, заканчивается в вершине $b \in S$ и помечена элементом $x \in X$ тогда и только тогда, когда в полугруппе S выполнено равенство $ax = b$.

Граф называется *внешнепланарным*, если он изоморфен плоскому графу, каждая вершина плоской укладки которого принадлежит внешней грани. Граф называется *обобщённым внешнепланарным*, если он изоморфен плоскому графу, каждое ребро плоской укладки которого принадлежит внешней грани хотя бы одной из своих вершин. Говорим, что *полугруппа* S *допускает* (*обобщённый*) *внешнепланарный граф Кэли*, если относительно некоторого множества образующих X *основа* $\mathrm{SCay}(S, X) = \big(S, \{\{a, b\} : a, b \in S, \exists x \in X\,(ax = b)\}\big)$ графа $\mathrm{Cay}(S, X)$, полученная из исходного путём удаления петель, меток и заменой всех дуг вида (a, x, b) и (b, y, a) для $a, b \in S$ и $x, y \in X$ одним ребром $\{a, b\}$, является (обобщённым) внешнепланарным графом.

Согласно критерию Седлачека, граф является обобщённым внешнепланарным тогда и только тогда, когда он не содержит подграфов, стягиваемых к запрещённым графам Седлачека G_1–G_{12}. Для сравнения, согласно критерию Чартрэндта — Харари, граф внешнепланарен тогда и только тогда, когда он не содержит подграфов, гомеоморфных графам Чартрэнда — Харари, $K_{2,3}$ или K_4.

2. Основной результат

В [6] доказано, что нециклическая полугруппа S с одним определяющим соотношением, допускающая полугрупповое тождество, имеет планарный граф Кэли тогда и только тогда, когда S антиизоморфна одной из полугрупп: $S_1 = \langle a, b \mid ab = ba\rangle$, $S_{2,k} = \langle a, b \mid ab = b^k\rangle$, $k = 1, 2, \ldots$, $S_3 = \langle a, b \mid aba = ba\rangle$, $S_4 = \langle a, b \mid aba = b\rangle$, $S_5 = \langle a, b \mid a^2 = b^2\rangle$, $S_6 = \langle a, b \mid aba^2 = ba\rangle$; или изоморфна одной из полугрупп: S_1, $S_{2,1}$, S_4, S_5. Заметим, что в силу [7] полугруппы S_1, $S_{2,k}$, $k = 1, 2, \ldots$, S_3, S_4, S_5 и S_6 — это все возможные полугруппы, которым может быть изоморфна или антиизоморфна нециклическая полугруппа с одним определяющим соотношением, допускающая полугрупповое тождество. Как оказалось, полугруппа S с одним определяющим соотношением и с тождеством допускает внешнепланарный граф Кэли тогда и только тогда, когда она допускает обобщённый внешнепланарный граф Кэли.

Теорема 1. Если S — это нециклическая полугруппа с единственным определяющим соотношением, допускающая полугрупповое тождество, то следующие условия эквивалентны:

1) полугруппа S допускает внешнепланарный граф Кэли;
2) полугруппа S допускает обобщенный внешнепланарный граф Кэли;
3) полугруппа S антиизоморфна одной из полугрупп: $S_{2,k} = \langle a, b \mid ab = b^k\rangle$, где $k \leqslant 3$; $S_3 = \langle a, b \mid aba = ba\rangle$; или изоморфна полугруппе $S_{2,1} = \langle a, b \mid ab = b\rangle$.

Доказательство. В [8, теорема 5.1] доказано, что нециклическая полугруппа с одним определяющим соотношением и с тождеством допускает внешнепланарный граф Кэли тогда и только тогда, когда она антиизоморфна одной из полугрупп $S_{2,k} = \langle a, b \mid ab = b^k\rangle$, где $k \leqslant 3$, или $S_3 = \langle a, b \mid aba = ba\rangle$; или изоморфна полугруппе $S_{2,1} = \langle a, b \mid ab = b\rangle$. Поэтому для доказательства теоремы достаточно выполнить дополнительную проверку допустимости обобщённого внешнепланарного графа Кэли перечисленных в [6] полугрупп, допускающих планарные графы Кэли.

Наличие обобщённой внешнеплоской укладки представлено в [8] на рис. 5.1.1 и 5.1.2 для $1 \leqslant k \leqslant 3$ и на рис. 5.1.3. Отсутствие обобщённой внешнеплоской укладки в оставшихся случаях рассматриваемых полугрупп с плоскими графами Кэли обосновывается наличием в основе их графа Кэли подграфов, стягиваемых к первому из графов Седлачека, G_1 в обозначениях из [9]. Более точно, для полугруппы, изоморфной или антиизоморфной S_1 или изоморфной S_4, наличие подграфа, стягиваемого к G_1 в основе, обеспечивается существованием трёх непересекающихся маршрутов от вершины a^2 до вершины a^2b^3, которым принадлежат вершины следующих множеств: $\{a^2, a, ab, ab^2, ab^3, a^2b^3\}$, $\{a^2, a^2b, a^2b^2, a^2b^3\}$, $\{a^2, a^3, a^3b, a^3b^2, a^3b^3, a^2b^3\}$. Для полугруппы, антиизоморфной $S_{2,k}$ при $k \geqslant 4$, наличие подграфа, стягиваемого к G_1 в основе, обеспечивается существованием трёх непересекающихся маршрутов от вершины $b^{2k-1}a$ до вершины b^2a (рис. 1). Здесь пунктиром изображён фрагмент маршрута, полученный последовательным умножением на b слева $(k-3)$ раза с учётом того, что для любого натурального числа n имеет место равенство $ab^na = abb^{n-1}a = b^kb^{n-1}a = b^{n+k-1}a$, которое используется при умножении на a слева.

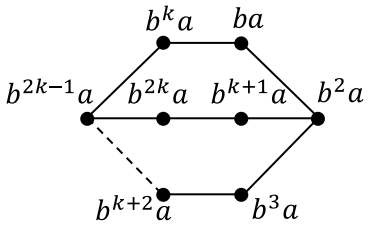


Рис. 1. Подграф основы графа Кэли полугруппы, антиизоморфной $S_{2,k} = \langle a, b \mid ab = b^k \rangle$, при $k \geqslant 4$ стягиваемый к первому графу Седлачека

В основе графа Кэли полугруппы, изоморфной S_5, обнаруживается подграф, стягиваемый к G_1 и состоящий из трёх непересекающихся маршрутов от вершины ba до a^5, восстанавливаемых на вершинах следующих множеств: $\{ba, bab, b^3a, b^5, a^6, a^5\}$, $\{ba, b^3, a^4, a^5\}$, $\{ba, b, a^2, a^3, a^3b, a^5\}$.

В основе графа Кэли полугруппы, антиизоморфной S_4, обнаруживается подграф, стягиваемый к G_1 и состоящий из трёх непересекающихся маршрутов от вершины b^2a^2 до bab, восстанавливаемых на вершинах следующих множеств: $\{b^2a^2, ba^2, ba, b, ab, bab\}$, $\{b^2a^2, b^2a, b^2, bab\}$, $\{b^2a^2, b^3a^2, b^3a, b^3, ab^3, bab\}$. Отметим попутно, что на соответствующем данной полугруппе рис. 11 в [6] содержится типографская неточность, вместо вершины ba следует читать bab.

В основе графа Кэли полугруппы, антиизоморфной S_5, обнаруживается подграф, стягиваемый к G_1 и состоящий из трёх непересекающихся маршрутов от вершины ab до a^5, восстанавливаемых на вершинах следующих множеств: $\{ab, bab, ab^3, b^5, a^6, a^5\}$, $\{ab, b^3, a^4, a^5\}$, $\{ab, b, a^2, a^3, ba^3, a^5\}$.

И наконец, в основе графа Кэли полугруппы, антиизоморфной S_6, обнаруживается подграф, стягиваемый к G_1 и состоящий из трёх непересекающихся маршрутов от вершины ba^4 до ba, восстанавливаемых на вершинах следующих множеств: $\{ba^4, a^4, a^3, a^2, a, ba\}$, $\{ba^4, ba^3, ba^2, ba\}$, $\{ba^4, b^2a^4, b^2a^3, b^2a^2, b^2a, ba\}$. Следовательно, графы Кэли полугрупп, допускающих планарные графы Кэли, но не удовлетворяющих условиям теоремы, не являются обобщёнными внешнепланарными. ■

Найти информацию о частично коммутативных полугруппах, необходимую для понимания следующей теоремы, можно в [10].

Теорема 2. Если $S(\Gamma)$ — это частично коммутативная свободная полугруппа, соответствующая графу коммутативности Γ множества образующих её элементов, то следующие условия эквивалентны:

1) полугруппа $S(\Gamma)$ допускает внешнепланарный граф Кэли;
2) полугруппа $S(\Gamma)$ допускает обобщённый внешнепланарный граф Кэли;
3) степень любой вершины в графе Γ равна нулю, то есть полугруппа $S(\Gamma)$ антикоммутативная.

Доказательство. Пусть V_Γ — множество вершин графа Γ. Тогда, как показано в [8, рис. 6.1.1], при наличии хотя бы одной пары коммутирующих элементов в множестве образующих основа графа Кэли содержит три попарно непересекающихся маршрута, восстанавливаемых на следующих множествах вершин: $\{sa^3b^2, sa^3b, sa^2b, sab, sb, sb^2\}$, $\{sa^3b^2, sa^2b^2, sab^2, sb^2\}$, $\{sa^3b^2, sa^3b^3, sa^2b^3, sab^3, sb^3, sb^2\}$, где s — слово из букв алфавита V_Γ, возможно пустое, завершающееся элементом, не коммутирующим с коммутирующими между собой a и b. Следовательно, основа графа Кэли такой полугруппы содержит подграф, стягиваемый к первому графу Сед-

лачека G_1. Поэтому полугруппа $S(\Gamma)$ допускает обобщённый внешнепланарный граф Кэли или допускает внешнепланарный граф Кэли тогда и только тогда, когда $S(\Gamma)$ некоммутативная. ■

Для полноты картины приведём ещё один результат, в классе n-веерных полурешёток описывающий полугруппы, свойство обобщённой внешнепланарности графа Кэли которых эквивалентно свойству его планарности.

Теорема 3. Если $S = S_k^n$ — это n-веерная полурешётка, то следующие условия эквивалентны:

1) полугруппа S допускает планарный граф Кэли;
2) полугруппа S допускает обобщённый внешнепланарный граф Кэли;
3) $\left|S^{(2)}\right| \leqslant 3$, где $S^{(2)}$ — множество всех ненулевых слов полугруппы S вида $a_i a_j$, $i \neq j$.

Доказательство. Характеристическому свойству планарности графа Кэли полугрупп S_k^n, доказанному в [10], удовлетворяют только S_k^1 (тогда основу графа Кэли формирует пустой граф первого порядка при любом $k \geqslant 1$), S_k^2 (тогда основу графа Кэли формирует звезда $K_{1,k}$ при любом $k \geqslant 1$) и S_3^3 (ориентированная основа графа Кэли в этом случае приведена в [8, рис. 6.2.2]). В каждом из этих случаев граф Кэли является обобщённым внешнепланарным. ■

Развивая идеи решения задачи о допустимости графов Понтрягина — Куратовского, взятых с некоторой ориентацией и пометкой рёбер в качестве графов Кэли полугрупп [11], и аналогично графам Чартренда — Харари [8, теорема 7.1], рассмотрим вопрос о допустимости графов Седлачека, взятых с некоторой ориентацией и пометкой рёбер, в качестве графов Кэли полугрупп.

Теорема 4. Если G — граф Кэли конечной полугруппы, то G не изоморфен ни одному из графов Седлачека G_i, $1 \leqslant i \leqslant 12$, с любой ориентацией и раскраской рёбер.

Доказательство. Число вершин n и число рёбер m графа Кэли полугруппы связаны с числом образующих её элементов t равенством $nt = m$. В противном случае не хватит рёбер для операции умножения на каждый из образующих либо не хватит вершин для получения результатов такого умножения. Параметры n и m графов Седлачека приведены в таблице.

i	1	2	3	4	5	6	7	8	9	10	11	12
n	8	8	8	8	6	7	7	7	8	5	6	6
m	9	9	9	9	8	9	9	9	10	9	9	9

Как видим, для любого i, $1 \leqslant i \leqslant 12$, число рёбер $m = |E(G_i)|$ не делится на число вершин $n = |V(G_i)|$. Следовательно, ни один из графов Седлачека, взятый с некоторой ориентацией и пометкой рёбер, не изоморфен графу Кэли какой-либо полугруппы. ■

Заключение

В результате проведённого исследования найдены: бесконечные серии частично коммутативных полугрупп и бесконечные серии нециклических полугрупп с единственным определяющим соотношением, допускающие полугрупповое тождество, свойство внешнепланарности графов Кэли которых эквивалентно свойству обобщённой внешнепланарности, но не эквивалентно свойству планарности; серии n-веерных полурешёток, свойство планарности графов Кэли которых эквивалентно свойству обобщённой внешнепланарности, но не эквивалентно внешнепланарности. Кроме того,

доказано, что граф Кэли любой конечной полугруппы не изоморфен ни одному из запрещённых графов Седлачека из критерия обобщённой внешнепланарности графов, взятых с любой ориентацией и пометкой рёбер.

Полученные результаты могут быть использованы для установления взаимного отношения между классами полугрупп, допускающих внешнепланарные графы Кэли, и классами полугрупп, допускающих обобщённые внешнепланарные графы Кэли. Последнее открывает перспективы исследования более сложных конструкций, в частности ординальных сумм прямоугольных полугрупп, допускающих внешнепланарные графы Кэли и их обобщения.

ЛИТЕРАТУРА

1. *Chartrand G. and Harary F.* Planar permutation graphs // Annales de l'I. H. P. Section B. 1967. V. 3. No. 4. P. 433–438.
2. *Sedláček J.* O jednom zobecněni vnějškově rovinných grafů // Časopis Pěst. Mat. 1988. V. 2. No. 113. P. 213–218. (in Czech)
3. *Schietgat L., Ramon J., and Bruynooghe M.* A polynomial-time maximum common subgraph algorithm for outerplanar graphs and its application to chemoinformatics // Ann. Math. Artif. Intell. 2013. V. 69. P. 343–376.
4. *Соломатин Д. В.* Исследования полугрупп с планарными графами Кэли: результаты и проблемы // Прикладная дискретная математика. 2021. № 54. С. 5–57.
5. *Соломатин Д. В.* Прямые произведения циклических моноидов, допускающие внешнепланарные графы Кэли и их обобщения // Вестник ТвГУ. Сер. Прикладная математика. 2023. № 4. С. 43–56.
6. *Соломатин Д. В.* Конечно порожденные полугруппы с одним определяющим соотношением и с тождеством, допускающие планарные графы Кэли // Математика и информатика: Наука и образование. 2007. № 6. С. 42–48.
7. *Шнеерсон Л. М.* Тождества в полугруппах с одним определяющим соотношением // Логика, алгебра и вычисл. матем. Иваново, 1972. № 1–2. С. 139–156.
8. *Соломатин Д. В.* Строение полугрупп, допускающих внешнепланарные графы Кэли // Сиб. электрон. матем. изв. 2011. Т. 8. С. 191–212.
9. *Almeria J. C. and Sevilla A. M.* A linear algorithm to recognize maximal generalized outerplanar graphs // Mathematica Bohemica. 1997. V. 122. No. 3. P. 225–230.
10. *Соломатин Д. В.* Свободные частично коммутативные полугруппы и n-веерные полурешетки с планарными графами Кэли // Математика и информатика: Наука и образование. 2009. № 8. С. 36–39.
11. *Соломатин Д. В.* О допустимости некоторых графов в качестве графов Кэли полугрупп // Математика и информатика: Наука и образование. 2004. № 4. С. 32–34.

REFERENCES

1. *Chartrand G. and Harary F.* Planar permutation graphs. Annales de l'I. H. P., section B, 1967, vol. 3, no. 4, pp. 433–438.
2. *Sedláček J.* O jednom zobecněni vnějškově rovinných grafů [A generalization of outerplanar graphs]. Časopis Pěst. Mat., 1988, vol. 2, iss. 113, pp. 213–218. (in Czech)
3. *Schietgat L., Ramon J., and Bruynooghe M.* A polynomial-time maximum common subgraph algorithm for outerplanar graphs and its application to chemoinformatics. Ann. Math. Artif. Intell., 2013, vol. 69, pp. 343–376.
4. *Solomatin D. V.* Issledovaniya polugrupp s planarnymi grafami Keli: rezul'taty i problemy [Researches of semigroups with planar Cayley graphs: Results and problems]. Prikladnaya Diskretnaya Matematika, 2021, no. 54, pp. 5–57. (in Russian)

5. *Solomatin D. V.* Pryamye proizvedeniya tsiklicheskikh monoidov, dopuskayushchie vneshneplanarnye grafy Keli i ikh obobshcheniya [Direct products of cyclic monoids admitting outerplanar Cayley graphs and their generalizations]. Vestnik TvGU, Ser. Prikladnaya Matematika, 2023, no. 4, pp. 43–56. (in Russian)
6. *Solomatin D. V.* Konechno porozhdennye polugruppy s odnim opredelyayushchim sootnosheniem i s tozhdestvom, dopuskayushchie planarnye grafy Keli [Finitely generated semigroups with one defining relation and identity, admitting planar Cayley graphs]. Matematika i Informatika: Nauka i Obrazovanie, 2007, no. 6, pp. 42–48. (in Russian)
7. *Shneerson L. M.* Tozhdestva v polugruppakh s odnim opredelyayushchim sootnosheniem [Identities in semigroups with one defining relation]. Logika, Algebra i Vychisl. Matem., Ivanovo, 1972, no. 1–2, pp. 139–156. (in Russian)
8. *Solomatin D. V.* Stroenie polugrupp, dopuskayushchikh vneshneplanarnye grafy Keli [Semigroups with outerplanar Cayley graphs]. Sib. Elektron. Matem. Izv., 2011, vol. 8, pp. 191–212. (in Russian)
9. *Almeria J. C. and Sevilla A. M.* A linear algorithm to recognize maximal generalized outerplanar graphs. Mathematica Bohemica, 1997, vol. 122, no. 3, pp. 225–230.
10. *Solomatin D. V.* Svobodnye chastichno kommutativnye polugruppy i n-veernye polureshetki s planarnymi grafami Keli [Free partially commutative semigroups and n-fan semilattices with planar Cayley graphs]. Matematika i Informatika: Nauka i Obrazovanie, 2009, no. 8, pp. 36–39. (in Russian)
11. *Solomatin D. V.* O dopustimosti nekotorykh grafov v kachestve grafov Keli polugrupp [On the admissibility of some graphs as Cayley graphs of semigroups]. Matematika i Informatika: Nauka i Obrazovanie, 2004, no. 4, pp. 32–34. (in Russian)

МАТЕМАТИЧЕСКИЕ МЕТОДЫ КРИПТОГРАФИИ

УДК 519.7 DOI 10.17223/20710410/64/3

STREEBOG AS A RANDOM ORACLE

L. R. Akhmetzyanova, A. A. Babueva, A. A. Bozhko

CryptoPro, Moscow, Russia

E-mail: {lah, babueva, bozhko}@cryptopro.ru

The random oracle model is an instrument used for proving that protocol has no structural flaws when settling with standard hash properties is impossible or fairly difficult. In practice, however, random oracles must be instantiated with some specific hash functions that are not random oracles. Therefore, in the real world an adversary has broader capabilities than considered in the random oracle proof: it can exploit the peculiarities of a specific hash function to achieve its goal. In a case when a hash function is based on some building block, one can go further and show that even if the adversary has access to that building block, the hash function still behaves like a random oracle under some assumptions made about the building block. Thereby, the protocol can be proved secure against more powerful adversaries under less complex assumptions. The notion of indifferentiability formalizes that approach. In this paper, we show that Streebog, a Russian standardized hash function, is indifferentiable from a random oracle under an ideal cipher assumption for the underlying block cipher.

Keywords: *Streebog, GOST, random oracle, indifferentiability.*

«СТРИБОГ» КАК СЛУЧАЙНЫЙ ОРАКУЛ

Л. Р. Ахметзянова, А. А. Бабуева, А. А. Божко

КриптоПро, г. Москва, Россия

Модель со случайным оракулом используется для доказательства стойкости криптографических протоколов в случае, когда стандартные предположения об использующейся хеш-функции не позволяют этого сделать. Однако на практике для реализации случайного оракула в конкретном протоколе используется некоторая детерминированная хеш-функция, которая, безусловно, не является случайным оракулом. Следовательно, в реальном мире нарушитель обладает более широкими возможностями, чем предполагалось в доказательстве — он может использовать особенности конструкции конкретной хеш-функции для осуществления угрозы. Если используемая хеш-функция строится на основе некоторого другого примитива (например, блочного шифра), можно рассмотреть нарушителя, который имеет доступ напрямую к этому примитиву, и показать, что даже относительного такого нарушителя используемая хеш-функция ведёт себя как случайный оракул в предположении об идеальности используемого примитива. Таким образом можно доказать стойкость протокола относительно более сильных нарушителей в менее сильных предположениях об использующихся примитивах. Хеш-функции,

при использовании которых можно достичь такого результата, называются неразличимыми от случайного оракула. В данной работе показано, что хеш-функция «Стрибог» неразличима от случайного оракула в модели идеального блочного шифра.

Ключевые слова: *Стрибог, ГОСТ, случайный оракул, неразличимость.*

1. Introduction

The random oracle model introduced in [1] assumes that each party of the protocol and an adversary has access to a random oracle, which is used instead of a hash function. A random oracle [1] is an ideal primitive that models a random function. It provides a random output for each new query, and identical input queries produce the same answer. The random oracle model makes it possible to prove that the protocol has no structural flaws in situations when it is impossible or very difficult to deal with standard hash properties, which is the case for many efficient and elegant solutions. For example, such protocols and mechanisms as TLS [2], IPSec [3], and Schnorr signature [4, 5] were analyzed in the random oracle model; Russian standardized versions of TLS [6] and IPSec [7], as well as SESPAKE protocol [8, 9], shortened ElGamal signature [10], to-be-standardized RSBS blind signature [11], and postquantum Shipovnik signature [12] are also analyzed in the random oracle model.

In practice, however, being idealized primitives, random oracles do not exist and have to be instantiated with some specific hash functions that are not random oracles. Therefore, in the real world, an adversary has broader capabilities than those considered in the random oracle proof: it can exploit the peculiarities of a specific hash function to achieve its goal. To address such a situation, one can go further and consider the design of the hash function to show that, under some less complex and more specific assumptions than the whole function being a random oracle, it behaves like a random oracle. To do that, one must first understand what "behaves like a random oracle" means and what assumptions you need to make.

These questions for a particular class of hash functions are addressed by J. S. Coron et al. in [13, 14]. They study the case when an arbitrary-length hash function is built from some fixed-length building block (like an underlying compression function or a block cipher). They propose a definition based on Maurer et al.'s notion of indifferentiability [15] of what it means to implement a random oracle with such a construction under the assumption that the building block itself is an ideal primitive. The definition is chosen in a way that any hash function satisfying it can securely instantiate a random oracle in a higher-level application[1] (under the assumption that the building block is an ideal primitive). Hence, idealized assumptions are made about less complex lower-level primitive and, as a result, more adversarial capabilities are taken into account.

In this paper, we study whether Streebog, a Russian standardized hash function [16], can instantiate a random oracle. We recall that Streebog has always been a popular target for analysis. An overview of the results which study standard properties of the algorithm can be found in [17]. A recent paper [18] studies keyed version of Streebog as a secure pseudorandom function in a related-key resilient PRF model for an underlying block cipher, highlighting some important high-level design features of Streebog.

[1] We note that, as shown in [19], it only directly applies to cryptographic protocols which admit the so-called "single-stage security proofs."

Since Streebog is a modified Merkle — Damgard construction based on LSX-style block cipher in Miyaguchi— Preneel mode, we adopt the notion of Coron et al. The paper's main result is presented in Section 3: we prove that Streebog is indifferentiable from a random oracle under an ideal cipher assumption for the underlying block cipher. We benefit greatly from the work done in [13, 14] since their analysis is focused on Merkle — Damgard constructions with a block cipher in Davis — Meyer mode. However, Streebog's design features and a different structure of the compression function do not allow us to use the paper's results and pose several challenges.

2. Definitions

Let $|a|$ be the bit length of the string $a \in \{0,1\}^*$, the length of an empty string is equal to 0. For a bit string a we denote by $|a|_n = \lceil |a|/n \rceil$ the length of the string a in n-bit blocks. Let 0^u be the string consisting of u zeroes.

For a string $a \in \{0,1\}^*$ and a positive integer $l \leqslant |a|$ let $\mathrm{msb}_\ell(a)$ be the string consisting of the leftmost l bits of a. For nonnegative integers l and i, let $\mathrm{str}_l(i)$ be l-bit representation of i with the least significant bit on the right, let $\mathrm{int}(M)$ be an integer i such that $\mathrm{str}_l(i) = M$. For bit strings $a \in \{0,1\}^{\leqslant n}$ and $b \in \{0,1\}^{\leqslant n}$ we denote by $a + b$ a string $\mathrm{str}_n((\mathrm{int}(a) + +\mathrm{int}(b)) \bmod 2^n)$. If the value s is chosen uniformly at random from a set S, then we denote it $s \xleftarrow{\mathcal{U}} S$.

A block cipher E with a block size n and a key size k is the permutation family $\big(E_K \in \mathrm{Perm}(\{0,1\}^n) : K \in \{0,1\}^k\big)$, where K is a key.

2.1. Streebog hash function

The Streebog hash function is defined in [16]. For the purposes of the paper, we will define Streebog as a modification of Merkle — Damgard construction, which is applied to a prefix-free encoding of the message; in that we follow the approach of [13, 14]. We will also make the use of the equivalent representation of Streebog from [20]. For Streebog the length of an internal state in Merkle — Damgard construction is $n = 512$ and the length of the output k is either 256 or 512.

Let us define a compression function $h : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$, which is based on 12-rounds LSX-like block cipher $E : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$, where the first argument is a key, in Miyaguchi — Preneel mode:

$$h(y,x) = E(y,x) \oplus x \oplus y.$$

We also define a prefix-free encoding $g : \{0,1\}^* \to (\{0,1\}^n, \{0,1\}^n)^*$, which takes as an input a message X:

$$g(X) = (x_1, \Delta_1) \| (x_2, \Delta_2) \| \ldots \| (x'_l \| 10^{n-1-|x'_l|}, \tilde{\Delta}_l) \| (L, 0) \| (\Sigma, 0),$$

where $L = |X|$, $l = \lfloor L/n \rfloor + 1$, $X = x_1 \| \ldots \| x'_l$, $x_1, \ldots, x_{l-1} \in \{0,1\}^n$, $x'_l \in \{0,1\}^{<n}$, and x'_l is an empty string if L is already divisible by n; $\Delta_i = \mathrm{str}_n(i\, n) \oplus \mathrm{str}_n((i-1)n)$, $\tilde{\Delta}_i = \mathrm{str}_n((i-1)n)$, and $\Sigma = \sum\limits_{i=1}^{l-1} x_i + (x'_l \| 10^{n-1-|x'_l|})$. The encoding pads the message with $10^{n-1-|x'_l|}$, then it splits the message in blocks of length n, computes the counter value for each block and appends two last blocks of the encoding, the bit length L and the checksum Σ, which correspond to the finalizing step of Streebog.

Finally, we define the hash function Streebog on Fig. 1, where IV, $|IV| = 512$, is a predefined constant, different for $k = 256$ and $k = 512$. On Fig. 2 Streebog is depicted schematically.

We will call a sequence of triples $(y_1, x_1, z_1), (y_2, x_2, z_2), \ldots, (y_{l+2}, x_{l+2}, z_{l+2})$, where $z_i = h(y_i, x_i) \oplus y_i \oplus x_i$, which appears during a computation of Streebog on an input X, a *computational chain* for X.

Streebog(X)

$l \leftarrow \lfloor |X|/n \rfloor + 1$
$(x_1, c_1) \| (x_2, c_2) \| \ldots \| (x_l, c_l) \| (x_{l+1}, c_{l+1}) \| (x_{l+2}, c_{l+2}) \leftarrow g(X)$
$y_1 \leftarrow IV$
for $i = 1 \ldots l + 2$ **do** :
  $y_{i+1} \leftarrow h(y_i, x_i) \oplus c_i$
return $\mathrm{msb}_k(y_{l+3})$

Fig. 1. Streebog hash function

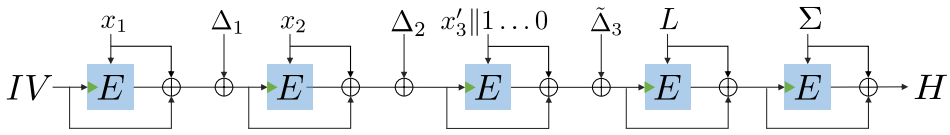


Fig. 2. Streebog computation, $l = 3$

2.2. Indifferentiability

The following strategy is often applied to prove the security of a cryptosystem with some component (or primitive). First, it is proven that the system is secure in case of using idealized primitive. Secondly, we prove that the real primitive is indistinguishable from an idealized one. Informally, two algorithms A and B are computationally indistinguishable if no (efficient) algorithm $\mathcal{D}$ is able to distinguish whether it is interacting with A or B.

We consider two types of the ideal primitives: random oracles and ideal ciphers. A random oracle [1] is an ideal primitive that models a random function. It provides a random output for each new query, identical input queries produce the same answer. An ideal cipher is an ideal primitive that models a random block-cipher $\mathcal{E} : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$, each key $K \in \{0,1\}^\kappa$ defines a random permutation on $\{0,1\}^n$. The ideal cipher provides oracle access to $\mathcal{E}$ and $\mathcal{E}^{-1}$; that is, on query $(+, K, x)$, it answers $c = E(K, x)$, and on query $(-, K, c)$, it answers x such that $c = E(K, x)$.

Obviously, a random oracle (ideal cipher) is easily distinguishable from a hash function (block cipher) if one knows its program and the public parameter. Thus, in [15] the extended notion of indistinguishability — *indifferentiability* — was introduced. It was proven, that if a component A is indifferentiable from B, then the security of any cryptosystem $C(A)$ based on A is not affected when replacing A by B. According to the authors, indifferentiability is the weakest possible property that allows security proofs of the generic type described above. Thus, to prove the security of some cryptosystem using hash function, we may prove its security in the random oracle model, and then prove that hash function is indifferentiable from a random oracle within some underlying assumptions. We assume that the base block cipher is modelled as an ideal cipher.

Let us define formally what the indifferentiability from an ideal primitive means. We give the definition directly for the hash function (based on the ideal cipher) and random oracle.

This definition is a particular case of more general indifferentiability notion introduced in [15].

Definition 1. A hash function H with oracle access to an ideal cipher $\mathcal{E}$ is said to be $(T_{\mathcal{D}}, q_H, q_E, \varepsilon)$-indifferentiable from a random oracle $\mathcal{H}$ if there exists a simulator S such that for any distinguisher $\mathcal{D}$ with binary output it holds that:

$$\left|\Pr\left[\mathcal{D}^{H,\mathcal{E}} \to 1\right] - \Pr\left[\mathcal{D}^{\mathcal{H},S} \to 1\right]\right| < \varepsilon.$$

The simulator has oracle access to $\mathcal{H}$. The distinguisher runs in time at most $T_{\mathcal{D}}$ and makes at most q_H and q_E queries to its oracles.

The indifferentability notion is illustrated in Fig. 3. The distinguisher interacts with two oracles, further we denote them by left and right oracles respectively. In one world, left oracle implements the hash function H (with oracle access to the ideal cipher), while the right oracle directly implements the ideal cipher $\mathcal{E}$. In another world, the left oracle implements the random oracle $\mathcal{H}$ and the right oracle is implemented by the simulator S. The task of the simulator is to model the ideal cipher using the oracle access to $\mathcal{H}$ so that no distinguisher could notice the difference. To achieve that, the output of S must match what the resolver can get from $\mathcal{H}$. Note that the simulator does not have access to the queries of the distinguisher to $\mathcal{H}$.

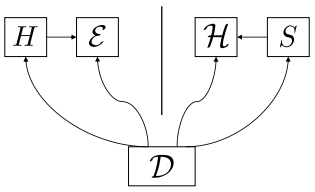


Fig. 3. The indifferentiability of hash function H and random oracle $\mathcal{H}$

3. Streebog indifferentiability

In this section, we present the main result of the paper, which shows that Streebog is indifferentiable from a random oracle in the ideal cipher model for the base block cipher.

First, we discuss the choice of the underlying assumption. Indeed, the straightforward solution is to prove Streebog indifferentiability in assumption that the compression function is a random oracle. Although such proof may be constructed much easier than in the ideal cipher model, we show that the Miyaguchi — Preneel compression function cannot be modeled as a random oracle. Indeed, for this function the following condition always holds:

$$x = E^{-1}(y, h(y, x) \oplus x \oplus y).$$

Thus, the distinguisher can easily identify whether it interacts with the real compression function or the random one by making the query (y, x) to the left oracle and the query $(-, y, h(y, x) \oplus x \oplus y)$ to the right oracle.

We give an indifferentiability theorem for Streebog. The full proof is provided for the Streebog variant with output size $k = 512$. For the shortened Streebog variant argumentation is completely similar. Formally, the only thing which has to be adjusted is the construction of the simulator; we will highlight the difference in the proof. The general structure of the proof and some techniques are adopted from [13, 14].

Theorem 1. The hash function Streebog with $k = 512$ or 256 using a cipher $E : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ is $(t_D, q_H, q_E, \varepsilon)$-indifferentiable from a random oracle in the ideal cipher model for E for any t_D with

$$\varepsilon = \frac{(1+l_m)q}{2^{n-4}} + \frac{(1+n+l_m)q^2}{2^{n-7}},$$

where $q = q_E + q_H(l_m + 2)$ and l_m is the maximum message length (in blocks, including padding) queried by the distinguisher to its left oracle.

Proof. The main goal of the proof is to show that no distinguisher can tell apart two words: in the first one, it has access to the Streebog construction using an ideal cipher as an underlying block cipher and to the ideal cipher itself; in the second one it has access to a random oracle and a simulator. The first step of the proof is to present a simulator for which it would be possible to achieve that goal.

Our simulator for the ideal cipher $\mathcal{E}$ is quite elaborate. On every distinguisher query, it tries to detect whether the distinguisher seeks to compute Streebog for some message itself. If this is the case, it chooses the answer consistently with the random oracle; otherwise, it chooses the answer randomly.

The simulator. Before we proceed with the simulator itself, let us define an auxiliary function $g_0 : \{0,1\}^* \to (\{0,1\}^n, \{0,1\}^n)^*$:

$$g_0(X) = (x_1, \Delta_1) \| (x_2, \Delta_2) \| \dots \| (x'_l \| 10^{n-1-|x'_l|}, \tilde{\Delta}_l) \| (L, 0),$$

where $L = |X|$, $l = \lfloor L/n \rfloor + 1$, $X = x_1 \| \dots \| x'_l$, $x_1, \dots, x_{l-1} \in \{0,1\}^n$, $x'_l \in \{0,1\}^{<n}$, and x'_l is an empty string if L is already divisible by n. Clearly, if $\Sigma = \sum\limits_{i=1}^{l-1} x_i + (x'_l \| 10^{n-1-|x'_l|})$, then $g_0(X) \| (\Sigma, 0) = g(X)$.

The simulator accepts two types of queries: either a forward ideal cipher query $(+, y, x)$, where $x \in \{0,1\}^n$ corresponds to a plaintext and $y \in \{0,1\}^n$ to a cipher key, on which it returns a ciphertext $z \in \{0,1\}^n$; or an inverse query $(-, y, z)$, on which it returns a plaintext x. The simulator maintains a table T, which contains triples $(y, x, z) \in \{0,1\}^n \times \times \{0,1\}^n \times \{0,1\}^n$.

Forward query. When the simulator gets a forward query $(+, y, x)$, it looks up the table T for a triple (y, x, z) for some z. It returns z if such a triple exists. If there is no such triple, the simulator chooses z randomly, puts the triple (y, x, z) in the table, and returns z to the distinguisher. Additionally, in that case the simulator proceeds with the following routine. It looks up the table for a sequence $(y_1, x_1, z_1), \dots, (y_l, x_l, z_l)$ of length $l = \lfloor \mathrm{int}(x)/n \rfloor + 1$ such that:

— there exists X such that $g_0(X) = (x_1, \Delta_1) \| (x_2, \Delta_2) \| \dots \| (x_l, \tilde{\Delta}_l) \| (x, 0)$;
— it is the case that $y_1 = IV$;
— for each $i = 2, \dots, l$, it is the case that $y_i = x_{i-1} \oplus y_{i-1} \oplus z_{i-1} \oplus \Delta_{i-1}$;
— it is the case that $y = x_l \oplus y_l \oplus z_l \oplus \tilde{\Delta}_l$.

If such sequence exists, the simulator forms a pair (y_{l+2}, x_{l+2}) such that $y_{l+2} = x \oplus$ $\oplus\, y \oplus z$ and $x_{l+2} = \sum\limits_{i=1}^{l-1} x_i + x'_l$, where $X = x_1 \| \dots \| x'_l$. It is easy to see that $g(X) =$ $= (x_1, \Delta_1) \| \dots \| (x_l, \tilde{\Delta}_l) \| (x, 0) \| (x_{l+2}, 0)$. The simulator does nothing if there already exists a triple (y_{l+2}, x_{l+2}, z') for some z' in the table T. Otherwise, it computes z' to form a triple (y_{l+2}, x_{l+2}, z'), which will be consistent with a random oracle output on X, in advance. To do

this, it queries the random oracle to get the output $Z = \mathcal{H}(X)$, computes $z' = Z \oplus x_{l+2} \oplus y_{l+2}$ and stores the triple (y_{l+2}, x_{l+2}, z') in the table T [2].

Inverse query. On an inverse query $(-, y, z)$ the simulator acts almost similarly. It looks up the table T for a triple (y, x, z) for some x. It returns x if such triple exists. If there is no such triple, the simulator chooses x randomly, puts the triple (y, x, z) in the table, and returns x to the distinguisher. In this case, it proceeds with completely the same routine as described above.

We will denote the number of entries in the table T by q. It is clear that $q_E \leqslant q \leqslant 2q_E$, since for each adversarial query to S, at most one additional record can be added to the table T besides the answer to the query itself.

Proof of indifferentiability. Due to the definition of indifferentiability, if the following inequality holds for every distinguisher $\mathcal{D}$:

$$\left|\Pr\left[\mathcal{D}^{H,\mathcal{E}} \to 1\right] - \Pr\left[\mathcal{D}^{\mathcal{H},S} \to 1\right]\right| \leqslant \varepsilon,$$

then the theorem follows. So we have to prove that no discriminator $\mathcal{D}$ can distinguish between these two words except with probability ε. We will do that using the game hopping technique, starting in the world with the random oracle $\mathcal{H}$ and the simulator S and moving through the sequence of indistinguishable games to the world with the Streebog construction and the ideal cipher $\mathcal{E}$.

Game 1 → Game 2. The Game 1 is the starting point, where $\mathcal{D}$ has access to the random oracle $\mathcal{H}$ and the simulator S. In the Game 2 (Fig. 4), we give $\mathcal{D}$ access to the relay algorithm R_0 instead of direct access to $\mathcal{H}$. R_0, in its turn, has access to the random oracle and on distinguisher's queries simply answers with $\mathcal{H}(X)$. Let us denote by G_i the events that $\mathcal{D}$ returns 1 in Game i. It is clear that $\Pr[G_1] = \Pr[G_2]$.

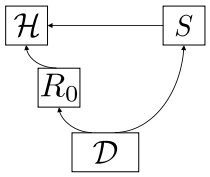


Fig. 4. Game 2

Game 2 → Game 3. In the Game 3, we modify the simulator S by introducing failure conditions. The simulator explicitly fails (i.e., returns an error symbol $\perp$) when answering to the distinguisher's query, if it computes the response satisfying one of the following failure conditions. Let S_0 denote the modified simulator.

We introduce two types of failure conditions. Each condition captures different relations between the simulator's answers that could be exploited by the distinguisher. By failing, the simulator "gives" the distinguisher an immediate win. Our longterm goal is to show that, unless the failure happens, distinguisher cannot tell apart Game 2 form the ideal cipher world. The simulator S_0 chooses response to the forward or inverse query similarly to the simulator S and then checks the resulting triple (y, x, z) for the conditions defined below. For each type of conditions we also provide a brief motivation behind it, i.e., how the distinguisher can exploit corresponding situations to tell apart two worlds.

[2] In the case of $k = 256$, the simulator first pads Z with 256 randomly chosen bits and then computes $z' = Z \oplus x_{l+2} \oplus y_{l+2}$.

Conditions of type 1. Conditions of type 1 are checked if the answer to the query was chosen randomly or the discriminator was first returned with a value selected by the simulator as corresponding to a random oracle and previously tabulated:

1) *Condition* B_{11}: $x \oplus y \oplus z = IV$.
2) *Condition* B_{12}: there exists $l \in \{1, \ldots, l_m\}$ such that $x \oplus y \oplus z \oplus \tilde{\Delta}_l = IV$.
3) *Condition* B_{13}: there exist a triple $(y', x', z') \in T$ and $i \in \{1, \ldots, l_m - 1\}$ such that $x \oplus y \oplus z = x' \oplus y' \oplus z' \oplus \Delta_i$. Note that $\left|\{\Delta_i : i \in \{1, 2, \ldots\}\}\right| \leqslant n$.
4) *Condition* B_{14}: there exist a triple $(y', x', z') \in T$ and $l \in \{1, \ldots, l_m\}$ such that $x \oplus y \oplus z = x' \oplus y' \oplus z' \oplus \tilde{\Delta}_l$.
5) *Condition* B_{15}: there exists a triple $(y', x', z') \in T$ such that $x \oplus y \oplus z = x' \oplus y' \oplus z'$.

The type 1 conditions correspond to the situation when the internal states of two Streebog computational chains for different messages collide. The distinguisher can exploit that situation in a number of ways, for example, it can force these two chains to end with the same block, which will give the same result for two different messages. From this, the distinguisher can easily distinguish between the two worlds by querying its left oracle with these messages. Other bad situations which correspond to this type of conditions are analyzed in the proof of Lemma 1.

Conditions of type 2. Conditions of type 2 are checked if only the answer to the query was chosen by the simulator randomly (i.e., the answer was not taken from the table):

1) *Condition* B_{21}: there exists a triple $(y', x', z') \in T$ such that $x \oplus y \oplus z = y'$.
2) *Condition* B_{22}: there exist a triple $(y', x', z') \in T$ and $i \in \{1, \ldots, l_m - 1\}$ such that $x \oplus y \oplus z = y' \oplus \Delta_i$.
3) *Condition* B_{23}: there exist a triple $(y', x', z') \in T$ and $l \in \{1, \ldots, l_m\}$ such that $x \oplus y \oplus z = y' \oplus \tilde{\Delta}_l$.

The conditions of type 2 correspond to a situation when some block in the computational chain is queried sometime after the query corresponding to the next block was made. In this case, this query can be made even after the query for the last block in the chain was. The distinguisher can then easily tell two worlds apart, because the simulator did not choose the answer to the last query to be consistent with the random oracle. Notice that conditions of that type are only checked when the simulator chooses the answer randomly itself. Otherwise, the distinguisher can easily force the failure event using the random oracle, for example, it can choose an arbitrary X, query the random oracle for $Z = \mathcal{H}(X)$, then query the right oracle with $(+, Z, x)$ for some x, and finally compute the Streebog construction for X using its right oracle. The simulator would then fail due to condition B_{21} when answering for the last block of the computational chain. However, such a situation will not help the distinguisher, since this is in a sense an extension of the computational message chain with new blocks, which will not lead to another valid computational chain due to our prefix-free encoding g. Bad situations which correspond to this type of conditions are analyzed in the proof of Lemma 2.

The probability of the event that the simulator fails due to one of the failure conditions is estimated as follows:

$$\Pr[S_0 \text{ fails}] \leqslant \frac{(1 + l_m)q_E}{2^{n-1}} + \frac{(1 + n + l_m)q_E^2}{2^{n-4}}.$$

That bound directly follows from Lemma 3 with $q_S = q_E$, which is given in Appendix Appendix A. The proof of this statement is rather technical and is also provided in Appendix Appendix A.

Since Game 2 and Game 3 are different only in situations, where the simulator S_0 fails, it is clear that

$$\big|\Pr[G_2] - \Pr[G_3]\big| \leqslant \Pr[S_0 \text{ fails}] \leqslant \frac{(1+l_m)q_E}{2^{n-1}} + \frac{(1+n+l_m)q_E^2}{2^{n-4}}.$$

Now, before we proceed to the next game, our aim is to show that unless the simulator fails, its outputs are always consistent with random oracle outputs, i.e., it does not matter if the distinguisher is computing the Streebog construction with its right oracle (maybe in some unsual way) or queries the random oracle, the results would be the same. To do this, we prove two lemmas, where Lemma 2 formalizes the outlined goal.

The first lemma states that in the table T there are no two sequences of triples corresponding to computational chains with two different inputs such that the last block of one chain is the first, middle, or last block of another, unless S_0 fails.

Lemma 1. If the simulator S_0 does not fail, then in the table T there are no two different sequences of triples (y_1, x_1, z_1), …, $(y_{l+2}, x_{l+2}, z_{l+2})$ and (y'_1, x'_1, z'_1), …, $(y'_{p+2}, x'_{p+2}, z'_{p+2})$, where $l, p \leqslant l_m$, such that the following conditions hold:

— there exist X and X' such that $g(X) = (x_1, \Delta_1)\| \ldots \|(x_{l+1}, 0)\|(x_{l+2}, 0)$ and $g(X') = (x'_1, \Delta_1)\| \ldots \|(x'_{p+1}, 0)\|(x'_{p+2}, 0)$;
— it is the case that $y_1 = y'_1 = IV$;
— for each $i = 2, \ldots, l$ and $j = 2, \ldots, p$, it is the case that $y_i = x_{i-1} \oplus y_{i-1} \oplus z_{i-1} \oplus \Delta_{i-1}$ and $y'_j = x'_{j-1} \oplus y'_{j-1} \oplus z'_{j-1} \oplus \Delta_{j-1}$;
— it is the case that $y_{l+1} = x_l \oplus y_l \oplus z_l \oplus \tilde{\Delta}_l$ and $y'_{p+1} = x'_p \oplus y'_p \oplus z'_p \oplus \tilde{\Delta}_l$;
— it is the case that $y_{l+2} = x_{l+1} \oplus y_{l+1} \oplus z_{l+1}$ and $y'_{p+2} = x'_{p+1} \oplus y'_{p+1} \oplus z'_{p+1}$;
— there exists $s \in \{1, \ldots, l+2\}$ such that $(y_s, x_s, z_s) = (y'_{p+2}, x'_{p+2}, z'_{p+2})$.

