ВЕСТНИК

ЮЖНО-УРАЛЬСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

2024
Т. 13, № 1

ISSN 2305-9052

СЕРИЯ

«ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА И ИНФОРМАТИКА»

Решением ВАК включен в Перечень научных изданий, в которых должны быть опубликованы результаты диссертаций на соискание ученых степеней кандидата и доктора наук

Учредитель — Федеральное государственное автономное образовательное учреждение высшего образования «Южно-Уральский государственный университет (национальный исследовательский университет)»

Тематика журнала:

- Вычислительная математика и численные методы
- Математическое программирование
- Распознавание образов
- Вычислительные методы линейной алгебры
- Решение обратных и некорректно поставленных задач
- Доказательные вычисления
- Численное решение дифференциальных и интегральных уравнений
- Исследование операций
- Теория игр
- Теория аппроксимации
- Информатика
- Искусственный интеллект и машинное обучение
- Системное программирование
- Перспективные многопроцессорные архитектуры
- Облачные вычисления
- Технология программирования
- Машинная графика
- Интернет-технологии
- Системы электронного обучения
- Технологии обработки баз данных и знаний
- Интеллектуальный анализ данных

BULLETIN
OF THE SOUTH URAL STATE UNIVERSITY

2024
vol. 13, no. 1

SERIES
"COMPUTATIONAL MATHEMATICS AND SOFTWARE ENGINEERING"

ISSN 2305-9052

Vestnik Yuzhno-Ural'skogo Gosudarstvennogo Universiteta.
Seriya "Vychislitel'naya Matematika i Informatika"

South Ural State University

The scope of the journal:

- Numerical analysis and methods
- Mathematical optimization
- Pattern recognition
- Numerical methods of linear algebra
- Reverse and ill-posed problems solution
- Computer-assisted proofs
- Numerical solutions of differential and integral equations
- Operations research
- Game theory
- Approximation theory
- Computer science
- Artificial intelligence and machine learning
- System software
- Advanced multiprocessor architectures
- Cloud computing
- Software engineering
- Computer graphics
- Internet technologies
- E-learning
- Database processing
- Data mining

Содержание

Contents

УДК 004.021, 519.713.2 **DOI: 10.14529/cmse240101**

ПРОГРАММНОЕ ИССЛЕДОВАНИЕ ПОЛУРЕШЕТОК, СВЯЗАННЫХ С АВТОМАТОМ ВАТЕРЛОО

© 2024 М.Э. Абрамян[1,2]

[1] *Университет МГУ-ППИ в Шэньчжэне*
(518172, КНР, провинция Гуандун, Шэньчжэнь, ул. Гоцзидасюеюань, д. 1),
[2] *Южный федеральный университет*
(344006, Россия, Ростов-на-Дону, ул. Большая Садовая, д. 105/42)
E-mail: m-abramyan@yandex.ru

Поступила в редакцию: 17.08.2023

Статья посвящена исследованию полурешеток, содержащих покрывающие автоматы для автомата Ватерлоо. В начальных разделах статьи описывается процесс построения покрывающих автоматов на основе подмножеств гридов исходного автомата (каждый грид представляет собой некоторое множество дуг, связанных с исходным автоматом), а также рассматриваются свойства полурешеток, образуемых покрывающими автоматами. Основным результатом статьи является полное описание полученных полурешеток с точки зрения эквивалентности входящих в них покрывающих автоматов исходному автомату Ватерлоо. Выделены три класса полурешеток, каждый из которых имеет особые свойства. Для полурешетки, построенной на базе минимального покрывающего автомата, получено графическое представление, которое позволяет наглядно отразить соотношения между ее наборами, состоящими из попарно эквивалентных автоматов. Кроме того, сформулирован критерий эквивалентности покрывающего автомата автомату Ватерлоо в терминах свойств подмножества гридов, определяющего покрывающий автомат. Исследование проводилось с применением библиотеки NFALib для анализа недетерминированных конечных автоматов, реализованной автором на языке C#. Актуальность изучения автомата Ватерлоо связана с его ролью в исследовании задачи вершинной минимизации недетерминированных конечных автоматов и разработке эвристических алгоритмов реального времени, используемых для ее решения.