Proof. Let us suppose that there exist two sequences $(y_1, x_1, z_1), \ldots, (y_{l+2}, x_{l+2}, z_{l+2})$ and $(y'_1, x'_1, z'_1), \ldots, (y'_{p+2}, x'_{p+2}, z'_{p+2})$ in the table T, which satisfy conditions of the lemma. Then there exists the maximum $r \in \{1, \ldots, \min(s, p+2)\}$ such that

$$(y_{s-i}, x_{s-i}, z_{s-i}) = (y'_{p-2-i}, x'_{p-2-i}, z'_{p-2-i}), \quad i = 0, \ldots, r-1.$$

In other words, r is the length of the subsequence of equal triples ending with $(y_s, x_s, z_s) = (y'_{p+2}, x'_{p+2}, z'_{p+2})$. We will now consider several cases depending on values of r and l. Notice that $r \leqslant s \leqslant l+2$.

T h e c a s e $r = 1$. Since it is true that $(y_s, x_s, z_s) = (y'_{p+2}, x'_{p+2}, z'_{p+2})$, we can deduce that one of the following equalities has to hold:

1) if $s = 1$, then $y_s = IV$. Hence, $x'_{p+1} \oplus y'_{p+1} \oplus z'_{p+1} = y'_{p+2} = y_s = IV$;
2) if $s \in \{2, \ldots, l\}$, then $y_s = x_{s-1} \oplus y_{s-1} \oplus z_{s-1} \oplus \Delta_{s-1}$. Hence, $x'_{p+1} \oplus y'_{p+1} \oplus z'_{p+1} = x_{s-1} \oplus y_{s-1} \oplus z_{s-1} \oplus \Delta_{s-1}$;
3) if $s = l+1$, then $y_s = x_{s-1} \oplus y_{s-1} \oplus z_{s-1} \oplus \tilde{\Delta}_{s-1}$. Hence, $x'_{p+1} \oplus y'_{p+1} \oplus z'_{p+1} = x_{s-1} \oplus y_{s-1} \oplus z_{s-1} \oplus \tilde{\Delta}_l$;
4) if $s = l+2$, then $y_s = x_{s-1} \oplus y_{s-1} \oplus z_{s-1}$. Hence, $x'_{p+1} \oplus y'_{p+1} \oplus z'_{p+1} = x_{s-1} \oplus y_{s-1} \oplus z_{s-1}$.

However, it is easy to see that the above equalities correspond to the failure conditions $B_{11}, B_{13}, B_{14}, B_{15}$, respectively. Therefore, one of these failure conditions would have been triggered if a forward or inverse query which corresponds to the triple $(y_{s-1}, x_{s-1}, z_{s-1})$ or $(y'_{p+1}, x'_{p+1}, z'_{p+1})$ (depending on which of them was made later) was made.

T h e c a s e $r \geqslant 2$, $l > 1$ and $r = 3$, $l = 1$. Since $r \geqslant 2$, it is easy to see that the same inequality holds for s. Thereof, from $y'_{p+2} = y_s$ and the lemma statement we have

that $x'_{p+1} \oplus y'_{p+1} \oplus z'_{p+1} \oplus 0 = x_{s-1} \oplus y_{s-1} \oplus z_{s-1} \oplus c$ for some $c \in \{\Delta_1, \ldots, \Delta_{l-1}, \tilde{\Delta}_l, 0\}$. However, since from $r \geqslant 2$ we have $(y_{s-1}, x_{s-1}, z_{s-1}) = (y'_{p+1}, x'_{p+1}, z'_{p+1})$, the constant c has to be equal to 0. It is also easy to see that none of the values $\{\Delta_1, \ldots, \Delta_{l-1}, \tilde{\Delta}_l\}$ is equal to 0 when $l > 1$. Hence, due to the encoding g, it is only possible that the triple (y_s, x_s, z_s) is the last one in the sequence and $s = l + 2$.

Thereof, $x_{l+1} = x'_{p+1}$, where, due to the definition of g, x_{l+1} and x'_{p+1} are equal to $|X|$ and $|X'|$ correspondingly. Consequently, since by definition $l = \lfloor |X|/n \rfloor + 1$ and $p = \lfloor |X'|/n \rfloor + 1$, we have that $p = l$.

Finally, consider triples $(y_{l+2-r}, x_{l+2-r}, z_{l+2-r}) \neq (y'_{l+2-r}, x'_{l+2-r}, z'_{l+2-r})$. Notice that $r < l + 2$ or else the considered sequences are equal (that excludes the $r = 3$, $l = 1$ case at all). Since $y_{l+2-r+1} = y'_{l+2-r+1}$, the following equality has to hold:

$$y_{l+2-r} \oplus x_{l+2-r} \oplus z_{l+2-r} \oplus c = y'_{l+2-r} \oplus x'_{l+2-r} \oplus z'_{l+2-r} \oplus c,$$

where c is equal either to Δ_{l+2-r} or $\tilde{\Delta}_{l+2-r}$. However, it is easy to see that in either way the equality matches the failure condition B_{15}. Therefore, it would have been triggered if a forward or inverse query which corresponds to the triple $(y_{l+2-r}, x_{l+2-r}, z_{l+2-r})$ or $(y'_{l+2-r}, x'_{l+2-r}, z'_{l+2-r})$ (depending on which of them was made later) was made.

The case $r = 2$ and $l = 1$. We have that $\tilde{\Delta}_l$ is equal to 0, hence two situations are possible. The first one is when $s = 3$, the reasoning here is exactly the same as in the last case, since equal triples are the last two triples in the sequences.

The second one is when $s = 2$. From that and since $r = 2$, we have that $(y_1, x_1, z_1) = (y'_{p+1}, x'_{p+1}, z'_{p+1})$. From the lemma statement, $y_1 = IV$ and $y'_{p+1} = x'_p \oplus y'_p \oplus z'_p \oplus \tilde{\Delta}_p$, thereof the following equality has to hold:

$$x'_p \oplus y'_p \oplus z'_p \oplus \tilde{\Delta}_p = IV.$$

However, it is easy to see that the equality matches the failure condition B_{12}. Hence, it would have been triggered, when a forward or inverse query which corresponds to the triple (y'_p, x'_p, z'_p) was made.

We have considered all possible pairs (r, l). Hence, we can conclude that no such sequences can exist if the simulator S_0 does not fail. ■

Now we prove that the outputs of the simulator are consistent with the random oracle unless it fails. To do this, we show that if the distinguisher at some point computes the Streebog construction itself, it has to do that block-by-block, with the last triple of the computational chain being consistent with the random oracle.

Lemma 2. Consider any sequence of triples (y_1, x_1, z_1), …, $(y_{l+2}, x_{l+2}, z_{l+2})$, where $l \leqslant l_m$, from the table T such that the following conditions hold:

— there exists X such that $g(X) = (x_1, \Delta_1) \| \ldots \| (x_{l+1}, 0) \| (x_{l+2}, 0)$;
— it is the case that $y_1 = IV$;
— for each $i = 2, \ldots, l$, it is the case that $y_i = x_{i-1} \oplus y_{i-1} \oplus z_{i-1} \oplus \Delta_{i-1}$;
— it is the case that $y_{l+1} = x_l \oplus y_l \oplus z_l \oplus \tilde{\Delta}_l$;
— it is the case that $y_{l+2} = x_{l+1} \oplus y_{l+1} \oplus z_{l+1}$.

If the simulator S_0 does not fail, then it must be the case the triples (y_1, x_1, z_1), …, $(y_{l+1}, x_{l+1}, z_{l+1})$ were put in the table T exactly in that order and answers to the corresponding queries were chosen randomly by the simulator. It is also necessary that the triple $(y_{l+2}, x_{l+2}, z_{l+2})$ was put in the table simultaneously with the triple $(y_{l+1}, x_{l+1}, z_{l+1})$, chosen to be consistent with the random oracle output $\mathcal{H}(X)$.

Proof. Let us suppose that there exists $i \in \{1, \ldots, l+1\}$ such that the triple (y_i, x_i, z_i) was put in the table as a result of the corresponding forward or inverse query, when the triple $(y_{i+1}, x_{i+1}, z_{i+1})$ already existed in the table T. For that pair of triples the following equality holds:

$$y_i \oplus x_i \oplus z_i \oplus c = y_{i+1},$$

where c is one of the values $\{\Delta_i, \tilde{\Delta}_i, 0\}$, depending on the value of i. From Lemma 1 it follows that the triple (y_i, x_i, z_i) could not be the last in the computational chain of some message $X' \neq X$. In other words, the answer to the corresponding query was not chosen to be consistent with the random oracle, but was chosen randomly by the simulator. Hence, on the query corresponding to the triple (y_i, x_i, z_i) one of the failure conditions of type 2 would have been triggered.

Thereby, when the query corresponding to the triple $(y_{l+1}, x_{l+1}, z_{l+1})$ is made, triples $(y_1, x_1, z_1), \ldots, (y_l, x_l, z_l)$ already exist in the table and the triple $(y_{l+2}, x_{l+2}, z_{l+2})$ does not. These triples satisfy the conditions of the simulator's routine and it has to choose the triple $(y_{l+2}, x_{l+2}, z_{l+2})$ to be consistent with the random oracle and put it in the table with the triple $(y_{l+1}, x_{l+1}, z_{l+1})$. ■

Game 3 → Game 4. In Game 4 (Fig. 5), we modify the relay algorithm R_0. Let R_1 denote the modified algorithm. It does not have access to the random oracle. On a distinguisher query X, R_1 applies the Streebog construction to X using the simulator for the block cipher E. Notice that now at most $q_E + q_H(l_m + 2)$ queries are made to S_0.

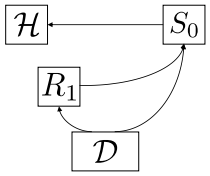


Fig. 5. Game 4

Let $fail3$ and $fail4$ denote the events when the simulator fails in the corresponding game. From Lemma 2 it follows that, unless the simulator does not fail, answers of the modified relay algorithm R_1 are exactly the outputs of the random oracle on corresponding messages, since the simulator's answers are consistent with the random oracle. Hence, if the simulator does not fail in either world, the view of the distinguisher remains unchanged from Game 3 to Game 4:

$$\Pr\left[G_3 \mid \overline{fail_3}\right] = \Pr\left[G_4 \mid \overline{fail_4}\right].$$

Probability of the event $fail_3$ was estimated earlier in the transition from Game 2 to Game 3. Probability of the event $fail_4$ is estimated from Lemma 3, where $q_S = q_E + q_H(l_m + 2)$. Thus, we have:

$$\begin{gathered}
\left|\Pr[G_3] - \Pr[G_4]\right| = \left|\Pr\left[G_3 \mid \overline{fail_3}\right]\Pr\left[\overline{fail_3}\right] + \Pr[G_3 \mid fail_3]\Pr[fail_3] - \right. \\
\left. - \Pr\left[G_4 \mid \overline{fail_4}\right]\Pr\left[\overline{fail_4}\right] - \Pr[G_4 \mid fail_4]\Pr[fail_4]\right| \leqslant \Pr\left[G_3 \mid \overline{fail_3}\right] \cdot \left|\Pr\left[\overline{fail_3}\right] - \right. \\
\left. - \Pr\left[\overline{fail_4}\right]\right| + \left|\Pr[G_3 \mid fail_3]\Pr[fail_3] - \Pr[G_4 \mid fail_4]\Pr[fail_4]\right| \leqslant \\
\leqslant \left|\Pr[fail_4] - \Pr[fail_3]\right| + \left|\Pr[G_3 \mid fail_3]\Pr[fail_3] - \Pr[G_4 \mid fail_4]\Pr[fail_4]\right| \leqslant \\
\leqslant \max(\Pr[fail_3], \Pr[fail_4]) + \max(1 \cdot \Pr[fail_3] - 0 \cdot \Pr[fail_4], 0 \cdot \Pr[fail_3] + 1 \cdot \Pr[fail_4]) \leqslant
\end{gathered}$$

$$\leqslant 2\max\left(\Pr[fail_3], \Pr[fail_4]\right) \leqslant 2\left(\frac{(1+l_m)(q_E+q_H(l_m+2))}{2^{n-1}} + \frac{(1+n+l_m)(q_E+q_H(l_m+2))^2}{2^{n-4}}\right).$$

Game 4 → Game 5. In Game 5 (Fig. 6) we modify the simulator. Let S_1 denote the modified simulator. It does not consult the random oracle when answering the query, it still maintains a table T of triples (x, y, z). On a forward query $(+, y, x)$, it searches the table T for a triple (y, x, z) for some z. It returns z if such triple exists. If there is no such triple, the simulator chooses z randomly, puts the triple (y, x, z) in the table and returns z to the distinguisher. It acts similarly to answer the inverse query $(-, y, z)$, but chooses a random x, if there is no corresponding triple.

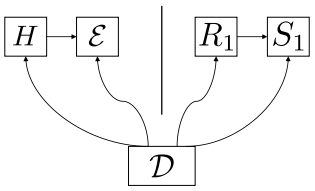


Fig. 6. The ideal cipher world and Game 5

The simulator responses in both games are identical except for the S_0 failure condition. This is true because even when S_0 chooses the answer using the random oracle, all its answers look uniformly distributed to the distinguisher as it does not have a direct access to the random oracle in Game 4. Hence, the view of the distinguisher is identical in both games if the simulator does not fail in Game 4, and if in Game 5 the simulator does not give a response, which would have led to failure in Game 4. The probabilities of these events are equal, since the number of queries to the simulators in both games is the same, and the distribution of the responses of the simulators is identical. Let us denote the event "S_1 should have failed" by $fail_5$. Hence, the following inequality holds:

$$\begin{aligned}
\left|\Pr[G_4] - \Pr[G_5]\right| &= \left|\Pr\left[G_4 \mid \overline{fail_4}\right]\Pr\left[\overline{fail_4}\right] + \Pr[G_4 \mid fail_4]\Pr[fail_4] - \right. \\
&\left. - \Pr\left[G_5 \mid \overline{fail_5}\right]\Pr\left[\overline{fail_5}\right] - \Pr[G_5 \mid fail_5]\Pr[fail_5]\right| = \\
&= \left|\Pr[G_4 \mid fail_4]\Pr[fail_4] - \Pr[G_5 \mid fail_5]\Pr[fail_5]\right| \leqslant \\
\leqslant \Pr[G_4 \mid fail_4]\Pr[fail_4] + \Pr[G_5 \mid fail_5]\Pr[fail_5] &\leqslant \Pr[fail_4] + \Pr[fail_5] = \\
= 2\Pr[fail_4] &\leqslant 2\left(\frac{(1+l_m)(q_E+q_H(l_m+2))}{2^{n-1}} + \frac{(1+n+l_m)(q_E+q_H(l_m+2))^2}{2^{n-4}}\right).
\end{aligned}$$

Game 5 → Game 6. In the final game we replace the simulator S_1 with the ideal cipher $\mathcal{E}$. Since the relay algorithm R_1 is the Streebog construction and now it uses the ideal cipher for E, the Game 6 is exactly the ideal cipher model.

We now have to show that the view of the distinguisher remains almost unchanged. The outputs of the ideal cipher and the simulator S_1 have different distributions: the ideal cipher is a permutation for each key and S_1 chooses its answers randomly. Hence, the distinguisher can tell apart two games only if forward/inverse outputs of the simulator collide for the same key. The probability of that event is at most the birthday bound through all queries. Thus, we have

$$\left|\Pr[G_5] - \Pr[G_6]\right| \leqslant \frac{(q_E + q_H(l_m+2))^2}{2^n}.$$

Finally, combining all the transitions and since Game 6 is exactly the ideal cipher model, we can deduce that

$$\left|\Pr\left[\mathcal{D}^{H,\mathcal{E}}\to 1\right]-\Pr\left[\mathcal{D}^{\mathcal{H},S}\to 1\right]\right|\leqslant\frac{(1+l_m)q_E}{2^{n-1}}+\frac{(1+n+l_m)q_E^2}{2^{n-4}}+$$
$$+4\left(\frac{(1+l_m)(q_E+q_H(l_m+2))}{2^{n-1}}+\frac{(1+n+l_m)(q_E+q_H(l_m+2))^2}{2^{n-4}}\right)+\frac{(q_E+q_H(l_m+2))^2}{2^n}.$$

The statement of Theorem 1 hence follows. ■

4. Conclusion

In the paper, we prove that the Streebog hash function is indifferentiable from a random oracle under the ideal cipher assumption for the underlying block cipher. From a practical point of view, under this assumption Streebog can be considered as a random oracle as long as computational power of the adversary remains much less than $2^{n/2}$ operations. However, it is still an open problem to determine if it is possible to prove indifferentiability of Streebog and other hash functions under idealized assumptions for even lower-level objects than a block cipher.

Acknowledgement

The authors are very grateful to Vitaly Kiryukhin for useful discussions and valuable comments, which greatly contributed to the quality of the paper, as well as for verifying the results.

REFERENCES

1. *Bellare M. and Rogaway P.* Random oracles are practical: A paradigm for designing efficient protocols. Proc. 1st ACM Conf. CCS'93, N.Y., ACM, 1993, pp. 62–73.
2. *Rescorla E.* The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, August 2018, `https://datatracker.ietf.org/doc/html/rfc8446`.
3. *Kaufman C., Hoffman P., Nir Y., et al.* Internet Key Exchange Protocol Version 2 (IKEv2). RFC 7296, October 2014, `https://datatracker.ietf.org/doc/html/rfc7296`.
4. *Schnorr C. P.* Efficient identification and signatures for smart cards. LNCS, 1990, vol. 435, pp. 239–252.
5. *Pointcheval D. and Stern J.* Security proofs for signature schemes. LNCS, 1996, vol. 1070, pp. 387–398.
6. *Smyshlyaev S., Alekseev E., Griboedova E., et al.* GOST Cipher Suites for Transport Layer Security (TLS) Protocol Version 1.3. RFC 9367, February 2023, `https://datatracker.ietf.org/doc/rfc9367`.
7. *Smyslov V.* Using GOST Ciphers in the Encapsulating Security Payload (ESP) and Internet Key Exchange Version 2 (IKEv2) Protocols. RFC 9227, March 2022, `https://datatracker.ietf.org/doc/rfc9227`.
8. *Smyshlyaev S., Alekseev E., Oshkin I., and Popov V.* The Security Evaluated Standardized Password-Authenticated Key Exchange (SESPAKE) Protocol. RFC 8133, March 2017, `https://datatracker.ietf.org/doc/html/rfc8133`.
9. *Alekseev E. K. and Smyshlyaev S. V.* O bezopasnosti protokola SESPAKE [On security of the SESPAKE protocol]. Prikladnaya Diskretnaya Matematika, 2020, no. 50, pp. 5–41. (in Russian)
10. *Akhmetzyanova L. R., Alekseev E. K., Babueva A. A., and Smyshlyaev S. V.* On methods of shortening ElGamal-type signatures. Mat. Vopr. Kriptogr., 2021, vol. 12, no. 2, pp. 75–91.

11. *Tessaro S. and Zhu C.* Short pairing-free blind signatures with exponential security. LNCS, 2022, vol. 13276, pp. 782–811.
12. *Vysotskaya V. V. and Chizhov I. V.* The security of the code-based signature scheme based on the Stern identification protocol. Prikladnaya Diskretnaya Matematika, 2022, no. 57, pp. 67–90.
13. *Coron J. S., Dodis Y., Malinaud C., and Puniya P.* Merkle-Damgård revisited: How to construct a hash function. LNCS, 2005, vol. 3621, pp. 430–448.
14. *Coron J. S., Dodis Y., Malinaud C., and Puniya P.* Merkle-Damgård revisited: How to construct a hash function. Full version, 2005. `https://cs.nyu.edu/~dodis/ps/merkle.pdf`.
15. *Maurer U. M., Renner R., and Holenstein C.* Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. LNCS, 2004, vol. 2951, pp. 21–39.
16. GOST R 34.11-2012. Informatsionnaya tekhnologiya. Kriptograficheskaya zashchita informatsii. Funktsiya kheshirovaniya [Information Technology. Cryptographic Data Security. Hash Function]. Moscow, Standartinform Publ., 2012. (in Russian)
17. *Smyshlyaev S. V., Shishkin V. A., Marshalko G. B., et al.* Obzor rezul'tatov analiza khesh-funktsii GOST R 34.11-2012 [Overview of hash-function GOST R 34.11-2012 cryptoanalysis]. Problemy Informatsionnoy Bezopasnosti. Komp'yuternye Sistemy, 2015, vol. 4, pp. 147–153. (in Russian)
18. *Kiryukhin V.* Keyed Streebog is a Secure PRF and MAC. 2022, Cryptology ePrint Archive, 2022. `https://eprint.iacr.org/2022/972`.
19. *Ristenpart T., Shacham H., and Shrimpton T.* Careful with composition: Limitations of the indifferentiability framework. LNCS, 2011, vol. 6632, pp. 487–506.
20. *Guo J., Jean J., Leurent G., et al.* The usage of counter revisited: Second-preimage attack on new Russian standardized hash function. LNCS, 2014, vol. 8781, pp. 195–211.

Appendix A. Probability of the simulator's failure event

Lemma 3. Let S_0 be a simulator defined in the proof of Theorem 1. Then the probability of the event that the simulator S_0 explicitly fails due to one of the failure conditions $B_{11}, \ldots, B_{23}$, defined in the proof of Theorem 1, satisfies the following bound:

$$\Pr[S_0 \text{ fails}] = \frac{(1+l_m)q_S}{2^{n-1}} + \frac{(1+n+l_m)q_S^2}{2^{n-4}},$$

where q_S is a number of queries made to the simulator.

Proof. Let us denote by q the maximum number of entries in the table T, $q_S \leqslant q \leqslant 2q_S$. To estimate the desired probability, we consider each failure condition and bound the probability that there exists a query to the simulator satisfying the condition. Let us begin with conditions of type 1.

— *Condition* B_{11}. It is the probability that one of at most q random n-bit strings (where the randomness is due to either the simulator's random choice or the random oracle output) is equal to fixed IV. Hence,

$$\Pr[\exists \textit{ query} \text{ satisfying } B_{11}] \leqslant \frac{q}{2^n}.$$

— *Condition* B_{12}. It is the probability that one of at most q random n-bit strings is equal to one of l_m strings $IV \oplus \tilde{\Delta}_l$, $l \in \{1, \ldots, l_m\}$:

$$\Pr[\exists \textit{ query} \text{ satisfying } B_{12}] \leqslant \frac{l_m\, q}{2^n}.$$

— *Condition* B_{13}. To estimate the probability of this event, we will consider three separate situations.
The first one is that there exists a query satisfying the condition, the answer to which was chosen by the simulator randomly. The probability of that situation is the probability that one of at most $q_S \leqslant q$ random n-bit strings is equal to one of less than nq strings $x' \oplus y' \oplus z' \oplus \Delta_i$, $(y', x', z') \in T$, $i \in \{1, \ldots, l_m - 1\}$ (recall that $|\{\Delta_i : i \in \{1, 2, \ldots\}| \leqslant n$). Hence,

$$\Pr[\exists \textit{ query} \text{ satisfying } B_{13} \text{ and Situation 1}] \leqslant \frac{n\,q^2}{2^n}.$$

The second one is that there exists a query satisfying the condition, the answer to which was chosen by the simulator to be consistent with the random oracle (then $x \oplus y \oplus z$ is exactly the random oracle output), and the triple $(y', x', z') \in T$ was constructed independently from the random oracle (the answer to the corresponding query was chosen randomly by the simulator itself). The probability of that situation is the probability that one of at most $q_S \leqslant q$ random oracle n-bit outputs is equal to one of less than nq strings $x' \oplus y' \oplus z' \oplus \Delta_i$, $(y', x', z') \in T$, $i \in \{1, \ldots, l_m - 1\}$. Hence,

$$\Pr[\exists \textit{ query} \text{ satisfying } B_{13} \text{ and Situation 2}] \leqslant \frac{n\,q^2}{2^n}.$$

The third one is that there exists a query satisfying the condition, the answer to which was chosen by the simulator to be consistent with the random oracle, and the triple $(y', x', z') \in T$ was also constructed to be consistent with the random oracle. Then both $x \oplus y \oplus z$ and $x' \oplus y' \oplus z'$ are the random oracle outputs on different messages X and X' (they are different since both triples have to be the last blocks of some computational chains and there is only one computational chain for every X). The probability of that situation is the probability that two random oracle outputs Z and Z' from at most $q_S \leqslant q$ satisfy any of the less than n equalities $Z \oplus Z' = \Delta_i$. Hence,

$$\Pr[\exists \textit{ query} \text{ satisfying } B_{13} \text{ and Situation 3}] \leqslant \frac{n\,q^2}{2^n}.$$

Finally, it is easy to see that

$$\Pr[\exists \textit{ query} \text{ satisfying } B_{13}] \leqslant \Pr[\exists \textit{ query} \text{ satisfying } B_{13} \text{ and Situation 1}] +$$
$$+ \Pr[\exists \textit{ query} \text{ satisfying } B_{13} \text{ and Situation 2}] + \Pr[\exists \textit{ query} \text{ satisfying } B_{13} \text{ and Situation 3}].$$

Hence,

$$\Pr[\exists \textit{ query} \text{ satisfying } B_{13}] \leqslant 3\,\frac{n\,q^2}{2^n}.$$

— *Condition* B_{14}. The probability of that event is estimated similarly to the previous one with the difference that $|\{\tilde{\Delta}_l : l = 1, \ldots, l_m\}| = l_m$. Hence,

$$\Pr[\exists \textit{ query} \text{ satisfying } B_{14}] \leqslant 3\,\frac{l_m\,q^2}{2^n}.$$

— *Condition* B_{15}. The probability of that event is estimated similarly to the previous two:

$$\Pr[\exists \textit{ query} \text{ satisfying } B_{15}] \leqslant 3\,\frac{q^2}{2^n}.$$

We proceed with conditions of type 2:

— *Condition* B_{21}. It is the probability that one of at most $q_S \leqslant q$ random n-bit strings, where the randomness is due to either the simulator's random choice or the random oracle output and is independent of the distinguisher's random tape, is equal to one of q strings y', $(y', x', z') \in T$, where all y' are chosen by the distinguisher. Hence,

$$\Pr[\exists\ query \text{ satisfying } B_{21}] \leqslant \frac{q^2}{2^n}.$$

— *Condition* B_{22}. The probability of that event is estimated similarly to the previous one, with the only difference that there are at most nq different strings $y' \oplus \Delta_i$, $(y', x', z') \in T$, $i \in \{1, \ldots, l_m - 1\}$. Hence,

$$\Pr[\exists\ query \text{ satisfying } B_{22}] \leqslant \frac{n\,q^2}{2^n}.$$

— *Condition* B_{23}. The probability of that event is estimated similarly to the previous ones, with the difference that there are at most $l_m\,q$ different strings $y' \oplus \tilde{\Delta}_l$, $(y', x', z') \in T$, $l \in \{1, \ldots, l_m\}$. Hence,

$$\Pr[\exists\ query \text{ satisfying } B_{23}] \leqslant \frac{l_m\,q^2}{2^n}.$$

Finally, we estimate the probability of the event that the simulator fails:

$$\Pr[S_0 \text{ fails}] \leqslant \Pr[\exists\ query \text{ satisfying some bad condition}] \leqslant$$
$$\leqslant \frac{(1 + l_m)q}{2^n} + \frac{(4 + 4n + 4l_m)q^2}{2^n} = \frac{(1 + l_m)q_S}{2^{n-1}} + \frac{(1 + n + l_m)q_S^2}{2^{n-4}},$$

where the last inequality is due to $q \leqslant 2q_S$. ■

МАТЕМАТИЧЕСКИЕ ОСНОВЫ НАДЁЖНОСТИ ВЫЧИСЛИТЕЛЬНЫХ И УПРАВЛЯЮЩИХ СИСТЕМ

УДК 681.32 DOI 10.17223/20710410/64/4

GRAPH METHODS FOR RECOGNITION OF CMOS GATES IN TRANSISTOR-LEVEL CIRCUITS

D.I. Cheremisinov, L.D. Cheremisinova

The United Institute of Informatics Problems of the National Academy of Sciences of Belarus, Minsk, Belarus

E-mail: {cher, cld}@newman.bas-net.by

The paper focuses on the decompilation of a flat transistor circuit in SPICE format into a hierarchical network of logic gates. The problem arises in VLSI layout verification as well as in reverse engineering transistor circuit to redesign integrated circuit and to detect untrusted attachments. The most general case is considered when the extraction of functional level structure from transistor-level circuit is performed without any predetermined cell library. Graph methods for solving some key tasks in this area are proposed. The presented graph methods have been implemented in C++ as a part of a decompilation program, which has been tested using practical transistor-level circuits.

Keywords: *CMOS transistor circuit, subcircuit extraction, logic gate recognition, graph isomorphism, SPICE format.*

ГРАФОВЫЕ МЕТОДЫ РАСПОЗНАВАНИЯ КМОП-ВЕНТИЛЕЙ В СХЕМАХ ТРАНЗИСТОРНОГО УРОВНЯ

Д.И. Черемисинов, Л.Д. Черемисинова

Объединенный институт проблем информатики НАН Беларуси, г. Минск, Беларусь

Рассматривается задача декомпиляции плоского описания транзисторной схемы в формате SPICE в иерархическое описание схемы на уровне логических элементов. Проблема декомпиляции возникает при верификации СБИС путём сравнения исходного описания для синтеза транзисторной схемы со схемой, восстановленной из топологии, а также при обратном инжиниринге для перепроектирования интегральных схем и обнаружения несанкционированных вложений. Рассматривается случай, когда при извлечении структуры функционального уровня из транзисторной схемы библиотека исходных логических элементов не известна. Предложены графовые методы для решения некоторых ключевых задач, возникающих при декомпиляции описания транзисторной схемы. Представленные методы реализованы на языке C++ как часть программы декомпиляции, которая протестирована на практических схемах транзисторного уровня.

Ключевые слова: *КМОП-схема из транзисторов, экстракция подсхем, распознавание логических вентилей, изоморфизм графов, формат SPICE.*

1. Introduction

Currently CMOS is the dominant technology used in most very large scale integrated (VLSI) circuit chips: more than 95 % of integrated circuits are fabricated in CMOS. Modern digital CMOS circuits contain up to a billion primitive elements at the transistor level, and the complexity of systems rapidly increases while time-to-market is imposed to decrease. Rapid verification of the software implementations and the detection of logical errors are becoming a major bottleneck in VLSI Computer-Aided Design (CAD). One of the most important steps in CAD VLSI circuit is to ensure that the final layout of the circuit geometry correctly represents the intended logic of the previous specification. Traditional test method, such as switch-level simulation, is an effective means for verifying MOS digital circuits, but it is very expensive in terms of the computing resources required, since transistor-level circuit simulators such as SPICE (Simulation Program with Integrated Circuit Emphasis) have proven to be very time consuming in terms of computer and human effort even for relatively small circuits. It is easier to verify circuit implementation at the higher level of its description (without unnecessary details) — functional or logical level.

The step to raise the level of circuit description is performed by decompiling transistor circuit. As with the decompilation of programs, the circuit is decompiled to replace its representation at a low (transistor) level with a higher-level representation (at the logic gate level). Tools for the recognition of high-level structures in transistor circuits can be used to support many tasks in integrated circuit design, such as functional verification [1], fault simulation [2], automatic test pattern generation [3], circuit reengineering [4], static timing analysis, etc.

In the paper, we consider the problem of extraction of logical networks from transistor-level circuit netlists in SPICE. The most general case is considered when the source cell library is unknown. Graphs are used to represent both the flat transistor circuit and hierarchical network of logic elements. This is undirected vertex-colored (or labeled) sparse graph of large size. In graph interpretation, the problem is formulated as recognition of subgraphs corresponding to logic gates and other subgraphs that frequently occur in a given graph. The complexity of frequent subgraph mining was thought to be tremendous due to the need to solve the subgraph isomorphism problem, which is NP-complete, many times. But VLSI transistor netlists tend to be sparse enough and have the specific structure, so runtimes did not grow unreasonably since sensible data structures and data processing methods have been adopted.

Graph methods are proposed to solve some key tasks in the problem of decompiling transistor-level circuits. The presented graph methods have been implemented in C++ as a part of a decompilation program, which was tested using practical transistor-level circuits. The presented experimental results show that the typical running time for large CMOS circuits is polynomial in the total number of transistors in netlists.

2. Related work

There were many attempts to solve the problem of extracting the hierarchy of large-scale subcircuits from a transistor level networks for various VLSI technologies, restrictions, solution methods. An overview of the approaches to this problem can be found in [5, 6]. Some of the methods for extracting logical networks are based on structure recognition and use a rule-based method in which CMOS gate structures are recognized in transistor-level circuits as channel connected sequences of MOS transistors [5, 7, 8]. Such algorithms are very fast and can easily find static CMOS logic gates, but cannot help to recognize other structures.

The other approach [9, 10] for solving subcircuit extraction problem is based on mapping transistor-level circuit into a graph and treating subcircuit pattern matching problem as subgraph isomorphism one. However, such algorithms are much slower than rule-based techniques because the subgraph matching problem is NP-complete in general case.

Some of the methods suppose that cell library is predefined. So the subgraphs to be found are known and the problem can be reduced to pattern recognition. This is done in [9, 10] and in the second step of extraction process for the gate-level structure in [5].

This paper presents the methods and the computer program for extracting the hierarchy of a large-scale digital circuit from its MOS transistor-level description for the most general case when any predefined cell library of logic gates is unknown. Moreover, the proposed method makes it possible to recognize subcircuits that implement the same logic functions but are not topologically isomorphic at the transistor level. The method is based on the solution of graph problems that are modified to process large transistor-level descriptions in a short time.

3. Transistor circuit graph representation

The source and resulting circuit netlists are presented in SPICE (Simulation Program with Integrated Circuit Emphasis) format [2]. It is one of the main formats for exchanging electrical circuits that allows to describe both the transistor- and gate-level circuits including hierarchical ones. This format is used in the developed decompilation program for source and resulting netlists.

The main part of the circuit netlist in SPICE format is the list of transistors, where each transistor terminal is indicated by the label of the net connecting it with the rest of the circuit. Each transistor has four terminals: drain, gate, source, and substrate, and so it has four connections. The general form of the netlist description of a unipolar transistor is as follows:

$$M\langle name\rangle\langle nd\rangle\langle ng\rangle\langle ns\rangle\langle nb\rangle\langle\text{model-name}\rangle[L = \text{value}]\,[W = \text{value}],$$

where M is the title of a transistor; nd, ng, ns, and nb are the labels of nets connected with the drain, gate, source, and substrate terminals of the corresponding transistor; "model-name" is the transistor type; L and W are the length and the width. For example, the transistor instance description «mp 2 1 3 3 P» is an abbreviated notation for the pairs (mp.d, 2), (mp.g, 1), (mp.s, 3), (mp.b, 3), where the name mp of the p-MOS transistor is taken out, the names of its terminals are omitted and set by a predetermined sequence of nets.

A simple model for an electrical circuit is a hypergraph, in which the vertices correspond to devices, and edges to their connections. But in a netlist format, an electrical circuit consists of elements that are connected to each other by nets, and a more convenient natural way to represent circuits is to use an undirected bipartite graph $G = (V_1, V_2, E)$, $V_1 \cap V_2 = \varnothing$, were vertices may be divided into two classes V_1 and V_2. The vertices of the first set V_1 correspond to transistor terminals and circuit ports (primary inputs and outputs), and the vertices of the other set V_2 correspond to connections between the terminals, i.e., nets. No edge exists between two transistor terminals and no edge exists between two nets. Examples of nets are power supply and ground nets, which are connected to a large number of circuit elements. Each edge $e \in E$ has one end in V_1 and the other in V_2.

Circuit representation in the form of a hypergraph requires significantly more memory during software implementation than the representation in the form of a bipartite graph. We can say that according to the memory requirement, the complexity of the first structure

is estimated as $O(n^2)$, while the complexity of the second is estimated as $O(n)$, where n is the number of circuit elements. In addition, a bipartite graph is a natural formal model for representing a circuit in SPICE format.

For sparse graphs (such as our bipartite graph), the optimal data structure that speeds up computation is an array of adjacency lists of graph vertices. The array is indexed by vertices, and each vertex of the graph corresponds to an adjacency list consisting of vertices adjacent to it. In a bipartite graph, which is a model of a CMOS circuit, the degrees of all vertices in the set V_1 of transistor terminals (and circuit ports) are 1, so for this graph each of the adjacency lists consists of the only element. In this case, the array is indexed by the vertices exclusively of the set V_1. The value of the i-th array element is the net connected to the i-th terminal. For example, the data structure representing a bipartite graph for the inverter circuit:

```
.subckt inverter 1 2 3
mp 2 1 3 3 mypmos
mn 2 1 0 0 mynmos
.ends
```

will be the array "2 1 3 3 2 1 0 0". The order of the nets in the array of adjacency is determined by the order of the transistor instances in the SPICE circuit description. The memory requirements for such structure are estimated as $O(n)$, where $n = |V_1|$.

4. Definitions and notation

As stated above, transistor circuits are modeled as a bipartite graph $G = (V_1, V_2, E)$, consisting of two subsets of vertices V_1 and V_2 corresponding to terminals (transistor terminals and circuit ports) and nets, and the set of edges E. We assume that the graph is undirected and vertex-colored, i.e., each vertex has a color associated with it, that is drawn from a predefined set of vertex colors $L(V)$. Each vertex of the graph is not required to have a unique color, and the same color can be assigned to several vertices in the same graph. Transistor-level circuits made by CMOS technology have several types of their nodes: terminals (drain, gate, source and substrate) of n-MOS and p-MOS transistors, power supply terminals (Vdd and Gnd), input/output ports (external nets), and internal nets. Thus each graph vertex corresponding to an n-MOS terminal is assigned by one of the first four colors, p-MOS terminal is assigned by one of next four colors. Then input/output ports, Vdd and Gnd nets, internal nets have unique colors.

A graph, which is a model for describing a MOS circuit, is connected (there is a path between any pair of vertices in the graph). Some other specific features of the bipartite graph: it is sparse and the degrees of all vertices from the set V_1 of transistor terminals and circuit ports are 1.

Two undirected graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ with $|V_1| = |V_2|$ are isomorphic if they are topologically identical to each other, that is, there is a one-to-one mapping f between vertices of V_1 and V_2 such that each edge in E_1 is mapped into a single edge in E_2 and vice versa, i.e., $(v, u) \in E_1 \iff (f(v), f(u)) \in E_2$.

In the case of colored graphs, the mapping f must also preserve the colors on the vertices. Two bipartite colored graphs $G^1 = (V_1^1, V_2^1, E^1)$ and $G^2 = (V_1^2, V_2^2, E^2)$ are isomorphic if there is a one-to-one mapping $f : V_1^1 \leftrightarrow V_1^2$ and $V_2^1 \leftrightarrow V_2^2$ between vertices of graphs such that $L(v) = L(f(v))$ for each $v \in V_1^1 \cup V_2^1$ and each edge in E^1 is mapped into a single edge in E^2 and vice versa, i.e., $(v, u) \in E^1 \iff (f(v), f(u)) \in E^2$.

The given graph $G_s = (V_s, E_s)$ is a subgraph of $G = (V, E)$ if $V_s \subseteq V$ and $E_s \subseteq E$. Two subgraphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ of a graph G are called edge-disjoint if they do

not share edges, i.e., they use different sets of edges from E: $E_1 \cap E_2 = \varnothing$. In the case of vertex-colored graphs, this mapping must also preserve the colors on the vertices.

Given two graphs $G = (V, E)$ and $G_1 = (V_1, E_1)$, the problem of subgraph isomorphism is to find an isomorphism between $G_1 = (V_1, E_1)$ and a subgraph of $G = (V, E)$, i.e., determine whether or not $G_1 = (V_1, E_1)$ is included in G. The subgraph isomorphism detection can be defined as follows: given a graph G and a pattern graph G_1 (that can have all its vertices and edges in G), find all the subgraphs of G which are isomorphic to G_1. Subgraph isomorphism has a wide range of practical applications.

Thus, graph isomorphism requires a strict correspondence among the two graphs being matched, and subgraph isomorphism requires an isomorphism between one of the compared graphs and a subgraph of the other. Subgraph isomorphism is more common than strict isomorphism in pattern recognition, but has been shown to be NP-complete for general case of graphs. However, for the problem of graph isomorphism, no efficient (polynomial) algorithm (suitable for arbitrary graphs) is known too, a lot of work has appeared on this topic, but little progress has been made [11].

5. Graph-based formulation of subcircuit extraction problem

The proposed subcircuit extraction application begins with the construction of a graph model from the SPICE description and hierarchical hash tables for storing the syntax elements of the analyzed circuit [12]. After this, the circuit is preprocessed, during which some standard fragments are searched. For example, each group of identical MOS transistors (with the same signals applied to their gate terminals), connected in series or in parallel, is replaced in the circuit with the single such transistor. Then, the identification of pass gates is fulfilled.

The goal of transistor circuit decompilation is to build a logic network that is functionally equivalent to it. The task consists in recognizing subcircuits, which implement logic gates. When there is no predefined cell library, it is necessary to extract subcircuits realizing logic functions or, if we cannot, to split the transistor circuit into sufficiently large subcircuits that look like as logic gates and are found quite often. In graph interpretation, the problem is solved by partitioning a graph into sufficiently large edge-disjoint subgraphs in such a way that they can be partitioned into the minimum number of classes of isomorphic graphs.

In MOS transistor circuit, not every subcircuit is correct. Correct subcircuits are among channel connected sequences of transistors. Thus, first, the proposed method of the subcircuit recognition uses the structural approach to divide the transistor-level circuit into subcircuits, which are channel connected sequences of transistors, as in [5, 7, 8]. In graph interpretation, the task is reduced to searching for connectivity components in a graph.

After this step, we get a set of possible correct subcircuits, which potentially can be standard CMOS gates. And, in a general case, in addition to the set of channel connected components of transistors, individual transistors or some other elements can remain. The set of such circuit elements forms the uncovered part of the circuit, they are no longer analyzed and are included in the resulting mixed gate-transistor-level circuit without changes.

In the second step, we have the set of possible subcircuits that are channel connected components. Among these subcircuits there are those that are standard CMOS gates. The task is to find such subcircuits and the functions that they realize. And finally, the set of all subcircuits, both implementing and not implementing CMOS gates, is partitioned into classes of topologically identical. Subcircuits of the same class represent the same functional block in resulting hierarchical description of the decompiled circuit. In graph interpretation, the task is to classify subcircuits into classes of isomorphic circuits.

As result of the mentioned steps performing, a hierarchical mixed gate-block-transistor netlist is generated. In the next step, the extraction of logic network from the hierarchical transistor-level circuit is done. That makes it possible to recognize more complex elements than gates. In graph interpretation, the task is to extract out of undirected graph connected subgraphs only with those vertices that correspond to CMOS gates, and convert the resulting undirected subgraphs into oriented ones.

6. Partitioning a graph into connected subgraphs

A static MOS circuit has a well-defined structure that allows it to be splitted into smaller subcircuits, each of which is a group of channel connected transistors. Such a circuit component consists of transistors connected by their source and drain terminals and provides a signal path between the power Vdd and Gnd terminals. A group of channel connected transistors is a cluster with three types of external connections:

- the cluster inputs are fed only to the transistor gates;
- the cluster outputs are connected only to the transistor gates of the other clusters;
- there are connections to the Vdd and Gnd terminals.

Figure 1 shows the example of grouping transistors into two channel connected components.

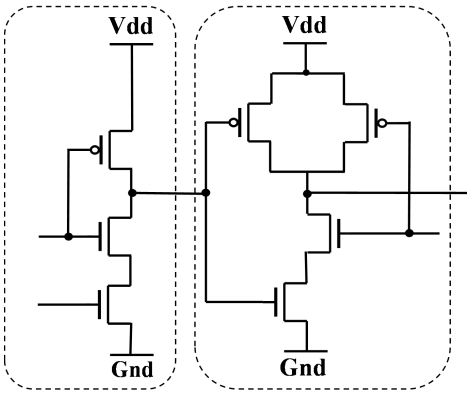


Fig. 1. Two groups of channel connected components of MOS transistor circuit

The task of recognizing clusters of the MOS transistor circuit is solved on a graph H, which is obtained from the previously introduced graph $G = (V_1, V_2, E)$, by:

- removing the circuit power terminals and transistor gate terminals;
- shorting the drain and source terminals for each transistor.

In graph interpretation, a channel connected group of MOS transistor circuit corresponds to a connectivity component of the graph H. All connectivity components are edge-disjoint subgraphs of the graph H. So the splitting transistor circuit into disjoint subcircuits of channel connected transistors is reduced to the search for connectivity components of the graph H. This is done by using the well-known Depth-First Search (DFS) algorithm, which starts at an arbitrary unconsidered vertex and explores paths from it as far as possible along each branch before backtracking. Reaching a backtracking results in a new connectivity component. When implementing the DFS algorithm, the initial graph $G = (V_1, V_2, E)$ was not transformed explicitly into the graph H. Instead, the DFS algorithm was tuned to the modification of data structure for storing a bipartite graph G.

It is worth noting that in order to search for groups of transistors connected by a current, it is necessary to know in advance which terminals of the initial transistor-level circuit correspond to Gnd and Vdd.

7. Structural recognition of logic gates

In static CMOS circuits, the MOS transistor can be regarded as a switch controlled by input voltage at its gate. The simplest digital circuit is a pass gate consisting of the only MOS transistor that controls the transmission of binary signals. This circuit is passive because it does not amplify the input signal. The amplification of binary signals is provided by a complementary MOS circuit (CMOS gate) in which, at any instant of time, gate output is connected either to a power circuit or to ground through a path with low resistance. The CMOS gate consists of two blocks separated by a connection net (output net) (Fig. 2). The first block is formed by n-MOS transistors (pull-down network — n-part of a CMOS gate), which are connected in series by their source/drain terminals. The block is placed between the connection node (output net) and Gnd. The second block is formed by p-MOS transistors (pull-up network — p-part of a CMOS gate), which are connected in parallel, by their source/drain terminals. The block is placed between Vdd and the connection node. When the block conductivities are complementary, no matter what the input signals (on transistors gates) are, there is a valid path to output node either from Gnd or from Vdd.

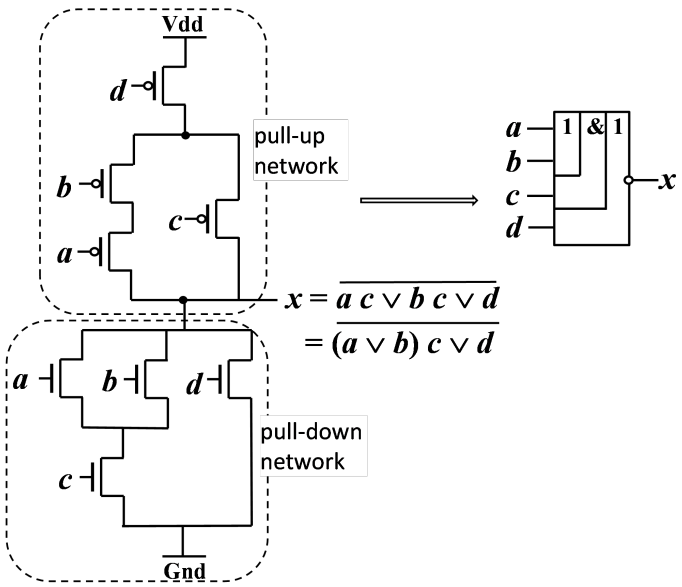


Fig. 2. CMOS gate: its transistor structure and implemented function

A CMOS gate is a group of channel connected transistors; the opposite is not always true. The necessary conditions for the group to belong to the class of CMOS gates are the following:

- the only chain connecting the p-part and the n-part of the group is the output (connection) node;
- all paths from the connection node go to the signal nets Gnd or Vdd;
- pull-down and pull-up networks have the same number of transistors;
- pull-down and pull-up networks implement mutually inverse functions.

For example, the right group of channel connected transistors of the two shown in Fig. 1 is CMOS NAND element, but the left one is not.

A logic function expression implemented by the pull-down (or pull-up) network is formed by tracing paths from the connection node to Gnd (or Vdd) terminal. Each path gives a conjunction of the conductivity variables fed to gate terminals of the transistors from the path. The OR of all such conjunctions yields the disjunctive normal form (DNF) for the expression. If the conductivity functions f_n and f_p of pull-down or pull-up networks are complementary ($f_n = \bar{f}_p$), then the analyzed channel connected group is a standard CMOS gate. To classify CMOS gates extracted from the transistor circuit, it is convenient to represent the recognized functions as parenthesized algebraic expressions. Such a form can be constructed by the algebraic factoring DNF of the Boolean function found [12, 13]. For the CMOS gate in Fig. 2, we have

$$f_n = ac \vee bc \vee d = (a \vee b)c \vee d, \quad f_p = \bar{a}\bar{b}\bar{d} \vee \bar{c}\bar{d} = (\bar{a}\bar{b} \vee \bar{c})\bar{d},$$

and $f_n = \bar{f}_p$. Thus, it is standard NOAO2 CMOS gate.

Channel connected groups of transistors, which are static CMOS gates, can be divided into classes of identical according to the formulae of implemented logic functions. Each class is made up of all instances that implement the same function formulae and therefore are functionally equivalent.

However, not always the only CMOS gate may be associated with the class of functionally equivalent channel connected groups. This is true if we are only interested in functional equivalence of circuits. The topological aspect requires dividing a class of functionally equivalent CMOS gates into subclasses of topologically equivalent CMOS gates. Some features of the topological implementation of circuits at the transistor level, which must be taken into account when combining or not combining subcircuits into a class of topologically equivalent, are given in [12]. For example, we should take into account the following specifics of the topological implementation of CMOS gates:

- asymmetry of the inputs of the topological implementation of a CMOS gate (although the gate implements a symmetric function);
- interchangeability of its drain and source.

The proposed algorithm distinguishes between the following groups of functionally equivalent CMOS gates:

1) CMOS gate implementations that differ from each other by exchange the drain and the source at least in one transistor. The interchangeability of the drain and source in a MOS transistor results in existence of topologically different subcircuits that implement the same logic function. For instance, there are four variants of subcircuits for a CMOS inverter. If, in a decompiled circuit, all variants of a logic gate subcircuit are represented by the same subcircuit, then the decompiled and original circuits will not be isomorphic.
2) CMOS gate implementations that differ from each other by permutation of their inputs (Fig. 3). Even if a CMOS gate implements a symmetric Boolean function, the permutation of the inputs of the CMOS circuit that implements it makes the circuit topologically nonisomorphic to the original one. This is because CMOS circuit has asymmetric inputs. However, logically, both CMOS circuits implementations are equivalent.

Topological equivalence of CMOS gate implementations can be established by checking whether the corresponding graphs are isomorphic or not.

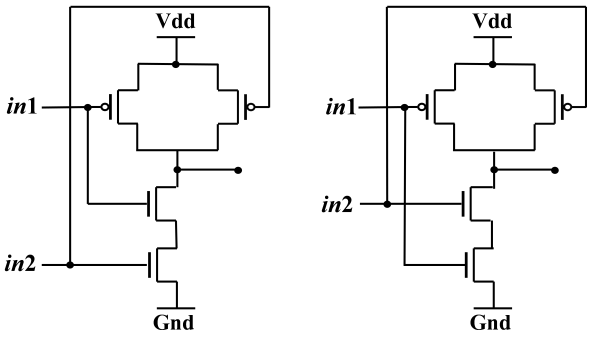


Fig. 3. Topologically nonequivalent implementations of the NAND gate

8. Graph isomorphism and canonical labeling

One of the key operations required to partition the set of subgraphs into classes of isomorphic ones consists in checking whether two subgraphs are identical or not. One way of performing this check is to perform a graph isomorphism operation. But in our case, when many such checks are required for the same set of subgraphs, a better way to perform the task is vertex canonical labeling [14]. It assigns to each graph a unique code (a sequence of bits) that is invariant on the ordering graph vertices and edges. Comparing whether or not two graphs have identical canonical labels allows you to say whether or not two graphs are identical. Moreover, by comparing the canonical labels we can partition the set of graphs into classes of pairwise isomorphic graphs. Thus, checking two arbitrary graphs for isomorphism is reduced to comparison of their canonical forms.

Calculating canonical labels is computationally equivalent to determining isomorphism between graphs; both canonical labeling and determining graph isomorphism are not known to be either in P- or in NP-complete class [15]. If a graph has $|V|$ vertices, the complexity of determining its canonical labeling using this method is in $O(|V|!)$ making it impractical even for moderate size graphs.

In our case, the complexity of determining a canonical labeling of a graph is reduced due to taking into account the special properties of subgraphs under classification: they are vertex-colored, sparse and small enough. By comparing canonical labels of graphs, it is possible to sort them in a unique and deterministic way.

Canonical labeling is done in an iterative manner in the process of building a sequence of partitions for the set of graph vertexes that defines an ordering of the graph vertices.

Suppose we have an ordered collection of subsets of the vertices $(V_1, V_2, \ldots, V_k)$ whose union is V. They say that all vertices from the same subset V_i have the same label i. The set of these subsets represents the partition of the set V of graph vertices, constructed from the initial partition that is specified by colors of vertices.

At first, the number and sizes of these subsets V_i must be the same for both compared graphs, i.e., the graphs have identical partitions of the set V. Then we repeatedly apply a relabeling step, which assigns to each vertex v a classifier $C(v) = (n_1, n_2, \ldots, n_k)$, where n_i is the number of vertices in subset V_i that are adjacent to v. Using these classifiers, each subset V_i can be partitioned into subsets, where each subset should include all vertices with the same classifier. These subsets are lexicographically ordered according to their classifiers. In this way, we may obtain a refinement of the original partition, which consists of subdividing the partition blocks. No refinement will be obtained if all vertices in each subset V_i have identical classifiers. If a refinement has been obtained, then the classifiers are recalculated (and vertices are relabeled) until there is no further refinement.

It is clear from the description that the essential idea is to relabel vertices so that each new classifier reflects information about a gradually increasing region around the vertex. In an ideal situation, after exhaustive application of the relabeling process, all subsets in the partition $(V_1, V_2, \ldots, V_k)$ will become singletons (containing exactly one member), such a graph canonical labeling is called discrete. Discrete canonical labeling defines graph canonical isomorph, which is given by an ordered set of vertex classifiers that represent a unique graph code. If two compared graphs have the same canonical labeling, then they are isomorphic with each other.

Today there exist several successful programs of computing the canonical isomorphs, they differ by refinement procedure associated with details of reducing the search tree built in the process of partition refinement (the pioneer work [16]). However, the fastest known algorithm for graph isomorphism (as well as graph canonization) is $2^{O(\sqrt{n \log n})}$ time, and no polynomial algorithm is known.

9. Graph-based subcircuit recognition method

After structural recognition of logic gates and pass gates, there are two main unrecognized groups of transistors. The first group includes structures that cannot be partitioned into gates or that are separate transistors. In resulting SPICE description they are given as ungrouped transistors. The second group includes found channel connected components of MOS transistors that have not been recognized as standard CMOS gates, so they are assigned to be pseudogates. Each of the pseudogates is represented by a bipartite undirected vertex-colored graph, which is sparse.

At this stage, the subcircuits associated with the pseudogates must be pattern matched using a user-defined library of cells, as was done, for example, in the Frosty program [5]. However, in our case, when there is no cell library, all we can do is to classify remaining pseudogates into classes of pairwise identical subcircuits.

In graph interpretation, the task consists in testing isomorphism between graphs by means of comparing their canonical labelings. To simplify the canonization problem, the subcircuit graphs are complemented with edges connecting all four terminals for each transistor. The "bliss" program (T. Yuntilla and P. Caski [17]) was chosen as a prototype of a program for calculating canonical isomorphs, which provides fast processing of large and sparse graphs. Our pseudogate graphs are represented with exactly such graphs. The experiments with the modified canonical graph labeling program have shown that applying the canonicalization process to pseudogate graphs results in a discrete canonical labeling.

The graphs of pseudogates with the same initial partitions on the set of colored vertices are considered one by one. For each of them, a canonical isomorph is generated and a hash of the canonized graph is computed. The hash value is a word-length bit string obtained by the transformation of a sequence of numbers representing graph vertex classifiers. Graphs with equal hashes are isomorphic and they are changed in a hierarchical SPICE description with their canonical isomorph.

10. Logic network construction

At this stage, we have a mixed circuit which, in addition to static CMOS gates, consists of pseudogates, pass transistors and ungrouped transistors. Now, the task is to recognize more complex elements than gates. Using the previously described graph-based subcircuit recognition method, we can recognize in logical network (consisting of CMOS gates) some library-defined patterns. So the next step is to extract a subcircuit from the mixed circuit

which consists only of gates, i.e., logical network. To specify a logical network means to specify its inputs and outputs, the structure of connections between its elements, and Boolean functions realized by the elements.

In graph interpretation, a logical network is a directed connected graph $H = (W, A)$. The set of vertices W is partitioned into three subsets: network inputs and outputs, and internal vertices. Each vertex is labeled with input or output variable, or, if it is internal vertex, with the function realized by the corresponding gate. An arc (directed edge) $a = (u, v) \in A$ goes from the source vertex u to the target vertex v $(u, v \in W)$. We further consider that graph $H = (W, A)$ is specified by the adjacency list, i.e., an array D of the length $|W|$, where each entry $D[i]$ is a pointer to a linked list of all out-neighbors of vertex $w_i \in W$.

The connected graph $H = (W, A)$ is extracted from the undirected bipartite graph $G = (V_1, V_2, E)$ corresponding to the object mixed circuit. Graph H is contained in G as the connected component C, including only the vertices corresponding to the CMOS gates. There can be more than one such a component in the graph G. Each undirected connected subgraph corresponding to a connected component in a bipartite undirected graph G is transformed into a directed connected graph $H_i = (W_i, A_i)$ of some logical network. The transformation is carried out in the process of traversing the subgraph along the paths in-going or out-going from the vertices labeled as CMOS gates.

The search for the next connected component C begins with any unconsidered vertex labeled as a CMOS gate and is done by the breadth-first search (BFS) method, considering only the vertices labeled as CMOS gates. BFS allows not only to find out a connected component C, but also to get its topological sorting, which orders the vertices so that the order corresponds to reachability. That is, if a vertex u is directly reachable from v, then the edge $(u, v) \in E$ generates arc $(v, u) \in A$, and if the vertex v belongs to the i-th graph rank, then the vertex u belongs to the $(i + 1)$-th rank.

The proposed method provides to extract logic network that is ranked lexicographically. From a lexicographically ordered network of logical gates, it is easy to pass to the formulas of logical equations that specify the output functions of the network.

The next task connected with the logic network extraction is to determine primary inputs and outputs of the network. It is solved by considering fan-ins and fan-outs for all vertices of the graph $H = (W, A)$. If all vertices from both fan-in and fan-out of some vertex $v \in W$ are labeled as CMOS gates, then the vertex v is an internal one. Non-internal vertices are assigned to primary inputs or primary outputs, depending on which of the fan-in and fan-out sets contains the non-internal vertex.

After the gate-level networks are extracted, more complex elements than gates can be recognized if the cell library is known.

11. Experimental results

Some experiments with the developed decompilation program have been performed. Decompiled transistor-level circuits implement digital devices, both combinational and sequential, with the complexity of several hundred thousand transistors.

The experiments have been carried out on a computer with Intel(R) Core(TM) i5-4460 3.20 GHz and 16 GB RAM. Table 1 shows how quickly the decompilation speed decreases with increasing transistor circuit complexity. Here, transistor circuit decompilation speed is estimated by the number of its transistors processed per second: n/t, where n is the number of transistors in a decompiled circuit, t is the circuit decompilation time.

Two types of experiments have been carried out with the developed programm. In the first experiment, CMOS circuits obtained by CAD system were used. In this case, the technology cell library was known. One hundred percent coverage of the transistor-level circuit by logic gates has been obtained. In the second experiment, transistor-level circuits extracted from layouts have been examined. For some of these circuits the hierarchical SPICE models were known, for others there was no additional information other than the transistor-level circuit. In some circuits, in addition to MOS transistors, there were other primitive elements (bipolar transistors, RC elements, etc.). Here, the coverage of the transistor-level circuit with logic gates at the level of 60–70 % was observed.

Table 1

Speed of transistor circuits decompilation

Number of transistors	Decompilation time: seconds	Number of found gates	Processing speed: transistors per second
1593	0.047	570	33893
11935	0.332	2727	35948
12566	0.398	3163	31572
38356	2.603	6179	14735
52408	6.085	9091	8612
62380	6.648	13664	9380
206896	90.182	34153	2294
345301	187.151	60033	1845

Some intermediate results of applying the proposed graph methods in the subcircuit extraction program are given in Table 2. The table shows how many:

- n-MOS and p-MOS transistors are contained in each decompiled circuit (the second column);
- the numbers of found pass gates (the third column);
- the numbers of all found CMOS gates in the circuit, the numbers of functionally and topologically identical CMOS gates (the forth column);
- the numbers of all found pseudogates and the numbers of classes containing topologically identical pseudogates (the fifth column).

Table 2

Intermediate experimental results

Circuit	Number of MOS transistors	Number of pass gates	Number of CMOS gates	Number of pseudogates
1	1682, 1269	0	528, 16, 16	154, 55
2	3016, 2381	89	1041, 15, 39	284, 88
3	5776, 5827	25	2392, 7, 8	615, 23
4	5962, 5947	661	2777, 17, 34	119, 71
5	9415, 9415	1374	6639, 16, 44	0, 0
6	22988, 16436	766	5915, 39, 64	1178, 252

12. Conclusion

In this paper, we present the graph-based methods for solving the task of extracting gate-level circuits from transistor-level descriptions for the most general case when any predefined cell library of logic gates is unknown. We have used well-known graph methods, modifying them so that they process large data of special types in a short time. The proposed methods have been implemented in C++ as a part of a decompilation program. The program has

been tested using practical and automatically designed transistor-level circuits. The tested circuits had up to 100000 transistors. Some results of experiments on the program execution and verification of the correctness of decompilation results can be found in [18].

Our future work is to extend the decompilation program with means of recognition memory elements in gate-level network without using pattern matching techniques.

REFERENCES

1. *Abadir M. S. and Ferguson J.* An Improved Layout Verification Algorithm (LAVA). Proc. EDAC, Glasgow, UK, 1990, pp. 391-395.
2. *Baker R.* CMOS Circuit Design, Layout, and Simulation. Third Ed. John Wiley & Sons, 2010.
3. *Kundu S.* A transistor to gate level model extractor for simulation, automatic test pattern generation and verification. Proc. Int. Test Conf. IEEE, Washington, 1998, pp. 372–381.
4. *Hunt V. D.* Reengineering: Leveraging the Power of Integrated Product Development. Vermont, Oliver Wight Publ., 1993. 282 p.
5. *Yang L. and Shi C-J. R.* FROSTY: A program for fast extraction of high-level structural representation from circuit description for industrial CMOS circuits. Integr. VLSI J., 2006, vol. 39, no. 4, pp. 311–339.
6. *Zhang N., Wunsch D. C., and Harary F.* The subcircuit extraction problem. IEEE Potentials, 2003, vol. 22, no. 3, pp. 22–25.
7. `http://www.silvaco.com/content/appNotes/iccad/2-003_LogicGates.pdf` — Logic Gate Recognition in Guardian LVS, Silvaco, 2009.
8. *Lester A., Bazargan-Sabet P., and Greiner A.* YAGLE, a second generation functional abstractor for CMOS VLSI circuits. Proc. ICM'98, Monastir, Tunisia, 1998, pp. 265–268.
9. *Ebeling E.* GeminiII: A second generation layout validation program. Proc. ICCAD-89, Santa Clara, CA, USA, 1988, pp. 322–325.
10. *Ohlrich M., Ebeling C., Ginting E., and Sather L.* SubGemini: Identifying subcircuits using a fast subgraph isomorphism algorithm. Proc. 30th ACM/IEEE Design Automation Conf., Dallas, TX, USA, 1993, pp. 31–37.
11. *Conte D., Foggia P., Sansone C., and Vento M.* Thirty years of graph matching in pattern recognition. Int. J. Pattern Recognit. Artif. Intell., 2004, vol. 18, pp. 265–298.
12. *Cheremisinov D. I. and Cheremisinova L. D.* Extracting a logic gate network from a transistor-level CMOS circuit. Russian Microelectronics, 2019, vol. 48, no. 3, pp. 187–196.
13. *Cheremisinova L. D.* Sintez i optimizatsiya kombinatsionnykh struktur SBIS [Synthesis and Optimization of Combinational Structures of VLSI]. UIIP NAS Belarus Publ., Minsk, 2005. (in Russian)
14. *Hartke S. G. and Radcliffe A. J.* McKay's Canonical Graph Labeling Algorithm. `https://api.semanticscholar.org/CorpusID:6454900`, 2008.
15. *Garey M. R. and Johnson D. S.* Computers and Intractability: A Guide to the Theory of NP-Completeness. NY, W. H. Freeman Publ., 1979.
16. *McKay B. D.* Practical graph isomorphism. Congressus Numerantium, 1981, vol. 30, pp. 45–87.
17. *Junttila T. and Kaski P.* Engineering an efficient canonical labeling tool for large and sparse graphs. Proc. ALENEX, New Orleans, LA, 2007, pp. 135–149.
18. *Cheremisinov D. and Cheremisinova L.* Subcircuit pattern recognition in transistor level circuits. Pattern Recognit. Image Anal., 2020, vol. 30, pp. 160–169.

ПРИКЛАДНАЯ ТЕОРИЯ ГРАФОВ

УДК 519.87:519.6:519.178 DOI 10.17223/20710410/64/5

АНАЛИЗ БАЗЫ ДАННЫХ ОПТИМАЛЬНЫХ ДВУХКОНТУРНЫХ КОЛЬЦЕВЫХ СЕТЕЙ[1]

Э. А. Монахова, О. Г. Монахов

Институт вычислительной математики и математической геофизики СО РАН, г. Новосибирск, Россия

E-mail: {emilia, monakhov}@rav.sscc.ru

Оптимальные циркулянтные сети вызывают практический интерес как модели надёжных с низкой задержкой сетей связи мультипроцессорных кластерных систем и сетей на кристалле. Авторами впервые построена большая база данных (датасет) оптимальных по диаметру двухконтурных кольцевых циркулянтных сетей до 50 тысяч узлов, содержащая полный набор образующих оптимальных графов. Проведён анализ датасета с целью исследования проблемы поиска аналитически задаваемых семейств оптимальных графов. Разработаны два новых алгоритма автоматизированного поиска аналитических, описываемых полиномами от диаметра, описаний семейств оптимальных графов. С помощью реализованных алгоритмов найдено большое количество новых аналитически описываемых семейств оптимальных сетей, проверенное с помощью валидации на всём диапазоне изменения диаметров графов датасета. Найденные семейства оптимальных сетей могут быть использованы при масштабировании алгоритмов передачи информации в двухконтурных кольцевых циркулянтных структурах.

Ключевые слова: *датасет оптимальных сетей, неориентированные двухконтурные кольцевые сети, циркулянтные сети, минимальный диаметр.*

DATABASE ANALYSIS OF OPTIMAL DOUBLE-LOOP NETWORKS

E. A. Monakhova, O. G. Monakhov

Institute of Computational Mathematics and Mathematical Geophysics SB RAS, Novosibirsk, Russia

Optimal circulant networks are of practical interest as models of reliable low-latency communication networks for multiprocessor cluster systems and on-chip networks. The authors are the first to construct a large dataset of optimal diameter double-loop circulant networks with up to 50 thousand nodes, containing a complete set of optimal graph generators. The analysis of the dataset has been carried out in order to study the problem of finding analytically defined families of optimal graphs. Two new algorithms for automatically finding analytical descriptions of optimal graphs families described by polynomials in diameter have been developed. Using the implemented

[1]Работа выполнена при финансовой поддержке бюджетным проектом ИВМиМГ СО РАН (код проекта FWNM-2022-0005).

algorithms, a large number of new analytically described families of optimal networks have been found and tested using validation over the entire range of changes in the diameters of the dataset graphs. The found families of optimal networks can be used when scaling information transmission algorithms in double-loop circulant structures.

Keywords: *dataset of optimal networks, undirected double-loop networks, circulant networks, minimum diameter.*

Введение

Неориентированные двухконтурные кольцевые сети являются объектом интенсивных исследований [1–11]. Благодаря высокой масштабируемости, надёжности и симметрии, они находят применение как сети связи в мультипроцессорных кластерных системах, в криптографии при построении совершенных кодов, исправляющих ошибки, а также в сетях на кристалле в качестве замены традиционно используемых в них двумерных решёток и торов, имеющих существенно большие задержки при одинаковом числе узлов.

Двухконтурная кольцевая сеть (undirected double-loop network) представляет собой неориентированный граф $C(N;1,s)$, $1 < s < N/2$, с множеством вершин $V = \{0, 1, \ldots, N-1\}$ и рёбер $E = \{(i,j) : i - j \equiv \pm 1 \pmod N, i - j \equiv \pm s \pmod N\}$, где $\{1, s\}$ — образующие; N — порядок графа. Пример двухконтурной кольцевой сети с числом узлов $N = 18$ представлен на рис. 1. Двухконтурные кольцевые сети степени четыре принадлежат к классу циркулянтных сетей [1–3, 12–16].

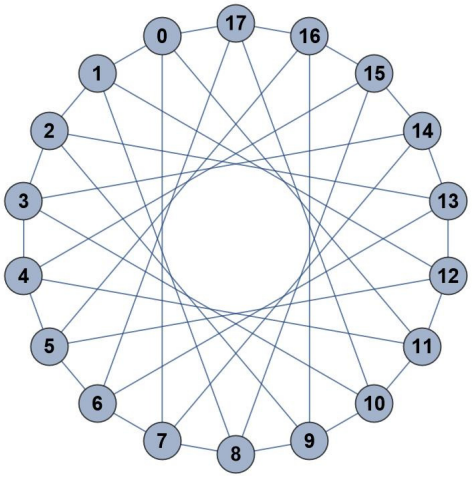


Рис. 1. Двухконтурная кольцевая сеть $C(18;1,7)$

Задержки при передаче информации в сети, а также при организации коллективных обменов в системе оцениваются диаметром графа (и/или средним расстоянием между вершинами) [3, 4, 16]. Диаметр графа $G = (V,E)$ есть параметр $\mathrm{d}(G) = \max\limits_{i,j \in V} \mathrm{d}(i,j)$, где $\mathrm{d}(i,j)$ — длина кратчайшего пути из вершины i в вершину j. Известно (см. ссылки в [1, 3]), что верхняя граница максимально возможного числа вершин в циркулянтных графах степени четыре с диаметром d равна $N_d = 2d^2 + 2d + 1$. Точная нижняя граница диаметра циркулянтов степени четыре получена в [17, 18]: $D(N) = \lceil (-1 + \sqrt{2N-1})/2 \rceil$.

Проблема синтеза оптимальных циркулянтных графов состоит в поиске графов с минимально возможным диаметром среди графов заданных степени и числа вершин. *Оптимальным* называется граф $C(N;1,s)$ с диаметром $\mathrm{d}(C(N;1,s)) = D(N)$, *субоптимальным* — граф с диаметром $D(N)+1$. В работе [19] выдвинута гипотеза, проверенная для всех значений $N \leqslant 8 \cdot 10^6$: по крайней мере, субоптимальные графы вида $C(N;1,s)$ существуют для любых $N > 4$.

Оптимальное семейство циркулянтных сетей любого порядка $N > 4$ и степени четыре найдено в [17] и переоткрыто в [18, 20]: $\{C(N;d,d+1) : d \geqslant 1\}$, где d — ближайшее целое к $(-1+\sqrt{2N-1})/2$. В [17] доказано, что все графы семейства одновременно имеют минимумы диаметра и среднего расстояния между вершинами. Для данного семейства сетей известны аналитические алгоритмы парной маршрутизации с константными оценками сложности [3, 20, 21].

Из описания циркулянтного графа вида $C(N;d,d+1)$ можно получить изоморфные описания путём умножения его образующих d и $d+1$ на элементы $t \leqslant \lfloor N/2 \rfloor$ приведённой системы вычетов по модулю N. Но такой метод не может быть использован для получения оптимальных описаний графов $C(N;1,s)$ при любых N, поскольку для некоторых N они либо не существуют, либо существуют, но не изоморфны описанию вида $(N;d,d+1)$. В настоящей работе предлагается метод автоматизации поиска аналитических описаний семейств оптимальных графов вида $C(N;1,s)$ на основе полученной авторами большой базы данных (датасета) параметров описаний оптимальных двухконтурных кольцевых сетей $C(N;1,s)$.

1. Подходы к построению оптимальных двухконтурных кольцевых сетей

Семейства двухконтурных кольцевых сетей, описанные в литературе, привлекают внимание в качестве сетей связи при изучении их структурных и коммуникативных свойств — алгоритмов маршрутизации различных видов, вложимости в чипы для сети на кристалле, структурной надёжности и др. Например, первое известное семейство оптимальных графов с аналитическим описанием [22] активно изучалось в теории кодирования [11] и как модель сети связи многопроцессорных систем:

$$\{C(N_d;1,2d+1) : d \geqslant 1\}.$$

В работе [23] поставлена следующая проблема: классифицировать все значения N, для которых оптимальные неориентированные графы $C(N;1,s)$ существуют. В литературе рассмотрены различные подходы к решению данной проблемы и получены некоторые бесконечные семейства графов $C(N;1,s)$ с аналитическим описанием.

В [23] доказано, что для семейства графов с числом вершин $N = N_d - 1$, где $d > 1$, минимально возможный диаметр равен $D(N)+1$. В большинстве работ, посвящённых поиску бесконечных семейств оптимальных графов $C(N;1,s)$, используются теоретические верхние оценки диаметра [19, 23–28]. В работах [8, 19, 23, 26, 28] найдены или исследуются семейства графов с линейными образующими вида $s = 2d \pm \alpha$, где d — диаметр; в [5, 29, 30] — семейства графов с квадратичными образующими от диаметра. Эффективные алгоритмы парной маршрутизации разработаны для ряда найденных семейств [26, 29, 31]. Известен [9] алгоритм парной маршрутизации для графов $C(N;1,s)$ сложности $O(\Delta)$, где $\Delta \leqslant d$. В [31, 32] получены «плотные» бесконечные семейства оптимальных графов, описания которых следуют из взаимной простоты чисел (N,d) или $(N,d+1)$. В [19] найдено множество мощности $O(\sqrt{d})$ бесконечных семейств оптимальных графов $C(N;1,s)$ для каждого интервала значений N,

где $N_{d-1} < N \leqslant N_d$, $d > 1$, с образующими вида $s = 2d \pm \alpha$. В [27] найдены три семейства оптимальных сетей, которые представимы как произведения Кронекера двух циклов. В [5] авторы получили шесть оптимальных и пять субоптимальных семейств графов $C(N; 1, s)$ для каждого интервала значений N, где $2d^2 + d < N < N_d - 1$, $d > 1$. В [33] реализованы генетические алгоритмы поиска семейств и построены 70 новых семейств оптимальных графов с линейными образующими видов $s = 4d \pm \alpha$ и $s = 6d \pm \alpha$. Авторы [34], используя предложенный ими алгоритм вычисления диаметра графов $C(N; 1, s)$, представили некоторые фрагменты результатов вычисления оптимальных и субоптимальных образующих для различных значений N, включая $N = 32\,000$. В [35], в связи с актуальностью применения циркулянтов в сетях на кристалле, получен датасет оптимальных циркулянтов различных степеней от 4 до 10 с числом вершин до 500, который включает в том числе двухконтурные кольцевые сети.

В [36] авторы построили датасет оптимальных (с минимально возможным диаметром при заданном порядке) графов $C(N; 1, s)$ до 50 тысяч вершин. Новый датасет содержит для каждого порядка графов все образующие, соответствующие оптимальным или субоптимальным (в случае отсутствия оптимальных) описаниям графа. Перечисление образующих для оптимальных графов в датасете позволило найти аналитические зависимости параметров, определяющих семейства оптимальных графов.

На рис. 2 приведено трёхмерное графическое изображение фрагмента датасета точек (N, s, d) с числом вершин $10 \leqslant N \leqslant 900$. Датасет получен с применением параллельного алгоритма исчерпывающего поиска, реализованным на C на кластере Kunpeng [36]. Точки (N, s, d) соответствуют параметрам описаний оптимальных графов $C(N; 1, s)$. Для каждого N показаны все образующие $s \leqslant N/2$, которые определяют граф минимально возможного диаметра d при данном N. Проведённый анализ графов из датасета показал существование значений N, для которых единственная оптимальная образующая может быть как линейного, так и квадратичного видов от диаметра. Весь датасет описаний параметров оптимальных графов с $N \leqslant 5 \cdot 10^4$ вершин содержит около 451 000 точек и представлен в открытом доступе в Интернете: `https://github.com/mila0411/Double-loop-networks/tree/main/Dataset`.

Первоначальный анализ датасета на открытие аналитически описываемых семейств оптимальных графов проведён в [36] с помощью подхода, основанного на темплейтах с недоопределёнными коэффициентами и использующего для поиска перспективных темплейтов алгоритмы метаэвристического поиска — муравьиной колонии и дифференциальной эволюции [37, 38]. Подробное описание алгоритмов поиска можно найти в [36]. Реализованные алгоритмы с применением пяти найденных темплейтов сгенерировали 200 семейств оптимальных двухконтурных кольцевых графов. На рис. 3 члены полученных семейств графов отмечены чёрными точками.

Далее рассмотрен другой подход к автоматизации поиска семейств оптимальных графов в датасете, основанный на последовательном делении с остатком параметров оптимальных графов и построении коэффициентов полиномов для их порядков и образующих, который: 1) обобщает темплейт-ориентированный подход; 2) содержит некоторые общие принципы построения семейств оптимальных графов; 3) позволяет открыть известные и получить новые семейства, покрывающие, как показали эксперименты, представительную часть точек датасета; 4) даёт возможность проводить теоретический анализ и практическую реализацию сетей, основанных на аналитических описаниях. На базе нового подхода разработаны два алгоритма автоматизированного поиска семейств оптимальных графов, отличающихся видом образующих.

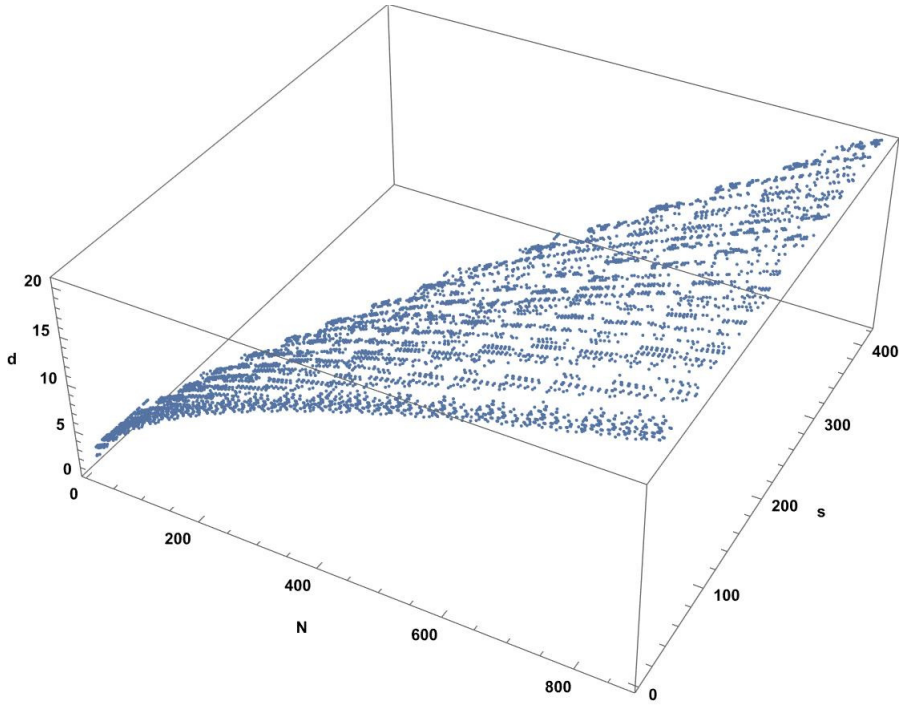


Рис. 2. Фрагмент датасета точек (N, s, d) оптимальных графов $C(N; 1, s)$

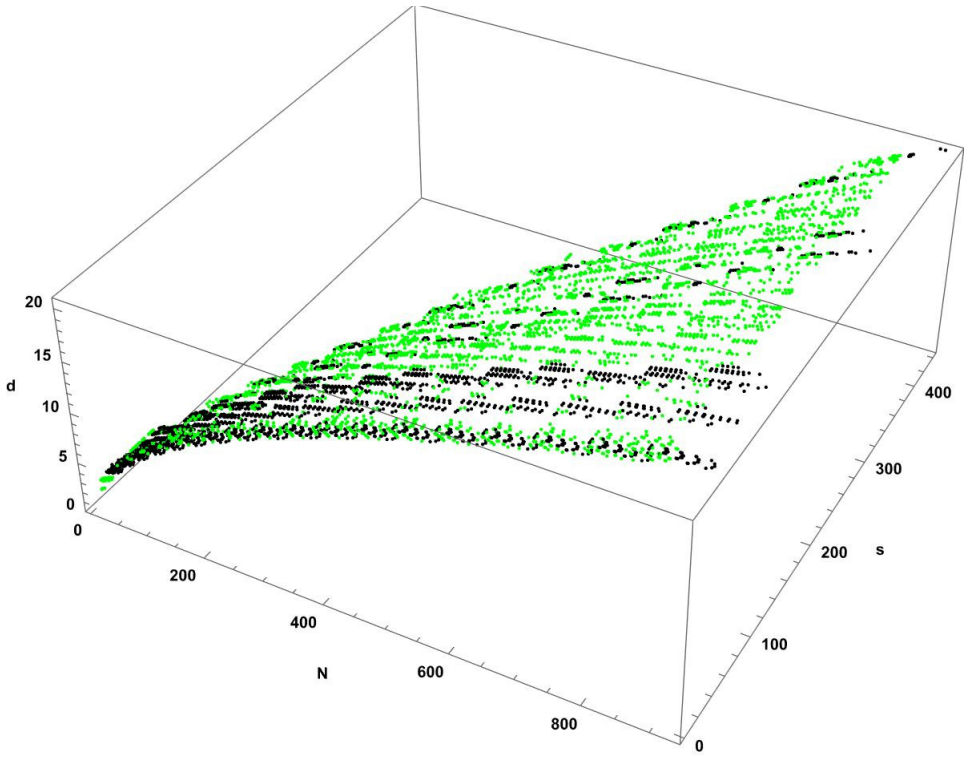


Рис. 3. Семейства оптимальных графов, полученные с помощью темплейт-ориентированного подхода

2. Поиск семейств оптимальных двухконтурных кольцевых сетей с квадратичными образующими

Чтобы автоматизировать процесс поиска семейств оптимальных графов, рассмотрим конкретный пример построения графов потенциально возможного семейства, основываясь на данных, взятых из датасета.

Введём параметр $p \geqslant 1$. Назовём его периодом повторяемости членов семейства. Параметр p равен разнице диаметров между соседними членами семейства. Анализ известных семейств оптимальных графов показал, что p может принимать различные значения — от $p = 1$, когда семейство существует при любом диаметре [22], до $p = 15$ [5, теорема 3.8], когда члены оптимального семейства существуют в классе диаметров по модулю 15. В настоящей работе мы рассматриваем случаи, когда разница диаметров между соседними членами семейства является постоянной.

Покажем, как, используя две точки датасета, можно получить аналитические формулы для построения образующих и порядков графов потенциального семейства.

Возьмём точку в датасете, например $(N_1, s_1, d_1) = (258, 48, 11)$. Пусть $p = 6$. Найдём в датасете точку (N_2, s_2, d_2), для которой, кроме условия $d_2 = d_1 + p$, выполняется следующее:

$$\left\lfloor \frac{N_1}{s_1} \right\rfloor = \left\lfloor \frac{N_2}{s_2} \right\rfloor. \tag{1}$$

Например, пусть это будет точка $(N_2, s_2, d_2) = (606, 108, 17)$. Выполнение условия (1) означает, что обе точки датасета находятся на общей линии с углом наклона, определяемым отношением $\lfloor N/s \rfloor$ (см. рис. 2). Найдём целые части и остатки от деления образующих графов, соответствующих данным точкам, на их диаметр: $s_1 = 4d_1 + 4$, $s_2 = 6d_2 + 6$. Введём параметры, которые определяют последовательности увеличения коэффициентов при одинаковых степенях d для образующих графов возможного семейства:

$$\begin{gathered} \Delta_1 = \left\lfloor \frac{s_2}{d_2} \right\rfloor - \left\lfloor \frac{s_1}{d_1} \right\rfloor, \\ \Delta_2 = s_2 \bmod d_2 - s_1 \bmod d_1. \end{gathered} \tag{2}$$

Для нашего примера $\Delta_1 = 2$, $\Delta_2 = 2$. Если продолжить эти последовательности, увеличивая диаметр графов на p, то общий вид образующих графов семейства будет следующим:

$$s = \left(\left\lfloor \frac{s_1}{d_1} \right\rfloor + \frac{(d - d_1)\Delta_1}{p} \right) d + s_1 \bmod d_1 + \frac{(d - d_1)\Delta_2}{p}.$$

Собирая коэффициенты при одинаковых степенях d, получим квадратичный полином от диаметра, определяющий образующие семейства оптимальных графов:

$$\begin{gathered} s = ed^2 + fd + g, \\ e = \frac{\Delta_1}{p}, \\ f = \left\lfloor \frac{s_1}{d_1} \right\rfloor - \frac{d_1\Delta_1}{p} + \frac{\Delta_2}{p}, \\ g = s_1 \bmod d_1 - \frac{d_1\Delta_2}{p}. \end{gathered} \tag{3}$$

Для нашего примера в силу (3) получаем $s = (d^2 + 2d + 1)/3$.

Теперь для двух рассмотренных точек датасета найдём формулу, которая определит порядки графов возможного семейства.

Имеем $N_1 = 5s_1 + N_1 \bmod s_1 = 5s_1 + 18$, $N_2 = 5s_2 + N_2 \bmod s_2 = 5s_2 + 66$. После деления остатков на диаметры получим $N_1 = 5s_1 + d_1 + 7$, $N_2 = 5s_2 + 3d_2 + 15$.

Введём параметры, которые определят последовательности увеличения коэффициентов при одинаковых степенях d в разложении остатков от деления на d остатка от деления N на s:

$$\begin{gathered} \Delta_3 = \lfloor (N_2 \bmod s_2)/d_2 \rfloor - \lfloor (N_1 \bmod s_1)/d_1 \rfloor, \\ \Delta_4 = (N_2 \bmod s_2) \bmod d_2 - (N_1 \bmod s_1) \bmod d_1. \end{gathered} \tag{4}$$

Для рассматриваемого примера $\Delta_3 = 2$, $\Delta_4 = 8$. Если продолжить эти последовательности, увеличивая диаметр графов на величину p, то общий вид порядков возможного семейства будет следующий:

$$N = \lfloor N_1/s_1 \rfloor s + \left(\lfloor (N_1 \bmod s_1)/d_1 \rfloor + \frac{(d-d_1)\Delta_3}{p} \right) d + (N_1 \bmod s_1) \bmod d_1 + \frac{(d-d_1)\Delta_4}{p}.$$

Подставляя формулу (3) для образующей s и собирая коэффициенты при одинаковых степенях d, получаем

$$\begin{gathered} N = ad^2 + bd + c, \\ a = e\lfloor N_1/s_1 \rfloor + \Delta_3/p, \\ b = f\lfloor N_1/s_1 \rfloor + \lfloor (N_1 \bmod s_1)/d_1 \rfloor - d_1\Delta_3/p + \Delta_4/p, \\ c = g\lfloor N_1/s_1 \rfloor + (N_1 \bmod s_1) \bmod d_1 - d_1\Delta_4/p. \end{gathered} \tag{5}$$

Таким образом, квадратичный полином от диаметра, который определяет порядки оптимальных графов потенциально возможного семейства, сформирован.

Для рассмотренного примера в силу (5) получаем $N = 2d^2 + 2d - 6$ и соответственно следующий вид семейства графов: $\{C(2d^2 + 2d - 6; 1, (d^2 + 2d + 1)/3) : d \equiv 5 \pmod 6\}$. Проверка в датасете показала существование всех графов семейства, начиная с $d = 5$ до 149.

Ниже описана общая схема алгоритма 1 — эвристического алгоритма автоматического поиска семейств оптимальных графов, основанного на рассмотренных выше принципах.

Следует отметить интересную особенность результатов работы алгоритма 1: некоторые из найденных семейств графов имеют образующие линейного вида от d. Это происходит в тех случаях, когда при выполнении необходимого условия (1) также выполняется условие $\lfloor s_1/d_1 \rfloor = \lfloor s_2/d_2 \rfloor$, что даёт $e = 0$. Чтобы автоматизировать процесс поиска семейств оптимальных графов с линейными образующими от диаметра, мы разработали отдельный алгоритм, основанный на аналогичных принципах.

Алгоритм 1. Алгоритм автоматического поиска семейств оптимальных графов с квадратичными образующими (общая схема)

Вход: Точки датасета (N, s, d); P — максимальное значение параметра p.

1: Выбираем начальное значение периода $p \in \{1, 2, 3, \ldots, P\}$ и диаметра $d = d_1$, где $d_1 \equiv 0 \pmod{p}$. Выбираем все точки (N_1, s_1, d_1) в датасете и среди точек вида (N_2, s_2, d_2), где $d_2 = d_1 + p$, выбираем точки (N_2, s_2, d_2), которые удовлетворяют условию (1).

Для точки (N_1, s_1, d_1) и очередной точки (N_2, s_2, d_2) генерируем общие формулы для s и N в виде квадратичных полиномов от диаметра для графов возможного семейства:

$$C(N = ad^2 + bd + c; 1, s = ed^2 + fd + g).$$

Для этого используется свойство повторяемости вида членов семейства через $k = (d - d_1)/p$ шагов, а именно: для коэффициентов e, f, g используются формулы (2) и (3); для a, b, c — формулы (3), (4) и (5). Таким образом, для точек (N_1, s_1, d_1) и (N_2, s_2, d_2) аналитический вид графов возможного семейства сформирован.

2: Проверяем присутствие членов сформированного семейства в датасете, увеличивая диаметр на величину p. Если существование следующих членов семейства подтверждено на выбранной части датасета, то считается, что семейство прошло тестирование и новое семейство графов найдено; оно добавляется в лист потенциальных семейств.

3: Процесс поиска семейств продолжается до тех пор, пока не рассмотрены, во-первых, все имеющиеся паросочетания графов с диаметрами d_1 и d_2 и, во-вторых, диаметры для всех вычетов по модулю p и затем — последовательно весь выделенный диапазон значений параметра p.

4: После этого все найденные семейства проверяются на той части датасета, которая не была включена в предварительно проведённое тестирование на шаге 2. Если результат положительный, то считается, что новое семейство найдено.

Выход: Множество аналитических описаний (формул $N(d)$ и $s(d)$) семейств оптимальных графов с квадратичными образующими, ограниченное рассмотренным диапазоном изменения параметра p.