Ключевые слова: недетерминированные конечные автоматы, универсальный автомат, грид, покрывающий автомат, алгоритмы эквивалентного преобразования, автомат Ватерлоо.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Абрамян М.Э. Программное исследование полурешеток, связанных с автоматом Ватерлоо // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2024. Т. 13, № 1. С. 5–21. DOI: 10.14529/cmse240101.

Введение

Для описания регулярного языка существуют разные полные инварианты: не только хорошо известные канонические автоматы [1–3], но и базисные автоматы [4], а также универсальные автоматы [5]. При построении базисных и универсальных автоматов необходимо построить канонические автоматы как для заданного регулярного языка, так и для его зеркального отражения. В процессе такого построения можно получить, среди прочих объектов, специальное бинарное отношение #, определенное на парах состояний этих двух канонических автоматов. Это отношение также является инвариантом (однако неполным) для рассматриваемого языка.

В настоящее время наиболее интересным для исследования является язык Ватерлоо и связанный с ним автомат Ватерлоо. Построенный для этого языка универсальный автомат обладает следующим свойством: среди соответствующих ему покрывающих автоматов [5, 6] существует ему неэквивалентный (см. далее раздел 1). Этот факт имеет непосредственное

отношение к вопросам, связанным с разработкой эффективных алгоритмов для решения задачи вершинной минимизации недетерминированных конечных автоматов. Следует заметить, что эта задача является NP-сложной [7]. Настоящая статья связана с эвристическими алгоритмами подобной минимизации. Однако методы, получаемые при решении именно этой задачи, подходы к ее решению (т. е. конкретные описания алгоритмов) находят применение и во многих других NP-сложных оптимизационных задачах.

Для любого автомата множество связанных с ним покрывающих автоматов образует полурешетку по объединению, однако можно показать, что оно, вообще говоря, не образует полурешетки по пересечению (см. далее раздел 2). Конкретнее, вместо одной полурешетки по пересечению образуется объединение нескольких таких полурешеток. Построение подобных конструкций фактически является основным этапом алгоритма минимизации для рассматриваемого автомата (отметим, что исследование полурешеток связано с оптимизационными задачами и для других дискретных структур; см., например, [8–10]). Настоящая статья посвящена рассмотрению полурешеток для автомата Ватерлоо. Она продолжает исследования, начатые в [11, 12].

Статья организована следующим образом. Раздел 1 статьи содержит предварительные сведения; приведена связанная с конечными автоматами терминология, причем в основном те термины, которые связаны с элементами универсального автомата, прежде всего с гридами, которые можно рассматривать как состояния последнего. Все понятия иллюстрируются на примере автомата Ватерлоо; при этом демонстрируется общий подход к программному исследованию автоматов для Ватерлоо-подобных языков с применением библиотеки для работы с недетерминированными конечными автоматами NFALib, реализованной автором на языке C#. В начале данного раздела приводится краткое описание библиотеки NFALib. В разделе 2 обсуждаются свойства полурешеток покрывающих автоматов и приводятся общие свойства полурешеток для автомата Ватерлоо. Разделы 3 и 4 содержат более детальное исследование различных полурешеток, связанное, прежде всего, с наличием в них покрывающих автоматов, не эквивалентных исходному автомату Ватерлоо. В разделе 5 приводится и обосновывается критерий эквивалентности покрывающего автомата автомату Ватерлоо в терминах гридов, определяющих покрывающий автомат. В заключении приводится краткая сводка результатов, полученных в работе, и указаны направления дальнейших исследований.