3. Поиск семейств оптимальных графов с линейными образующими

При анализе полученного датасета было замечено, что линейные образующие с нечётным коэффициентом при d не дают устойчивых оптимальных семейств на больших диапазонах изменения диаметра. Поэтому для поиска семейств оптимальных графов с линейными образующими будем рассматривать образующие вида $s = \gamma d + \alpha$, где γ может принимать только чётные, а α — любые целые значения. Для разработки процесса автоматизации поиска семейств оптимальных графов рассмотрим пример построения семейства оптимальных графов на основании данных, взятых из датасета. Покажем, как по двум точкам датасета можно получить аналитические формулы для порядков графов семейства и образующих линейного вида.

Возьмём в датасете точку $(N_1, s_1, d_1) = (295, 69, 12)$. Пусть $p = 3$ — разность диаметров соседних членов возможного семейства. Найдём в датасете точку (N_2, s_2, d_2), для которой, кроме условия $d_2 = d_1 + p$, выполняется ещё условие

$$\left\lfloor \frac{s_1}{d_1} \right\rfloor = \left\lfloor \frac{s_2}{d_2} \right\rfloor. \tag{6}$$

Например, это точка $(N_2, s_2, d_2) = (459, 87, 15)$. Условие (6) выполнено, значит, коэффициент γ не зависит от диаметра. Найдём целые части и остатки от деления образующих графов на их диаметры, при этом коэффициент при d должен быть чётным, то есть:

$$\begin{gathered} s = fd + g, \\ f = \begin{cases} \lfloor s_1/d_1 \rfloor, & \text{если } \lfloor s_1/d_1 \rfloor \text{ чётное}, \\ \lceil s_1/d_1 \rceil & \text{в противном случае}, \end{cases} \\ g = \begin{cases} s_1 \bmod d_1, & \text{если } \lfloor s_1/d_1 \rfloor \text{ чётное}, \\ s_1 - \lceil s_1/d_1 \rceil\, d_1 & \text{в противном случае}. \end{cases} \end{gathered} \tag{7}$$

Для нашего примера $s_1 = 6d_1 - 3$, $s_2 = 6d_2 - 3$. Таким образом, $f = 6$, $g = -3$ и $s = 6d - 3$.

Введём три параметра, которые определяют последовательности приращения коэффициентов при одинаковых степенях d в разложении N на s и в разложении остатка от деления на d остатка от деления N на s:

$$\begin{gathered} \Delta_1 = \lfloor N_2/s_2 \rfloor - \lfloor N_1/s_1 \rfloor, \\ \Delta_2 = \lfloor (N_2 \bmod s_2)/d_2 \rfloor - \lfloor (N_1 \bmod s_1)/d_1 \rfloor, \\ \Delta_3 = (N_2 \bmod s_2) \bmod d_2 - (N_1 \bmod s_1) \bmod d_1. \end{gathered} \tag{8}$$

Для рассмотренного примера имеем $N_1 = 4s_1 + 19 = 4s_1 + d_1 + 7$, $N_2 = 5s_2 + 24 = 5s_2 + d_2 + 9$. Таким образом, $\Delta_1 = 1$, $\Delta_2 = 0$, $\Delta_3 = 2$. Если продолжить эти последовательности, наращивая диаметр графов на величину p, то общий вид порядков возможного семейства будет равен

$$\begin{gathered} N = (\lfloor N_1/s_1 \rfloor + (d - d_1)\,\Delta_1/p)s + (\lfloor (N_1 \bmod s_1)/d_1 \rfloor + (d - d_1)\,\Delta_2/p)\,d + \\ + (N_1 \bmod s_1) \bmod d_1 + (d - d_1)\,\Delta_3/p. \end{gathered}$$

Подставив формулу (7) для образующей s и собрав коэффициенты при одинаковых степенях d, получим

$$\begin{gathered} N = ad^2 + bd + c, \\ a = f\Delta_1/p + \Delta_2/p, \\ b = f\,\lfloor N_1/s_1 \rfloor + \lfloor (N_1 \bmod s_1)/d_1 \rfloor - fd_1\Delta_1/p + g\Delta_1/p - d_1\Delta_2/p + \Delta_3/p, \\ c = \ g\,\lfloor N_1/s_1 \rfloor + (N_1 \bmod s_1) \bmod d_1 - gd_1\Delta_1/p - d_1\Delta_3/p. \end{gathered} \tag{9}$$

Таким образом, сформирован квадратичный полином от диаметра, задающий порядки оптимальных графов семейства.

Для нашего примера получаем следующий вид возможного семейства графов: $\{C(2d^2 + 2d/3 - 1; 1, 6d - 3) : d \equiv 0 \pmod 3\}$. Проверка в датасете показала существование всех графов семейства с $d = 3$ до 150.

Ниже представлена общая схема алгоритма 2 — эвристического алгоритма автоматического поиска семейств оптимальных двухконтурных кольцевых графов с линейными образующими, основанного на описанных принципах.

Алгоритм 2. Алгоритм автоматического поиска семейств оптимальных графов с линейными образующими (общая схема)

Вход: Точки датасета (N, s, d); P — максимальное значение параметра p.

1: Выбираем начальное значение периода $p \in \{2, 3, 4, \ldots, P\}$ и диаметра $d = d_1$, где $d_1 \equiv 0 \pmod{p}$. Выбираем в датасете все точки (N_1, s_1, d_1) и среди точек вида (N_2, s_2, d_2), где $d_2 = d_1 + p$, выбираем точки, удовлетворяющие условию (6).
Для точки (N_1, s_1, d_1) и очередной точки (N_2, s_2, d_2) создаём общие формулы для s и N в виде полиномов от диаметра для возможного семейства графов:

$$C(N = ad^2 + bd + c; 1, s = fd + g).$$

Для этого используется свойство повторяемости вида членов семейства через $k = (d - d_1)/p$ шагов, а именно: для коэффициентов f, g используются формулы (7), для a, b, c — формулы (8) и (9). Таким образом, для точек (N_1, s_1, d_1) и (N_2, s_2, d_2) аналитический вид графов возможного семейства сформирован.

2: Проверяем наличие членов сформированного семейства в датасете при увеличении диаметра на p. Если на выбранном фрагменте данных существование следующих членов семейства подтверждено, то считается, что семейство прошло тестирование и новое семейство найдено, оно записывается в список новых семейств.

3: Процесс поиска семейств продолжаем до тех пор, пока не рассмотрены, во-первых, все имеющиеся паросочетания графов с диаметрами d_1 и d_2 и, во-вторых, диаметры для всех вычетов по модулю p, затем — последовательно весь выделенный диапазон значений параметра p.

4: После этого происходит валидация полученных семейств, для чего используется часть массива данных, не участвовавшая в тестировании на шаге 2. При положительном результате считается, что новое семейство найдено и его аналитическое описание пополняет список найденных семейств.

Выход: Множество аналитических описаний (формул $N(d)$ и $s(d)$) семейств оптимальных графов с линейными образующими, ограниченное рассмотренным диапазоном изменения параметра p.

4. Экспериментальные результаты реализации алгоритмов

Алгоритм 1 реализован в системе Wolfram Mathematica 10 для $15 \leqslant d \leqslant 150$ и $1 \leqslant p \leqslant 7$. Общее число полученных аналитически описываемых семейств оптимальных графов равно 1944.

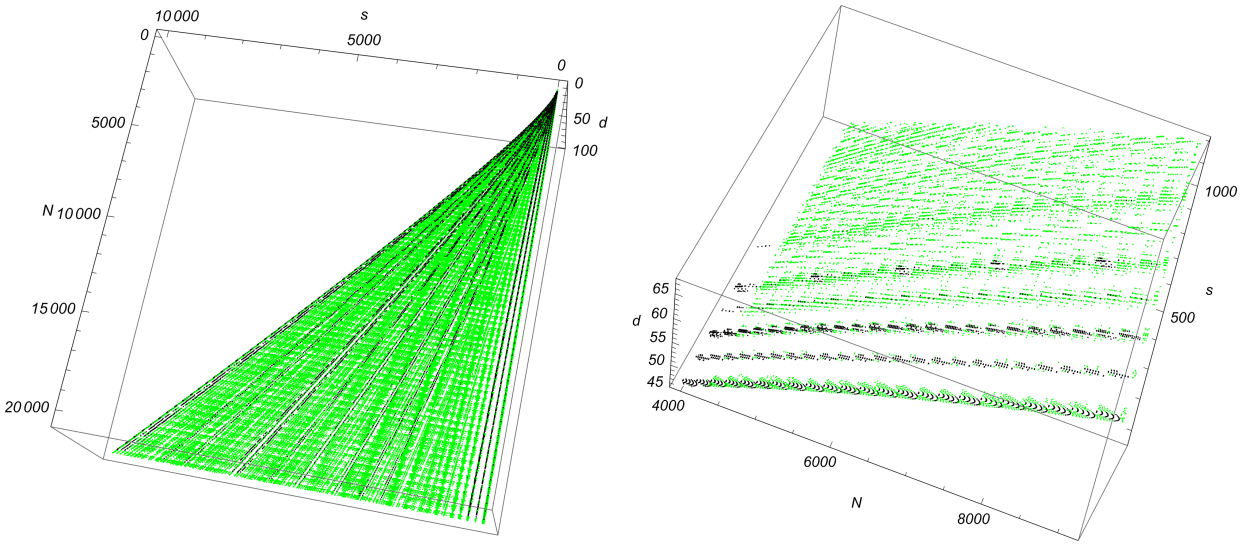
Алгоритм 2 также реализован в системе Wolfram Mathematica 10. Для чётных и нечётных диаметров и значений $2 \leqslant p \leqslant 6$ получено 293 аналитически описываемых семейств оптимальных графов с линейными образующими. В таблице приведён фрагмент описаний оптимальных графов семейств вместе с их периодом повторяемости и типом диаметров, полученных при реализации алгоритмов 1 и 2.

На рис. 4 показаны фрагменты найденных семейств оптимальных двухконтурных кольцевых сетей с квадратичными и линейными образующими. Члены найденных семейств отмечены чёрными точками. Описания всех оптимальных семейств, полученных при реализации алгоритмов, приведены в разделе датасета.

Предложенные алгоритмы автоматического поиска семейств основаны на темплейтах описаний известных оптимальных семейств и проведённом интеллектуальном анализе датасета. Этот подход позволил переоткрыть известные семейства и найти боль-

Примеры описаний семейств оптимальных графов

N	s	p	$d \bmod p$	N	s	p	$d \bmod p$
$d + 2d^2$	$3 + 4d$	5	2	$-5 + 2d^2$	$-3 + 2d$	4	2
$1 + d + 2d^2$	$2d$	5	2	$-5 + 2d^2$	$3 + 2d$	4	2
$-7 + 2d + 2d^2$	$-3 + 2d$	5	2	$-4 + 2d^2$	$-3 + 2d$	4	2
$-7 + 2d + 2d^2$	$5 + 2d$	5	2	$-4 + 2d^2$	$3 + 2d$	4	2
$-1 + 2d + 2d^2$	$-1 + 2d$	5	2	$-3 + 2d^2$	$-3 + 2d$	4	2
$-1 + 2d + 2d^2$	$3 + 2d$	5	2	$-3 + 2d^2$	$3 + 2d$	4	2
$1 + 2d + 2d^2$	$1 + 2d$	5	2	$-3 + 2d^2$	$-5 + 4d$	4	2
$-2 + 2d^2$	$-2 - d + d^2$	5	2	$-1 + 2d^2$	$-1 + 2d$	4	2
$-2 + 2d^2$	$-d + d^2$	5	2	$-1 + 2d^2$	$1 + 2d$	4	2
$2d^2$	$-1 - d + d^2$	5	2	$-1 + 2d^2$	$-3 + 4d$	4	2



а *б*

Рис. 4. Семейства оптимальных графов, полученные с помощью алгоритмов 1 (*а*) и 2 (*б*)

шое количество новых, но он не исчерпывает все возможные виды аналитических выражений для описания семейств. Применение других подходов к анализу датасета, в том числе с использованием моделей глубокого обучения, позволит выявить другие возможные темплейты для описаний оптимальных семейств и алгоритмов их построения и, если это возможно, покрыть сетью аналитики весь построенный датасет в реализованных границах изменения диаметра сетей. Для генерации новых оптимальных семейств с использованием полученных алгоритмов мы предполагаем в дальнейшем также увеличивать область изменения параметра p. Для будущих исследований остаются открытыми следующие вопросы: какие ещё принципы построения семейств оптимальных (субоптимальных) двухконтурных кольцевых графов могут быть реализованы при анализе полученного датасета; как распределяются начальные значения диаметров, при которых порождаются новые семейства; возможный вид функций для параметра p. В частности, для автоматизации поиска оптимальных семейств планируется рассмотреть случаи, когда период повторяемости членов семейства может быть не константой, а функцией линейного вида от другого параметра, как это, например, имеет место для семейства, найденного в [5, следствие 3.6].

Заключение

Предложен новый метод открытия аналитических зависимостей параметров описаний семейств оптимальных двухконтурных кольцевых циркулянтных графов, представляющих практический интерес при моделировании систем связи для сетей на кристалле и кластеров мультипроцессорных систем. При анализе большого датасета параметров оптимальных по диаметру двухконтурных кольцевых графов были замечены некоторые закономерности в их появлении. Чтобы найти новые семейства оптимальных графов, разработаны и реализованы в системе Wolfram Mathematica алгоритмы их автоматизированного поиска в датасете. Найденные с их помощью аналитически описываемые семейства могут составлять базу для конструирования масштабируемых по числу элементов больших многоуровневых вычислительных систем с унификацией по диаметру алгоритмов маршрутизации. Все найденные семейства графов будут представлены в отдельном разделе доступного в Интернете датасета `https://github.com/mila0411/Double-loop-networks/tree/main/Dataset`. Полученный датасет можно также использовать при анализе соотношений между семействами оптимальных графов и их характеристиками, такими, как структурные задержки, определяемые средним расстоянием между узлами, надёжность при отказах элементов сети, пропускная способность, алгоритмы маршрутизации.

ЛИТЕРАТУРА

1. *Bermond J.-C., Comellas F., and Hsu D. F.* Distributed loop computer networks: a survey // J. Parallel Distrib. Comput. 1995. No. 24(1). P. 2–10.
2. *Hwang F. K.* A survey on multi-loop networks // Theoret. Comput. Sci. 2003. V. 299. P. 107–121.
3. *Монахова Э. А.* Структурные и коммуникативные свойства циркулянтных сетей // Прикладная дискретная математика. 2011. № 3. С. 92–115.
4. *Huang X., Ramos A. F., and Deng Y.* Optimal circulant graphs as low-latency network topologies // J. Supercomput. 2022. V. 78. P. 13491–13510.
5. *Chen B.-X., Meng J.-X., and Xiao W.-J.* Some new optimal and suboptimal infinite families of undirected double-loop networks // Discrete Math. Theor. Comput. Sci. 2006. V. 8. P. 299–311.
6. *Romanov A. Y.* Development of routing algorithms in networks-on-chip based on ring circulant topologies // Heliyon. 2019. V. 5. Iss. 4. Article e01516.
7. *Chen B.-X., Xiao W.-J., and Parhami B.* Diameter formulas for a class of undirected double-loop networks // Networks. 2005. V. 6(1). P. 1–15.
8. *Jha K. P.* Tight-optimal circulants vis-a-vis twisted tori // Discrete Appl. Math. 2014. V. 175. P. 24–34.
9. *Huanping L. and Yixian Y.-J.* On the fault-tolerant routing in distributed loop networks // J. Electronics. 2000. V. 17. P. 84–89.
10. *Bian Q.-F., Hang T., Liu H., and Fang M.* Research on the diameter and average diameter of undirected double-loop networks // Int. Conf. Grid Cloud Computing. Nanjang, Jiangsu China, 2010. P. 461–466.
11. *Martinez C., Beivide R., Stafford E., et al.* Modeling toroidal networks with the Gaussian integers // IEEE Trans. Comput. 2008. V. 57. No. 8. P. 1046–1056.
12. *Monakhova E. A., Monakhov O. G., and Romanov A. Yu.* Routing algorithms in optimal degree four circulant networks based on relative addressing: Comparative analysis for networks-on-chip // IEEE Trans. Netw. Sci. Eng. 2023. V. 10. No. 1. P. 413–425.

13. *Perez-Roses H., Bras-Amoros M., and Seradilla-Merinero J. M.* Greedy routing in circulant networks // Graphs Combinatorics. 2022. V. 38. Article 86.
14. *Lewis R. R.* Analysis and Construction of Extremal Circulant and Other Abelian Cayley Graphs. Ph.D. Thesis. The Open University, London, UK, 2022.
15. *Pai K.-J., Yang J.-S., Chen G.-Y., and Chang J.-M.* Configuring protection routing via completely independent spanning trees in dense Gaussian on-chip networks // IEEE Trans. Netw. Sci. Eng. 2022. V. 9. No. 2. P. 932–946.
16. *Monakhov O. G., Monakhova E. A., Romanov A. Y., et al.* Adaptive dynamic shortest path search algorithm in networks-on-chip based on circulant topologies // IEEE Access. 2021. V. 9. P. 160836–160846.
17. *Монахова Э. А.* Об аналитическом описании оптимальных двумерных диофантовых структур однородных вычислительных систем // Вычислительные системы. 1981. № 90. С. 81–91.
18. *Boesch F. and Wang J.-F.* Reliable circulant networks with minimum transmission delay // IEEE Trans. Circuits Syst. 1985. V. 32. No. 12. P. 1286–1291.
19. *Tzvieli D.* Minimal diameter double-loop networks. 1. Large infinite optimal families // Networks. 1991. V. 21. P. 387–415.
20. *Liu H., Li X., and Wang S.* Construction of dual optimal bidirectional double-loop networks for optimal routing // Mathematics. 2022. V. 10. Article 4016.
21. *Monakhova E. A., Romanov A. Y., and Lezhnev E. V.* Shortest path search algorithm in optimal two-dimensional circulant networks: Implementation for networks-on-chip // IEEE Access. 2020. V. 8. P. 215010–215019.
22. *Монахова Э. А.* Синтез оптимальных диофантовых структур // Вычислительные системы. 1979. № 80. С. 18–35.
23. *Du D.-Z., Hsu D. F., Li Q., and Xu J.* A combinatorial problem related to distributed loop networks // Networks. 1990. V. 20. P. 173–180.
24. *Li Y., Chen Y., Tai W., and Wang R.* The minimum distance diagram and diameter of undirected double-loop networks // Proc. ICMEMTC. Taiyuan, China, 2016. P. 1682–1687.
25. *Loudiki L., Kchikech M., and Essaky E. H.* A New Approach for Computing the Distance and the Diameter in Circulant Graphs. `https://arxiv.org/abs/2210.11116`. 2022.
26. *Jha P. K.* Dense bipartite circulants and their routing via rectangular twisted torus // Discrete Appl. Math. 2014. V. 166. P. 141–158.
27. *Jha P. K. and Smith J. D. H.* Cycle Kronecker products that are representable as optimal circulants // Discrete Appl. Math. 2015. V. 181. P. 130–138.
28. *Liu H., Yang Y., and Hu M.* Tight optimal infinite families of undirected double-loop networks // Systems Eng. Theory Practice. 2002. V. 1. P. 75–79.
29. *Jha P. K.* Dimension-order routing algorithms for a family of minimal-diameter circulants // J. Interconn. Networks. 2013. V. 14. No. 1. Article 1350002. 24 p.
30. *Bermond J.-C. and Tzvieli D.* Minimal diameter double-loop networks: Dense optimal families // Networks. 1991. V. 21. P. 1–9.
31. *Chen B.-X., Meng J.-X., and Xiao W.-J.* A constant time optimal routing algorithm for undirected double-loop networks // LNCS. 2005. V. 3794. P. 308–316.
32. *Monakhova E. A.* Optimal circulant computer networks // Proc. PaCT'91. Novosibirsk, USSR, 1991. P. 450–458.
33. *Монахова Э. А., Монахов О. Г.* Эволюционный синтез семейств оптимальных двумерных циркулянтных сетей // Вестник СибГУТИ. 2014. № 2. С. 72–81.
34. *Zerovnik J. and Pisanski T.* Computing the diameter in multiple-loop networks // J. Algorithms. 1993. V. 14. P. 226–243.

35. *Romanov A.* The dataset for optimal circulant topologies // Big Data Cogn. Comput. 2023. V. 7. P. 80.
36. *Monakhova E. A. and Monakhov O. G.* Generation and analysis of optimal double-loop circulant networks dataset // Proc. CEUR Workshop SibDATA. 2023. (to be published)
37. *Storn R. and Price K.* Differential evolution — A simple and efficient heuristic for global optimization over continuous spaces // J. Global Optimization. 1997. V. 11. P. 341–359.
38. *Zaheer H., Pant M., Monakhov O., and Monakhova E.* Simple and efficient co-operative approach for solving multi modal problems // Proc. ICEEOT. Chennai, Tamilnadu, India, 2016. P. 731–737.

REFERENCES

1. *Bermond J.-C., Comellas F., and Hsu D. F.* Distributed loop computer networks: a survey. J. Parallel Distrib. Comput., 1995, no. 24(1), pp. 2–10.
2. *Hwang F. K.* A survey on multi-loop networks. Theoret. Comput. Sci., 2003, vol. 299, pp. 107–121.
3. *Monakhova E. A.* Strukturnye i kommunikativnye svoystva tsirkulyantnykh setey [Structural and communicative properties of circulant networks]. Prikladnaya Diskretnaya Matematika, 2011, no. 3, pp. 92–115. (in Russian)
4. *Huang X., Ramos A. F., and Deng Y.* Optimal circulant graphs as low-latency network topologies. J. Supercomput., 2022, vol. 78, pp. 13491–13510.
5. *Chen B.-X., Meng J.-X., and Xiao W.-J.* Some new optimal and suboptimal infinite families of undirected double-loop networks. Discrete Math. Theor. Comput. Sci., 2006, vol. 8, pp. 299–311.
6. *Romanov A. Y.* Development of routing algorithms in networks-on-chip based on ring circulant topologies. Heliyon, 2019, vol. 5, iss. 4, article e01516.
7. *Chen B.-X., Xiao W.-J., and Parhami B.* Diameter formulas for a class of undirected double-loop networks. Networks, 2005, vol. 6(1), pp. 1–15.
8. *Jha K. P.* Tight-optimal circulants vis-a-vis twisted tori. Discrete Appl. Math., 2014, vol. 175, pp. 24–34.
9. *Huanping L. and Yixian Y.-J.* On the fault-tolerant routing in distributed loop networks. J. Electronics, 2000, vol. 17, pp. 84–89.
10. *Bian Q.-F., Hang T., Liu H., and Fang M.* Research on the diameter and average diameter of undirected double-loop networks. Int. Conf. Grid Cloud Computing, Nanjang, Jiangsu China, 2010, pp. 461–466.
11. *Martinez C., Beivide R., Stafford E., et al.* Modeling toroidal networks with the Gaussian integers. IEEE Trans. Comput., 2008, vol. 57, no. 8, pp. 1046–1056.
12. *Monakhova E. A., Monakhov O. G., and Romanov A. Yu.* Routing algorithms in optimal degree four circulant networks based on relative addressing: Comparative analysis for networks-on-chip. IEEE Trans. Netw. Sci. Eng., 2023, vol. 10, no. 1, pp. 413–425.
13. *Perez-Roses H., Bras-Amoros M., and Seradilla-Merinero J. M.* Greedy routing in circulant networks. Graphs Combinatorics, 2022, vol. 38, article 86.
14. *Lewis R. R.* Analysis and Construction of Extremal Circulant and Other Abelian Cayley Graphs. Ph.D. Thesis, The Open University, London, UK, 2022.
15. *Pai K.-J., Yang J.-S., Chen G.-Y., and Chang J.-M.* Configuring protection routing via completely independent spanning trees in dense Gaussian on-chip networks. IEEE Trans. Netw. Sci. Eng., 2022, vol. 9, no. 2, pp. 932–946.

16. *Monakhov O. G., Monakhova E. A., Romanov A. Y., et al.* Adaptive dynamic shortest path search algorithm in networks-on-chip based on circulant topologies. IEEE Access, 2021, vol. 9, pp. 160836–160846.
17. *Monakhova E. A.* Ob analiticheskom opisanii optimal'nykh dvumernykh diofantovykh struktur odnorodnykh vychislitel'nykh sistem [On analytical representation of optimal two-dimensional Diophantine structures of homogeneous computer systems]. Vychislitel'nye Sistemy, 1981, no. 90, pp. 81–91. (in Russian)
18. *Boesch F. and Wang J.-F.* Reliable circulant networks with minimum transmission delay. IEEE Trans. Circuits Syst., 1985, vol. 32, no. 12, pp. 1286–1291.
19. *Tzvieli D.* Minimal diameter double-loop networks. 1. Large infinite optimal families. Networks, 1991, vol. 21, pp. 387–415.
20. *Liu H., Li X., and Wang S.* Construction of dual optimal bidirectional double-loop networks for optimal routing. Mathematics, 2022, vol. 10, article 4016.
21. *Monakhova E. A., Romanov A. Y., and Lezhnev E. V.* Shortest path search algorithm in optimal two-dimensional circulant networks: Implementation for networks-on-chip. IEEE Access, 2020, vol. 8, pp. 215010–215019.
22. *Monakhova E. A.* Sintez optimal'nykh diofantovykh struktur [Synthesis of optimal Diophantine structures]. Vychislitel'nye Sistemy, 1979, no. 80, pp. 18–35. (in Russian)
23. *Du D.-Z., Hsu D. F., Li Q., and Xu J.* A combinatorial problem related to distributed loop networks. Networks, 1990, vol. 20, pp. 173–180.
24. *Li Y., Chen Y., Tai W., and Wang R.* The minimum distance diagram and diameter of undirected double-loop networks. Proc. ICMEMTC, Taiyuan, China, 2016, pp. 1682–1687.
25. *Loudiki L., Kchikech M., and Essaky E. H.* A New Approach for Computing the Distance and the Diameter in Circulant Graphs. `https://arxiv.org/abs/2210.11116`, 2022.
26. *Jha P. K.* Dense bipartite circulants and their routing via rectangular twisted torus. Discrete Appl. Math., 2014, vol. 166, pp. 141–158.
27. *Jha P. K. and Smith J. D. H.* Cycle Kronecker products that are representable as optimal circulants. Discrete Appl. Math., 2015, vol. 181, pp. 130–138.
28. *Liu H., Yang Y., and Hu M.* Tight optimal infinite families of undirected double-loop networks. Systems Eng. Theory Practice, 2002, vol. 1, pp. 75–79.
29. *Jha P. K.* Dimension-order routing algorithms for a family of minimal-diameter circulants. J. Interconn. Networks, 2013, vol. 14, no. 1, article 1350002, 24 p.
30. *Bermond J.-C. and Tzvieli D.* Minimal diameter double-loop networks: Dense optimal families. Networks, 1991, vol. 21, pp. 1–9.
31. *Chen B.-X., Meng J.-X., and Xiao W.-J.* A constant time optimal routing algorithm for undirected double-loop networks. LNCS, 2005, vol. 3794, pp. 308–316.
32. *Monakhova E. A.* Optimal circulant computer networks. Proc. PaCT'91, Novosibirsk, USSR, 1991, pp. 450–458.
33. *Monakhova E. A. and Monakhov O. G.* Evolyutsionnyy sintez semeystv optimal'nykh dvumernykh tsirkulyantnykh setey [Evolutionary synthesis of families of optimal two-dimensional circulant networks.] Vestnik SibSUTIS, 2014, no. 2, pp. 72–81. (in Russian)
34. *Zerovnik J. and Pisanski T.* Computing the diameter in multiple-loop networks. J. Algorithms, 1993, vol. 14, pp. 226–243.
35. *Romanov A.* The dataset for optimal circulant topologies. Big Data Cogn. Comput., 2023, vol. 7, pp. 80.
36. *Monakhova E. A. and Monakhov O. G.* Generation and analysis of optimal double-loop circulant networks dataset. Proc. CEUR Workshop SibDATA, 2023. (to be published)

37. *Storn R. and Price K.* Differential evolution — A simple and efficient heuristic for global optimization over continuous spaces. J. Global Optimization, 1997, vol. 11, pp. 341–359.
38. *Zaheer H., Pant M., Monakhov O., and Monakhova E.* Simple and efficient co-operative approach for solving multi modal problems. Proc. ICEEOT, Chennai, Tamilnadu, India, 2016, pp. 731–737.

МАТЕМАТИЧЕСКИЕ ОСНОВЫ ИНФОРМАТИКИ И ПРОГРАММИРОВАНИЯ

УДК 510.52 DOI 10.17223/20710410/64/6

О ГЕНЕРИЧЕСКОЙ СЛОЖНОСТИ РЕШЕНИЯ УРАВНЕНИЙ НАД НАТУРАЛЬНЫМИ ЧИСЛАМИ СО СЛОЖЕНИЕМ[1]

А. Н. Рыбалов

Институт математики им. С. Л. Соболева СО РАН, г. Омск, Россия

E-mail: alexander.rybalov@gmail.com

Изучается генерическая сложность проблемы проверки совместности систем уравнений над натуральными числами со сложением. Доказывается NP-полнота этой проблемы, предлагается полиномиальный генерический алгоритм её решения. Доказывается, что при P $\neq$ NP и P = BPP для проблемы проверки совместности систем уравнений над натуральными числами с нулём не существует сильно генерического полиномиального алгоритма. Для сильно генерического полиномиального алгоритма нет эффективного метода случайной генерации входов, на которых этот алгоритм не может решить проблему.

Ключевые слова: *генерическая сложность, линейные уравнения, натуральные числа.*

ON THE GENERIC COMPLEXITY OF SOLVING EQUATIONS OVER NATURAL NUMBERS WITH ADDITION

A. N. Rybalov

Sobolev Institute of Mathematics, Omsk, Russia

We study the general complexity of the problem of determining the solvability of equations systems over natural numbers with the addition. The NP-completeness of this problem is proved. A polynomial generic algorithm for solving this problem is proposed. It is proved that if P $\neq$ NP and P = BPP, then for the problem of checking the solvability of systems of equations over natural numbers with zero there is no strongly generic polynomial algorithm. For a strongly generic polynomial algorithm, there is no efficient method for random generation of inputs on which the algorithm cannot solve the problem. To prove this theorem, we use the method of generic amplification, which allows us to construct generically hard problems from problems that are hard in the classical sense. The main feature of this method is the cloning technique, which combines the input data of a problem into sufficiently large sets of equivalent input data. Equivalence is understood in the sense that the problem is solved similarly for them.

Keywords: *generic complexity, linear equations, natural numbers.*

[1]Работа выполнена в рамках госзадания ИМ СО РАН, проект FWNF-2022-0003.

Введение

Решение уравнений и систем уравнений над вещественными, комплексными, рациональными, целыми числами является классической темой исследований в различных областях математики в течение уже нескольких тысяч лет. Классическая алгебраическая геометрия изучает множества решений алгебраических уравнений над полями вещественных и комплексных чисел. В рамках диофантовой геометрии и диофантова анализа изучаются решения алгебраических уравнений над целыми и рациональными числами. В XX веке большую роль начали играть вычислительные аспекты этих теорий. Изучение алгоритмических проблем, связанных с определением наличия решения у систем уравнений, а также с нахождением и описанием множества решений, является темой многочисленных теоретических и практических исследований.

Как правило, проблема определения совместности систем уравнений над различными алгебраическими системами является либо неразрешимой, либо имеет большую вычислительную сложность. Даже над нетривиальными конечными алгебраическими системами, например над конечными полями, эта проблема оказывается NP-полной. Это означает, что при условии $\mathrm{P} \neq \mathrm{NP}$ для неё не существует полиномиальных алгоритмов. Поэтому актуальным является изучение генерической сложности [1] данных проблем. В рамках генерического подхода алгоритмическая проблема рассматривается не на всём множестве входов, а на некотором подмножестве «почти всех» входов. Такие входы образуют так называемое генерическое множество. Понятие «почти все» формализуется введением асимптотической плотности на множестве входных данных. Результаты об асимптотической плотности совместных уравнений в свободных группах [2] и в нильпотентных группах [3] получены Р. Гилманом, А. Г. Мясниковым и В. А. Романьковым. Генерическая сложность проблемы совместности уравнений над кольцом целых чисел изучалась А. Н. Рыбаловым [4], а над конечными группами, полями и полугруппами — А. Н. Рыбаловым и А. Н. Шевляковым [5]. Уравнения над моноидом натуральных чисел по сложению в рамках универсальной алгебраической геометрии исследовались в работах А. Н. Шевлякова [6, 7]. С. Л. Кривым [8] рассмотрены критерии совместности системы линейных диофантовых уравнений над множеством натуральных чисел, а также даны верхние оценки для компонент минимального множества решений и алгоритмы построения минимальных порождающих наборов решений для всех таких систем.

В данной работе изучается генерическая сложность проблемы проверки совместности систем уравнений над натуральными числами со сложением.

1. Предварительные сведения

Пусть $\mathbb{N} = \{1, 2, 3, \ldots\}$ — множество натуральных чисел, начинающееся с 1, а $\omega = \{0, 1, 2, \ldots\}$ — множество натуральных чисел с нулём. Будем рассматривать две алгебраические системы $\mathfrak{N}_0 = \langle \omega, + \rangle$ и $\mathfrak{N}_1 = \langle \mathbb{N}, + \rangle$. Нетрудно показать, что любую систему уравнений над натуральными числами со сложением можно привести к эквивалентной системе, в которой каждое уравнение является равенством одного из следующих типов:

1) $x_i = x_j + x_k$;
2) $x_i = 1$.

При этом в k-м уравнении системы могут встречаться только переменные x_i, где $i \leqslant 3k$. Под *системой уравнений* будем понимать систему описанного вида. Размер

такой системы — это число уравнений в ней. Обозначим через $\mathcal{S}$ множество всех систем уравнений.

Лемма 1. Число систем уравнений размера n равно

$$|\mathcal{S}_n| = \prod_{k=1}^{n} (27k^3 + 3k).$$

Доказательство. Для t-го уравнения в системе $S \in \mathcal{S}_n$ существует $(3t)^3$ вариантов выбрать уравнение вида $x_i = x_j + x_k$ и $3t$ вариантов выбрать уравнение вида $x_i = 1$. Итого для t-го уравнения есть $27t^3 + 3t$ вариантов; для всей системы из n уравнений имеем $|\mathcal{S}_n| = \prod_{k=1}^{n} (27k^3 + 3k)$ вариантов. ■

Основные определения генерического подхода можно найти в [1] или в [4]. Определения вычислительных классов P, NP и BPP содержатся в [9].

2. NP-полнота

Теорема 1. Проблемы проверки совместности систем уравнений над $\mathfrak{N}_1$ и $\mathfrak{N}_0$ являются NP-полными.

Доказательство. Принадлежность этих проблем к классу NP следует из того, что в качестве подсказки (решения) можно взять значения переменных, которые делают все уравнения системы истинными. То, что размер минимального решения в двоичной записи ограничен полиномиально от размера системы, доказывается, например, в [10].

Докажем, что к проблеме проверки совместности систем уравнений над $\mathfrak{N}_1$ полиномиально сводится известная NP-полная проблема о выполнимости 3-КНФ, которая заключается в следующем.

3-КНФ $\Phi(x_1, \ldots, x_n)$ — это конъюнкция дизъюнкций вида $(\alpha_1 \vee \alpha_2 \vee \alpha_3)$, где α_i, $i = 1, 2, 3$, есть либо булева переменная x_k, либо её отрицание $\overline{x_k}$. Нужно определить, является ли заданная 3-КНФ $\Phi(x_1, \ldots, x_n)$ выполнимой, то есть существуют ли значения $a_1, \ldots, a_n \in \{0, 1\}$, такие, что $\Phi(a_1, \ldots, a_n) = 1$.

Построим по 3-КНФ $\Phi(x_1, \ldots, x_n)$ систему уравнений над $\mathfrak{N}_1$, которая имеет решение тогда и только тогда, когда $\Phi(x_1, \ldots, x_n)$ выполнима. Для этого каждой булевой переменной из Φ сопоставим две натуральнозначные переменные x и y, а также запишем систему уравнений с новыми вспомогательными переменными t_1, t_2, t_3:

$$\begin{cases} t_1 = 1, \\ t_2 = t_1 + t_1, \\ t_3 = t_1 + t_2, \\ t_3 = x + y. \end{cases}$$

Эта система гарантирует, что x и y могут принимать значения только 1 или 2. Число 2 выполняет роль логической единицы (истина), а 1 — логического нуля (ложь). У переменных роли таковы: x моделирует соответствующую логическую переменную, а y — её отрицание.

Далее смоделируем дизъюнкцию $(\alpha_1 \vee \alpha_2 \vee \alpha_3)$. Для этого возьмём переменные z_1, z_2, z_3 над $\mathbb{N}$, соответствующие логическим переменным (или их отрицаниям) из дизъюнкции, и запишем систему

$$\begin{cases} u_1 = 1, \\ u_2 = u_1 + u_1, \\ u_3 = u_1 + u_2, \\ s = u + u_3, \\ r = z_1 + z_2, \\ s = r + z_3, \end{cases}$$

где u, u_1, u_2, u_3, s, r — новые вспомогательные переменные. Заметим, что эта система совместна над $\mathbb{N}$ тогда и только тогда, когда хотя бы одна из переменных z_1, z_2, z_3 равна 2, что соответствует истинности соответствующей дизъюнкции.

Теперь по каждой дизъюнкции 3-КНФ $\Phi(x_1, \ldots, x_n)$ строим подобные системы уравнений. Затем нумеруем все переменные так, чтобы в k-м уравнении системы встречались только переменные x_i, где $i \leqslant 3k$. В итоге получаем систему S_Φ, которая совместна над $\mathbb{N}$ тогда и только тогда, когда 3-КНФ $\Phi(x_1, \ldots, x_n)$ выполнима. Полиномиальность данной сводимости следует из процесса построения.

Аналогично доказывается NP-полнота для $\mathfrak{N}_0$. ■

3. Решение уравнений над $\mathfrak{N}_1$

Теорема 2. Проблема проверки совместности систем уравнений над $\mathfrak{N}_1$ генерически разрешима за полиномиальное время.

Доказательство. Соответствующий генерический полиномиальный алгоритм работает на системе S следующим образом:

1) ищет в системе S уравнение вида $x_i = x_i + x_j$;
2) если такое уравнение найдётся, то система несовместна над $\mathbb{N}$ и алгоритм выдаёт ответ 0;
3) иначе выдаёт ответ «?».

Покажем, что этот алгоритм даёт неопределённый ответ на пренебрежимом множестве систем уравнений. Обозначим через $\mathcal{L}$ семейство систем, в которых отсутствуют уравнения вида $x_i = x_i + x_j$. Число вариантов выбрать k-е уравнение в системе из множества $\mathcal{L}$ равно $27k^3 + 3k - 9k^2$. Поэтому

$$|\mathcal{L}_n| = \prod_{k=1}^{n} (27k^3 + 3k - 9k^2).$$

По лемме 1

$$|\mathcal{S}_n| = \prod_{k=1}^{n} (27k^3 + 3k).$$

Определение асимптотической плотности ρ и ρ_n можно найти в [1, 4]. Получаем

$$\rho_n(\mathcal{L}) = \frac{|\mathcal{L}_n|}{|\mathcal{S}_n|} = \frac{\prod\limits_{k=1}^{n} (27k^3 + 3k - 9k^2)}{\prod\limits_{k=1}^{n} (27k^3 + 3k)} = \prod_{k=1}^{n} \frac{9k^3 + k - 3k^2}{9k^3 + k} = \prod_{k=1}^{n} \Big(1 - \frac{3k^2}{9k^3 + k}\Big) <$$

$$< \prod_{k=1}^{n} \Big(1 - \frac{3k^2}{9k^3 + k^3}\Big) = \prod_{k=1}^{n} \Big(1 - \frac{3}{10k}\Big) < \prod_{k=1}^{n} \Big(1 - \frac{1}{4k}\Big).$$

Чтобы оценить сверху последнее произведение, возведём его в четвёртую степень:

$$\prod_{k=1}^{n}\Big(1-\frac{1}{4k}\Big)^4 < \prod_{k=1}^{n}\bigg(\Big(1-\frac{1}{4k}\Big)\Big(1-\frac{1}{4k+1}\Big)\Big(1-\frac{1}{4k+2}\Big)\Big(1-\frac{1}{4k+3}\Big)\bigg) = \prod_{k=4}^{4n}\Big(1-\frac{1}{k}\Big) =$$
$$= \prod_{k=4}^{4n}\Big(\frac{k-1}{k}\Big) = \frac{3}{4}\cdot\frac{4}{5}\cdot\ldots\cdot\frac{4n-2}{4n-1}\cdot\frac{4n-1}{4n} = \frac{3}{4n}.$$

Итого получаем

$$\rho(\mathcal{L}) = \lim_{n\to\infty}\frac{|\mathcal{L}_n|}{|\mathcal{S}_n|} \leqslant \lim_{n\to\infty}\sqrt[4]{\frac{3}{4n}} = 0.$$

Теорема 2 доказана. ■

4. Решение уравнений над $\mathfrak{N}_0$

Для произвольной системы уравнений $S = \{S_1, \ldots, S_m\}$ рассмотрим множество систем $eq(S)$, которые получаются добавлением к системе S произвольных уравнений $S_{m+1}, \ldots, S_n$, где l-е уравнение имеет вид $x_i = x_j + x_k$, причём $3m < i, j, k < 3(l+m)$. Очевидно, что любая система из $eq(S)$ совместна над $\mathfrak{N}_0$ тогда и только тогда, когда совместна над $\mathfrak{N}_0$ система S.

Лемма 2. Для любой системы S размера m и любого $n > m$ имеет место оценка

$$\frac{|eq(S)_n|}{|\mathcal{S}_n|} > \frac{1}{2(30n^3)^m}.$$

Доказательство. Пусть $n > m$. Для t-го добавленного к S уравнения вида $x_i = x_j + x_k$, где $3m < i, j, k < 3(t+m)$, имеется $(3t)^3 = 27t^3$ вариантов. Поэтому

$$|eq(S)_n| = \prod_{t=1}^{n-m}(27t^3).$$

По лемме 1

$$\rho_n(eq(S)) = \frac{|eq(S)_n|}{|\mathcal{S}_n|} = \frac{\prod\limits_{k=1}^{n-m}(27k^3)}{\prod\limits_{k=1}^{n}(27k^3+3k)} = \prod_{k=1}^{n-m}\Big(\frac{27k^2}{27k^2+3}\Big)\prod_{k=n-m+1}^{n}\frac{1}{27k^3+3k}.$$

Оценим снизу сначала первое произведение:

$$\prod_{k=1}^{n-m}\Big(\frac{27k^2}{27k^2+3}\Big) = \prod_{k=1}^{n-m}\Big(1-\frac{1}{9k^2+1}\Big) > \prod_{k=2}^{n-m+1}\Big(1-\frac{1}{k^2}\Big) = \prod_{k=2}^{n-m+1}\frac{(k-1)(k+1)}{k^2} =$$
$$= \frac{1\cdot 3}{2^2}\cdot\frac{2\cdot 4}{3^2}\cdot\frac{(n-m-1)(n-m+1)}{(n-m)^2}\cdot\frac{(n-m)(n-m+2)}{(n-m+1)^2} = \frac{n-m+2}{2(n-m+1)} > \frac{1}{2}.$$

Теперь оценим второе произведение:

$$\prod_{k=n-m+1}^{n}\frac{1}{27k^3+3k} > \frac{1}{(27n^3+3n)^m}.$$

Итого получаем

$$\rho_n(eq(S)) = \frac{|eq(S)_n|}{|\mathcal{S}_n|} > \frac{1}{2(27n^3+3n)^m} > \frac{1}{2(30n^3)^m}.$$

Лемма 2 доказана. ■

Теорема 3. Пусть P $\neq$ NP и P = BPP. Тогда для проблемы проверки совместности систем уравнений над $\mathfrak{N}_0$ не существует сильно генерического полиномиального алгоритма.

Доказательство. Допустим, что существует сильно генерический полиномиальный алгоритм $\mathcal{A}$, определяющий совместность систем уравнений над $\mathfrak{N}_0$. Построим вероятностный полиномиальный алгоритм $\mathcal{B}$, решающий эту проблему на всём множестве входов. На системе S размера n алгоритм $\mathcal{B}$ работает следующим образом:

1) генерирует случайно равновероятно систему S' из множества $eq(S)$ размера n^2;
2) запускает алгоритм $\mathcal{A}$ на системе S';
3) если $\mathcal{A}(S') \neq ?$, то алгоритм правильно определяет, совместна ли система S', а вместе с ней и система S;
4) если $\mathcal{A}(S') = ?$, то выдаёт ответ «НЕТ».

Заметим, что полиномиальный вероятностный алгоритм $\mathcal{B}$ выдаёт правильный ответ на шаге 3, а на шаге 4 может выдать неправильный ответ. Надо доказать, что вероятность того, что ответ выдаётся на шаге 4, меньше 1/3.

Оценим вероятность выдачи ответа на шаге 4. Вероятность того, что для S' имеет место $\mathcal{A}(S') = ?$, не больше

$$\frac{|\{S' \in \mathcal{S} : \mathcal{A}(S') = ?\}_{n^2}|}{|eq(S)_{n^2}|} = \frac{|\{S' \in \mathcal{S} : \mathcal{A}(S') = ?\}_{n^2}|}{|\mathcal{S}_{n^2}|} \cdot \frac{|\mathcal{S}_{n^2}|}{|eq(S)_{n^2}|}.$$

Так как множество $\{S' \in \mathcal{S} : \mathcal{A}(S') = ?\}$ сильно пренебрежимое, существует константа $\alpha > 0$, такая, что

$$\frac{|\{S' \in \mathcal{S} : \mathcal{A}(S') = ?\}_{n^2}|}{|\mathcal{S}_{n^2}|} < \frac{1}{2^{\alpha n^2}}$$

для любого n. По лемме 2

$$\frac{|\mathcal{S}_{n^2}|}{|eq(S)_{n^2}|} < 2(30n^6)^n.$$

Поэтому искомая вероятность ответа на шаге 4 не больше

$$\frac{2(30n^6)^n}{2^{\alpha n^2}} \cdot \frac{2^{1+\log 30n+6n\log n}}{2^{\alpha n^2}} = \frac{1}{2^{\alpha n^2 - 6n\log n - \log 30n - 1}} < \frac{1}{3}$$

при больших n.

Таким образом, проблема проверки совместности систем уравнений над $\mathfrak{N}_0$ принадлежит классу BPP. А так как BPP = P, то она принадлежит классу P. А это, по теореме 1, противоречит условию P $\neq$ NP. ■

Автор выражает искреннюю благодарность рецензенту за полезные замечания и предложения по улучшению текста статьи.

ЛИТЕРАТУРА

1. *Kapovich I., Miasnikov A., Schupp P., and Shpilrain V.* Generic-case complexity, decision problems in group theory and random walks // J. Algebra. 2003. V. 264. No. 2. P. 665–694.
2. *Gilman R. H., Myasnikov A., and Roman'kov V.* Random equations in free groups // Groups Complexity Cryptology. 2011. V. 3. No. 2. P. 257–284.
3. *Gilman R. H., Myasnikov A., and Roman'kov V.* Random equations in nilpotent groups // J. Algebra. 2012. V. 352. No. 1. P. 192–214.
4. *Rybalov A.* Generic complexity of the Diophantine problem // Groups Complexity Cryptology. 2013. V. 5. No. 1. P. 25–30.

5. *Rybalov A. and Shevlyakov A.* Generic complexity of solving of equations in finite groups, semigroups and fields // J. Physics: Conf. Ser. 2021. V. 1901. Article 012047. 8 p.
6. *Shevlyakov A.* Algebraic geometry over the additive monoid of natural numbers: The classifcation of coordinate monoids // Groups Complexity Cryptology. 2010. V. 2. No. 1. P. 91–111.
7. *Шевляков А. Н.* Алгебраическая геометрия над моноидом натуральных чисел. Неприводимые алгебраические множества // Труды Института математики и механики УрО РАН. 2010. Т. 16. № 2. С. 258–269.
8. *Kryvyi S. L.* Compatibility of systems of linear constraints over the set of natural numbers // Cybernetics Systems Analysis. 2002. V. 38. No. 1. P. 17–29.
9. *Китаев А., Шень А., Вялый М.* Классические и квантовые вычисления. М.: МЦНМО, ЧеРо, 1999. 192 с.
10. *Схрейвер А.* Теория линейного и целочисленного программирования. М.: Мир, 1991. 360 с.

REFERENCES

1. *Kapovich I., Miasnikov A., Schupp P., and Shpilrain V.* Generic-case complexity, decision problems in group theory and random walks. J. Algebra, 2003, vol. 264, no. 2, pp. 665–694.
2. *Gilman R. H., Myasnikov A., and Roman'kov V.* Random equations in free groups. Groups Complexity Cryptology, 2011, vol. 3, no. 2, pp. 257–284.
3. *Gilman R. H., Myasnikov A., and Roman'kov V.* Random equations in nilpotent groups. J. Algebra, 2012, vol. 352, no. 1, pp. 192–214.
4. *Rybalov A.* Generic complexity of the Diophantine problem. Groups Complexity Cryptology, 2013, vol. 5, no. 1, pp. 25–30.
5. *Rybalov A. and Shevlyakov A.* Generic complexity of solving of equations in finite groups, semigroups and fields. J. Physics: Conf. Ser., 2021, vol. 1901, Article 012047, 8 p.
6. *Shevlyakov A.* Algebraic geometry over the additive monoid of natural numbers: The classifcation of coordinate monoids. Groups Complexity Cryptology, 2010, vol. 2, no. 1, pp. 91–111.
7. *Shevlyakov A. N.* Algebraicheskaya geometriya nad monoidom natural'nykh chisel. Neprivodimye algebraicheskie mnozhestva [Algebraic geometry over the monoid of natural numbers. Irreducible algebraic sets]. Trudy Instituta Matematiki i Mekhaniki UrO RAN, 2010, vol. 16, no. 2, pp. 258–269. (in Russian)
8. *Kryvyi S. L.* Compatibility of systems of linear constraints over the set of natural numbers. Cybernetics Systems Analysis, 2002, vol. 38, no. 1, pp. 17–29.
9. *Kitaev A., Shen' A., and Vyalyy M.* Klassicheskie i kvantovye vychisleniya [Classical and Quantum Computations]. Moscow, MCCME Publ., 1999. 192 p. (in Russian)
10. *Schrijver A.* Theory of Linear and Integer Programming. Wiley, 1998. 484 p.

ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ В ДИСКРЕТНОЙ МАТЕМАТИКЕ

УДК 519.651 DOI 10.17223/20710410/64/7

УПРОЩЁННАЯ ФОРМУЛА СУММИРОВАНИЯ ДИСКРЕТНЫХ ЗНАЧЕНИЙ НЕКОТОРЫХ ФУНКЦИЙ

В.Р. Осипов

г. Москва, Россия

E-mail: osvktr@bk.ru

Получен упрощённый вариант формулы суммирования Эйлера — Маклорена

$$\sum_{k=0}^{m} f(a+kh) = \frac{1}{h}\int_{y_0}^{y_m} f(y)\,dy - \sum_{k} h^{2k-1} b_k \big(f^{(2k-1)}(y_m) - f^{(2k-1)}(y_0)\big),$$

где $y_0 = a - h/2$; $y_m = a + (m+1/2)h$. Формула включает в себя интегральную оценку суммы дискретных отсчётов функции и поправку к ней в виде суммы ряда весовых граничных значений её нечётных производных. Упрощением является исключение из результата суммирования полусуммы граничных значений функции и достигается путём смещения hr отсчётов внутрь отрезков интегрирования. Доказывается, что оптимальным является смещение каждого отсчёта в середину отрезка $r = 1/2$. Это смещение задаёт пределы интегральной оценки y_0, y_m и значения весовых коэффициентов производных поправочного ряда. Найдено аналитическое выражение этих коэффициентов и их производящая функция

$$b_k = \frac{1-2^{1-2k}}{(2k)!}B_{2k},\ \Psi_b(t) = 1 - \frac{t}{2}\operatorname{cosech}\frac{t}{2} = \sum_{k=1}^{\infty} b_k t^{2k},$$

где B_{2k} — числа Бернулли. На примерах получения точных выражений сумм $\sum\limits_{k=1}^{m} k^n$, $\sum\limits_{k=k_0}^{m} a^{hk}$, где m, n — целые положительные числа, показана справедливость полученной формулы и производящей функции её коэффициентов. Формула была использована для получения приближённых выражений для дзета-функции Римана, пси-функции, полигамма функций, а также сумм бесконечных обратностепенных рядов и гармонического ряда. На основании анализа погрешности этих выражений показаны преимущества упрощённой формулы перед формулой Эйлера — Маклорена в точности и краткости.

Ключевые слова: *сумма, ряд, коэффициент, производящая функция, число Бернулли, поправка, погрешность.*

SIMPLIFIED FORMULA FOR SUMMING DISCRETE VALUES OF SOME FUNCTIONS

V. R. Osipov

Moscow, Russia

The simplified variant of Euler — Maclaurin summation formula is obtained:

$$\sum_{k=0}^{m} f(a+kh) = \frac{1}{h}\int_{y_0}^{y_m} f(y)\,dy - \sum_k h^{2k-1} b_k \big(f^{(2k-1)}(y_m) - f^{(2k-1)}(y_0)\big),$$

where $y_0 = a - h/2$, $y_m = a + (m+1/2)h$. The formula includes an integrated estimation of the sum of function discrete samples and the correction to it in the form of the series sum of weight boundary values of its odd derivatives. Simplification is the exception a half-sum of boundary function values from summing result and is reached by shiftihg hr samples inside the integrated segments. The optimal value of this shift for each sample to the middle of the segment $r = 1/2$ is proven. This shift specifies integrated estimation limits y_0, y_m, and values of correction series weighting coefficients. The analytical expression of these coefficients and their generating function have been found: $b_k = \dfrac{1-2^{1-2k}}{(2k)!} B_{2k}$, $\Psi_b(t) = 1 - \dfrac{t}{2}\operatorname{cosech}\dfrac{t}{2} = \sum\limits_{k=1}^{\infty} b_k t^{2k}$, where B_{2k} are Bernoulli numbers. Using examples of obtaining exact expressions for the sums $\sum\limits_{k=1}^{m} k^n$, $\sum\limits_{k=k_0}^{m} a^{hk}$, where $m, n \in \mathbb{N}$, the validity of the resulting formula and the generating function of its coefficients is shown. The formula was used to obtain approximate expressions for the Riemann zeta function, psi function, polygamma function, as well as sums of infinite inverse power series and harmonic series. Based on an analysis of the error of these expressions, the advantages of the simplified formula over the Euler — Maclaurin formula in terms of accuracy and brevity are shown.

Keywords: *sum, row, coefficient, generating function, Bernoulli's number, correction, error.*

Введение

Решения задач прикладной математики и асимптотические оценки некоторых специальных функций иногда сводятся к результату в виде суммы параметрического функционального ряда

$$S_m(a,h) = \sum_{k=0}^{m} f(a+kh), \tag{1}$$

где $f(y)$ — функция члена ряда, суммируемая в узлах равномерной сетки.

Сумму (1) можно оценить определённым интегралом от функционального члена в пределах значений k при условии его существования. Фундаментальной основой такого способа является формула суммирования Эйлера — Маклорена. Приведём вариант этой формулы из авторитетного справочника [1, (23.1.30)]:

$$\begin{aligned}\sum_{k=0}^{m} F(a+kh) = \frac{1}{h}\int_a^b F(t)dt + \frac{1}{2}(F(a)+F(b)) + \sum_{k=1}^{n-1}\frac{h^{2k-1}}{(2k)!}B_{2k}\big(F^{(2k-1)}(b) - F^{(2k-1)}(a)\big) + \\ \ldots + \frac{h^{2n}}{(2n)!}B_{2n}\sum_{k=k_0}^{m-1} F^{(2n)}(a+kh+\theta h), \qquad h = \frac{b-a}{m},\end{aligned} \tag{2}$$

где B_{2k}, B_{2n} — числа Бернулли.

Формула (2) связывает результат суммирования $(m+1)$ значений (отсчётов) функции с интегралом от этой функции в заданных пределах и содержит, кроме него, полусумму крайних значений и поправочный ряд с остаточным n-м членом.

Целью работы является уточнение интегральной оценки и упрощение формулы суммирования за счёт избавления её от потенциально сравнимого с конечным результатом значения полусуммы.

1. Вывод упрощённой формулы суммирования

Для монотонных на интервале функций поставленную цель можно достигнуть, используя теорему о среднем значении на основе её утверждения о равенстве определённого интеграла на этом интервале значению функции в его промежуточной точке, умноженной на длину отрезка. При этом для разных интервалов смещение промежуточной точки относительно границ будет неодинаковым, зависящим от самой функции. Для одинакового значения этого положения внутри всех интервалов его следует оптимизировать с целью минимизации погрешности интегральной оценки.

1.1. Оптимизация положения функциональных отсчетов внутри интервалов

Представим непрерывную функцию ряда (1) как функцию дискретных значений (отсчётов) целочисленного аргумента, помещённых внутрь $(m - k_0 + 1)$ единичных интервалов. Представим результат суммирования этих отсчётов в виде разности интегральной оценки по всем интервалам и суммы дискретных поправок для каждого единичного интервала:

$$\sum_{k=k_0}^{m} f(k) = \int_{k_0-r}^{m+1-r} f(k)\,dk - \sum_{k=k_0}^{m} \big(\varphi(k+1-r) - \varphi(k-r)\big), \tag{3}$$

где $\varphi(y)$ — функция поправки; $r < 1$ — одинаковое для всех интервалов смещение от их нижних границ.

Найдём значение дискретной поправки к интегральной оценке на единичном интервале:

$$\Delta\varphi(k,r) = \varphi(k+1-r) - \varphi(k-r) = \int_{k-r}^{k+1-r} f(k)\,dk - f(k). \tag{4}$$

Потребуем от подынтегральной функции, чтобы она была аналитической в пределах каждого из единичных интервалов, и представим её в виде разложения в степенной ряд Маклорена с радиусом сходимости $r < 1$:

$$f(k+t) = f(k) + \sum_{n=1}^{\infty} \frac{f^{(n)}(k)}{n!} t^n.$$

Подставим это разложение в (4) и получим выражение для поправки в виде ряда

$$\Delta\varphi(k,r) = \int_{k-r}^{k+1-r} \left(f(k) + \sum_{n=1}^{\infty} \frac{f^{(n)}(k)}{n!} t^n \right) dt - f(k) = \sum_{n=1}^{\infty} \frac{f^{(n)}(k)}{(n+1)!} \big((1-r)^{n+1} - (-r)^{n+1}\big). \tag{5}$$

Проанализируем значение общего множителя этих поправок $\theta_n(r) = (1-r)^{n+1} - (-r)^{n+1}$, для чего исследуем производную квадрата этой зависимости:

$$\big(\theta_n(r)^2\big)' = 2(n+1)\left((1-r)^{n+1} - (-r)^{n+1}\right)\left((1-r)^n - (-r)^n\right).$$

Легко видеть, что при $r = 1/2$ эта производная для любого n равна 0, что соответствует минимуму значения зависимости. При этом $\theta_n(1/2) = 0$ для всех нечётных n, $\theta_n(1/2) = (1/2)^n$ для всех чётных n, а при $r = 0$ или 1 для любых n имеет максимальное значение $|\theta_n(r)| = 1$. Таким образом, значение $r = 1/2$ обеспечивает минимизацию общего множителя, а также способствует сокращению числа членов разложения (5) за счёт исключения из него членов с нечётными индексами и обеспечивает сокращение радиуса его сходимости. С этой точки зрения, полагая оптимальным положение дискретного отсчёта $r = 1/2$ в середине единичного интервала, подставим это значение в (5) и произведя замену $n \to 2n$, получим

$$\Delta\varphi(k, 1/2) = \varphi(k + 1/2) - \varphi(k - 1/2) = \sum_{n=1}^{\infty} \frac{f^{(2n)}(k)}{2^{2n}(2n + 1)!}. \tag{6}$$

1.2. Определение аналитического выражения функции поправки

Очевидно, что функция поправки $\varphi(y)$ должна быть связана с суммируемой функцией $f(y)$. Представим эту связь в виде функционального ряда

$$\varphi(y) = \sum_{l=0}^{\infty} b_l f^{(l)}(y), \tag{7}$$

где b_l — весовые коэффициенты производных $f^{(l)}(y)$. Тогда функциональное выражение для дискретной поправки примет вид

$$\varphi(k + 1/2) - \varphi(k - 1/2) = \sum_{l=0}^{\infty} b_l \big(f^{(l)}(k + 1/2) - f^{(l)}(k - 1/2)\big). \tag{8}$$

Представим граничные значения производных в (8) в виде степенного ряда, используя формулы производных функции $f(y)$:

$$f^{(l)}(k \pm 1/2) = \sum_{q=0}^{\infty} \frac{f^{(l+q)}(k)}{q!} (\pm 1/2)^q.$$

Подставим эти разложения в (8) и приравняем результат подстановки к правой части (6):

$$\sum_{l=0}^{\infty} b_l \sum_{q=0}^{\infty} \frac{f^{(l+q)}(k)}{q!} \left((1/2)^q - (-1/2)^q\right) = \sum_{n=1}^{\infty} \frac{f^{(2n)}(k)}{2^{2n}(2n + 1)!}. \tag{9}$$

Очевидно, что условием независимости коэффициентов b_l от значений производных функции является индексное равенство $l + q = 2n$. Поскольку $n \geqslant 1$, то $l + q \geqslant 2$. При этом выражение $(1/2)^q - (-1/2)^q = 2/2^q$ имеет ненулевое значение только для нечётных q, следовательно, индекс l также должен быть нечётным, а функция поправки (7) может быть представлена, как и в формуле Эйлера — Маклорена, функциональным рядом только с нечётными производными:

$$\varphi(y) = \sum_{l=1}^{\infty} b_l f^{(2l-1)}(y). \tag{10}$$

Составим для преобразованных индексов новое уравнение $2q - 1 + 2l - 1 = 2n$. Решив его относительно q, получим условие $q = n - l + 1$, ограничивающее число значений

индекса l числом n. Подставим преобразованные значения индексов в (9) и приравняем члены рядов правой и левой части с одинаковым индексом n:

$$\sum_{l=1}^{n} b_l \frac{f^{(2n)}(k)(1/2)^{2n-2l}}{(2n-2l+1)!} = \frac{f^{(2n)}(k)}{2^{2n}(2n+1)!}.$$

Сокращая в правой и левой части отношение $f^{(2n)}(k)/2^{2n}$, получаем независимую от суммируемой функции систему уравнений

$$\sum_{l=1}^{n} \frac{c_l}{(2n-2l+1)!} = \frac{1}{(2n+1)!}, \tag{11}$$

где $c_l = 2^{2l} b_l$. Последовательно задавая значения $l = 1, 2, 3\ldots$, вычисляем $b_l = \dfrac{1}{24}$, $-\dfrac{7}{5760}$, $\dfrac{31}{967680}$, $\ldots$

Для получения общего выражения для коэффициентов b_l найдём производящие функции коэффициентов степенных рядов правой и левой части (11). Для коэффициентов правой части используем разложение $\operatorname{sh} t = \sum\limits_{n=1}^{\infty} \dfrac{t^{2n-1}}{(2n-1)!}$. Вычитаем из него первый член, в результате имеем

$$\operatorname{sh} t - t = \sum_{n=1}^{\infty} \frac{t^{2n+1}}{(2n+1)!}. \tag{12}$$

Производящую функцию левой части, ввиду присутствия в ней двух индексов, представим в виде произведения функций $\eta(t)\Psi_c(t)$ с нулевым значением при $t = 0$ и приравняем результат произведения их разложений в степенные ряды ряду (12):

$$\sum_{q=1}^{\infty} \frac{\eta^{(q)}(0)}{q!} t^q \sum_{l=1}^{\infty} \Psi_c^{(l)}(0)\, t^l = \sum_{q=1}^{\infty} \sum_{l=1}^{\infty} \Psi_c^{(l)}(0) \frac{\eta^{(q)}(0)}{q!}\, t^{q+l} = \sum_{n=1}^{\infty} \frac{t^{2n+1}}{(2n+1)!}. \tag{13}$$

Сгруппируем члены ряда левой части (13) со степенью t в соответствии с уравнением $q + l = 2n + 1$. Решив уравнение относительно q, получим $q = 2n - l + 1$. Приравняем члены рядов правой и левой части (13) с одинаковой степенью t. Сократив в них t^{2n+1}, получим

$$\sum_{l=1}^{2n} \Psi_c^{(l)}(0) \frac{\eta^{(2n-l+1)}(0)}{(2n-l+1)!} = \frac{1}{(2n+1)!}.$$

Очевидно, что условиями идентичности этого выражения с (11) являются чётное значение индекса l, независимость от индекса n значения производных $\eta^{(2n-2l+1)}(0) = \mathrm{C}$ и равенство $\Psi_c^{(2l)}(0) = c_l/\mathrm{C}$, где C — константа. Таким образом, индекс q должен быть нечётным числом и разложение функции $\eta(t)$ является рядом Маклорена из нечётных степеней t с постоянным значением при $t = 0$ и нулевых значениях чётных производных. Этим требованиям соответствует разложение $\operatorname{sh} t$, в котором $\mathrm{C} = 1$. Отсюда следует равенство $\Psi_c^{(2l)}(0) = c_l$, и из (12) и (13) получаем уравнение $\operatorname{sh} t\, \Psi_c(t) = \operatorname{sh} t - t$, решив которое, получим производящую функцию для коэффициентов c_l из (11):

$$\Psi_c(t) = \sum_{l=1}^{\infty} c_l t^{2l} = 1 - t \operatorname{cosech} t. \tag{14}$$

Далее, воспользовавшись разложением в степенной ряд [1, (4.5.65)]

$$\operatorname{cosech} z = \frac{1}{z} - \frac{z}{6} + \frac{7}{360}z^3 - \frac{31}{15120}z^5 + \ldots - \frac{2(2^{2n-1}-1)B_{2n}}{(2n)!}z^{2n-1} + \ldots \quad (|z| < \pi),$$

где B_{2n} — числа Бернулли, получим из (14)

$$\Psi_c(t) = 1 - t\left[\frac{1}{t} - \frac{t}{6} + \frac{7}{360}t^3 - \ldots - \frac{2(2^{2l-1}-1)B_{2l}}{(2l)!}t^{2l-1} + \ldots\right] = \sum_{l=1}^{\infty}\frac{2(2^{2l-1}-1)B_{2l}}{(2l)!}t^{2l}.$$

Очевидно, что общий коэффициент этого ряда является коэффициентом c_l разложения в степенной ряд (14). Подставив его выражение в соотношение между коэффициентами из (11), получим

$$b_l = 2^{-2l}c_l = 2^{-2l}\frac{2(2^{2l-1}-1)B_{2l}}{(2l)!} = \frac{1-2^{1-2l}}{(2l)!}B_{2l}, \tag{15}$$

или, более кратко, на основании соотношения $B_n(1/2) = -(1-2^{1-n})B_n$ [1, (23.1.21)]

$$b_l = -\frac{B_{2l}(1/2)}{(2l)!}.$$

Для достаточно больших $l > 3$ формулу (15) можно упростить, используя ограничения для чисел Бернулли [1, (23.1.15)]:

$$\frac{2(2n)!}{(2\pi)^{2n}}\frac{1}{1-2^{1-2n}} > (-1)^{n+1}B_{2n} > \frac{2(2n)!}{(2\pi)^{2n}}.$$

Согласно соотношению между b_l и B_{2l} в (15), изменив обозначение l на n, умножим обе части этого неравенства на $\dfrac{1-2^{1-2n}}{(2n)!}$. В результате получим

$$\frac{2}{(2\pi)^{2n}} > (-1)^{n+1}b_n > \frac{2}{(2\pi)^{2n}}(1-2^{1-2n}).$$

Отсюда для достаточно больших n получим приблизительное равенство

$$b_n \simeq \frac{2(-1)^{n+1}}{(2\pi)^{2n}}. \tag{16}$$

Из соотношения между коэффициентами c_l и b_l нетрудно получить производящую функцию для коэффициентов b_l. Для этого воспользуемся преобразованием

$$\Psi_b(t) = \sum_{l=1}^{\infty}b_l t^{2l} = \sum_{l=1}^{\infty}\frac{c_l}{2^{2l}}t^{2l} = \sum_{l=1}^{\infty}c_l\left(\frac{t}{2}\right)^{2l} = \Psi_c\left(\frac{t}{2}\right).$$

Отсюда из (14) окончательно получаем

$$\Psi_b(t) = \sum_{l=1}^{\infty}b_l t^{2l} = 1 - \frac{t}{2}\operatorname{cosech}\frac{t}{2} = 1 - \frac{t\,\mathrm{e}^{t/2}}{\mathrm{e}^t - 1}. \tag{17}$$

Подставив в (10) выражение для весовых коэффициентов b_l (15), получим аналитическое выражение для функции поправки:

$$\varphi(y) = \sum_{l=1}^{\infty}\frac{1-2^{1-2l}}{(2l)!}B_{2l}f^{(2l-1)}(y). \tag{18}$$

Представление функции поправки рядом из нечётных производных не противоречит представлению дискретной поправки (6) рядом высших чётных производных, так как последняя является разностью между граничными значениями функции поправки в единичном интервале.

Просуммируем дискретные поправки в (3), учитывая взаимное уничтожение их крайних значений на границе соседних интервалов $-\varphi((k+1)-1/2)+\varphi(k+1/2)=0$, и получим поправку к интегральной оценке в виде разности граничных значений функции $\varphi(y)$:

$$\sum_{k=k_0}^{m}\left(\varphi(k+1/2)-\varphi(k-1/2)\right)=\varphi(m+1/2)-\varphi(k_0-1/2).$$

1.3. Аналитическое выражение упрощённой формулы суммирования

Результатом оптимизации положения дискретных отсчётов суммируемой функции в середине единичных интервалов $r=1/2$ является определение как коэффициентов (15), так и границ интегральной оценки. Используя это значение и аналитическое выражение функции поправки (18), приведём формулу суммирования дискретных значений функций от целочисленного аргумента (3) к виду

$$\sum_{k=k_0}^{m} f(k)=\int_{k_0-1/2}^{m+1/2} f(y)dy-\sum_{k}\frac{1-2^{1-2k}}{(2k)!}B_{2k}\big(f^{(2k-1)}(m+1/2)-f^{(2k-1)}(k_0-1/2)\big), \quad (19)$$

где $k=1,2,3,\ldots$; B_{2k} — числа Бернулли.

Общее выражение для формулы суммирования, сходное по виду с формулой Эйлера — Маклорена (2), можно получить, раскрывая параметрическую зависимость аргумента $y_k=a+kh$, где $a=a_0+k_0h$:

$$\sum_{k=0}^{m} f(a+kh)=\frac{1}{h}\int_{a-h/2}^{a+mh+h/2} f(y)\,dy-\ldots$$
$$-\sum_{k}^{n-1} h^{2k-1}\frac{1-2^{1-2k}}{(2k)!}B_{2k}\left(f^{(2k-1)}(a+m\,h+h/2)-y^{(2k-1)}(a-h/2)\right)+R_n. \quad (20)$$

Для монотонных функций, поскольку все производные нечётного порядка $f^{(2k-1)}(y)$ не меняют знак, поправочный ряд в формулах (2) и (20) является знакопеременным, так как числа Бернулли поочерёдно меняют знак в соответствии с равенством $B_{2k}=(-1)^{k-1}|B_{2k}|$ [2, (9.611)]. Если при этом последний член этих рядов при неограниченном росте его индекса стремится к нулю, то, согласно признаку сходимости Лейбница знакопеременного ряда с монотонно убывающими членами, абсолютное значение остатка его суммирования не превышает абсолютного значения первого отбрасываемого в (20) члена $|R_n|\leqslant|A_n|$ [2, (0.227)], где $A_n=h^{2n-1}b_n\left(f^{(2n-1)}(a+m\,h+h/2)-y^{(2n-1)}(a-h/2)\right)$. В остальных случаях без исследования остатка суммирования поправочного ряда R_n знак равенства в формулах (19) и (20) является условным. Однако если производные а) $f^{(2n+2)}(y)$ и б) $f^{(2n+4)}(y)$ не меняют знак на всём интервале интегрирования, то, согласно свойствам поправочного ряда формулы Эйлера — Маклорена, $|R_n|=\theta|A_n|$, где $\theta<2$ в случае (а) и $\theta<1$ в случае (а+б) [1, примечание к формуле (25.4.7); 3, формула (21*), с. 542–543]. Это свойство можно применить в упрощённых формулах (19) и (20), используя сходство последней

с (2), а также мажорирующее значение коэффициентов в (2) к коэффициентам b_k в (20). Их отношение $1/(1-2^{1-2k})$ больше 1 и стремится к 1 при $k \to \infty$. При этом первый коэффициент поправочного ряда в (20) вдвое меньше первого коэффициента формулы (2), что может быть существенным для приближённых оценок.

Таким образом, при сформулированных ограничениях для функции остаток суммирования поправочного ряда в (20) можно оценить выражением

$$|R_n| = \theta \left| b_n h^{2n-1} \left(f^{(2n-1)}(y_m) - f^{(2n-1)}(y_0) \right) \right|, \tag{21}$$

где b_n — коэффициент поправочного ряда (15); $\theta < 2$ при однозначности производной $f^{(2n+2)}(y)$ на всём интервале $[y_0, y_m]$ и $\theta < 1$ при дополнительной однозначности производной $f^{(2n+4)}(y)$.

Замечание 1. В случае равенств $f(k_0) = 0$ в (19) или $f(a) = 0$ в (20), не влияющих на значения итоговых сумм, использование приведённых в формулах нижних границ приведёт к ошибке. Для определения границ упрощённой формулы следует использовать только крайние значащие отсчёты суммируемой функции.

2. Применение упрощённой формулы для решения прикладных задач

Подтвердим справедливость полученной формулы и получим на её основе новые приближённые формулы.

2.1. Получение точных аналитических выражений

Найденную упрощённую формулу можно проверить, не прибегая к вычислениям, на примерах получения с её помощью точных аналитических выражений. Очевидно, что точное аналитическое выражение можно получить для функций с конечным числом членов поправочного ряда, то есть функций с ограниченным числом производных. Такой является, например, степенная функция $f(k) = (a + kh)^n$ с целочисленным показателем степени $n > 0$. Согласно формулам (19) и (20), можно получить точное выражение для суммы такого ряда с любым показателем n. При этом количество членов поправочного ряда, входящих в выражение, равно $n/2$ для чётного n и $(n+1)/2-1$ для нечётного.

В качестве примера с помощью формулы (19) при $k_0 = 1$ найдём суммы целочисленных степенных рядов и сравним результаты суммирования с известными [2, (0.121)]:

$$\sum_{k=1}^{n} k = \int_{1/2}^{n+1/2} k\,dk = \frac{1}{2}\left[\left(n+\frac{1}{2}\right)^2 - \left(\frac{1}{2}\right)^2\right] = \frac{n(n+1)}{2},$$

$$\sum_{k=1}^{n} k^2 = \frac{1}{3}\left[\left(n+\frac{1}{2}\right)^3 - \left(\frac{1}{2}\right)^3\right] - \frac{1}{24}\left[2\left(n+\frac{1}{2}\right) - \frac{2}{2}\right] = \frac{n(n+1)(2n+1)}{6},$$

$$\sum_{k=1}^{n} k^3 = \frac{1}{4}\left[\left(n+\frac{1}{2}\right)^4 - \left(\frac{1}{2}\right)^4\right] - \frac{1}{24}\left[3\left(n+\frac{1}{2}\right)^2 - \frac{3}{2^2}\right] = \left[\frac{n(n+1)}{2}\right]^2,$$

$$\ldots$$

Дальнейшие результаты суммирования при повышении степени n подтверждают справедливость как границ интегральной оценки, так и значений весовых коэффициентов b_l при производных поправочного ряда.

В справедливости производящей функции коэффициентов b_l можно убедиться на примере суммирования членов геометрической прогрессии [2, (0.112)]:

$$\sum_{k=0}^{m} d^{ck} = \frac{d^{c(m+1)} - 1}{d^c - 1}.$$

Применим для суммы значений функции d^{ck} формулу (20), преобразовав её к виду $d^{ck} = \mathrm{e}^{hk}$, где $h = c \ln d$. Интегрируя в (20) преобразованную функцию при $a = 0$, получаем

$$\sum_{k=0}^{m} \mathrm{e}^{hk} = (1/h)(\mathrm{e}^{h(m+1/2)} - \mathrm{e}^{-h/2}) - \varphi(h(m+1/2)) + \varphi(-h/2),$$

где $\varphi(hy)$ — функция поправки. Использовав равенство функции поправки ряду (10) и выражение для производящей функции (17) при $t = h$, приведём её к виду

$$\varphi(hy) = \mathrm{e}^{hy} \sum_{l=1}^{\infty} b_l h^{2l-1} = \mathrm{e}^{hy} \frac{1}{h} \sum_{l=1}^{\infty} b_l h^{2l} = \mathrm{e}^{hy} \frac{1}{h} \Psi_b(h).$$

Таким образом, формулу искомой суммы можно определить, применяя выражение для производящей функции коэффициентов b_l:

$$\sum_{k=0}^{m} \mathrm{e}^{hk} = \frac{\mathrm{e}^{h(m+1/2)} - \mathrm{e}^{-h/2}}{h} - \frac{\mathrm{e}^{h(m+1/2)} - \mathrm{e}^{-h/2}}{h} \Psi_b(h) = \frac{\mathrm{e}^{h(m+1/2)} - \mathrm{e}^{-h/2}}{h} (1 - \Psi_b(h)).$$

С помощью формулы (17) приведём выражение в скобках к виду $1 - \Psi_b(h) = \dfrac{h\,\mathrm{e}^{h/2}}{\mathrm{e}^h - 1}$ и подставим этот результат в предыдущее выражение. После несложных преобразований, используя подстановку $h = c \ln d$, окончательно получаем

$$\sum_{k=0}^{m} \mathrm{e}^{hk} = \frac{\mathrm{e}^{h(m+1/2)} - \mathrm{e}^{-h/2}}{h} \frac{h\,\mathrm{e}^{h/2}}{\mathrm{e}^h - 1} = \frac{\mathrm{e}^{h(m+1)} - 1}{\mathrm{e}^h - 1} = \frac{d^{c(m+1)} - 1}{d^c - 1}.$$

Этот результат совпадает с выражением для суммы геометрической прогрессии, что доказывает справедливость найденного выражения для производящей функции весовых коэффициентов поправочного ряда.

С помощью производящей функции можно получить точные суммы линейных комбинаций показательной функции, например гиперболических функций $\mathrm{sh}\,(hk)$, $\mathrm{ch}\,(hk)$.

2.2. Общее выражение приближённых формул суммирования и их погрешность

Вычисление с помощью полученных формул приблизительных числовых значений сумм ограниченных функциональных рядов с заданными параметрами не имеет смысла, поскольку можно получить практически точный результат, используя компьютер. Приближённые аналитические выражения могут потребоваться для установления упрощённой функциональной связи результата суммирования с его параметрами, а также для получения его числовой оценки с требуемой точностью при неограниченном числе членов суммируемого ряда. Эти приближения неизбежны в случае бесконечного числа членов поправочного ряда в формулах (19) и (20) и ограничение их числа приведёт к погрешности итогового результата формул, равной остатку его суммирования. Из-за возможного большого различия граничных значений производных в формулах (19) и (20) целесообразно разделить поправочный ряд на два, с верхним и нижним значениями производных и разным количеством учитываемых членов с соответствующим остатком.

Разделяя ряд, представим формулу (20) в виде

$$\sum_{k=0}^{m} f(y_k) = \widehat{S_q^m}\big(f(y)\big) - \varphi_J(y_m) + \varphi_I(y_q) + \Delta_J(y_m) - \Delta_I(y_q).$$

Здесь приближённая интегральная оценка

$$\widehat{S_q^m}\big(f(y)\big) = \sum_{k=0}^{q} f(y_k) + \frac{1}{h}\int\limits_{y_q}^{y_m} f(y)\,dy; \tag{22}$$

$\varphi_J(y_m), \varphi_I(y_q)$ — поправка к оценке, представленная ограниченными рядами поправочной функции (18):

$$\varphi_I(y_q) = \sum_{i=1}^{I-1} b_i f^{(2i-1)}(y_q), \quad \varphi_J(y_m) = \sum_{j=1}^{J-1} b_j f^{(2j-1)}(y_m); \tag{23}$$

$\Delta_I(y_q), \Delta_J(y_m)$ — погрешности интегральной оценки, равные остаткам суммирования рядов (23), взятым с противоположным знаком, $y_q = a + (q + 1/2)h$, где q определяет вклад значения $\Delta_I(y_q)$ в погрешность интегральной оценки. Задавая граничные индексы $I, J = 1, 2, 3, \ldots$ и раскрывая числовые значения коэффициентов $b_{I,J+1}$ из (15), эти погрешности можно представить виде

$$-\Delta_I(y_q) = \left\{ \frac{hf'(y_q)}{24}, -\frac{7h^3 f'''(y_q)}{5760}, \frac{31h^5 f^V(y_q)}{967680}, \cdots \right\}; \tag{24}$$

$$\Delta_J(y_m) = \left\{ \frac{hf'(y_m)}{24}, -\frac{7h^3 f'''(y_m)}{5760}, \frac{31h^5 f^V(y_m)}{967680}, \cdots \right\}. \tag{25}$$

Значений погрешностей (24) и (25) интегральной оценки недостаточно для определения её точности. О ней можно судить, лишь вычислив относительную погрешность. При численно неопределённых параметрах суммирования эту погрешность можно определить приблизительно:

$$\widehat{\delta}_q^m(I, J) \simeq \frac{\Delta_J(y_m) - \Delta_I(y_q)}{\widehat{S_q^m}\big(f(y)\big)}. \tag{26}$$

В случаях известного результата суммирования относительную погрешность приближённой интегральной оценки можно определить с любой точностью:

$$\delta_q^m(I, J) = \frac{\widehat{S_q^m}\big(f(y)\big)}{\sum\limits_{k=0}^{m} f(y_k)} - 1. \tag{27}$$

2.3. Суммирование дискретных значений функции $f(y) = y^{-p}$

Очевидно, что при бесконечном числе отсчётов функции y^{-p} сумма существует только для $p > 1$. В остальных случаях при проведении оценки следует ограничивать число отсчётов.

Исследуем сходимость поправочного ряда, для чего найдём предел отношения значений соседних его членов $d_k = \dfrac{b_{k+1} f^{(2k+1)}(y)}{b_k f^{(2k-1)}(y)}$. Общее выражение для нечётных производных этой функции можно привести к виду

$$f^{(2k-1)}(y) = -\left(\frac{1}{y}\right)^{p+2k-1} \prod_{l=1}^{2k-1} (p + l - 1). \tag{28}$$

Отсюда, задавая $k = 1, 2, 3, \ldots$ и используя выражения для коэффициентов b_k (15) и (16), получаем

$$|d_k| = \left\{ \frac{(1+p)(2+p)}{34{,}2857y^2}, \frac{(3+p)(4+p)}{37{,}9355y^2}, \frac{(5+p)(6+p)}{39{,}0551y^2}, \ldots, \frac{(p+2k-1)(p+2k)}{39{,}4784y^2} \right\}.$$

Из выражения k-го члена этой последовательности следует $\lim\limits_{k\to\infty} |d_k| = \infty$, что противоречит признаку сходимости Даламбера $\lim\limits_{k\to\infty} |d_k| < 1$ [2, (0.222)]. Это означает, что поправочный ряд расходится для любых $y < \infty$. Поскольку все нечётные производные (28) отрицательны, поправочный ряд на их основе является знакопеременным из-за противоположности знаков b_n его соседних членов. При этом все чётные производные имеют положительный знак во всём интервале $[y_0, y_m]$. Отсюда, согласно (21), погрешность можно оценить первым отбрасываемым членом функции поправки при $\theta < 1$.

Из выражения для производной (28) видно, что погрешность оценки $\Delta_I(y_q)$ тем меньше, чем больше единицы нижняя граница интегральной оценки y_0. Поэтому если $y_0 < 1$, целесообразно просуммировать в соответствии с (22) начальные члены с $y_k < 1$ с целью установления нижней границы $y_q > 1$. Если число суммируемых отсчётов ограничено, а параметры a, h в (20) столь малы, что приводят к значительному количеству членов начальной суммы с $y_k < 1$, формулу (20) можно свести к оценке суммы отсчётов функции от целочисленного аргумента путём умножения каждого отсчёта на h^p и деления результата суммирования на h^{p+m+1} при $a > 0$ и на h^{p+m} при $a = 0$.

При $p \neq 1$ упрощённая интегральная оценка (23) примет вид

$$\widehat{S_q^m}(y^{-p}) = \sum_{k=0}^{q} y_k^{-p} + \frac{1}{(p-1)\, y_q^{p-1}} - \frac{1}{(p-1)\, y_m^{p-1}}, \tag{29}$$

где $y_q = a + (q + 1/2)h$; $y_m = a + (m + 1/2)h$; оценка по формуле Эйлера — Маклорена равна

$$S_q^{\mathrm{EM}}\left(y^{-p}\right) = \sum_{k=0}^{q} y_k^{-p} + \frac{1}{2y_q^p} + \frac{1}{(p-1)y_q^{p-1}} - \frac{1}{(p-1)y_m^{p-1}}, \tag{30}$$

где $y_q = a + (q + 1)h$; $y_m = a + mh$.

Для численных расчётов с заданными параметрами p, a, h, m значение q определяется заданной погрешностью, а начальную сумму можно рассчитать на компьютере при практически неограниченном числе её членов. Тогда без учёта поправки ожидаемая погрешность в соответствии с (21) будет определяться первым членом поправочных рядов (23):
для упрощённой оценки

$$\Delta_1(y_{q,m}) = \frac{p}{24y_{q,m}^{p+1}}; \tag{31}$$

для формулы Эйлера — Маклорена

$$\Delta_1^{\mathrm{EM}}(y_{q,m}) = \frac{p}{12y_{q,m}^{p+1}}.$$

Поскольку $\Delta_1(y_m) \to 0$ при $m \to \infty$, увеличивая число членов начальной суммы q, можно обеспечить любую точность численной интегральной оценки.

Суммирование гармонического ряда $S_1^m(k^{-1}) = \sum\limits_{k=1}^{m} k^{-1}$

Этот ряд является расходящимся, поэтому число отсчётов m должно быть ограниченным. Известный ещё в начале прошлого века результат суммирования имеет вид [2, (0.131)]

$$\sum_{k=1}^{n}\frac{1}{k} = C + \ln(n) + \frac{1}{2n} - \sum_{k=2}^{\infty}\frac{A_k}{n(n+1)\dots(n+k-1)}, \tag{32}$$

где $C = \gamma$ — постоянная Эйлера; $A_2 = A_3 = 1/12$; $A_4 = 19/80$; $A_5 = 9/20$, ...

Представим в соответствии с (22) приближённую интегральную оценку этого ряда в виде

$$\widehat{S_q^m}\left(k^{-1}\right) = C_q + \ln(m+1/2), \text{ где } C_q = \sum_{k=1}^{q}\frac{1}{k} - \ln(q+1/2).$$

Задавая значения $q = 1, 2, 3, 4, \dots$, получим $C_q \simeq 0{,}594535, 0{,}583709, 0{,}58057, 0{,}579256, 0{,}577593, \dots$ При дальнейшем увеличении q получим, согласно [1, (6.1.3)], $\lim\limits_{q\to\infty} C_q = \gamma$, где $\gamma = 0{,}577\,215\,664\,901\,532\,5\dots$ — постоянная Эйлера.

Очевидно, что для такой оценки поправка $\varphi_I(y_q) = 0$ и погрешность $\Delta_I(y_q) = 0$. Отсюда, согласно (19), используя при $p = 1$ выражение для производной (28) $f^{(2k-1)}(y) = -y^{-2k}(2k-1)!$, представим упрощённую формулу суммирования гармонического ряда в виде

$$S_1^m\left(k^{-1}\right) = \gamma + \ln(m+1/2) + \sum_{k=1}^{\infty}\frac{(1-2^{1-2k})B_{2k}}{2k(m+1/2)^{2k}}. \tag{33}$$

Аналогичным способом можно получить из (2) формулу

$$S_{\mathrm{EM}}^m(k^{-1}) = \gamma + \ln m + \frac{1}{2m} - \sum_{k=1}^{\infty}\frac{B_{2k}}{2km^{2k}}. \tag{34}$$

Отметим, что интегральные оценки формул (32) и (34) совпадают, причём и первые члены поправочного ряда при больших m, n практически равны. Получим из (33) и (34) приближённые выражения, удерживая в них не более двух членов, зависящих от верхнего индекса m:

$$\widehat{S}_{10}^m\left(k^{-1}\right) = \gamma + \ln(m+1/2); \tag{35}$$

$$\widehat{S}_{11}^m\left(k^{-1}\right) = \gamma + \ln(m+1/2) + \frac{1}{24(m+1/2)^2}; \tag{36}$$

$$\widehat{S}_{\mathrm{EM}}^m\left(k^{-1}\right) = \gamma + \ln(m) + \frac{1}{2m}. \tag{37}$$

Взятый с обратным знаком третий член формулы (36), в соответствии с (21), определяет приближённую погрешность формулы (35) $\Delta_1(m) = -\dfrac{1}{24(m+1/2)^2}$.

Найдём, согласно (26) и (27), относительные погрешности приближённых формул (35)–(37) и сравним их между собой:

$$\widehat{\delta}_{01}(m) = \frac{\Delta_1(m)}{\widehat{S}_{10}^m\left(k^{-1}\right)},\ \delta_{10}(m) = \frac{\widehat{S}_{10}^m\left(k^{-1}\right)}{\sum\limits_{k=1}^{m} k^{-1}} - 1,\ \delta_{11}(m) = \frac{\widehat{S}_{11}^m\left(k^{-1}\right)}{\sum\limits_{k=1}^{m} k^{-1}} - 1,\ \delta_{EM}(m) = \frac{S_{EM}^m\left(k^{-1}\right)}{\sum\limits_{k=1}^{m} k^{-1}} - 1.$$

Результаты вычислений этих погрешностей приведены в табл. 1.

Таблица 1

Относительные погрешности интегральных оценок сумм гармонического ряда

m	3	10	100	1000
$\delta_{\mathrm{EM}}(m)$	$5 \cdot 10^{-3}$	$2{,}84 \cdot 10^{-4}$	$1{,}61 \cdot 10^{-6}$	$1{,}11 \cdot 10^{-8}$
$\widehat{\delta}_{10}(m)$	$-1{,}86 \cdot 10^{-3}$	$-1{,}29 \cdot 10^{-4}$	$-7{,}95 \cdot 10^{-7}$	$-5{,}56 \cdot 10^{-9}$
$\delta_{10}(m)$	$-1{,}83 \cdot 10^{-3}$	$-1{,}29 \cdot 10^{-4}$	$-7{,}95 \cdot 10^{-7}$	$-5{,}56 \cdot 10^{-9}$
$\delta_{11}(m)$	$2{,}55 \cdot 10^{-5}$	$2{,}04 \cdot 10^{-7}$	$1{,}38 \cdot 10^{-11}$	$8{,}88 \cdot 10^{-16}$

Сравнив табличные результаты, можно сделать следующие выводы:

— все приближённые формулы (35)–(37) обеспечивают хорошую точность суммирования даже для малых значений m;
— приближённая $\widehat{\delta}_{10}(m)$ и реальная $\delta_{10}(m)$ погрешности формулы (35) практически совпадают;
— приближённая формула (35), представленная одним зависящим от индекса m членом, более чем в 2 раза точнее формулы (37), полученной из формулы Эйлера — Маклорена и представленной двумя такими членами;
— приближённая формула (36), представленная двумя зависящими от m членами, на несколько порядков точнее формул (35) и (37).