1. Основные понятия и связанные с ними примеры для автомата Ватерлоо

В данном разделе приводятся необходимые понятия, связанные с конечными автоматами, и иллюстрируем их на примере автомата Ватерлоо W (рис. 1). Интерес к этому автомату обусловлен тем, что он позволяет выявить важные особенности алгоритма вершинной минимизации недетерминированных конечных автоматов. Автомат Ватерлоо был впервые описан в [13]. Именно объекты, связанные с автоматом Ватерлоо, будут основным предметом рассмотрения в последующих разделах.

Поскольку в статье описываются результаты программного исследования автоматов, основанного на применении библиотеки NFALib, при описании каждого объекта указывается метод библиотеки, позволяющий получить данный объект.

Библиотека NFALib разработана на языке C# и предназначена для исследования недетерминированных конечных автоматов и связаных с ними объектов. Основным классом

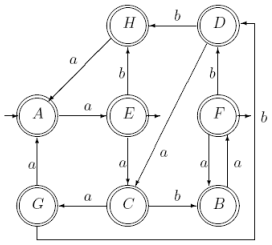


Рис. 1. Автомат Ватерлоо W

библиотеки является класс `NFA`, позволяющий определять недетерминированный конечный автомат, отображать его в текстовом виде, получать его характеристики и строить на его основе ряд специальных конечных автоматов, в том числе детерминированный автомат (полученный в результате процедуры детерминизации исходного автомата), зеркальный автомат, канонический автомат, базисный автомат, универсальный автомат COM, покрывающий автомат (далее приводятся методы, позволяющие получить большинство из перечисленных автоматов).

Кроме того, библиотека содержит классы, связанные со специальным отношением #; использование этих классов также будет проиллюстрировано далее. На основе перечисленных классов в библиотеке NFALib реализован итерационный алгоритм нахождения псевдооптимальных решений задачи об определении минимального покрывающего набора полных гридов для матрицы отношения #. Важность этой задачи обусловлена тем, что она связана с основным этапом алгоритма вершинной минимизации недетерминированных конечных автоматов [14].

Анализируемый автомат K (определяющий некоторый регулярный язык L) создается в виде объекта типа `NFA` на основе информации из текстового файла. На рис. 2 приводится содержимое файла, соответствующее автомату Ватерлоо. Смысл обозначений должен быть ясен из сравнения этого рисунка с рис. 1.

```
      a   b
==> A E   -
    B F   -
    C G   B
    D C   H
<== E C   H
<== F B   D
    G A   D
    H A   -
```

Рис. 2. Текстовое описание автомата W

Алгоритм вершинной минимизации основан на анализе *бинарного отношения* # [3, разд. 3.3], связывающего множества состояний X и Y двух канонических автоматов, которые построены на основе анализируемого недетерминированного конечного автомата K и зеркального к нему автомата K^R. Автомат Ватерлоо является детерминированным автоматом, который не изменяется в результате его канонизации. Канонический автомат для зеркального автомата после переобозначения состояний принимает вид, приведенный на рис. 3.

Для получения зеркальных и канонических автоматов в библиотеке NFALib предусмотрены методы `GetMirror` и `GetCanonicalDFA`.

```
        a   b   % States
==> X   Y   -   % E-F
<== Y   Z   U   % A-B
    Z   V   W   % F-G-H
    U   W   -   % C
    V   P   U   % B-C
    W   Q   R   % D-E
    P   Y   R   % D-E-F
<== Q   S   -   % A
    R   V   -   % F-G
    S   U   W   % G-H
```

Рис. 3. Канонический автомат для автомата W^R

В столбце `States` на рис. 3 приводятся множества состояний до их переобозначения; каждое такое состояние соответствует множеству состояний исходного зеркального автомата, полученному в результате детерминизации этого автомата.

На основе канонических автоматов строится *матрица отношения* $\#$, строки которой соответствуют состояниям канонического автомата K, а столбцы — состояниям автомата, канонического к зеркальному автомату K^R. Вид матрицы отношения $\#$ для автомата W приведен на рис. 4. Для получения матрицы отношения $\#$ некоторого автомата в библиотеке NFALib достаточно обратиться к его свойству `CurrentSharpRelation`.

```
   X  Y  Z  U  V  W  P  Q  R  S
A  -  #  -  -  -  -  -  #  -  -
B  -  #  -  -  #  -  -  -  -  -
C  -  -  -  #  #  -  -  -  -  -
D  -  -  -  -  -  #  #  -  -  -
E  #  -  -  -  -  #  #  -  -  -
F  #  -  #  -  -  -  #  -  #  -
G  -  -  #  -  -  -  -  -  #  #
H  -  -  #  -  -  -  -  -  -  #
```

Рис. 4. Матрица отношения $\#$ для автомата W

Элементы матрицы, помеченные символом `#`, определяются следующим образом: в строке, соответствующей некоторому состоянию s канонического автомата для исходного автомата (в нашем случае для автомата W), помечаются элементы для тех состояний канонического автомата для зеркального автомата, которые содержат состояние s в столбце `States`. Например, для строки `A` помечаются столбцы `Y` и `Q`, поскольку состояние Y соответствует множеству состояний $\{A, B\}$, а состояние Q — множеству состояний $\{A\}$ (рис. 3).

С отношением $\#$ связывается набор *гридов* [3, разд. 3.4]. Каждый грид определяется парой подмножеств $X_0 \in X$ и $Y_0 \in Y$, где X — множество строк матрицы отношения $\#$ (совпадающее с множеством состояний канонического автомата для автомата K), а Y — множество столбцов матрицы отношения $\#$ (совпадающее с множеством состояний канони-

ческого автомата для автомата K^R). Подмножества X_0 и Y_0 должны удовлетворять двум условиям: (1) для любых состояний $x \in X_0$ и $y \in Y_0$ выполняется $x\#y$; (2) подмножества X_0 и Y_0 нельзя расширить с сохранением условия (1). Такой грид будем обозначать $X_0 \times Y_0$.

Множество M гридов называется *покрывающим*, если для любых элементов $x \in X$, $y \in Y$ таких, что $x\#y$, найдется грид $X_0 \times Y_0$ из M, для которого $x \in X_0$ и $y \in Y_0$. Очевидно, что полный набор гридов, построенных по отношению #, является покрывающим множеством. На рис. 5 приводится полный набор из 14 гридов для отношения #, соответствующего автомату W. Для получения полного набора гридов в библиотеке NFALib предусмотрен метод `GetCompleteGrids`, основанный на полном переборе.

```
{ A } x { Y, Q }  % 1
{ A, B } x { Y }  % 2
{ B } x { Y, V }  % 3
{ B, C } x { V }  % 4
{ C } x { U, V }  % 5
{ D, E } x { W, P }  % 6
{ E } x { X, W, P }  % 7
{ E, F } x { X, P }  % 8
{ F } x { X, Z, P, R }  % 9
{ F, G } x { Z, R }  % 10
{ G } x { Z, R, S }  % 11
{ G, H } x { Z, S }  % 12
{ F, G, H } x { Z }  % 13
{ D, E, F } x { P }  % 14
```

Рис. 5. Полный набор гридов для автомата W

В [5] описан алгоритм, позволяющий по полному набору гридов построить *универсальный автомат* $COM(K)$, который определяет тот же регулярный язык L, что и исходный автомат K, и при этом каждый грид соответствует некоторому состоянию автомата $COM(K)$. Автомат $COM(W)$ приведен на рис. 6. Для его получения предназначен метод `GetCOM(completeGrids)` библиотеки NFALib.

Как было отмечено выше, построенный автомат $COM(W)$ является эквивалентным исходному автомату W, что можно показать, выполнив его канонизацию с помощью метода `GetCanonicalDFA` и переименовав его состояния требуемым образом.

На основе автомата $COM(K)$ можно определить семейство *покрывающих автоматов*, каждый из которых получается путем удаления некоторых состояний автомата $COM(K)$, причем оставшиеся состояния соответствуют гридам, образующим покрывающее множество.