Численные расчёты сумм бесконечных рядов k^{-p} с показателем степени $p > 1$

Формулы для этих расчётов можно получить из выражений (29)–(31), задавая параметры $a = 0$, $h = 0$ и $m = \infty$:

$$\widehat{S}_q(k^{-p}) = \sum_{k=1}^{q} k^{-p} + \frac{1}{(p-1)(q+1/2)^{p-1}}; \tag{38}$$

$$S_q^{\mathrm{EM}}(k^{-p}) = \sum_{k=1}^{q} k^{-p} + \frac{1}{2(q+1)^p} + \frac{1}{(p-1)(q+1)^{p-1}}; \tag{39}$$

$$\Delta_{1q}(k^{-p}) = -\frac{p}{24(q+1/2)^{p+1}}. \tag{40}$$

Формула (40) задаёт приближённую погрешность формулы (38), зависящую от количества членов начального ряда q. Задавая q, можно сколько угодно уменьшать эту погрешность, однако её использование в качестве реальной следует проверить. Реальную погрешность можно вычислить, применяя известные точные значения сумм рядов, например рядов с чётным целочисленным показателем p [2, (0.233.3)]:

$$S_1^{\infty}(k^{-p}) = \sum_{k=1}^{\infty} k^{-2n} = \frac{2^{2n-1}\pi^{2n}}{(2n)!} \left| B_{2n} \right|.$$

Используя значения этих сумм, можно вычислить реальную погрешность (27) формулы (38) и сравнить её с относительной погрешностью (26) и с реальной погрешностью формулы (39), полученной из формулы Эйлера — Маклорена (2):

$$\delta_q^p = \frac{\widehat{S}_q(k^{-p})}{S_1^{\infty}(k^{-p})} - 1, \ \widehat{\delta}_q^p = \frac{\Delta_{1q}(k^{-p})}{\widehat{S}_q(k^{-p})}, \ \delta_q^{\mathrm{EM}} = \frac{S_q^{EM}(k^{-p})}{S_1^{\infty}(k^{-p})} - 1.$$

На рис. 1 представлены графики десятичного логарифма приближённой относительной погрешности $\widehat{\delta}_q^p$ вычисления сумм бесконечных рядов k^{-p} по формуле (38)

при $p \in [1{,}5, 14]$ и количестве членов начальной суммы $q = 10, 100, 1000$. Маркерами отмечены реальные относительные погрешности δ_q^p для чётных значений $p = 2, \ldots, 12$. Для тех же чётных p в табл. 2 приведены числовые значения реальной погрешности δ_q^p формулы (38), отношения её приближённой погрешности к реальной $\widehat{k}_q^p = \widehat{\delta}_q^p / \delta_q^p$ и отношения реальных погрешностей $k_q^{\mathrm{EM}} = \delta_q^{\mathrm{EM}} / \delta_q^p$.

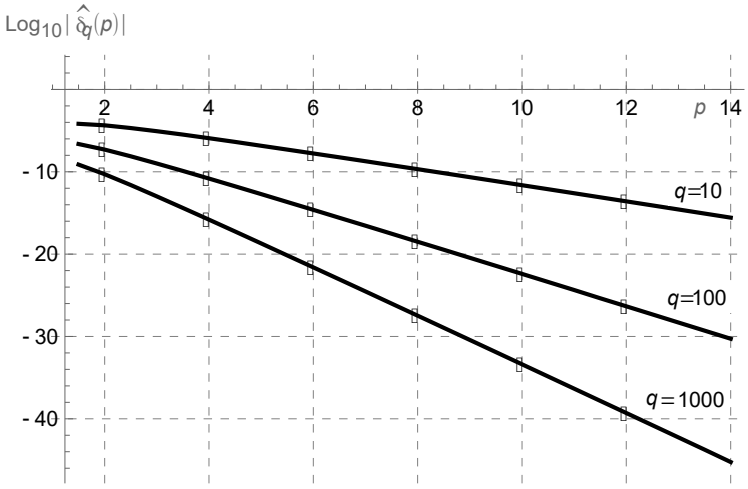


Рис. 1. Относительные погрешности формул суммирования бесконечных рядов k^{-p}

Таблица 2

Относительные погрешности вычисления сумм бесконечных рядов k^{-p} с чётным показателем p

q	Формула	Значение суммы					
		$\pi^2/6$	$\pi^4/90$	$\pi^6/945$	$\pi^8/9450$	$\pi^{10}/93555$	$691\pi^{12}/638512875$
		p					
		2	4	6	8	10	12
10	δ_q^p	$4{,}36 \cdot 10^{-5}$	$1{,}20 \cdot 10^{-6}$	$1{,}72 \cdot 10^{-8}$	$2{,}09 \cdot 10^{-10}$	$2{,}35 \cdot 10^{-12}$	$2{,}53 \cdot 10^{-14}$
	$\widehat{k}_q^p$	1,00	1,01	1,01	1,02	1,03	1,05
	k_q^{EM}	−0,574	−0,629	−0,688	−0,752	−0,821	−0,895
100	δ_q^p	$4{,}99 \cdot 10^{-8}$	$1{,}50 \cdot 10^{-11}$	$2{,}37 \cdot 10^{-15}$	$3{,}17 \cdot 10^{-19}$	$3{,}94 \cdot 10^{-23}$	$4{,}68 \cdot 10^{-27}$
	$\widehat{k}_q^p$	1,00	1,00	1,00	1,00	1,00	1,00
	k_q^{EM}	−0,507	−0,513	−0,518	−0,523	−0,528	−0,533
1000	δ_q^p	$5{,}06 \cdot 10^{-11}$	$1{,}54 \cdot 10^{-16}$	$2{,}45 \cdot 10^{-22}$	$3{,}30 \cdot 10^{-28}$	$4{,}14 \cdot 10^{-34}$	$4{,}97 \cdot 10^{-40}$
	$\widehat{k}_q^p$	1,00	1,00	1,00	1,00	1,00	1,00
	k_q^{EM}	−0,501	−0,501	−0,502	−0,502	−0,503	−0,503

Данные графиков и результаты расчётов позволяют сделать следующие выводы:

— если значение суммы неизвестно, то погрешность её расчёта можно задавать при достаточно большом количестве членов начальной суммы, благодаря практически равным при этом приближённой и реальной погрешностей;

— увеличивая количество членов начальной суммы, можно задавать практически любую точность результата суммирования;

— формула (38), несмотря на более короткую форму, почти вдвое точнее формулы (39), полученной из формулы Эйлера — Маклорена.

Приближённые формулы суммирования бесконечных рядов k^{-p} при $p > 1$

Задачей приближённых формул является упрощение связи результата суммирования с параметрами суммируемой функции. Для создания приближённых формул сумм бесконечных рядов k^{-p} можно использовать формулы (38) и (39) с небольшим количеством членов начальной суммы. Удерживая не более двух таких членов, из них получим:

— (38) при $q = 1$, без поправки, с приближённой погрешностью $\Delta_{10} = \dfrac{p}{24(3/2)^{p+1}}$:

$$\widehat{S_{10}}(k^{-p}) = 1 + \frac{1}{(p-1)(3/2)^{p-1}}; \tag{41}$$

— (38) при $q = 2$, без поправки, с приближённой погрешностью $\Delta_{20} = \dfrac{p}{24(5/2)^{p+1}}$:

$$\widehat{S_{20}}(k^{-p}) = 1 + \frac{1}{2^p} + \frac{1}{(p-1)(5/2)^{p-1}}; \tag{42}$$

— (38) при $q = 1$, с первым членом поправки, с приближённой погрешностью $\Delta_{11} = \dfrac{7p(p+1)(p+2)}{5760(3/2)^{p+3}}$:

$$\widehat{S_{11}}(k^{-p}) = 1 + \frac{1}{(p-1)(3/2)^{p-1}} - \frac{p}{24(3/2)^{p+1}}; \tag{43}$$

— (39) при $q = 1$, без поправки, с приближённой погрешностью $\Delta_{\text{em}} = \dfrac{p}{12 \cdot 2^{p+1}}$:

$$S_{10}^{\text{EM}}(k^{-p}) = 1 + \frac{1}{2^{p+1}} + \frac{1}{(p-1)2^{p-1}}. \tag{44}$$

По этим формулам, согласно (26), рассчитаны приближённые относительные погрешности. Для их сравнения с реальной погрешностью с помощью (38) и (39)) при $q = 1000$ рассчитаны уточнённые суммы и, в соответствии с (27), уточнённые относительные погрешности. Результаты расчётов приведены на рис. 2, где пунктиром показаны приближённые погрешности, сплошной линией — уточнённые.

По результатам проведённых расчётов можно сделать следующие выводы:

— кривые приближённой погрешности имеют сходную с соответствующей уточнённой погрешностью траекторию и находятся в её пределах, что доказывает справедливость формулы (21) для оценки остатка суммирования поправочного ряда;
— наилучшее приближение $\delta_{20}(p) < 0{,}3\,\%$ даёт формула (42), наихудшее $\delta_{10}(p) < 1{,}7\,\%$ — самая простая (41);
— формула (43), содержащая поправку, не имеет преимуществ в точности при $p > 2{,}5$;
— приближённые погрешности и их близость к уточнённым сильно зависят от количества членов начальной суммы q;
— сравнение погрешностей $\delta_{20}(p)$ формулы (42) и $\delta_{\text{em}}(p)$ формулы Эйлера — Маклорена (44), содержащих по два зависящих от p члена, вновь указывает на преимущество упрощённой формулы суммирования.

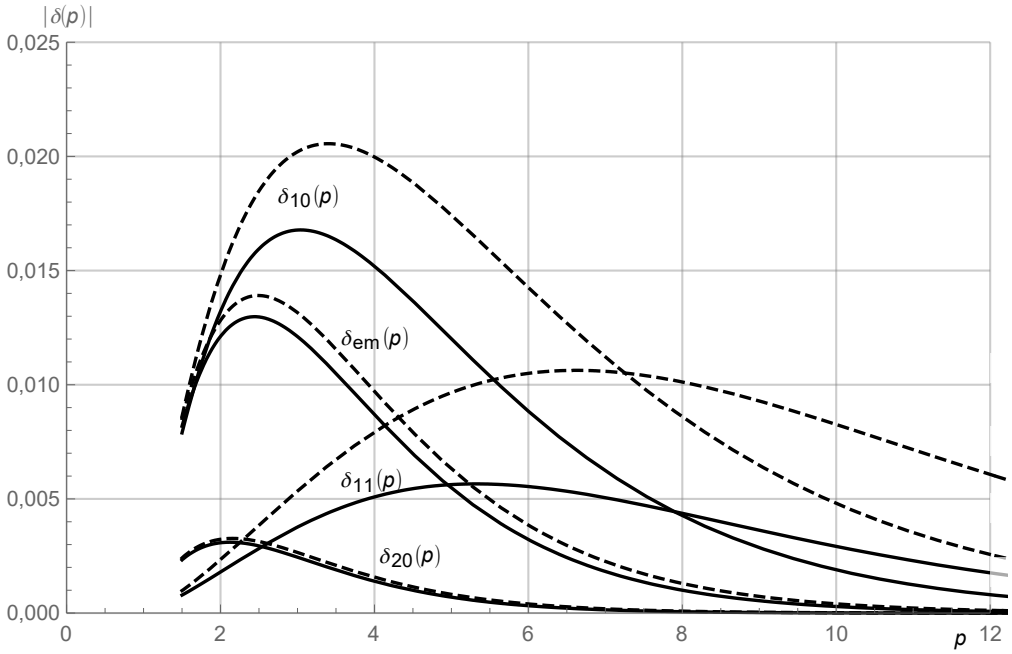


Рис. 2. Относительные погрешности приближённых формул суммирования бесконечных рядов k^{-p}

2.4. Упрощённые формулы некоторых специальных функций

На конечных и бесконечных суммах обратностепенных числовых рядов основаны выражения и оценки для некоторых специальных функций.

Приближённое выражение для дзета-функции Римана

Классическое представление этой функции в виде ряда Дирихле имеет вид [4, с. 58]

$$\zeta(s) = \sum_{n=1}^{\infty} n^{-s},$$

где $s = \sigma + it$ и ряд сходится к аналитической функции при $\sigma > 1$.

В п. 2.3 проведён сравнительный анализ приближённых формул для такого ряда в отсутствие мнимого аргумента ($t = 0$). Проверим их справедливость при $t \neq 0$. Для этого используем формулу (42), содержащую два члена начальной суммы:

$$\widehat{\zeta}_{20}(s) = 1 + \frac{1}{2^s} + \frac{1}{(s-1)(5/2)^{s-1}}, \tag{45}$$

и более точную, содержащую поправку к ней:

$$\widehat{\zeta}_{21}(s) = 1 + \frac{1}{2^s} + \frac{1}{(s-1)(5/2)^{s-1}} - \frac{s}{24(5/2)^{s+1}}. \tag{46}$$

Рис. 3 и 4, где приведены графики зависимости погрешностей абсолютных значений $|\widehat{\zeta}_{20}(s)|$ и $|\widehat{\zeta}_{21}(s)|$ от комплексного аргумента относительно истинного абсолютного значения дзета-функции $|\zeta(s)|$, указывают на вполне удовлетворительную ($|\widehat{\zeta}_{20}(s)| < 10\,\%$ для (45) и $|\widehat{\zeta}_{21}(s)| < 5\,\%$ для (46)) точность оценки значений дзета-функции в пределах изменения аргументов $|s| \in [1, 10]$, $\varphi_s \in [-\pi/2, \pi/2]$. Приближение s к мнимому значению $\varphi_s \to \pm\pi/2$ значительно увеличивает погрешность формул при увеличении $|s|$.

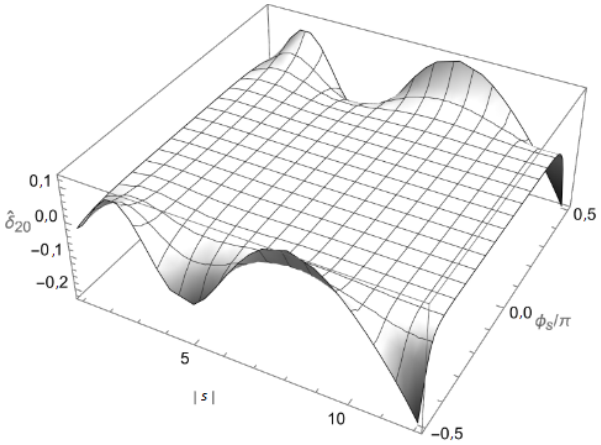


Рис. 3. Относительная погрешность приближённого абсолютного значения дзета-функции (45) на комплексной плоскости

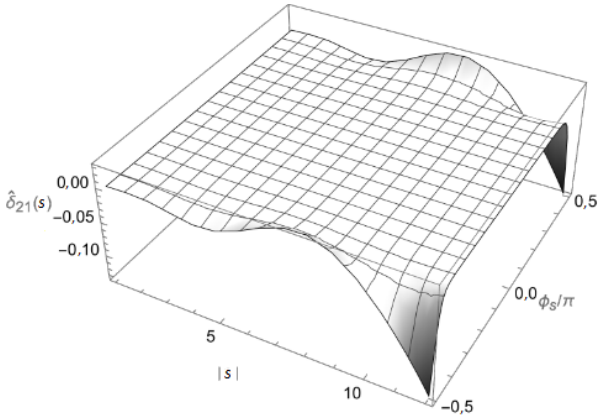


Рис. 4. Относительная погрешность приближённого абсолютного значения дзета-функции (46) на комплексной плоскости

Приближённые формулы для пси-функции

Для вывода приближённого выражения пси-функции (или дигамма-функции) из всего многообразия её представлений воспользуемся формулой для целочисленного аргумента [1, (6.3.2)]

$$\psi(n) = \begin{cases} -\gamma, & \text{если } n = 1, \\ -\gamma + \sum\limits_{k=1}^{n-1} k^{-1}, & \text{если } n \geqslant 2, \end{cases} \tag{47}$$

где γ — постоянная Эйлера.

В этой формуле присутствует сумма гармонического ряда, для которого найдены приближённые выражения — простейшее (35) и более точное (36). С их помощью получим следующие приближённые формулы:

$$\widehat{\psi}_0(n) = \ln(n - 1/2); \tag{48}$$

$$\widehat{\psi}_1(n) = \ln(n - 1/2) + \frac{1}{24(n - 1/2)^2}. \tag{49}$$

Проверим справедливость этих формул для любых действительных $n = z > 1/2$. На рис. 5 представлены графики этих зависимостей в сравнении с реальным значением пси-функции $\psi(z)$. Как видно из графиков, зависимости значительно отклоняются от реальной при $z \to 1/2$, поскольку при этом значении аргумента в формулах (48) и (49) возникает неопределённость. От этого недостатка можно избавиться, прибавляя к сумме в (47) член $1/n$ и вычитая его из итоговой оценки. Модифицированные таким образом формулы (48), (49) при $n = z$ примут вид

$$\widehat{\psi}_{01}(z) = \ln(z + 1/2) - \frac{1}{z}; \tag{50}$$

$$\widehat{\psi}_{11}(z) = \ln(z + 1/2) - \frac{1}{z} + \frac{1}{24(z + 1/2)^2}. \tag{51}$$

На рис. 6 приведены графики относительной погрешности формул (50), (51) при значениях аргумента $0{,}01 \leqslant z \leqslant 0{,}5$.

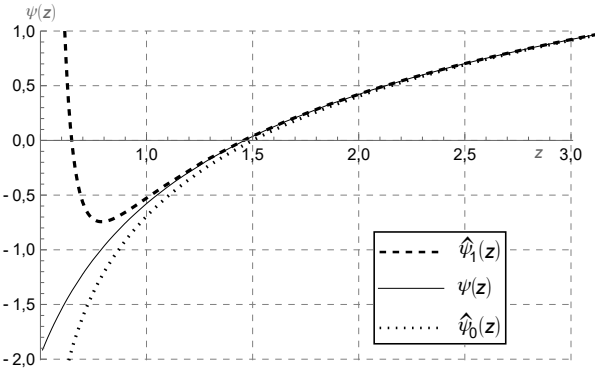


Рис. 5. Приближённые формулы пси-функции (48), (49) в сравнении с реальным её значением

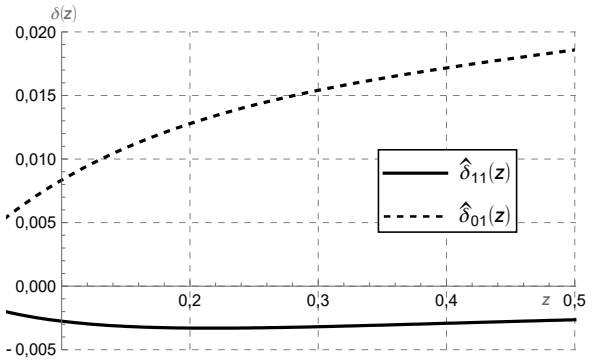


Рис. 6. Относительная погрешность приближённых формул (50), (51) при значениях аргумента $z \leqslant 1/2$

Все рассмотренные приближённые зависимости с ростом z асимптотически стремятся к значению $\ln z$, что соответствует свойствам пси-функции [1, (6.3.18)].

Приближённые формулы для полигамма-функций

Полигамма-функцией порядка n является n-я производная пси-функции [1, (6.4.1)]

$$\psi^{(n)}(z) = \frac{d^n}{dz^n}\psi(z). \tag{52}$$

Для целого аргумента этих функций существует выражение [1, (6.4.2)]

$$\psi^{(m)}(n+1) = (-1)^m m! \left[-\zeta(m+1) + 1 + \frac{1}{2^{m+1}} + \ldots + \frac{1}{n^{m+1}}\right], \tag{53}$$

где $\zeta(m+1)$ — дзета-функция Римана.

Выражение в квадратных скобках можно преобразовать, используя определение дзета-функции

$$-\zeta(m+1) + 1 + \frac{1}{2^{m+1}} + \ldots + \frac{1}{n^{m+1}} = -\sum_{k=1}^{\infty}\frac{1}{k^{m+1}} + \sum_{k=1}^{n}\frac{1}{k^{m+1}} = -\sum_{k=n+1}^{\infty}\frac{1}{k^{m+1}}.$$

Вынося из суммы первый член $1/(n+1)^{m+1}$ и преобразуя аргумент $n+1 \to n$, из (53) получаем

$$\psi^{(m)}(n) = (-1)^{m-1} m! \left[\frac{1}{n^{m+1}} + \sum_{k=n+1}^{\infty}\frac{1}{k^{m+1}}\right].$$

Подставляя в это выражение вместо суммы её упрощённую интегральную оценку $\displaystyle\int_{m+1/2}^{\infty}\frac{1}{k^{m+1}} = \frac{1}{m(n+1/2)^m}$, имеем

$$\widehat{\psi}^{(m)}(n) = (-1)^{m-1} m! \left[\frac{1}{n^{m+1}} + \frac{1}{m(n+1/2)^m}\right]. \tag{54}$$

Проинтегрировав (54) при $m=1$, $n=z$ по z, получим результат $\ln(z+1/2) - 1/z$, полностью совпадающий с упрощённой интегральной оценкой пси-функции (50). Вынос члена суммы $1/(n+1)^{m+1}$ понадобился, как в (50), для устранения неопределённости в приближённых выражениях полигамма-функций при $z = 1/2$.

Проверим возможность использования приближённых выражений пси-функции (50), (51) для получения выражений полигамма-функций, взяв в соответствие с (52) их производные

$$\widehat{\psi}_{x1}^{(n)}(z) = \frac{d^n}{dz^n}\widehat{\psi}_{x1}. \tag{55}$$

На рис. 7 и 8 приведены графики погрешности производных (55) приближённых выражений пси-функции (50), (51) относительно реальных значений соответствующих полигамма-функций. Максимальная относительная погрешность формулы (55) относительно значений полигамма-функций для приближённого выражения пси-функции $\widehat{\psi}_{01}^{(n)}(z)$ (50) до десятого порядка не превышает 2,5 %, для пси-функции $\widehat{\psi}_{11}^{(n)}(z)$ (51) — 0,7 %.

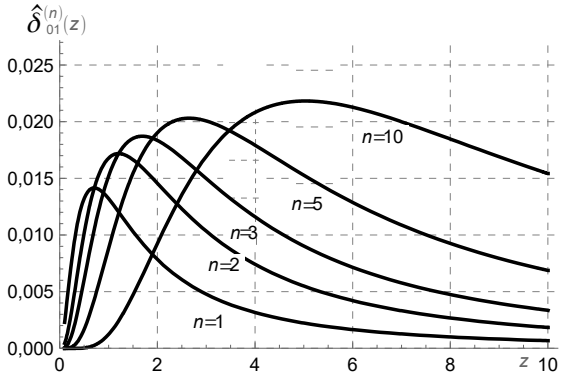


Рис. 7. Относительная погрешность формулы (55) для функции $\widehat{\psi}_{01}^{(n)}(z)$ (50)

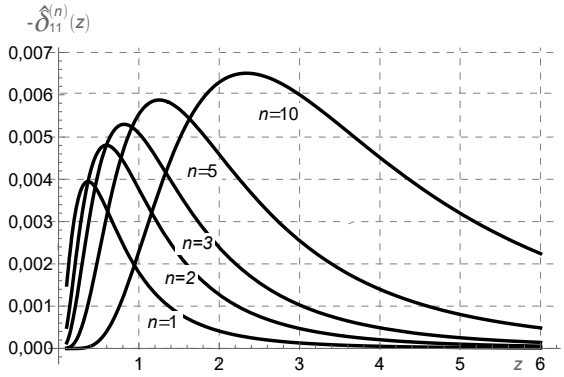


Рис. 8. Относительная погрешность формулы (55) для функции $\widehat{\psi}_{11}^{(n)}(z)$ (51)

Заключение

Полученные приближённые аналитические выражения для дзета-функции Римана, пси-функции, полигамма-функций, а также сумм обратно степенных рядов и гармонического ряда демонстрируют непрямой порядок применения упрощённой формулы суммирования. При их выводе были использованы начальное суммирование для уменьшения погрешности оценки и прибавление дополнительного члена к оцениваемой сумме с вычитанием его из итоговой оценки для устранения в ней неопределённости. Полученные формулы выявили некоторое преимущество применения упрощённой формулы перед формулой Эйлера — Маклорена (2) в краткости и погрешности. Этим она оправдывает своё прикладное значение и существование в качестве ещё одного варианта формулы суммирования Эйлера — Маклорена. Однако, в отличие от последней, она неприменима для оценки интегралов в пределах, совпадающих с позицией граничных отсчётов функций.

ЛИТЕРАТУРА

1. Справочник по специальным функциям с формулами, графиками и математическими таблицами / под ред. М. Абрамовица и И. Стиган. М.: Наука, 1979. 830 с.
2. *Градштейн И. С., Рыжик И.М.* Таблицы интегралов, сумм, рядов и произведений. 4-е изд. М.: Физматгиз, 1963. 1108 с.
3. *Фихтенгольц Г. М.* Курс дифференциального и интегрального исчисления. Т. II. 7-е изд. М.: Наука, Физматгиз, 1970. 800 с.
4. *Уитеккер Э. Т., Ватсон Дж. Н.* Курс современного анализа. Ч. 2. Трансцендентные функции. 2-е изд. М.: Физматгиз, 1963. 500 с.

REFERENCES

1. Handbook of Mathematical Functions with Formulas, Graphs and Mathematical Tables. M. Abramowitz and I. A. Stegun (eds.). N.Y., Dover Publ., 1964.
2. *Gradshteyn I. S. and Ryzhik I.M.* Tablitsy integralov, summ, ryadov i proizvedeniy [Tables of Integrals, Sums, Series, and Products]. Moscow, Fizmatgiz Publ., 1963. 1108 p. (in Russian)
3. *Fikhtengol'ts G. M.* The Fundamentals of Mathematical Analysis. Elsevier Ltd., 1965.
4. *Whittaker E. T. and Watson G. N.* A Course of Modern Analysis. 4th Ed. Cambridge, Cambridge University Press, 1927.

УДК 519.8 DOI 10.17223/20710410/64/8

ЗАДАЧА МИНИМИЗАЦИИ ОБЩЕГО ВРЕМЕНИ ОБРАБОТКИ ИДЕНТИЧНЫХ ДЕТАЛЕЙ[1]

А. А. Романова*, В. В. Сервах*,**, В. Ю. Тавченко*

*Омский государственный университет им. Ф. М. Достоевского, г. Омск, Россия
**Институт математики им. С. Л. Соболева СО РАН, г. Омск, Россия

E-mail: RomanovaAA@omsu.ru, svv_usa@rambler.ru, nikapolicheva@mail.ru

Рассматривается задача минимизации общего времени обработки идентичных деталей со сложным технологическим маршрутом, когда возможно неоднократное поступление деталей на некоторые машины. Исследуются вопросы вычислительной сложности данной задачи, доказана её NP-трудность в обычном смысле. При фиксированном числе деталей доказана псевдополиномиальная разрешимость задачи. Исследуется вопрос использования циклических расписаний при построении приближённых решений.

Ключевые слова: *расписание, идентичные детали, NP-трудность, псевдополиномиальный алгоритм, теория сложности.*

MAKESPAN MINIMIZATION IN REENTRANT FLOW SHOP PROBLEM WITH IDENTICAL JOBS

A. A. Romanova*, V. V. Servakh*,**, V. Yu. Tavchenko*

*Dostoevsky Omsk State University, Omsk, Russia
**Sobolev Institute of Mathematics, Omsk, Russia

We consider the reentrant flow shop problem $F|\text{reentrant}, p_{ij} = p_i|C_{\max}$ with identical jobs and makespan criterion. We prove its NP-hardness in the ordinary sense. The proof is performed by polynomial reduction of the problem $J3|n = 3|C_{\max}$ to the problem $F|\text{reentrant}, p_{ij} = p_i|C_{\max}$ with three identical jobs. Using the input data of the problem $J3|n = 3|C_{\max}$, we have constructed a special type of job for the problem $F|\text{reentrant}, p_{ij} = p_i|C_{\max}$. In the proof, we analyze all possible variants of critical paths in the constructed instance. We also propose a dynamic programming algorithm to find the optimal solution of the problem $F|\text{reentrant}, p_{ij} = p_i|C_{\max}$. Analysis of the time complexity of the algorithm showed that the problem with fixed number of jobs is pseudopolynomially solvable. Next, we study the use of cyclic schedules to construct approximate solutions. A cyclic schedule with a minimum cycle time can be found in polynomial time. We propose a polynomial algorithm for finding an approximate solution. This algorithm is based on the construction of a cyclic schedule with a minimum cycle time and its subsequent compaction at the beginning and at the end. We construct an upper bound of the makespan of cyclic schedules. This bound depends on the number of jobs processed in one cycle. The paper provides numerous examples characterizing cyclic schedules from both the positive and negative sides to solve the problem $F|\text{reentrant}, p_{ij} = p_i|C_{\max}$.

[1]Работа выполнена в рамках госзадания ИМ СО РАН, проект FWNF-2022-0020.

Keywords: *schedule, identical jobs, NP-hardness, pseudopolynomial algorithm, theory of NP-completeness.*

Введение

Рассматривается задача обработки идентичных деталей со сложным технологическим маршрутом на современной роботизированной производственной линии. В общепринятой номинации [1] эта задача обозначается как $F|\text{reentrant}, p_{ij} = p_i|C_{\max}$, подразумевая, что все детали проходят одинаковый технологический маршрут и длительности соответствующих операций равны. Тем самым все детали идентичны, а термин «reentrant» означает, что при обработке деталь может поступать на некоторые машины неоднократно. Задача $F|\text{reentrant}, p_{ij} = p_i|C_{\max}$ является обобщением классической одномаршрутной задачи Flow-Shop и впервые была сформулирована в [2]. Большинство известных результатов можно найти в [3]. Данная задача является NP-трудной даже для случая двух машин. Фактически большинство теоретических вопросов, связанных с общей постановкой, так или иначе закрыты. Если же детали идентичны, то возникает множество математически интересных постановок. Ключевым является вопрос о вычислительной сложности задачи $F|\text{reentrant}, p_{ij} = p_i|C_{\max}$. В литературе рассматриваются различные частные, в том числе полиномиально разрешимые случаи. Однако, как отмечено в обзоре [4], вопрос о полиномиальной разрешимости задачи с идентичными деталями даже для единичных длительностей работ остаётся открытым.

Будем придерживаться следующей постановки. Имеется заказ на выпуск N идентичных деталей. Для их обработки имеется m различных машин $M_1, M_2, \ldots, M_m$. Все детали в процессе обработки проходят одинаковый технологический маршрут, который состоит из n последовательно выполняемых операций $O_1, O_2, \ldots, O_n$. Операция O_i выполняется на машине M_{m_i} в течение $p_i \in \mathbb{Z}^+$ единиц времени, $i = 1, \ldots, n$. Прерывания операций запрещены. Машина не может выполнять более одной операции одновременно. В технологическом маршруте машины могут повторяться. Требуется минимизировать общее время выполнения заказа.

В классической одномаршрутной задаче число операций n совпадает с числом машин m, а технологический маршрут задаётся последовательностью $(M_1, M_2, \ldots, M_m)$. Для идентичных деталей с таким маршрутом получаем обычный конвейер, и задача решается тривиально. Неоднократное использование машин в технологическом маршруте приводит к различным постановкам задач. В литературе рассматривались случаи V-маршрута $(M_1, M_2, \ldots, M_m, M_{m-1}. \ldots, M_2, M_1)$ [5], циклического $(M_1, M_2, \ldots, M_m, M_1, M_2, \ldots, M_m, \ldots, M_1, M_2, \ldots, M_m)$ [6, 7], замкнутого $(M_1, M_2, \ldots, M_m, M_1)$ [8], с постоянным возвратом на первую машину $(M_1, M_2, M_1, M_3, M_1, \ldots, M_1, M_m, M_1)$ [9, 10] и т. д. [11–13]. Особенно интересен последний случай, в котором машина M_1 рассматривается как транспортное средство, перемещающее деталь между операциями [14, 15].

Важным аспектом является взаимосвязь рассматриваемой задачи с задачей построения циклических расписаний. Циклические расписания строятся за полиномиальное время и обеспечивают ритмичность производства, равномерный выход продукции, более удобную логистику. Наиболее значимым является результат из [16], где показано, что с ростом количества деталей циклическое расписание с минимальным временем цикла обеспечивает асимтотически точное решение для критерия $C_{\max}$. Однако такой подход не решает проблем загрузки оборудования на стадии запуска и стадии завершения обработки партии деталей в производство. Поэтому, если производственная но-

менклатура предприятия разнообразна и партии идентичных деталей относительно невелики, вопрос о взаимосвязи этих задач остается актуальным.

Структура работы следующая. В п. 1 изложен основной результат: доказано, что задача является NP-трудной в обычном смысле. В п. 2 приведён алгоритм построения точного решения, который является псевдополиномиальным при фиксированном числе деталей. В п. 3 обсуждается вопрос об использовании циклических расписаний при минимизации $C_{\max}$, в том числе при наличии дополнительных ограничений на расписание.

1. Вычислительная сложность задачи

Некоторые подходы к анализу сложности задачи предложены в [17]. Входные параметры задачи — это число деталей N, число операций n и их длительности p_i, $i = 1, 2, \ldots, n$. Длина входа равна $O(\log_2 N + n \log_2 p_{\max})$, где $p_{\max}$ — максимальная длительность операций. Далее доказана NP-трудность задачи и сделан анализ её сложности в зависимости от входных параметров.

Теорема 1. Задача $F|\text{reentrant}, p_{ij} = p_i|C_{\max}$ минимизации общего времени обработки однотипных деталей является NP-трудной.

Доказательство. Рассмотрим NP-трудную разномаршрутную задачу $J3|n = 3|C_{\max}$ с тремя машинами M_1, M_2, M_3 и тремя деталями $J1, J2, J3$ [18]. Сформулируем её как задачу распознавания. Задано число d. Существует ли допустимое расписание выполнения работ длины не более d? Полиномиально сведём её к задаче $F|\text{reentrant}, p_{ij} = p_i|C_{\max}$ с тремя идентичными деталями.

Пусть M_1 и M_2 — машины, на которых выполняются соответственно первая и последняя операция детали $J2$. Если эти операции выполняются на одной машине, то обозначим её M_1. Добавим машину M_0 и три фиктивные машины M_3, M_4 и M_5, которые в дальнейшем будут заменены на технологические маршруты деталей $J1$, $J2$ и $J3$. Рассмотрим деталь со следующим технологическим маршрутом и соответствующими длительностями:

O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8	O_9	O_{10}	O_{11}	O_{12}	O_{13}	O_{14}	O_{15}
M_1	M_0	M_1	M_2	M_3	M_2	M_0	M_4	M_0	M_1	M_5	M_1	M_2	M_0	M_2
$4d$	d	$4d$	$4d$	d	$4d$	$4d$	d	$4d$	$4d$	d	$4d$	$4d$	d	$4d$

Пусть необходимо обработать три таких детали. Оптимальное расписание приведено на рис. 1. Его длина равна $63d$. Номера машин подписаны. Наиболее загруженными являются машины M_1 и M_2, и в расписании они представлены симметрично относительно операции обращения времени. Операции каждой детали обрабатываются без простоев, и операция O_1 второй детали выполняется только после завершения операции O_3 первой детали. При любой другой последовательности операций время обработки будет не менее $66d$. Такая конструкция позволяет локализовать операции O_5, O_8 и O_{11} в оптимальном расписании.

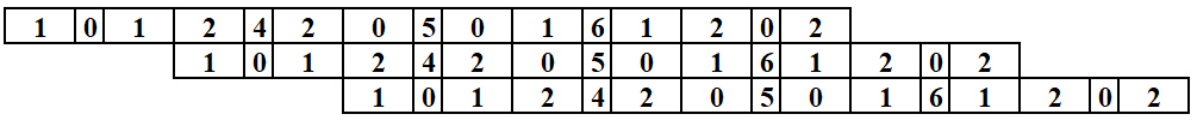


Рис. 1. Оптимальное расписание для трёх деталей

Операции O_5, O_8 и O_{11} заменим соответственно на последовательности операций деталей $J1, J2, J3$. Получим деталь со следующей последовательностью операций:

$$O_1,\ O_2,\ O_3,\ O_4,\ J1,\ O_6,\ O_7,\ J2,\ O_9,\ O_{10},\ J3,\ O_{12},\ O_{13},\ O_{14},\ O_{15}.$$

Заметим, что при замене суммарная длительность операций не увеличилась, так как суммарная длительность операций каждой из деталей $J1, J2, J3$ не превосходит d. Рассмотрим задачу составления расписания для трёх таких деталей. Порядок выполнения операций в расписании минимальной длины показан на схеме рис. 2.

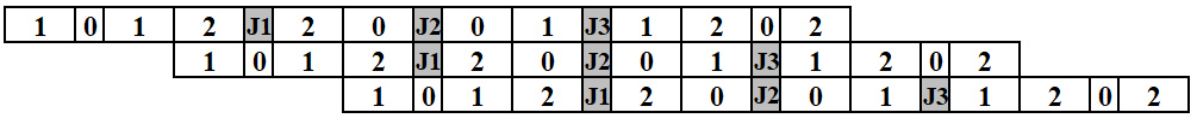


Рис. 2. Схема расположения операций деталей $J1, J2$ и $J3$ в оптимальном расписании

Покажем, что для этой задачи расписание длины не более $63d$ существует тогда и только тогда, когда для разномаршрутной задачи с деталями $J1, J2, J3$ существует расписание длины не более чем d.

Доказательство достаточности очевидно. Если для деталей $J1, J2$ и $J3$ существует расписание длины не более d, то для пар $J1, J2$ и $J2, J3$ тем более существует расписание длины не более d. Следовательно, расписание на рис. 2 имеет длину не более $63d$.

Для доказательства необходимости проведём более детальный анализ. Длину оптимального расписания определяет критический путь — последовательность операций, задержка при выполнении которых приводит к увеличению длины расписания. Операции, входящие в критический путь, обычно называют критическими. На рис. 3 тёмным цветом выделены операции, которые в той или иной ситуации могут входить в критический путь; примеры критических путей приведены на рис. 4 и 5.

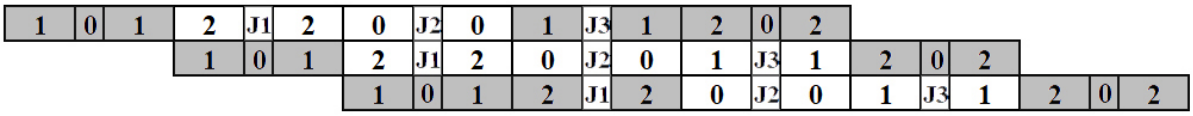


Рис. 3. Множество возможных критических операций

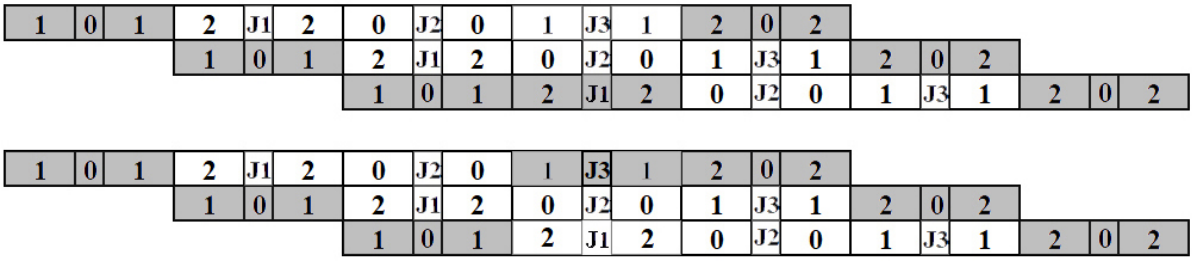


Рис. 4. Критические пути для вариантов $J1 - J1$ и $J3 - J3$

Пусть для деталей $J1, J2$ и $J3$ длина оптимального расписания превосходит d. Покажем, что тогда и длина расписания в задаче с идентичными деталями будет больше чем $63d$. Рассмотрим возможные варианты начала и завершения критического пути в разномаршрутной задаче. Напомним, что первая операция детали $J2$ выполняется на машине M_1, а последняя либо на M_2, либо на M_1. В первом случае имеется

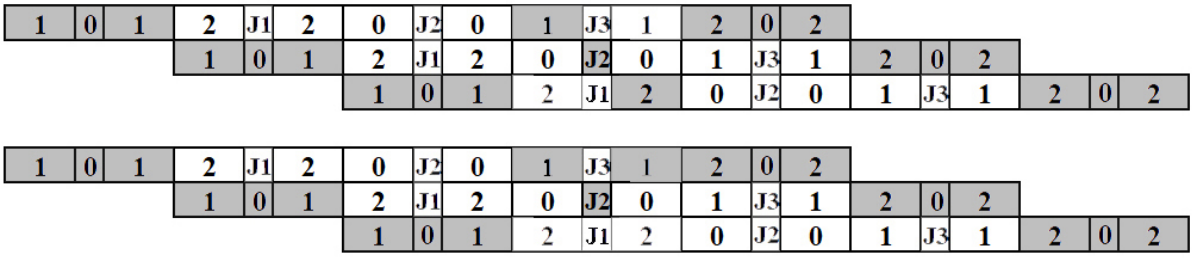


Рис. 5. Критический путь для двух вариантов $J2 - J2$

9 комбинаций первой и последней операций критического пути. К ним добавятся ещё 3 варианта во втором случае. С учётом этого общее количество различных вариантов концов критического пути равно 12. Рассмотрим некоторые из них подробнее.

Пусть критический путь для задачи с деталями $J1, J2, J3$ содержит первую и последнюю операции детали $J1$. Назовём его $J1 - J1$. Тогда выделенная на первой части рис. 4 последовательность операций имеет длину больше чем $63d$. Вариант $J3 - J3$ аналогичен и приведён на второй части рисунка.

Вариант $J2 - J2$. Первая операция критического пути выполняется на машине M_1, последняя — на машине M_2. Выделенная на рис. 5 последовательность операций имеет длину больше чем $63d$. Критический путь в случае, если последняя операция детали $J2$ выполняется на машине M_1, приведён на второй части рисунка.

Все остальные комбинации также дают критический путь, длина которого больше чем $63d$. Действительно, в любом случае слева критический путь упирается в предшествующую операцию, выполняемую на машине 1 или 2, эти операции невозможно сдвинуть на более ранний срок. Справа также всё ограничивают операции, сдвиг которых увеличивает длину критического пути. ■

Отметим, что доказана только обычная NP-трудность рассматриваемой задачи. Ниже предложен алгоритм построения точного решения задачи, который при фиксированном числе деталей является псевдополиномиальным. Более того, на основе этого алгоритма может быть построена вполне полиномиальная аппроксимационная схема [19]. Таким образом, полученный результат является неулучшаемым для случая фиксированного числа работ. Однако вопрос о вычислительной сложности задачи в зависимости от числа работ остаётся открытым.

2. Точный алгоритм решения задачи

Опишем алгоритм построения точного решения рассматриваемой задачи, идея которого предложена в [20] для задачи календарного планирования с ограниченными ресурсами. Далее этот алгоритм адаптирован для задачи с идентичными деталями $F|\text{reentrant}, p_{ij} = p_i|C_{\max}$.

В основе алгоритма лежит геометрическая интерпретация задачи, в которой каждой детали сопоставляется ось в N-мерном пространстве и каждая точка параллелепипеда $[0, P]^N$ соответствует промежуточному состоянию обработки деталей, где $P = \sum_{i=1}^{n} p_i$ — суммарная длительность операций одной детали. Текущее состояние процесса обработки детали j будем задавать величиной x_j. Значение x_j определяет, сколько времени, без учёта простоев, деталь j уже обрабатывалась, и требуется ещё $P - x_j$ единиц времени для завершения обработки. Каждое расписание обработки деталей отображается ломаной линией между точками $\mathbf{0} = (0, 0, \ldots, 0)$, когда ни одна деталь

не начала обработку, и $\mathbf{S} = (P, P, \ldots, P)$, когда все детали обработаны. Любое звено ломаной проходит между точками $(x_1, x_2, \ldots, x_N)$ и $(x_1 + \tau_1, x_2 + \tau_2, \ldots, x_N + \tau_N)$, где $\tau_i = 0$ или $\tau_i = \tau$. Такое звено соответствует одновременной обработке в течение времени τ тех деталей, для которых $\tau_j = \tau$. Остальные детали в этот период не обрабатываются. Длина этого участка ломаной полагается равной τ, а длина расписания равна сумме длин всех звеньев ломаной.

Так как p_i — целые, существует оптимальное расписание, в котором все операции начинаются и заканчиваются в целочисленные моменты времени и для каждого целого t на полуинтервале $(t-1, t]$ множество обрабатываемых деталей не изменяется. Поэтому достаточно рассмотреть только такие расписания, для которых ломаная меняет направление только в целочисленных точках. Целочисленный вектор $\mathbf{x} = (x_1, x_2, \ldots, x_N)$ будем называть *состоянием* обработки деталей. Множество всех состояний обозначим через $X = \{\mathbf{x} = (x_1, x_2, \ldots, x_N) : x_j \in \{0, 1, \ldots, P\}\}$. В такой ситуации ломаную можно разбить на участки единичной длины. Каждый из них соединяет целочисленные точки и соответствует переходу из состояния $\mathbf{x}$ в состояние $\mathbf{x} + \delta$, где $\delta = (\delta_1, \delta_2, \ldots, \delta_N)$, $\delta_j \in \{0, 1\}$, $j = 1, 2, \ldots, N$. Процесс перехода соответствует одновременному выполнению в единичном временном интервале операций тех деталей, для которых $\delta_j = 1$.

Идея алгоритма основана на схеме динамического программирования и заключается в переборе всевозможных состояний, а для каждого состояния по рекуррентной формуле вычисляется лучшее допустимое частичное расписание.