Алгоритм минимизации исходного автомата K состоит в том, чтобы выбрать покрывающее множество гридов M_0 минимального размера, для которого построенный на его основе покрывающий автомат является эквивалентным автомату K, т. е. определяет тот же регулярный язык L.

К сожалению, *не всякое покрывающее множество гридов позволяет получить покрывающий автомат, эквивалентный исходному.* Примером является автомат Ватерлоо. Минимальным покрывающим множеством для него является множество M_0 из 7 элементов,

```
        a              b       % Grids
==> 1   6,7,8,14       -       % { A } x { Y, Q }
==> 2   8,14           -       % { A, B } x { Y }
    3   8,9,10,13,14   -       % { B } x { Y, V }
    4   10,13          -       % { B, C } x { V }
    5   10,11,12,13    2,3,4   % { C } x { U, V }
    6   4,5            12,13   % { D, E } x { W, P }
<== 7   4,5            12,13   % { E } x { X, W, P }
<== 8   4              -       % { E, F } x { X, P }
<== 9   2,3,4          6,14    % { F } x { X, Z, P, R }
    10  2              6,14    % { F, G } x { Z, R }
    11  1,2            6,14    % { G } x { Z, R, S }
    12  1,2            -       % { G, H } x { Z, S }
    13  2              -       % { F, G, H } x { Z }
    14  4              -       % { D, E, F } x { P }
```

Рис. 6. Автомат $COM(W)$

содержащее следующие гриды из полного множества: 1, 3, 5, 6, 8, 10, 12. Для нахождения минимальных покрывающих множеств предусмотрен метод `GetMinGridCovers`.

На рис. 7 приведен покрывающий автомат W_0, построенный на основе минимального покрывающего множества M_0 с помощью метода `GetCovering(M0)`, примененного к автомату $COM(W)$.

```
        a       b    % Grids
==> 1   6,8     -    % { A } x { Y, Q }
    3   8,10    -    % { B } x { Y, V }
    5   10,12   3    % { C } x { U, V }
    6   5       12   % { D, E } x { W, P }
<== 8   -       -    % { E, F } x { X, P }
    10  -       6    % { F, G } x { Z, R }
    12  1       -    % { G, H } x { Z, S }
```

Рис. 7. Минимальный покрывающий автомат W_0 для автомата $COM(W)$

После процедуры канонизации и переименования состояний автомат W_0 принимает вид, приведенный на рис. 8. *Данный автомат не является эквивалентным исходному автомату Ватерлоо* (ср. с рис. 2; отличающаяся строка соответствует состоянию F):

2. Полурешетки покрывающих автоматов

Поскольку минимальный покрывающий автомат, как показывает пример автомата Ватерлоо, не обязательно будет эквивалентным исходному автомату, представляет интерес исследование всей совокупности покрывающих автоматов, которые могут быть получены из автомата COM для различных покрывающих множеств гридов.

Отметим, что все покрывающие множества гридов, согласно приведенным в предыдущем разделе определениям, образуют *полурешетку* по объединению. Это очевидно, поскольку объединение любых двух покрывающих множеств также является покрывающим.

		a	b	% States
==>	A	E	-	% 1
	B	F	-	% 3
	C	G	B	% 5
	D	C	H	% 6
<==	E	C	H	% 6-8
<==	F	-	D	% 8-10
	G	A	D	% 10-12
	H	A	-	% 12

Рис. 8. Канонический автомат для минимального покрывающего автомата с переименованными состояниями

В этом смысле можно говорить и о полурешетке по объединению для всех покрывающих автоматов, понимая под операцией объединения объединение подмножеств гридов, на основе которых построены данные покрывающие автоматы.

Однако полный набор покрывающих множеств гридов не образует полурешетку по пересечению. Для доказательства этого факта достаточно рассмотреть простую матрицу $\mathcal{A}$ размера 3×3, приведенную на рис. 9.

	X	Y	Z
A	#	#	-
B	#	-	#
C	-	#	#

Рис. 9. Матрица $\mathcal{A}$

Заметим, что матрица $\mathcal{A}$ действительно является матрицей отношения # для некоторого автомата, в частности, *полного автомата* [15], который строится непосредственно по данной матрице (вид полного автомата для матрицы $\mathcal{A}$ приведен на рис. 10).

		a	b	c	d	e	f	g	h	i
<=>	A	A	B	C	A	B	C	-	-	-
<==	B	A	B	C	-	-	-	A	B	C
	C	-	-	-	A	B	C	A	B	C

Рис. 10. Полный автомат, построенный по матрице $\mathcal{A}$

Нетрудно убедиться в том, что по каждой строке и каждому столбцу матрицы $\mathcal{A}$ можно построить грид, причем данная матрица не содержит других гридов. Если упорядочить гриды в лексикографическом порядке их обозначений, то будет получен полный набор, приведенный на рис. 11 (в этом наборе гриды 1, 4, 6 соответствуют строкам матрицы $\mathcal{A}$, а гриды 2, 3, 5 — ее столбцам).

Таким образом, имеется два минимальных покрывающих множества гридов $\{1, 4, 6\}$ и $\{2, 3, 5\}$, но, конечно, их пересечение (пустое множество) покрывающим множеством не является.

```
{ A } x { X, Y }  % 1
{ A, B } x { X }  % 2
{ A, C } x { Y }  % 3
{ B } x { X, Z }  % 4
{ B, C } x { Z }  % 5
{ C } x { Y, Z }  % 6
```

Рис. 11. Полный набор гридов для матрицы $\mathcal{A}$

Несмотря на то, что полный набор покрывающих множеств гридов не образует, вообще говоря, полурешетку по пересечению, из него можно выделить подмножества покрывающих множеств, которые подобную полурешетку образуют. Каждая такая *полурешетка по пересечению* будет включать некоторое базовое покрывающее множество гридов, полное множество и все промежуточные множества, обладающим тем свойством, что любое их попарное пересечение также принадлежит этой полурешетке. Будем рассматривать только такие полурешетки по пересечению, которые нельзя расширить. В этом случае для приведенного выше примера будет получено пять полурешеток по пересечению. Две из них (с базовыми покрывающими множествами гридов $\{1, 4, 6\}$ и $\{2, 3, 5\}$) содержат по 8 элементов, а остальные (с базовыми покрывающими множествами $\{1, 2, 5, 6\}$, $\{1, 3, 4, 5\}$ и $\{2, 3, 4, 6\}$) — по 4 элемента. Все эти полурешетки изображены на рис. 12. Следует заметить, что каждая такая полурешетка может интерпретироваться как *гиперкуб* соответствующей размерности.

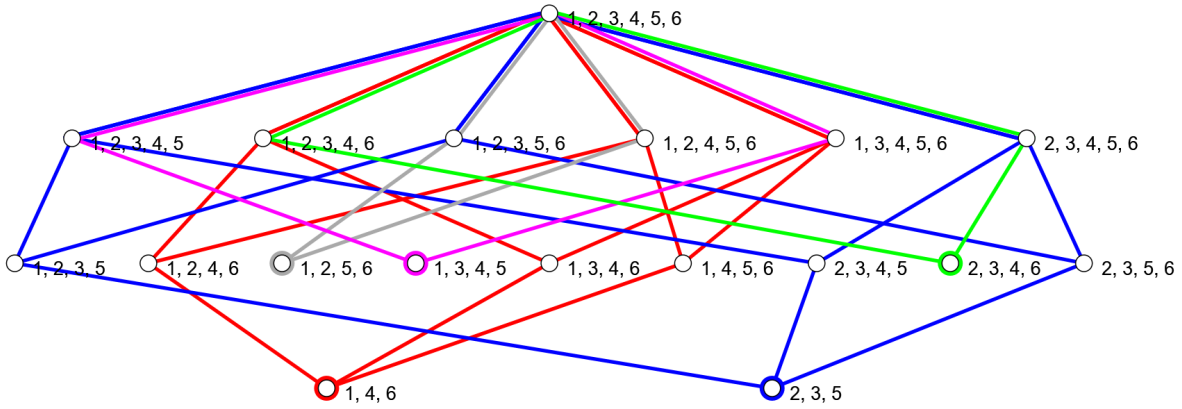


Рис. 12. Полурешетки множеств гридов для матрицы $\mathcal{A}$

В дальнейшем для краткости *полурешетками* будем называть полурешетки по пересечению, которые нельзя расширить.