Имеется два типа ограничений, которые необходимо учесть. Первое — это непрерывность выполнения операций. Для состояния $\mathbf{x}$ выполнение условия

$$\sum_{k=1}^{i-1} p_k < x_j < \sum_{k=1}^{i-1} p_k + p_i$$

означает, что операция i детали j была начата и ещё не завершилась. Тогда, в силу непрерывности выполнения операций, δ_j должно быть равно 1. Поэтому для такого $\mathbf{x}$ переход $\mathbf{x} \to \mathbf{x} + \delta$, у которого $\delta_j = 0$, недопустим.

Второе — это запрет на одновременное выполнение двух операций на одной машине. Состояние $\mathbf{x}$ и единичные компоненты перехода δ однозначно определяют набор операций, которые находятся в состоянии выполнения. Если хотя бы одна пара из этого набора выполняется на одной и той же машине, то переход δ недопустим.

Множество допустимых переходов, приводящих в состояние $\mathbf{x} \in \mathbf{X}$, обозначим через $\Delta_{\mathbf{x}}$. Каждый переход соответствует выполнению некоторого набора операций в течение единичного интервала времени. Необходимо найти кратчайшую последовательность допустимых переходов из состояния $\mathbf{0}$ в состояние $\mathbf{S}$.

Пусть $L(\mathbf{x})$ — наименьшее число переходов из состояния $\mathbf{0}$ в состояние $\mathbf{x}$. Выпишем рекуррентное соотношение для поиска кратчайшего пути:

$$L(\mathbf{x}) = \min_{\delta \in \Delta_{\mathbf{x}}} \{L(\mathbf{x} - \delta) + 1\}, \quad \mathbf{x} \in X \setminus \{\mathbf{0}\}.$$

Алгоритм начинает работу с начального состояния $\mathbf{0}$ и в порядке лексикографического возрастания перебирает все отмеченные состояния $\mathbf{x} \in X$, вычисляя значения $L(\mathbf{x})$. При этом запоминает вектор $\delta(\mathbf{x}) \in \Delta_{\mathbf{x}}$, на котором достигается минимум $L(\mathbf{x})$. Величина $L(\mathbf{S})$ определяет оптимальное значение целевой функции задачи.

Для восстановления решения первоначально полагаем $\mathbf{x} = \mathbf{S}$ и рассматриваем переход $\delta(\mathbf{S})$. Во временном интервале $(L(\mathbf{S}) - 1, L(\mathbf{S})]$ выполняются операции тех деталей,

для которых $\delta_j(\mathbf{S}) = 1$. Затем переходим в состояние $\mathbf{x} = \mathbf{S} - \delta(\mathbf{S})$. Во временном интервале $(L(\mathbf{S})-2, L(\mathbf{S})-1]$ выполняются операции тех деталей, для которых $\delta_j(\mathbf{x}) = 1$. И так далее, пока не дойдём до состояния $\mathbf{x} = \mathbf{0}$. Зная множества операций, выполняемых в каждом из интервалов $(t-1, t]$, $t = 1, 2, \ldots, L(\mathbf{S})$, определяем моменты начала их выполнения.

Трудоёмкость алгоритма линейно зависит от количества состояний. Для состояния $\mathbf{x}$ нужно выбрать минимум из $\Delta_{\mathbf{x}}$ чисел. Верхняя оценка трудоёмкости составит $O(2^N P^N) = O((2P)^N)$. Для идентичных деталей эту оценку можно улучшить. Без ограничения общности детали запускаются в порядке возрастания номеров. Очевидно, что в оптимальном расписании такой же порядок сохранится при выполнении соответствующих операций всех деталей. Поэтому в любой момент имеет место $x_1 > x_2 > \ldots > x_N$. Это позволяет существенно сократить число перебираемых состояний. В случае $N < P$ верхняя оценка может быть уменьшена до $O((2P)^N/N!)$.

Временная сложность алгоритма экспоненциально зависит от количества деталей N. Число машин M не влияет на трудоёмкость, так как в алгоритме для каждой пары операций делается простая проверка, обрабатываются эти операции на одной машине или нет. Если число деталей фиксировать, то предложенный алгоритм становится псевдополиномиальным. Таким образом, справедлива следующая

Теорема 2. Задача минимизации общего времени обработки партии идентичных деталей при условии, что количество обрабатываемых деталей ограничено константой, разрешима за псевдополиномиальное время.

Отметим, что этот вывод может быть также получен из [21] на основе псевдополиномиальной разрешимости задачи $J|N = \text{const}\,|C_{\max}$. При $N = 2$ задача полиномиально разрешима. Уже при $N = 3$ рассматриваемая задача становится NP-трудной в обычном смысле, что показано выше.

В реальных задачах NP-трудность в зависимости от длительностей не является критической, так как трудно представить ситуацию очень длинных технологических маршрутов при обработке детали. А вот количество деталей может меняться и составлять несколько сотен, а возможно, и тысячи [17]. Поэтому важно исследование вычислительной сложности задачи в зависимости от параметра N. Для задачи с идентичными деталями длина входа по этому параметру составляет $O(\log_2 N)$. То есть полиномиальность по этому параметру предполагает наличие полиномиального алгоритма от величины $O(\log_2 N)$. Как вариант, трудоёмкость алгоритма может не зависеть от N. В следующем пункте исследуем возможность использования при минимизации $C_{\max}$ циклических расписаний. Алгоритм построения расписаний с минимальным временем цикла является полиномиальным и его трудоёмкость не зависит от N.

3. Использование циклических расписаний при минимизации общего времени

Расписание называется циклическим, если выполнение соответствующих операций любых двух последовательно обрабатываемых деталей происходит через промежуток времени C, который называется длиной цикла. Минимально возможная длина цикла равна суммарной длительности операций одной детали на самой загруженной машине. Цикл называется полным, если внутри него выполняются все операции $\{1, 2, \ldots, n\}$, возможно, разных деталей. Циклические расписания обеспечивают ритмичность производства, равномерный выход продукции, более удобную логистику. При минимальном времени цикла C и росте числа идентичных деталей эффективность производства возрастает и его производительность стремится к максимально возможной. Более того,

циклическое расписание с минимальным временем цикла может быть найдено за полиномиальное время. Естественным образом возникает идея использования малотрудоёмкого алгоритма построения циклических расписаний для решения NP-трудной задачи минимизации общего времени выполнения заказа. На основе таких расписаний в [16] построен асимптотически точный алгоритм для критерия $C_{\max}$. Недостатком этого алгоритма и циклических расписаний является то, что максимальная загрузка оборудования достигается только в полном цикле. При запуске партии деталей и её завершении формируются неполные циклы и происходит простой оборудования. Для сложного технологического маршрута обработки детали такие простои могут быть довольно значительными. При больших размерах партии (более нескольких тысяч) они не так чувствительны, так как их доля при росте N стремится к нулю. Но когда требуется обработать несколько десятков или сотен сложных деталей, то временные потери могут составлять существенную часть общего времени обработки. В этом случае асимптотика ещё не дает эффекта, а описанный выше псевдополиномиальный алгоритм уже не работает из-за высокой трудоёмкости. Приведём небольшой пример:

$$\begin{pmatrix} O_1 & O_2 & O_3 & O_4 & O_5 & O_6 & \dots & O_{2k-1} & O_{2k} & O_{2k+1} \\ M_0 & M_1 & M_0 & M_2 & M_0 & M_3 & \dots & M_0 & M_k & M_0 \\ 1 & p_1 & 1 & p_2 & 1 & p_3 & \dots & 1 & p_k & 1 \end{pmatrix}.$$

Здесь $p = \sum\limits_{i=1}^{k} p_i \leqslant 1$. Минимальное значение длины цикла определяется загрузкой машины M_0 и равно $C^* = k + 1$. Но полные циклы, в которых выполняется весь набор операций, начинаются только с $(k+1)$-й детали. Выполнив первую и вторую операцию первой детали, мы не может выполнять третью операцию на машине M_0, хотя она свободна. Ждём k единиц времени и только тогда запускаем вторую деталь. Только когда для первых k деталей будут завершены операции O_1, начнёт выполнение операция O_3 первой детали. Далее загрузка машины M_0 будет максимально возможной. Но в итоге самая дефицитная машина в начальной стадии процесса, как и в заключительной стадии, простаивает в общей сложности k^2 единиц времени. Таким образом, главным недостатком циклических расписаний являются потери, связанные с запуском и завершением всей партии деталей.

Естественно возникает желание уплотнить неполные циклы. В рассмотренном примере сжатие циклического расписания действительно даёт эффект и приводит к оптимальному значению $C_{\max}$. Далее описан алгоритм 1 уплотнения циклического расписания.

Алгоритм 1 является полиномиальным. Так как алгоритм построения расписания с минимальной длиной цикла также полиномиален, то получить таким образом оптимум с критерием $C_{\max}$ невозможно, это противоречило бы теореме 1. Более того, если рассмотреть всё множество расписаний с минимальным временем цикла и каждое из них уплотнить, то ни одно из этих расписаний может не дать оптимума. Пример очень простой:

$$\begin{pmatrix} O_1 & O_2 & O_3 & O_4 \\ M_1 & M_2 & M_3 & M_1 \\ 1 & 1 & 1 & 1 \end{pmatrix}.$$

Минимальная длина цикла $C = 2$. При $N \geqslant 4$ любое циклическое расписание (с учётом возможного сжатия) имеет длину $2N + 2$. А оптимум достигается на расписании рис. 6 и имеет длину $2N$. При $N = 4$ отклонение от оптимума — в 1,25 раза. Это означает, что

Алгоритм 1. Уплотнение циклического расписания

1: Строим циклическое расписание с минимальным временем цикла.
2: Выделяем первый цикл с полным набором операций.
3: Операции, входящие в этот и предшествующие циклы, упорядочиваем по невозрастанию времени окончания.
4: Просматриваем список с конца: каждую операцию сдвигаем как можно позднее, пока не нарушится отношение предшествования или две операции не поступят на одну машину. Получим плотное расписание, когда ни одна из операций не может быть сдвинута на более поздний срок.
5: Аналогично при завершении заказа. Берём последний цикл с полным набором операций. Операции этого и всех последующих циклов упорядочиваем по возрастанию моментов их начала.
6: Просматриваем список с начала, каждую операцию сдвигаем как можно раньше, до первого нарушения отношения предшествования или поступления двух операций на одну машину. Получим плотное расписание, когда ни одна операция не может быть сдвинута на более ранний срок.

для критерия минимизации общего времени обработки деталей в классе циклических расписаний нельзя гарантировать решение с оценкой точности лучше чем 1,25.

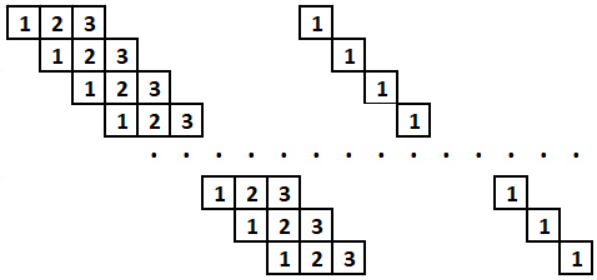


Рис. 6. Оптимальное расписание

Если исследовать вопрос о приближении, то значение $C_{\max}$ можно оценить сверху величиной $CN + C(n-1)$, так как количество неполных циклов не превосходит $2C(n-1)$, а полных циклов будет $(N-n+1)$. Эта оценка может быть улучшена до значения $CN + C(k-1)$, где k — количество обрабатываемых деталей в одном цикле. Логично было бы предположить, что чем меньше деталей задействовано в цикле, тем лучше должно быть расписание после уплотнения. Это хорошо согласуется с прежними результатами из [22], где доказана псевдополиномиальная разрешимость задачи минимизации числа деталей при условии оптимальной длины цикла. Однако удалось построить пример, когда это не так.

Пусть в предыдущем примере длительности операций равны 2, 3, 3 и 2 единицы соответственно. На первой части рис. 7 приведено циклическое расписание для $N = 5$ с минимально возможным числом деталей в цикле, равным трём. Циклы отделены вертикальными линиями. Два первых и два последних цикла неполные. На второй части рис. 7 отображено расписание после уплотнения. Его длина равна 25. На рис. 8 приведено расписание, когда в цикле минимальной длины обрабатываются четыре детали. После его уплотнения длина расписания равна 21. Тем самым число деталей в цикле

не является определяющим для поиска лучшего приближения в классе циклических расписаний.

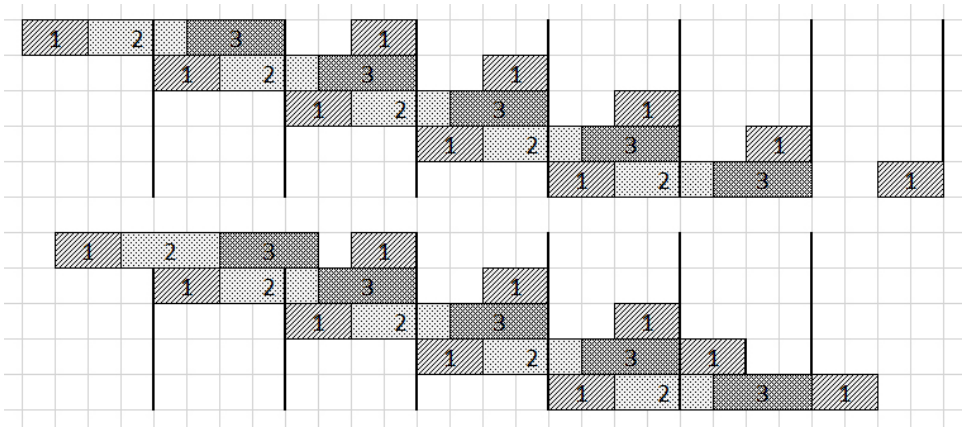


Рис. 7. Расписание для трёх деталей в цикле

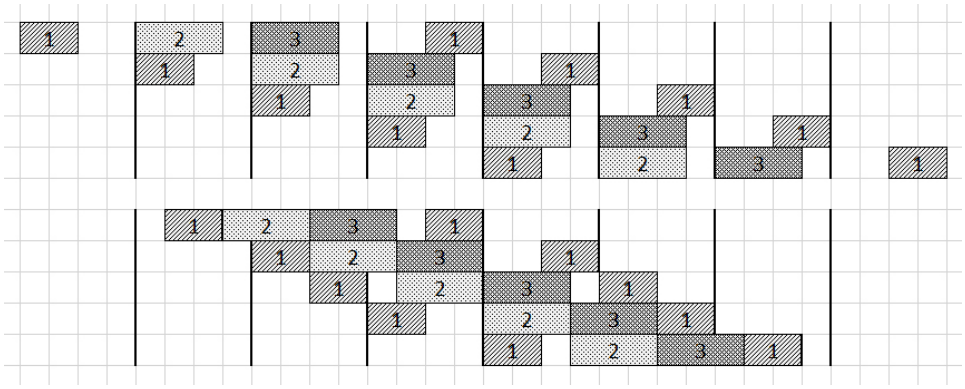


Рис. 8. Расписание для четырёх деталей в цикле

Пока непонятно, какое циклическое расписание приведёт к самой плотной упаковке. Но, как сказано ранее, в классе алгоритмов, включающих построение циклического расписания с дальнейшим уплотнением, достижение оптимума не гарантировано. Это косвенно говорит о том, что невозможно построить алгоритм минимизации $C_{\max}$, трудоёмкость которого не зависит от числа деталей.

Циклические расписания могут существенно помочь, когда количество деталей, одновременно находящихся в обработке, ограничено некоторой величиной [23]. Приведённый в начале пункта пример с $(2k+1)$ деталями особенно важен в случае, когда число деталей, одновременно находящихся в обработке, ограничено некоторой величиной H. Для обеспечения наибольшей производительности линии необходимо запускать в производство сразу $k+1$ деталей, и эта величина может существенно превосходить H. Пример циклического расписания для детали с шестью операциями при $H=2$ приведён на рис. 9. Операции O_1, O_3, O_5 выполняются на первой машине, а O_2, O_4, O_6 — на второй. Минимально возможная длина цикла равна суммарной длительности операций одной детали на самой загруженной машине.

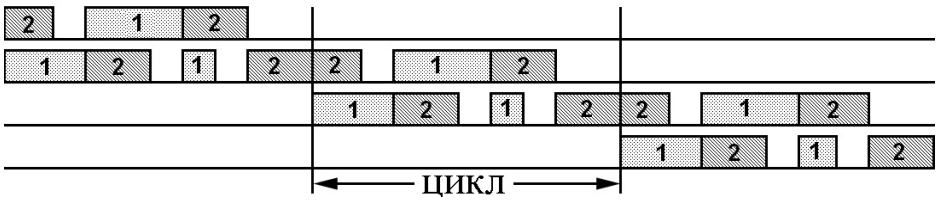


Рис. 9. Фрагмент циклического расписания

Теорема 3. При $H = 2$ циклическое расписание с минимальным временем цикла после уплотнения даёт оптимальное решение для критерия $C_{\max}$.

Доказательство основано на том, что при $H = 2$ операции, не вошедшие в основной цикл, удаётся уплотнить, и при этом достигается нижняя оценка оптимального времени обработки партии деталей. При $H = 2$ задача полиномиально разрешима.

Следствие 1. Задача $F|\text{reentrant}, p_{ij}{=}p_i, H{=}2|C_{\max}$ полиномиально разрешима.

При $H = 3$ операции, не вошедшие в полный цикл, не всегда удаётся уплотнить до состояния, когда будет достигнута нижняя оценка. Кроме того, вопрос о вычислительной сложности задачи построения оптимального циклического расписания для $H = 3$ до сих пор остаётся открытым.

Заключение

Исследуется вычислительная сложность задачи минимизации общего времени обработки идентичных деталей со сложным технологическим маршрутом, когда возможно неоднократное поступление деталей на некоторые машины. Доказано, что задача $F|\text{reentrant}, p_{ij} = p_i|C_{\max}$ является NP-трудной в обычном смысле. Предложен алгоритм построения точного решения задачи. При фиксированном числе деталей N алгоритм является псевдополиномиальным и на его основе может быть построена вполне полиномиальная аппроксимационная схема. Если зафиксировать технологический маршрут и длительности операций, то в задаче остаётся единственный параметр — число деталей N, и длина входа составит $O(\log_2 N)$. Такая ситуация более характерна для реального производства. В этом случае вопрос о вычислительной сложности остаётся открытым. Исследуется взаимосвязь данной задачи с задачей построения циклических расписаний с минимальным временем цикла, которая полиномиально разрешима. Построены оценки возможного отклонения решения, полученного в классе циклических расписаний, от оптимального значения для критерия $C_{\max}$.

ЛИТЕРАТУРА

1. *Pinedo M. L.* Scheduling. Theory, Algorithms, and Systems. Springer, 2008. 671 p.
2. *Graves S. C., Meal H. M., Stefek D., and Zeghmi A. H.* Scheduling of re-entrant flow shops // J. Oper. Management. 1983. V. 3. No. 4. P. 197–207.
3. *Emmons H. and Vairaktarakis G.* Flow Shop Scheduling: Theoretical Results, Algorithms, and Applications. Springer Science & Business Media, 2012. 334 p.
4. *Shufan E., Grinshpoun T., Ikar E., and Ilani H.* Reentrant flow shop with identical jobs and makespan criterion // J. Production Res. 2021. V. 61. No. 1. P. 183–197.
5. *Lev V. and Adiri I.* V-shop scheduling // Europ. J. Oper. Res. 1984. V. 18. No. 1. P. 51–56.
6. *Pan J. C.-H. and Chen J.-S.* Minimizing makespan in re-entrant permutation flow-shops // J. Oper. Res. Soc. 2003. V. 54. No. 6. P. 642–653.
7. *Chen J.-S.* A branch and bound procedure for the reentrant permutation flowshop scheduling problem // Intern. J. Adv. Manufacturing Technology. 2006. V. 29. P. 1186–1193.

8. *Wang M. Y., Suresh P. S., and van de Velde S. L.* Minimizing makespan in a class of reentrant shops // Oper. Res. 1997. V. 45. No. 5. P. 702–712.
9. *Kubiak W., Lou Sh. X. C., and Wang Y.* Mean flow time minimization in reentrant job shops with a hub // Oper. Res. 1996. V. 44. No. 5. P. 743–753.
10. *Xie X., Tang L., and Li Y.* Scheduling of a hub reentrant job shop to minimize makespan // Intern. J. Adv. Manufacturing Technology. 2011. V. 56. P. 743–753.
11. *Middendorf M. and Timkovsky V. G.* On scheduling cycle shops: Classification, complexity and approximation // J. Scheduling. 2002. V. 5(2). P. 135–169.
12. *Timkovsky V. G.* Cycle Shop Scheduling / Leung J. Y.-T. (ed.). Handbook of Scheduling. Ch. 7. Boca Raton; London; N.Y.; Washington, CRC Press, 2004.
13. *Yu T.-S. and Pinedo M. L.* Flow shops with reentry: Reversibility properties and makespan optimal schedules // Europ. J. Oper. Res. 2021. V. 282(2). P. 478–490.
14. *Kats V. and Levner E.* Minimizing the number of robots to meet a given cyclic schedule // Ann. Oper. Res. 1997. V. 69. P. 209–226.
15. *Kats V. and Levner E.* Cyclic scheduling in a robotic production line // J. Scheduling. 2002. V. 5(1). P. 23–41.
16. *Boudoukh T., Penn M., and Weiss G.* Scheduling jobshops with some identical or similar jobs // J. Scheduling. 2001. V. 4(4). P. 177–199.
17. *Межецкая М. А., Сервах В. В.* Задачи обработки деталей со сложным технологическим маршрутом // Современные проблемы науки и образования. 2013. № 1. `https://science-education.ru/ru/article/view?id=8407`.
18. *Sotskov Y. N. and Shakhlevich N. V.* NP-hardness of shop-scheduling problems with three jobs // Discrete Appl. Math. 1995. V. 59(3). P. 237–266.
19. *Servakh V. V. and Shcherbinina T. A.* A fully polynomial time approximation scheme for two project scheduling problems // IFAC Proc. Volumes. 2006. V. 39. Iss. 3. P. 131–135.
20. *Сервах В. В.* Эффективно разрешимый случай задачи календарного планирования с возобновимыми ресурсами // Дискретн. анализ исслед. опер. Сер. 2. 2000. Т. 7. № 1. С. 75–82.
21. *Middendorf M. and Timkovsky V. G.* Transversal graphs for partially ordered sets: Sequencing, merging and scheduling problems // J. Combin. Optimization. 1999. V. 3. No. 4. P. 417–435.
22. *Romanova A. A. and Servakh V. V.* Optimization of identical jobs production on the base of cyclic schedules // J. Appl. Industr. Math. 2009. V. 3. No. 4. P. 496–504.
23. *Боброва Е. А., Романова А. А., Сервах В. В.* Сложность задачи построения циклических расписаний обработки однотипных деталей // Дискретн. анализ исслед. опер. 2013. Т. 20. № 4. С. 3–14.

REFERENCES

1. *Pinedo M. L.* Scheduling. Theory, Algorithms, and Systems. Springer, 2008. 671 p.
2. *Graves S. C., Meal H. M., Stefek D., and Zeghmi A. H.* Scheduling of re-entrant flow shops. J. Oper. Management, 1983, vol. 3, no. 4, pp. 197–207.
3. *Emmons H. and Vairaktarakis G.* Flow Shop Scheduling: Theoretical Results, Algorithms, and Applications. Springer Science & Business Media, 2012. 334 p.
4. *Shufan E., Grinshpoun T., Ikar E., and Ilani H.* Reentrant flow shop with identical jobs and makespan criterion. J. Production Res., 2021, vol. 61, no. 1, pp. 183–197.
5. *Lev V. and Adiri I.* V-shop scheduling. Europ. J. Oper. Res., 1984, vol. 18, no. 1, pp. 51–56.
6. *Pan J. C.-H. and Chen J.-S.* Minimizing makespan in re-entrant permutation flow-shops. J. Oper. Res. Soc., 2003, vol. 54, no. 6, pp. 642–653.

7. *Chen J.-S.* A branch and bound procedure for the reentrant permutation flowshop scheduling problem. Intern. J. Adv. Manufacturing Technology, 2006, vol. 29, pp. 1186–1193.
8. *Wang M. Y., Suresh P. S., and van de Velde S. L.* Minimizing makespan in a class of reentrant shops. Oper. Res., 1997, vol. 45, no. 5, pp. 702–712.
9. *Kubiak W., Lou Sh. X. C., and Wang Y.* Mean flow time minimization in reentrant job shops with a hub. Oper. Res., 1996, vol. 44, no. 5, pp. 743–753.
10. *Xie X., Tang L., and Li Y.* Scheduling of a hub reentrant job shop to minimize makespan. Intern. J. Adv. Manufacturing Technology, 2011, vol. 56, pp. 743–753.
11. *Middendorf M. and Timkovsky V. G.* On scheduling cycle shops: Classification,complexity and approximation. J. Scheduling, 2002, vol. 5(2), pp. 135–169.
12. *Timkovsky V. G.* Cycle Shop Scheduling. Leung J. Y.-T. (ed.). Handbook of Scheduling. Ch. 7. Boca Raton; London; N.Y.; Washington, CRC Press, 2004.
13. *Yu T.-S. and Pinedo M. L.* Flow shops with reentry: Reversibility properties and makespan optimal schedules. Europ. J. Oper. Res., 2021, vol. 282(2), pp. 478–490.
14. *Kats V. and Levner E.* Minimizing the number of robots to meet a given cyclic schedule. Ann. Oper. Res., 1997, vol. 69, pp. 209–226.
15. *Kats V. and Levner E.* Cyclic scheduling in a robotic production line. J. Scheduling, 2002, vol. 5(1), pp. 23–41.
16. *Boudoukh T., Penn M., and Weiss G.* Scheduling jobshops with some identical or similar jobs. J. Scheduling, 2001, vol. 4(4), pp. 177–199.
17. *Mezhetskaya M. A. and Servakh V. V.* Zadachi obrabotki detaley so slozhnym tekhnologicheskim marshrutom [The shop scheduling problems with complex technological route]. Sovremennye Problemy Nauki i Obrazovaniya, 2013, no. 1, `https://science-education.ru/ru/article/view?id=8407`. (in Russian)
18. *Sotskov Y. N. and Shakhlevich N. V.* NP-hardness of shop-scheduling problems with three jobs. Discrete Appl. Math., 1995, vol. 59(3), pp. 237–266.
19. *Servakh V. V. and Shcherbinina T. A.* A fully polynomial time approximation scheme for two project scheduling problems. IFAC Proc. Volumes, 2006, vol. 39, iss. 3, pp. 131–135.
20. *Servakh V. V.* Effektivno razreshimyy sluchay zadachi kalendarnogo planirovaniya s vozobnovimymi resursami [An efficiently solvable case of the scheduling problem with renewable resources]. Diskretnyy Analiz i Issledovaniye Operatsiy, Ser. 2, 2000, vol. 7, no. 1, pp. 75–82. (in Russian)
21. *Middendorf M. and Timkovsky V. G.* Transversal graphs for partially ordered sets: Sequencing, merging and scheduling problems. J. Combinat. Optimization, 1999, vol. 3, no. 4, pp. 417–435.
22. *Romanova A. A. and Servakh V. V.* Optimization of identical jobs production on the base of cyclic schedules. J. Appl. Industr. Math., 2009, vol. 3, no. 4, pp. 496–504.
23. *Bobrova E. A., Romanova A. A., and Servakh V. V.* Slozhnost zadachi postroyeniya tsiklicheskikh raspisaniy obrabotki odnotipnykh detaley [Complexity of cyclic scheduling for identical jobs.] Diskretnyy Analiz i Issledovaniye Operatsiy, 2013, vol. 20, no. 4, pp. 3–14. (in Russian)

УДК 519.17+157 DOI 10.17223/20710410/64/9

ПОДХОД К АНАЛИЗУ И ПОСТРОЕНИЮ АЛГОРИТМОВ РЕШЕНИЯ ОДНОЙ ЗАДАЧИ КЛАСТЕРИЗАЦИИ НА ЗНАКОВЫХ ГРАФАХ[1]

А. А. Солдатенко, Д. В. Семенова, Э. И. Ибрагимова

Сибирский федеральный университет, г. Красноярск, Россия

E-mail: ASoldatenko@sfu-kras.ru, DVSemenova@sfu-kras.ru, IbragimovaEI@mail.ru

Рассматривается NP-трудная оптимизационная задача корреляционной кластеризации для неориентированных и невзвешенных знаковых графов без кратных рёбер и петель, где функционал ошибки представляет собой линейную комбинацию межкластерной и внутрикластерной ошибок. Предложен системный подход построения и анализа алгоритмов, основанных на структуре графа, для решения этой задачи. Подход представлен в виде общей схемы, состоящей из шести взаимосвязанных блоков, отражающих основные этапы решения задачи корреляционной кластеризации. С использованием данной схемы проанализированы шесть существующих алгоритмов. Согласно общей схеме построен новый алгоритм `CarVeR`, который является модификацией алгоритма $\texttt{SGClust}_\alpha$ с помощью потенциальных функций. Топология общей схемы открывает возможности для анализа и доказательства вычислительной сложности алгоритмов, что продемонстрировано в теореме о вычислительной сложности алгоритма `CarVeR`. Представлены вычислительные эксперименты на синтетических данных для сравнения пяти алгоритмов. Результаты экспериментов показали конкурентную способность алгоритма `CarVeR` как по времени выполнения, так и по минимизации значения функционала ошибки.

Ключевые слова: *знаковый граф, корреляционная кластеризация, систематизация алгоритмов, потенциальные функции.*

APPROACH TO ANALYSIS AND CONSTRUCTION OF ALGORITHMS FOR SOLVING ONE CLUSTERING PROBLEM ON SIGNED GRAPHS

A. A. Soldatenko, D. V. Semenova, E. I. Ibragimova

Siberian Federal University, Krasnoyarsk, Russia

We consider the NP-hard correlation clustering problem for undirected and unweighted signed graphs without multiple edges and loops, where the error functional is a linear combination of intercluster and intracluster errors. In this paper, we propose a systematic approach for constructing and analyzing graph structure based algorithms to solve this problem. The approach is presented in the form of a general scheme consisting of six interrelated blocks reflecting the main stages of solving the correlation clustering problem. Six existing algorithms have been analyzed using this scheme. According to the general scheme, a new algorithm `CarVeR` has been constructed, which is a modification of the $\texttt{SGClust}_\alpha$ algorithm using potential functions. The topology

[1]Работа поддержана Красноярским математическим центром, финансируемым Минобрнауки РФ (Соглашение № 075-02-2024-1429).

of the general scheme opens up the possibility of analyzing and proving the computational complexity of the algorithms, which is demonstrated in the computational complexity theorem of the CarVeR algorithm. This paper presents computational experiments on synthetic data to compare five algorithms. The experimental results show the competitive ability of the CarVeR algorithm both in terms of execution time and minimization of the value of the error functional.

Keywords: *signed graph, correlation clustering, algorithm systematization, potential functions.*

Введение

На протяжении десятилетий исследователи активно изучают задачу корреляционной кластеризации и предлагают различные методы ее решения. В западной литературе данная задача носит название Correlation Clustering problem. Исторический обзор по данной проблеме приведён в [1], а достаточно подробный обзор существующих методов решения представлен в [2].

Алгоритмы решения задачи корреляционной кластеризации можно условно разделить на три группы [2]. Первая группа алгоритмов учитывает структуру графа [2, 3]. Для второй группы характерно представление задачи корреляционной кластеризации как задачи математического программирования (например, линейного, целочисленного линейного, полуопределённого и др.) и использование соответствующих методов и алгоритмов для решения задачи [2, 4–8]. Третья группа основана на различных матричных представлениях графа, что позволяет применять в алгоритмах аппарат матричной алгебры [2, 9]. Далее внимание акцентировано на первой группе алгоритмов.

Структура работы следующая. В п. 1 приведены необходимые для дальнейшего изложения определения и формулировка задачи корреляционной кластеризации знаковых графов. В п. 2 исследованы популярные алгоритмы её решения, основанные на структуре графа, и предложена общая схема построения и анализа таких алгоритмов. В п. 3 исследуется новый алгоритм CaRVeR. Результаты вычислительных экспериментов по сравнению алгоритма CaRVeR с некоторыми известными алгоритмами представлены в п. 4.

1. Постановка задачи

1.1. Знаковый граф

В работе исследуются знаковые графы вида $\Sigma = (G, \sigma)$, где $G = (V, E)$ является неориентированным невзвешенным графом без кратных рёбер и петель с множеством вершин V, $|V| = n \geqslant 2$, и множеством рёбер E, $|E| = m \geqslant 1$. В графе G каждое ребро однозначно представляется неупорядоченной парой $e = (u, v)$, где $e \in E$, $u, v \in V$. В этом случае говорят, что ребро e инцидентно вершинам u и v, а вершины u и v смежны. Обозначим множество вершин, смежных с v, как $\Gamma(v) = \{u\colon (v, u) \in E\}$. Под степенью вершины v понимается число рёбер, инцидентных ей. Очевидно, что $\delta(v) = |\Gamma(v)|$; степенью графа будем считать $\Delta = \max\limits_{v \in V} \delta(v)$. На рёбрах $(u, v) \in E$ графа G задана функция знака $\sigma : E \to \{+, -\}$, которая порождает разбиение множества рёбер графа $E = E^+ \cup E^-$, где E^+ — множество положительных, E^- — множество отрицательных рёбер. Для ребра $e = (u, v)$ функция знака представима в виде

$$\sigma(u, v) = \operatorname{sign}\Big(\big[(u, v) \in E^+\big] - \big[(u, v) \in E^-\big]\Big),$$

где $[\cdot]$ — скобки Айверсона [10].

Знаковый граф называется k-сбалансированным, если множество его вершин можно разбить на k попарно непересекающихся непустых подмножеств так, что все положительные рёбра находятся внутри, а отрицательные — между подмножествами [11].

1.2. Задача корреляционной кластеризации

Обозначим систему множеств, образующих разбиение множества вершин V на k подмножеств, как

$$\mathcal{C} = \left\{ C_i \subseteq V : \bigcup_{i=1}^{k} C_i = V,\ C_i \cap C_j = \varnothing, i \neq j;\ i = 1, \ldots, k \right\}. \tag{1}$$

Известно, что для произвольного знакового графа свойство k-сбалансированности может не выполняться. В этом случае интересен поиск такого разбиения множества вершин графа, для которого число отрицательных рёбер внутри подмножеств и число положительных рёбер между подмножествами будут минимальны. Данная задача рассматривается как задача кластеризации графа со специальным видом функционала ошибки. Элементы разбиения $C_i \in \mathcal{C}$ будем называть кластерами.

Под положительной ошибкой $P(\mathcal{C})$ разбиения (1) будем понимать число положительных рёбер между подмножествами $C_1, \ldots, C_k$. Заметим, что $P(\mathcal{C})$ — это межкластерная ошибка, вычисляемая по формуле

$$P(\mathcal{C}) = \sum_{i=1}^{k} \sum_{u \in C_i} \sum_{v \in V \setminus C_i} [(u, v) \in E^+]\,. \tag{2}$$

Под отрицательной ошибкой $N(\mathcal{C})$ будем понимать число отрицательных рёбер внутри подмножеств для разбиения (1). Отрицательная ошибка — это внутрикластерная ошибка, вычисляемая по формуле

$$N(\mathcal{C}) = \sum_{i=1}^{k} \sum_{\{u,v\} \subseteq C_i} [(u, v) \in E^-]\,. \tag{3}$$

В [12] авторы предлагают представлять суммарную ошибку в виде выпуклой комбинации положительной и отрицательной ошибок, зависящей от параметра $\alpha \in [0, 1]$:

$$Q_\alpha(\mathcal{C}) = \alpha N(\mathcal{C}) + (1 - \alpha) P(\mathcal{C}). \tag{4}$$

Заметим, что функционал ошибки (4) всегда удовлетворяет неравенству

$$0 \leqslant Q_\alpha(\mathcal{C}) \leqslant \alpha |E^-| + (1 - \alpha) |E^+|.$$

Задачу кластеризации знакового графа будем рассматривать в следующей постановке [13, 14].

Correlation Clustering problem (задача CC)

Условие: задан знаковый граф $\Sigma = (G, \sigma)$, где $G = (V, E)$ — неориентированный граф; $n = |V| \geqslant 2$; $m = |E| \geqslant 1$.

Вопрос: для заданного $\alpha \in [0, 1]$ требуется найти разбиение $\mathcal{C}$ множества вершин V знакового графа Σ с минимальной суммарной ошибкой $Q_\alpha(\mathcal{C})$.

В работе [13] показано, что задача корреляционной кластеризации знаковых графов с функционалом ошибки в виде (4) при $\alpha = 0{,}5$ в распознавательной форме является NP-полной.

Решением задачи является множество кластеров $\mathcal{C}^*$, доставляющих минимум функционалу ошибки (4):

$$\mathcal{C}^* = \arg\min_{\mathcal{C}\in\Phi}\big[\alpha N(\mathcal{C}) + (1-\alpha)P(\mathcal{C})\big], \tag{5}$$

где $\Phi = \bigcup\limits_{k=1}^{n} \Phi_k$ — множество всех возможных разбиений V; Φ_k — множество разбиений на k подмножеств. Мощность пространства решений Φ равна числу Белла B_n. Следует отметить, что решение (5) может быть не единственным.

При $\alpha = 0$ и 1 данная задача вырождается в полиномиально разрешимые случаи минимизации межкластерной (2) и внутрикластерной (3) ошибок соответственно.

Одна из стратегий поиска нетривиального решения задачи корреляционной кластеризации знаковых графов для $\alpha = 0$ формулируется следующим образом. Множество кластеров $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ формируется из исходного знакового графа Σ путём нахождения компонент связности $C_1, C_2, \ldots, C_k$ порождённого графа $\Sigma^+ = (V, E^+)$. Полученное разбиение имеет ошибку $Q_0(\mathcal{C}) = 0$. Такая стратегия может быть реализована алгоритмами поиска в глубину или ширину, временна́я сложность которых составляет $\mathcal{O}(n + |E^+|)$ [15]. Для поиска нетривиального решения с параметром $\alpha = 1$ можно применить следующую стратегию. Изначально полагается, что все вершины находятся в одном кластере. Далее на каждом шаге вершина, инцидентная наибольшему числу отрицательных рёбер, выделяется в отдельный кластер. В результате для любого ребра $e = (u, v) \in E^-$ выполняется, что $u \in C_i$ и $v \in C_j$, где $i \neq j$. Данная процедура приводит к разбиению $\mathcal{C}$ с ошибкой $Q_1(\mathcal{C}) = 0$. Такая стратегия выполнима за время, не превышающее $\mathcal{O}(n^2 + nm)$.

2. Общая схема алгоритмов, основанных на структуре графа

Исследование алгоритмов, основанных на структуре графа, выявило общую схему в организации вычислений для решения задачи корреляционной кластеризации, которая представлена на рис. 1.

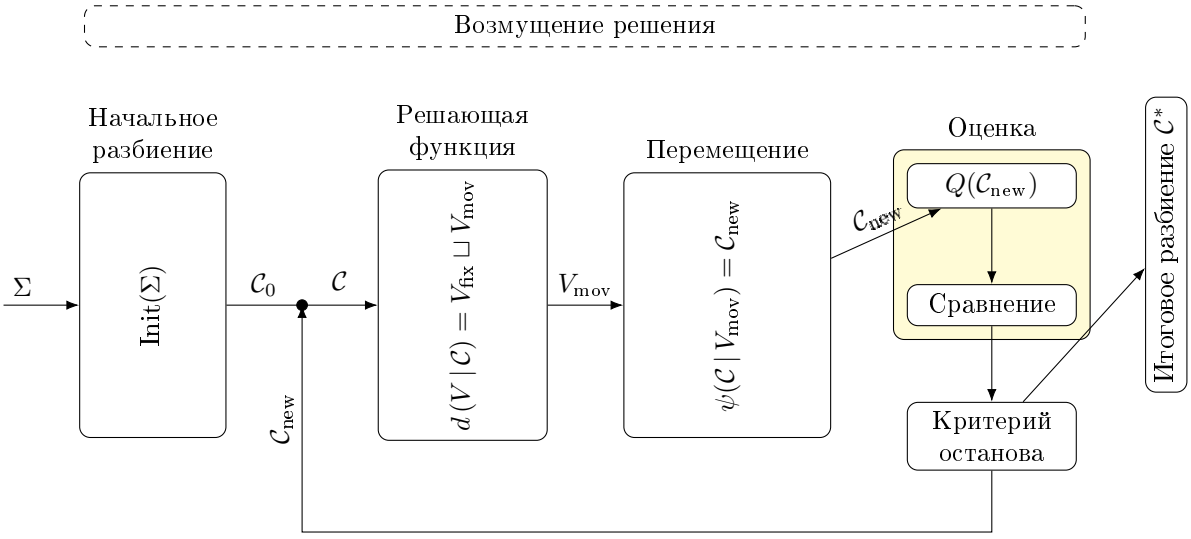


Рис. 1. Общая схема алгоритмов, основанных на структуре графа

Блок «Начальное разбиение» содержит функцию $\mathrm{Init}(\Sigma)$, осуществляющую первоначальное разбиение множества вершин V знакового графа Σ по некоторому правилу. Тривиальным случаем такого разбиения будем считать случайное разбиение V на

фиксированное или нефиксированное количество кластеров. Данный блок зачастую выполняется единожды при запуске алгоритма и формирует первоначальное разбиение $\mathcal{C}_0 \in \Phi$ вида (1).

Блок «Решающая функция» является подготовительным для блока «Перемещение». Решающая функция осуществляет разделение множества вершин V с учётом текущего разбиения $\mathcal{C}$ на два подмножества

$$d\,(V \,|\, \mathcal{C}) = V_{\text{fix}} \sqcup V_{\text{mov}}, \tag{6}$$

где V_{fix} — множество заблокированных для перемещения между кластерами вершин; V_{mov} — множество допустимых для перемещения вершин. Тривиальной будем считать решающую функцию, возвращающую $V_{\text{mov}} = V$.

Блок «Перемещение» содержит функцию $\psi(\mathcal{C} \,|\, V_{\text{mov}})$, которая отвечает за перемещение вершин из множества V_{mov} между кластерами текущего разбиения $\mathcal{C}$ и тем самым формирует новое разбиение $\mathcal{C}_{\text{new}} \in \Phi$. Вершины могут перемещаться не только между существующими кластерами, но и образовывать новые кластеры.

Блок «Оценка» состоит из двух этапов. На первом этапе вычисляется функционал ошибки $Q(\mathcal{C}_{\text{new}})$. На втором этапе проводится сравнение текущего разбиения с ранее найденными по значению функционала ошибки. Данный блок присутствует и в алгоритме, который находит последовательно или параллельно несколько разбиений.

Блок «Критерий останова» позволяет исключить перебор по всему множеству Φ. В качестве критерия останова могут выступать: время, число итераций, значение функционала ошибки, невязка функционала ошибки и т. п.

Блок «Возмущение решения» нацелен на выход из локального минимума функционала ошибки путём перемешивания вершин текущего разбиения, при этом способ перемешивания может определять основную идею алгоритма. Данный блок может следовать после любого другого блока общей схемы и повторяться многократно.

В табл. 1 известные алгоритмы решения задачи корреляционной кластеризации представлены в виде последовательностей блоков из схемы рис. 1. Символами «+»/«−» обозначается соответственно присутствие или отсутствие блока. Используются также следующие обозначения: alg — результат работы другого алгоритма; special — специальным образом; trivial — тривиальный случай; time — время; iter — число итераций; $|V|$ — просмотрены все вершины. В столбце «Перемещение»: i — алгоритм одномоментно перемещает i вершин и при этом может создавать новые кластеры; ① — алгоритму запрещено создавать новые кластеры в процессе перемещения одной вершины. Число вершин i может быть фиксированным в алгоритме либо являться его входным параметром, что обозначается как r.

Рассмотрим структуру алгоритмов из табл. 1 в соответствии со схемой рис. 1.

Алгоритм Relocation heuristic (RH) предложен в [14] и относится к классу эвристических алгоритмов. Количество кластеров разбиения является входным параметром алгоритма. Первоначальное разбиение $\mathcal{C}_0$ строится случайным образом, что соответствует тривиальному случаю блока «Начальное разбиение». Множество допустимых для перемещения вершин V_{mov} между кластерами совпадает со всем множеством вершин V, что соответствует тривиальному случаю для функции (6) в блоке «Решающая функция». Блоку «Перемещение» в табл. 1 соответствует символ ①, что означает перемещение между кластерами ровно одной вершины без образования новых кластеров. Для каждой вершины оцениваются все её возможные перемещения между кластерами. Реализуется перемещение, при котором ошибка будет наименьшей. Процесс повторяется до тех пор, пока не истечёт заданное время.

Таблица 1

Представление алгоритмов решения задачи корреляционной кластеризации по фазам схемы рис. 1

Алгоритм	Авторы	Начальное разбиение	Решающая функция	Перемещение	Критерий останова	Возмущение решения
Relocation heuristic (RH)	P. Doreian, A. Mrvar (1996)	trivial	trivial	①	time	–
Tabu search	M. J. Brusco, P. Doreian (2019)	alg	special	①	time	–
Variable neighborhood search	M. J. Brusco, P. Doreian (2019)	alg	trivial	①	time	+
KwikCluster	N. Ailon, M. Charikar, A. Newman (2008)	special	–	–	$\|V\|$	–
Iterated local search (ILS)	M. Levorato, L. Drummond, Y. Frota, R. Figueiredo (2015)	special	trivial	$1 \dots r$	iter, time	+
$\texttt{SGClust}_\alpha$	Э. И. Ибрагимова, Д. В. Семенова, А. А. Солдатенко (2023)	special	special	1	$\|V\|$	–

Метаэвристический алгоритм Tabu search предложен в [16]. Количество кластеров разбиения является входным параметром алгоритма. Начальное разбиение $\mathcal{C}_0$ является результатом работы алгоритма RH. Во избежании полного перебора авторами вводится решающая функция вида (6), где множество фиксированных вершин V_{fix} определяется списком *tabu*. Список *tabu* формируется из пар (v, it_v), где v — перемещённая на текущей итерации вершина, а it_v — число итераций, на которое вершина v помещается в список *tabu*. Все последующие блоки аналогичны алгоритму RH.

Метаэвристический алгоритм Variable neighborhood search предложен в [16]. Алгоритм представляет собой модификацию RH. Суть модификации состоит в добавлении блока «Возмущение решения» для разбиения, подающегося на вход блоку «Решающая функция», и в изменении блока «Оценка». Решение возмущается следующим образом. Для каждой вершины графа разыгрывается случайная бернуллиевская величина с заданным параметром вероятности успеха *ypert*. В случае наступления успеха данная вершина перемещается из текущего кластера в другой случайный кластер. Далее запускается алгоритм RH с начальным разбиением, соответствующим возмущённому решению. В блоке «Оценка» результат работы RH сравнивается с предыдущим разбиением. Если ошибка не уменьшилась, то вероятность перемещения *ypert* увеличивается на шаг *ystep*, заданный параметром алгоритма, а полученное разбиение забывается. В противном случае решение становится новым текущим разбиением.

Эвристический алгоритм KwikCluster предложен в [3]. Согласно схеме рис. 1, алгоритм содержит только блок «Начальное разбиение». Построение разбиения осуществляется следующим образом: случайно выбирается вершина из множества непросмотренных вершин; вершины, соединённые с ней положительным ребром, помещаются в тот же кластер и отмечаются как просмотренные. Процесс повторяется, пока не бу-

дут просмотрены все вершины. Алгоритм в ходе работы не выполняет дальнейшего перемещения вершин.

Метаэвристический алгоритм Iterated local search (ILS) предложен в [17]. Блок «Начальное разбиение» строится следующим образом. Вводится функция дисбаланса специального вида для ранжирования вершин графа, что позволяет сформировать упорядоченный список вершин. Далее строится α-срез списка вершин, из которого случайным образом выбирается вершина и размещается в кластере согласно функции дисбаланса. Процедура повторяется до тех пор, пока все вершины не будут размещены по заданному числу кластеров. Глубина среза α является входным параметром алгоритма. Блок «Возмущение решения» применяется к разбиению, передаваемому в блок «Решающая функция», и заключается в следующем. Выбирается случайная вершина из случайного кластера и перемещается в другой случайный кластер. Данная процедура выполняется t раз. Множество допустимых для перемещения вершин V_{mov} между кластерами совпадает с множеством вершин V, что соответствует тривиальному случаю для функции (6) в блоке «Решающая функция». В табл. 1 блок «Перемещение» содержит обозначение $1 \dots r$, что соответствует перемещению между кластерами от одной до r вершин с возможностью образования новых кластеров. Перемещения вершин перебираются до тех пор, пока не будет найдено первое улучшение функционала ошибки. Данное перемещение будет результатом блока. Блок «Оценка» сравнивает полученное разбиение с предыдущим. Если ошибка не уменьшилась, то число возмущений t увеличивается на единицу. Возмущения осуществляются до тех пор, пока t не достигнет значения соответствующего параметра алгоритма. Авторы предлагают запускать данный алгоритм многократно, согласно значению параметра iter, либо до истечения заданного времени работы time. В качестве итогового разбиения выбирается наилучшее в смысле функционала ошибки среди всех решений.

Эвристический алгоритм $\texttt{SGClust}_\alpha$ предложен в работах [18, 19]. В качестве первоначального разбиения выбираются компоненты связности в графе Σ^+, который получается из графа Σ путём удаления всех отрицательных рёбер. Решающая функция (6) зависит от внутрикластерной ошибки каждой вершины и обновляемого на каждой итерации закрытого списка *closeList*. Если внутрикластерная ошибка вершины отлична от нуля и вершина не содержится в списке *closeList*, то она помещается в множество V_{mov}. Блоку «Перемещение» в табл. 1 соответствует символ 1, что означает перемещение между кластерами ровно одной вершины с возможностью образования новых кластеров. Вершина с наибольшей внутрикластерной ошибкой выбирается из множества V_{mov}. Данная вершина поочередно присоединяется к каждому из кластеров текущего разбиения, включая пустой. Реализуется перемещение, обеспечивающее наименьшее значение функционала ошибки. Перемещённая вершина добавляется в закрытый список *closeList*. Процесс повторяется, пока не будут просмотрены все вершины из множества V либо на текущей итерации множество допустимых для перемещения вершин не станет пустым ($V_{\text{mov}} = \varnothing$).

3. Алгоритм CaRVeR с потенциальными функциями

Рассмотрим модификацию алгоритма $\texttt{SGClust}_\alpha$, именуемую CaRVeR (***Car****eful* ***Ve****rtex* ***R****elocator*), впервые изложенную в [20]. Суть модификации заключается в применении потенциальных функций в блоке «Решающая функция». Исследуем данный алгоритм согласно блокам схемы рис. 1.

3.1. Начальное разбиение

Алгоритм может работать с любым начальным разбиением, однако целесообразно строить некоторое разбиение, отличное от тривиального случая. В алгоритме используется следующий простой двухшаговый метод построения начального разбиения. На первом шаге строится граф Σ^+ путём удаления всех отрицательных рёбер в исходном графе Σ. Этот шаг требует времени $\mathcal{O}(m)$. На втором шаге алгоритмом поиска в ширину в графе Σ^+ выделяются компоненты связности. Множество вершин в компоненте связности графа Σ^+ является кластером в графе Σ. Множество $\mathcal{C}$ всех таких кластеров образует первоначальное разбиение $\mathcal{C}_0$. Этот шаг требует времени не более чем $\mathcal{O}(n+m)$ [15]. Количество кластеров определяется числом компонент связности $|\mathcal{C}_0| = k(\Sigma^+)$.

Из этих рассуждений вытекает следующая лемма:

Лемма 1. Сложность выполнения блока «Начальное разбиение» для алгоритма `CaRVeR` составляет $\mathcal{O}(n+m)$.

Отличительное свойство такого начального разбиения заключается в том, что оно обладает межкластерной ошибкой $P(\mathcal{C}) = 0$. Данная стратегия обоснована тем, что алгоритм `CaRVeR` в ходе работы уменьшает внутрикластерную ошибку без увеличения суммарной ошибки.

3.2. Решающая функция

Решающая функция (6) зависит от значения потенциальной функции для каждой вершины при текущем разбиении $\mathcal{C}$. Значение потенциальной функции вершины v, принадлежащей кластеру $C \in \mathcal{C}$, будем вычислять следующим образом:

$$\begin{aligned}\pi(v) = \alpha \left(\sum_{u \in \Gamma(v) \cap C} [(v,u) \in E^-] - \sum_{u \in \Gamma(v) \setminus C} [(v,u) \in E^-] \right) + \\ +(1-\alpha) \left(\sum_{u \in \Gamma(v) \setminus C} [(v,u) \in E^+] - \sum_{u \in \Gamma(v) \cap C} [(v,u) \in E^+] \right).\end{aligned} \tag{7}$$

В (7) первое слагаемое определяется как разность текущего вклада вершины v в отрицательную ошибку и числа корректных отрицательных рёбер. Второе слагаемое есть разность текущего вклада вершины v в межкластерную ошибку и числа корректных положительных рёбер. Следовательно, потенциальная функция (7) состоит из межкластерной и внутрикластерной ошибки вершины v и носит следующий смысл: «может ли при идеальных условиях вершина v иметь меньший вклад в ошибку, чем сейчас в кластере C?» Выбор вершины v и её перемещение в другой кластер однозначно увеличат ошибку пропорционально числу корректных положительных рёбер, инцидентных v, в текущей кластеризации, при этом некоторое число корректных отрицательных рёбер может сохраниться.

Таким образом, будем говорить, что вершина v не подлежит перемещению, т. е. $v \in V_{\text{fix}}$, если $\pi(v) < 0$ либо она была перемещена ранее, и $v \in V_{\text{mov}}$, если $\pi(v) \geqslant 0$. Аналогично алгоритму $\texttt{SGClust}_\alpha$, в множестве V_{mov} не могут содержаться вершины из закрытого списка *closeList*.

Для вычисления потенциальных функций $\pi(v)$ для всех вершин $v \in V$ требуется проверить кластер для каждой вершины из окрестности $\Gamma(v)$. Известно, что сумма степеней всех вершин в графе равна $2m$. Тогда справедлива следующая

Лемма 2. Сложность выполнения блока «Решающая функция» для алгоритма `CaRVeR` составляет $\mathcal{O}(m)$.

Отметим, что вид формулы (7) позволяет выполнять пересчёт значения для вершины v только при изменении кластера самой вершины v или смежной с ней вершины.

3.3. Перемещение

Данный блок описывается функцией $\psi(\mathcal{C} \mid V_{\text{mov}})$, которая выбирает единственную вершину $v \in V_{\text{mov}}$ с наибольшим значением потенциальной функции $\pi(v)$. Если таких вершин несколько, то из них выбирается случайная. Далее определяется кластер, в котором вершина v даст наименьший вклад в ошибку. Ошибку вершины v относительно кластера C_i в разбиении $\mathcal{C}$ будем определять по формуле

$$\tau(C_i, v) = \alpha \sum_{u \in \Gamma(v) \cap C_i} [(v,u) \in E^-] + (1-\alpha) \sum_{u \in \Gamma(v) \setminus C_i} [(v,u) \in E^+]\,. \tag{8}$$

Вершина v перемещается в кластер C_i, доставляющий минимум функции $\tau(C_i, v)$.

Вычисление формулы (8) требует не более чем Δ времени для каждого кластера C_i, где Δ — степень графа. Очевидно, что число кластеров не может превосходить числа вершин n. Из этих рассуждений вытекает следующая лемма:

Лемма 3. Сложность выполнения блока «Перемещение» для алгоритма `CaRVeR` составляет $\mathcal{O}(nm)$.

3.4. Оценка и критерий останова

Блок «Оценка» организован стандартным образом и содержит вычисление функционала ошибки (4) на последнем шаге алгоритма.

В алгоритме `CaRVeR` блок «Критерий останова» организован следующим образом. В конце каждой итерации алгоритма обновляется закрытый список *closeList*, в который помещается вершина, выбранная в блоке «Перемещение» на текущей итерации. Останов алгоритма происходит в том случае, если в блоке «Решающая функция» $V_{\text{mov}} = \varnothing$.

Фиксация вершины от дальнейших перемещений выполнима за константное время $\mathcal{O}(1)$, а критерий останова формирует основной цикл алгоритма с числом итераций, не превосходящим n.

Из лемм 1–3 и критерия останова выводится итоговая оценка сложности алгоритма `CaRVeR`.

Теорема 1. Временная сложность алгоритма `CaRVeR` составляет $\mathcal{O}(n^2\, m)$.

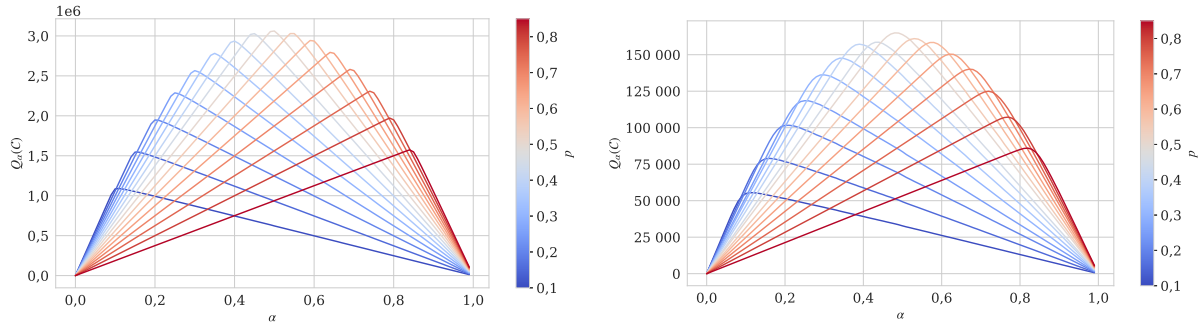
4. Вычислительные эксперименты

Опишем две серии экспериментов на синтетических данных.

Первая серия экспериментов предназначена для изучения поведения алгоритма `CaRVeR` в зависимости от параметра графа p и значения α функционала ошибки (4). Исследования проводились на графах с фиксированным числом вершин, равным 5000. Рассматривались два типа графов: полные графы и графы, смоделированные методом Ваксмена [21] с параметрами 0,15 и 0,4. Знаки рёбер для каждого графа формировались по схеме Бернулли с параметром p из набора $\{0{,}10; 0{,}15; \ldots; 0{,}85\}$. Параметр p обозначает вероятность положительного знака ребра. Значения параметра α функционала ошибки (4) выбирались из диапазона от 0 до 1 с шагом 0,01.

На рис. 2 представлено типичное поведение функционала ошибки в зависимости от значения параметра α для графов с разной долей положительных рёбер p. В каждом случае функционал ошибки $Q_\alpha(\mathcal{C})$ достигает своего максимального значения при $\alpha \approx p$. Рис. 2, *б* отражает поведение функционала ошибки для неполных графов, сгенерированных по методу Ваксмена. Аналогичное поведение характерно для полных

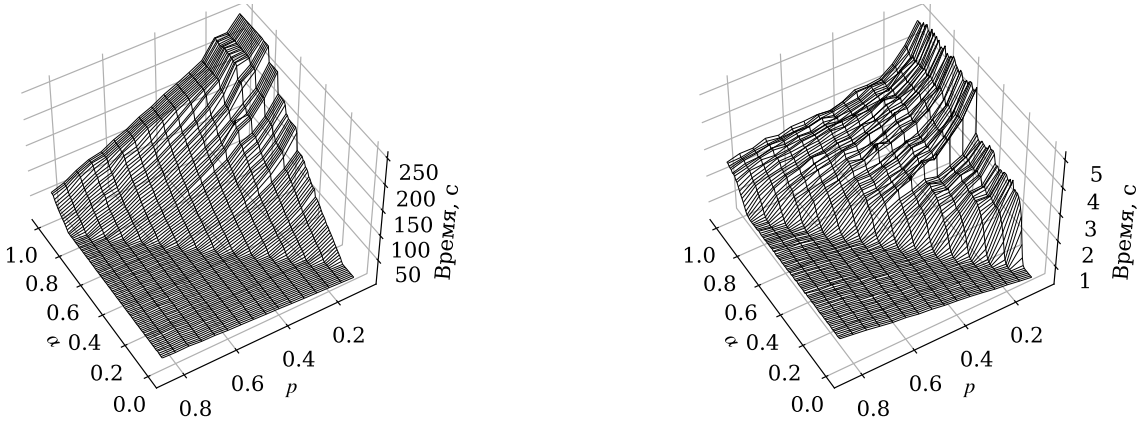
графов (рис. 2, *а*). Стоит отметить, что для неполных графов число рёбер в среднем составляет 708827, а для полных графов оно равно 12497500.



а *б*

Рис. 2. Поведение функционала ошибки алгоритма `CaRVeR` в зависимости от значения параметра α на графах с разной долей положительных рёбер p: *а*) для полных графов; *б*) для графов, сгенерированных по методу Ваксмена

С ростом параметра α трудоёмкость алгоритма `CaRVeR` возрастает, при этом высокая доля положительных рёбер p в графе замедляет рост затрачиваемого времени. На полных графах время выполнения алгоритма существенно возрастает, это следует из специфики блока «Решающая функция», поскольку требуется обновлять окрестность каждой вершины. Данные результаты представлены на рис. 3.

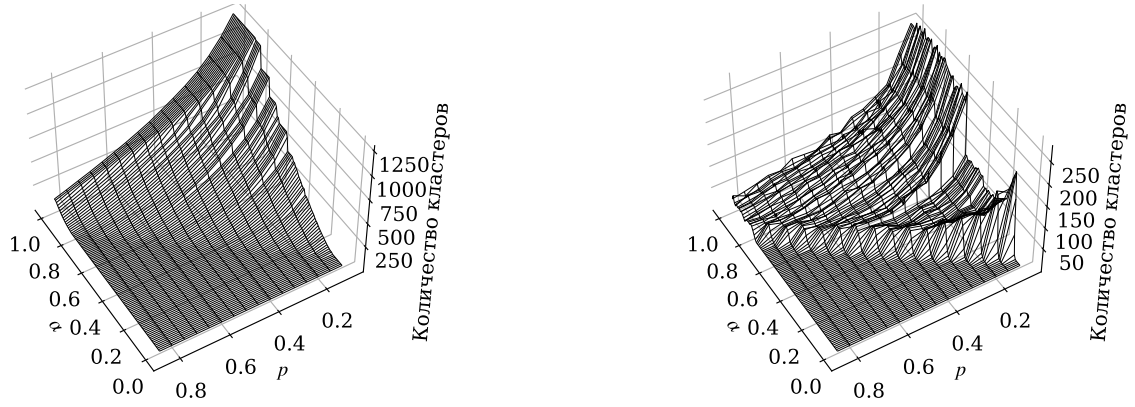


а *б*

Рис. 3. Время работы алгоритма `CaRVeR` при различных параметрах α на графах с разной долей положительных рёбер p: *а*) для полных графов; *б*) для графов, сгенерированных по методу Ваксмена

Число кластеров, выделяемых алгоритмом `CaRVeR`, представлено на рис. 4. Для полных графов число кластеров растёт с уменьшением доли положительных рёбер p (рис. 4, *а*), что естественно для стратегии, применяемой алгоритмом `CaRVeR`.

Вторая серия экспериментов представляет сравнение алгоритма `CaRVeR` с алгоритмами KwikCluster, $\texttt{SGClust}_\alpha$, RH, Tabu search. Алгоритмы сравнивались по времени работы и значению функционала ошибки $Q_\alpha(\mathcal{C})$ при $\alpha = 0{,}5$. Начальное разбиение для алгоритмов RH и Tabu search задавалось случайным образом, а число кластеров — таким же, что было получено алгоритмом `CaRVeR` для данного графа. Сравне-



а *б*

Рис. 4. Количество выделяемых кластеров алгоритмом `CaRVeR` при различных параметрах α на графах с разной долей положительных рёбер p: *а*) для полных графов; *б*) для графов, сгенерированных по методу Ваксмена

ние проводилось на графах с разной долей положительных рёбер $p = \{0{,}25; 0{,}5; 0{,}75\}$. Для каждой доли p были сгенерированы 100 графов методом Ваксмена с параметрами 0,15 и 0,4.

Результаты сравнения алгоритмов по времени представлены в табл. 2. Алгоритм `CaRVeR` ведёт себя достаточно стабильно по времени выполнения. Заметим, что представленные алгоритмы работают значительно быстрее на графах с большей долей положительных рёбер p.

Таблица 2

Сравнение алгоритмов по времени работы

p	Алгоритм	Среднее время, с	Минимальное время, с	Максимальное время, с	Среднеквадратичное отклонение
0,25	`CaRVeR`	0,016	0,015	0,018	0,001
	KwikCluster	0,001	0,001	0,001	0
	RH	30,667	26,145	36,995	2,039
	$\texttt{SGClust}_\alpha$	0,018	0,017	0,02	0,001
	Tabu search	71,322	60,041	79,819	5,312
0,5	`CaRVeR`	0,008	0,007	0,009	0,001
	KwikCluster	0,001	0,001	0,001	0
	RH	16,158	10,642	19,663	1,623
	$\texttt{SGClust}_\alpha$	0,016	0,014	0,017	0,001
	Tabu search	64,701	60,031	69,755	2,892
0,75	`CaRVeR`	0,004	0,004	0,005	0,001
	KwikCluster	0,001	0,001	0,001	0
	RH	1,636	0,405	3,221	0,587
	$\texttt{SGClust}_\alpha$	0,008	0,007	0,01	0,001
	Tabu search	60,294	60,0	60,981	0,242

На рис. 5 и 6 представлена скрипичная диаграмма значений функционала ошибки $Q_\alpha(\mathcal{C})$. Из рис. 5 видно, что на полных графах алгоритм $\texttt{SGClust}_\alpha$ и его модификация `CaRVeR` в среднем демонстрируют наилучший результат среди тестируемых алгоритмов. Аналогичный вывод справедлив и для графов, сгенерированных методом

Ваксмена (рис. 6). Следует отметить, что алгоритм KwikCluster демонстрирует удовлетворительные результаты по значению функционала ошибки на неполных графах и, учитывая его высокую скорость работы, может быть использован как алгоритм первоначального разбиения. Наибольший разброс в значениях функционала ошибки показывает алгоритм Tabu search. Алгоритмы $\texttt{SGClust}_\alpha$ и `CaRVeR` менее чувствительны к изменению доли положительных рёбер p, что видно из рис. 6.

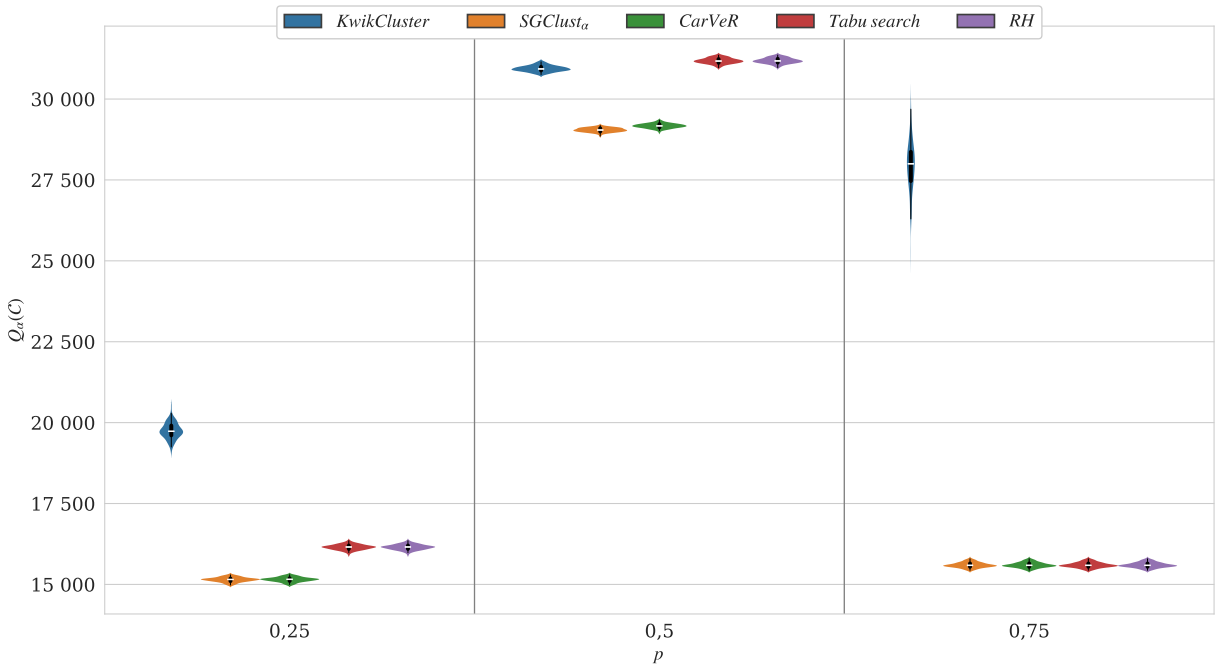


Рис. 5. Сравнение алгоритмов по функционалу ошибки $Q_\alpha(\mathcal{C})$ для полных графов

На рис. 7 и 8 отражены средние доли в процентах межкластерной $P(\mathcal{C})$ и внутрикластерной $N(\mathcal{C})$ ошибок в найденном решении, что в некотором смысле характеризует стратегию тестируемых алгоритмов. В алгоритме `CaRVeR`, в сравнении с алгоритмом $\texttt{SGClust}_\alpha$, происходит балансировка внутрикластерной и межкластерной ошибок, что обусловлено спецификой потенциальной функции (рис. 7 и 8). Для полных графов (рис. 7) алгоритм KwikCluster находит в среднем равные доли межкластерной и внутрикластерной ошибок, а для неполных (рис. 8) доля межкластерной ошибки значительно превосходит долю внутрикластерной ошибки.

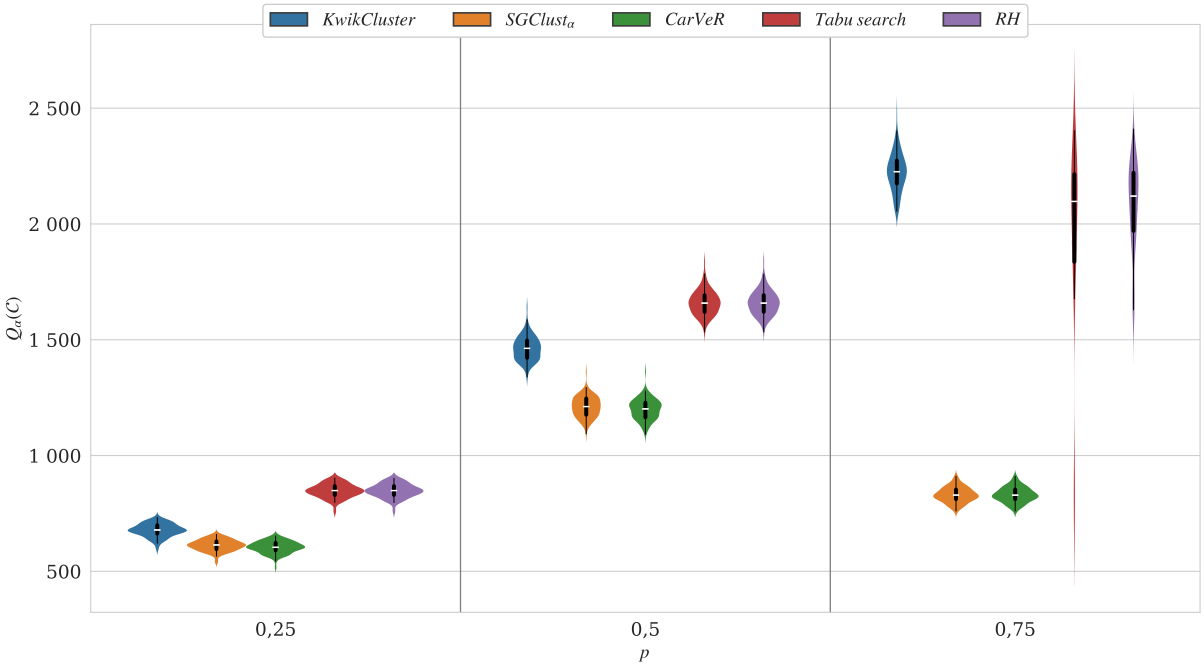


Рис. 6. Сравнение алгоритмов по функционалу ошибки $Q_\alpha(\mathcal{C})$ для графов, сгенерированных методом Ваксмена

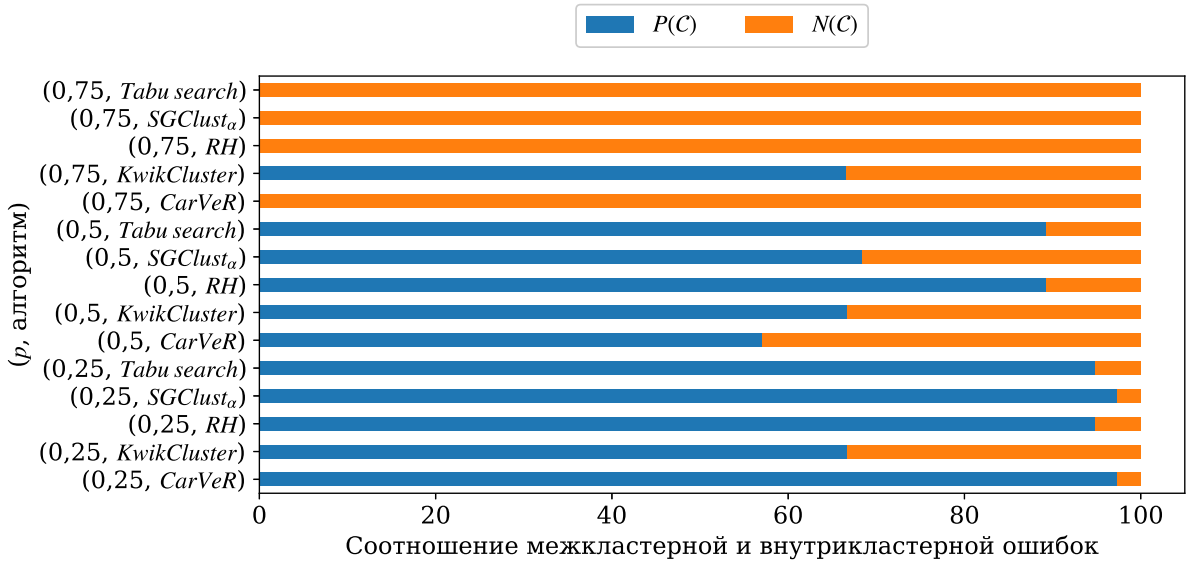


Рис. 7. Сравнение алгоритмов по долям p в процентах межкластерной $P(\mathcal{C})$ и внутрикластерной $N(\mathcal{C})$ ошибок для найденного решения на полных графах

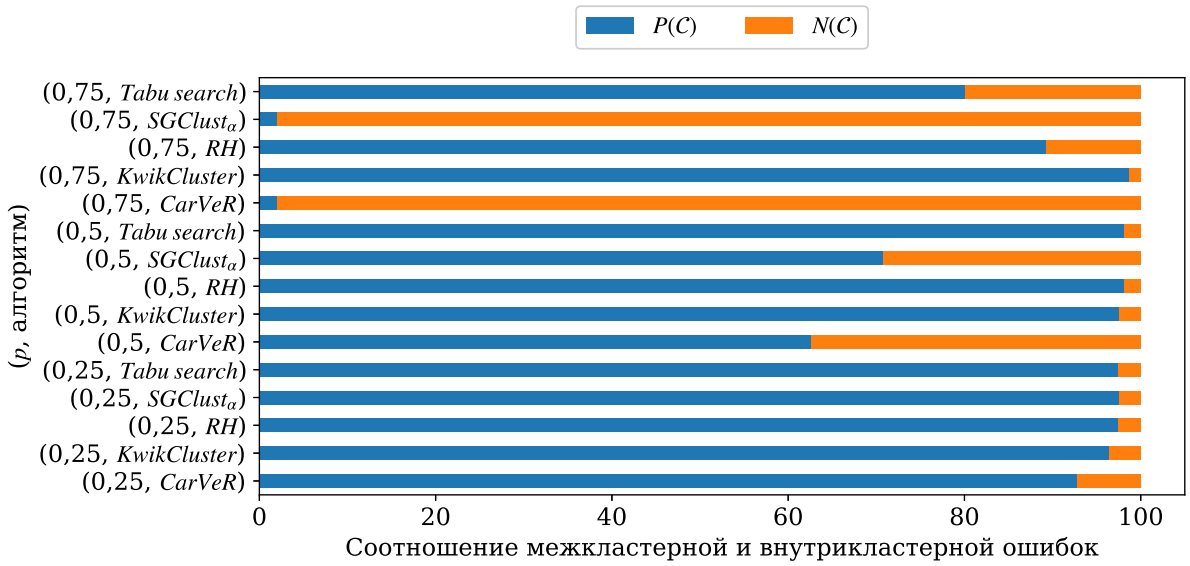


Рис. 8. Сравнение алгоритмов по долям p в процентах межкластерной $P(\mathcal{C})$ и внутрикластерной $N(\mathcal{C})$ ошибок для найденного решения на графах, сгенерированных методом Ваксмена

Заключение

Проведённый системный анализ совокупности алгоритмов решения задачи корреляционной кластеризации, основанных на структуре знакового графа, позволил выявить общую концептуальную схему конструирования и анализа таких алгоритмов. Компонентами схемы являются шесть основных блоков, характеризующих возможные этапы решения, что продемонстрировано на примере описания известных алгоритмов KwikCluster, $\texttt{SGClust}_\alpha$, Relocation heuristic, Tabu search, Variable neighborhood search, Iterated local search. Такой общий взгляд на структуру алгоритмов позволяет модифицировать существующие алгоритмы путём внесения изменений в один или несколько блоков схемы. Предложен новый алгоритм `CaRVeR`. Согласно общей схеме, он является улучшенной модификацией алгоритма $\texttt{SGClust}_\alpha$ в блоке «Решающая функция». Более того, топология общей схемы открывает возможности для анализа и доказательства вычислительной сложности алгоритмов, что представлено в теореме 1.

Серии вычислительных экспериментов на синтетических данных показали результативность алгоритма `CaRVeR` в сравнении с алгоритмами KwikCluster, $\texttt{SGClust}_\alpha$, Relocation heuristic, Tabu search как по времени работы, так и по значению функционала ошибки.

Перспективны дальнейшие исследования потенциальных функций на блоке «Решающая функция» общей схемы, а также формирование базового алгоритма, содержащего все блоки общей схемы, для осуществления различных модификаций любого из блоков.

ЛИТЕРАТУРА

1. *Il'ev V., Il'eva S., and Kononov A.* Short survey on graph correlation clustering with minimization criteria // LNCS. 2016. V. 9869. P. 25–36.
2. *Wahid D. F. and Hassini E.* A literature review on correlation clustering: Cross-disciplinary taxonomy with bibliometric analysis // Oper. Res. Forum. 2022. V. 3. Article 47.

3. *Ailon N., Charikar M., and Newman A.* Aggregating inconsistent information: Ranking and clustering // J. ACM. 2008. V. 55. Iss. 5. Article 23. P. 1–27.
4. *Demaine E. D. and Immorlica N.* Correlation clustering with partial information // LNCS. 2003. V. 2764. P. 1–13.
5. *Swamy C.* Correlation clustering: Maximizing agreements via semidefinite programming // Proc. SODA'04. New Orleans, Louisiana, 2004. P. 526–527.
6. *Bonizzoni P., Vedova D. G., Dondi R., and Jiang T.* Correlation clustering and consensus clustering // LNCS. 2005. V. 3827. P. 226—235.
7. *Figueiredo R. and Moura G.* Mixed integer programming formulations for clustering problems related to structural balance // Social Networks. 2013. No. 35. P. 639–651.
8. *Queiroga E., Subramanian A., Figueiredo R., and Frota Yu.* Integer programming formulations and efficient local search for relaxed correlation clustering // J. Glob. Optim. 2021. No. 81. P. 919–966.
9. *Doreian P. and Mrvar A.* Partitioning signed social networks // Soc. Networks. 2009. No. 31. P. 1–11.
10. *Graham R., L., Knuth D., E., and Patashnik O.* Concrete Mathematics: A Foundation for Computer Science. 2nd ed. Massachusetts, USA: Addison-Wesley, 1994. 657 p.
11. *Cartwright D. and Harary F.* Structural balance: A generalization of Heider's theory // Psychol. Rev. 1956. V. 63. No. 5. P. 227–293.
12. *Doreian P. and Mrvar A.* Structural Balance and Partitioning Signed Graphs. `http://mrvar2.fdv.si/pajek\/SignedNetworks/Bled94.pdf`. 1996.
13. *Bansal N., Blum A., and Chawla S.* Correlation clustering // Machine Learning. 2004. No. 56. P. 89–113.
14. *Doreian P. and Mrvar A.* A partitioning approach to structural balance // Soc. Networks. 1996. No. 18. P. 149–168.
15. *Kormen T. H., Leiserson C. E., Rivest R. L., and Stein C.* Introduction to Algorithms. 3rd ed. Cambridge: MIT Press, 2009. 1312 p.
16. *Brusco M. J. and Doreian P.* Partitioning signed networks using relocation heuristics, tabu search, and variable neighborhood search // Soc. Networks. 2019. No. 56. P. 70–80.
17. *Levorato M., Figueiredo R., Frota Yu., and Drummond L.* Evaluating balancing on social networks through the efficient solution of correlation clustering problems // EURO J. Comput. Optim. 2017. V. 5. P. 467–498.
18. *Ibragimova E., Semenova D., and Soldatenko A.* Comparison of two heuristic algorithms for correlation clustering problem solving // 5th Intern. Conf. PCI. Baku, Azerbaijan, 2023. P. 1–4.
19. *Soldatenko A., Semenova D., and Ibragimova E.* On heuristic algorithm with greedy strategy for the correlation clustering problem solution // LNCS. 2024. V. 14123. P. 462–477.
20. *Солдатенко А. А., Семенова Д. В., Ибрагимова Э. И.* Алгоритм с потенциальными функциями для задачи разбиения знаковых графов // Информационные технологии и математическое моделирование. Томск, 2023. С. 238–244.
21. *Waxman B. M.* Routing of multipoint connections // IEEE J. Selected Areas Commun. 1988. V. 6. No. 9. P. 1617–1622.

REFERENCES

1. *Il'ev V., Il'eva S., and Kononov A.* Short survey on graph correlation clustering with minimization criteria. LNCS, 2016, vol. 9869, pp. 25–36.
2. *Wahid D. F. and Hassini E.* A literature review on correlation clustering: Cross-disciplinary taxonomy with bibliometric analysis. Oper. Res. Forum, 2022, vol. 3, article 47.

3. *Ailon N., Charikar M., and Newman A.* Aggregating inconsistent information: Ranking and clustering. J. ACM, 2008, vol. 55, iss. 5, article 23, pp. 1–27.
4. *Demaine E. D. and Immorlica N.* Correlation clustering with partial information. LNCS, 2003, vol. 2764, pp. 1–13.
5. *Swamy C.* Correlation clustering: Maximizing agreements via semidefinite programming. Proc. SODA'04, New Orleans, Louisiana, 2004, pp. 526–527.
6. *Bonizzoni P., Vedova D. G., Dondi R., and Jiang T.* Correlation clustering and consensus clustering. LNCS, 2005, vol. 3827, pp. 226—235.
7. *Figueiredo R. and Moura G.* Mixed integer programming formulations for clustering problems related to structural balance. Soc. Networks, 2013, no. 35, pp. 639–651.
8. *Queiroga E., Subramanian A., Figueiredo R., and Frota Yu.* Integer programming formulations and efficient local search for relaxed correlation clustering. J. Glob. Optim., 2021, no. 81, pp. 919–966.
9. *Doreian P. and Mrvar A.* Partitioning signed social networks. Soc. Networks, 2009, no. 31, pp. 1–11.
10. *Graham R., L., Knuth D., E., and Patashnik O.* Concrete Mathematics: A Foundation for Computer Science. 2nd ed. Massachusetts, USA, Addison-Wesley, 1994. 657 p.
11. *Cartwright D. and Harary F.* Structural balance: A generalization of Heider's theory. Psychol. Rev., 1956, vol. 63, no. 5, pp. 227–293.
12. *Doreian P. and Mrvar A.* Structural Balance and Partitioning Signed Graphs. `http://mrvar2.fdv.si/pajek\/SignedNetworks/Bled94.pdf`, 1996.
13. *Bansal N., Blum A., and Chawla S.* Correlation clustering. Machine Learning, 2004, no. 56, pp. 89–113.
14. *Doreian P. and Mrvar A.* A partitioning approach to structural balance. Soc. Networks, 1996, no. 18, pp. 149–168.
15. *Kormen T. H., Leiserson C. E., Rivest R. L., and Stein C.* Introduction to Algorithms. 3rd ed. Cambridge, MIT Press, 2009. 1312 p.
16. *Brusco M. J. and Doreian P.* Partitioning signed networks using relocation heuristics, tabu search, and variable neighborhood search. Soc. Networks, 2019, no. 56, pp. 70–80.
17. *Levorato M., Figueiredo R., Frota Yu., and Drummond L.* Evaluating balancing on social networks through the efficient solution of correlation clustering problems. EURO J. Comput. Optim., 2017, vol. 5, pp. 467–498.
18. *Ibragimova E., Semenova D., and Soldatenko A.* Comparison of two heuristic algorithms for correlation clustering problem solving. 5th Intern. Conf. PCI, Baku, Azerbaijan, 2023, pp. 1–4.
19. *Soldatenko A., Semenova D., and Ibragimova E.* On heuristic algorithm with greedy strategy for the correlation clustering problem solution. LNCS, 2024, vol. 14123, pp. 462–477.
20. *Soldatenko A. A., Semenova D. V., Ibragimova E. I.* Algoritm s potentsial'nymi funktsiyami dlya zadachi razbieniya znakovykh grafov [Algorithm with potential functions for the problem of partitioning signed graphs]. Informatsionnye Tekhnologii i Matematicheskoe Modelirovanie, Tomsk, 2023, pp. 238–244. (in Russian)
21. *Waxman B. M.* Routing of multipoint connections IEEE J. Selected Areas Commun.. 1988, vol. 6, no. 9, pp. 1617–1622.

СВЕДЕНИЯ ОБ АВТОРАХ

АХМЕТЗЯНОВА Лилия Руслановна — кандидат физико-математических наук, заместитель начальника отдела криптографических исследований ООО «КРИПТО-ПРО», г. Москва. E-mail: **lah@cryptopro.ru**

БАБУЕВА Александра Алексеевна — ведущий инженер-аналитик отдела криптографических исследований ООО «КРИПТО-ПРО», г. Москва.
E-mail: **babueva@cryptopro.ru**

БОЖКО Андрей Алексеевич — инженер-аналитик отдела криптографических исследований ООО «КРИПТО-ПРО», г. Москва. E-mail: **bozhko@cryptopro.ru**

ИБРАГИМОВА Эллада Ибрагимовна — аспирантка Сибирского федерального университета, г. Красноярск. E-mail: **IbragimovaEI@mail.ru**

КУДЫК Иван Дмитриевич — аспирант Омского государственного университета им. Ф. М. Достоевского, г. Омск. E-mail: **volume8091@mail.ru**

МОНАХОВ Олег Геннадьевич — кандидат технических наук, ведущий научный сотрудник Института вычислительной математики и математической геофизики СО РАН, г. Новосибирск. E-mail: **monakhov@rav.sscc.ru**

МОНАХОВА Эмилия Анатольевна — кандидат технических наук, доцент, ведущий научный сотрудник Института вычислительной математики и математической геофизики СО РАН, г. Новосибирск. E-mail: **emilia@rav.sscc.ru**

НИКИТИН Алексей Юрьевич — соискатель Омского государственного университета им. Ф. М. Достоевского, г. Омск. E-mail: **nikitinlexey@gmail.com**

ОСИПОВ Виктор Ростиславович — г. Москва. E-mail: **osvktr@bk.ru**

РОМАНОВА Анна Анатольевна — кандидат физико-математических наук, доцент, доцент кафедры фундаментальной и прикладной математики Омского государственного университета им. Ф. М. Достоевского, г. Омск.
E-mail: **RomanovaAA@omsu.ru**

РЫБАЛОВ Александр Николаевич — кандидат физико-математических наук, старший научный сотрудник лаборатории комбинаторных и вычислительных методов алгебры и логики Института математики им. С. Л. Соболева СО РАН, г. Омск.
E-mail: **alexander.rybalov@gmail.com**

СЕМЕНОВА Дарья Владиславовна — кандидат физико-математических наук, доцент, доцент Института математики и фундаментальной информатики Сибирского федерального университета, г. Красноярск. E-mail: **DVSemenova@sfu-kras.ru**

СЕРВАХ Владимир Вицентьевич — доктор физико-математических наук, старший научный сотрудник Института математики им. С. Л. Соболева СО РАН, профессор кафедры фундаментальной и прикладной математики Омского государственного университета им. Ф. М. Достоевского, г. Омск. E-mail: **svv_usa@rambler.ru**

СОЛДАТЕНКО Александр Александрович — кандидат физико-математических наук, доцент Института математики и фундаментальной информатики Сибирского федерального университета, г. Красноярск. E-mail: **ASoldatenko@sfu-kras.ru**

СОЛОМАТИН Денис Владимирович — кандидат физико-математических наук, доцент, доцент кафедры математики и методики обучения математике Омского государственного педагогического университета, г. Омск.
E-mail: **solomatin_dv@omgpu.ru**

ТАВЧЕНКО Вероника Юрьевна — аспирантка Омского государственного университета им. Ф. М. Достоевского, г. Омск. E-mail: **nikapolicheva@mail.ru**

ЧЕРЕМИСИНОВ Дмитрий Иванович — кандидат технических наук, доцент, ведущий научный сотрудник Объединённого института проблем информатики НАН Беларуси, г. Минск. E-mail: **cher@newman.bas-net.by**

ЧЕРЕМИСИНОВА Людмила Дмитриевна — доктор технических наук, профессор, главный научный сотрудник Объединённого института проблем информатики НАН Беларуси, г. Минск. E-mail: **cld@newman.bas-net.by**

Журнал «Прикладная дискретная математика» входит в перечень ВАК рецензируемых научных изданий, в которых должны быть опубликованы основные результаты диссертаций на соискание учёной степени кандидата и доктора наук по специальностям 2.3.5. «Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей» (технические науки), 2.3.6. «Методы и системы защиты информации, информационная безопасность» (физико-математические и технические науки), 1.1.5. «Математическая логика, алгебра, теория чисел и дискретная математика» (физико-математические науки), 1.2.3. «Теоретическая информатика, кибернетика» (физико-математические науки), а также в перечень журналов, рекомендованных ФУМО ВО ИБ в качестве учебной литературы по специальности «Компьютерная безопасность».

Журнал индексируется в базах данных Web of Science (Emerging Sources Citation Index (ESCI) и Russian Science Citation Index (RSCI)), Scopus, MathSciNet и Zentralblatt MATH. По решению ВАК от 21.12.2023 он отнесён к первой категории (К1) научных журналов, входящих в Перечень ВАК.

Журнал «Прикладная дискретная математика» распространяется по подписке; его подписной индекс 38696 в объединённом каталоге «Пресса России». Полнотекстовые электронные версии вышедших номеров журнала доступны на его сайте journals.tsu.ru/pdm и на Общероссийском математическом портале www.mathnet.ru. На сайте журнала можно найти также правила подготовки рукописей статей для публикации в журнале.

Тематика публикаций журнала:

- *Теоретические основы прикладной дискретной математики*
- *Математические методы криптографии*
- *Математические методы стеганографии*
- *Математические основы компьютерной безопасности*
- *Математические основы надёжности вычислительных и управляющих систем*
- *Прикладная теория кодирования*
- *Прикладная теория автоматов*
- *Прикладная теория графов*
- *Логическое проектирование дискретных автоматов*
- *Математические основы информатики и программирования*
- *Вычислительные методы в дискретной математике*
- *Математические основы интеллектуальных систем*
- *Исторические очерки по дискретной математике и её приложениям*