Для большинства автоматов рассмотрение наборов покрывающих автоматов и связанных с ними полурешеток интереса не представляет, поскольку все эти автоматы являются эквивалентными. Иная ситуация возникает при аналогичном исследовании автомата Ватерлоо W. Как было показано в предыдущем разделе, для матрицы отношения $\#$, связанной с данным автоматом, можно построить 14 гридов, из которых можно выделить единственное минимальное покрывающее множество M_0 размера 7, причем покрывающий автомат W_0, построенный на основе множества M_0, не является эквивалентным исходному автомату W.

Численное исследование гридов матрицы отношения $\#$ для автомата Ватерлоо показывает, что имеется 260 различных покрывающих множеств гридов, размеры которых ме-

няются от 7 до 14. На основе различных покрывающих множеств гридов можно построить 8 различных полурешеток (напомним, что рассматриваются полурешетки по пересечению, которые нельзя расширить). Характеристики этих полурешеток приведены в табл. 1.

Таблица 1. Полурешетки для автомата W

Полурешетка	Базовое покрывающее множество гридов	Количество элементов (покрывающих автоматов)	Количество покрывающих автоматов, не эквивалентных W
A	$\{1, 3, 5, 6, 8, 10, 12\}$	128	48
B	$\{1, 2, 4, 5, 6, 8, 10, 12\}$	64	0
C	$\{1, 3, 5, 6, 7, 9, 10, 12\}$	64	0
D	$\{1, 3, 5, 6, 7, 9, 11, 12\}$	64	0
E	$\{1, 3, 5, 6, 8, 9, 11, 12\}$	64	0
F	$\{1, 2, 4, 5, 6, 7, 9, 10, 12\}$	32	8
G	$\{1, 2, 4, 5, 6, 7, 9, 11, 12\}$	32	12
H	$\{1, 2, 4, 5, 6, 8, 9, 11, 12\}$	32	8

Особый интерес представляет последний столбец таблицы, в котором указано количество покрывающих автоматов из данной полурешетки, не эквивалентных исходному автомату Ватерлоо. В последующих разделах обсуждаются особенности различных полурешеток, связанные с наличием в них неэквивалентных покрывающих автоматов.

3. Исследование максимальной полурешетки для автомата Ватерлоо

В данном разделе описываются дополнительные свойства полурешетки A из табл. 1. Данная полурешетка, во-первых, имеет максимальный размер (128 элементов), во-вторых, ее базовым элементом является покрывающий автомат W_0, построенный по минимальному покрывающему множеству гридов и, в-третьих, она содержит наибольшее число покрывающих автоматов, которые не являются эквивалентными автомату Ватерлоо.

Дополнительное исследование полурешетки A показало, что в нее входят четыре множества попарно эквивалентных автоматов. Полученные результаты в графическом виде приведены на рис. 13.

Вершины изображенного графа соответствуют различным покрывающим множествам гридов (и, соответственно, различным покрывающим автоматам) полурешетки A. Эти вершины, будучи элементами полурешетки размера 128, можно интерпретировать как вершины 7-мерного гиперкуба. Ребра соединяют соседние вершины гиперкуба, отличающиеся ровно на один грид, входящий в большее множество. Каждый горизонтальный ряд вершин на рис. 13 соответствует множествам с одинаковым количеством гридов (от 7 — единственная вершина в нижнем ряду, соответствующая множеству M_0, до 14 — единственная вершина в верхнем ряду, соответствующая полному множеству M). Покрывающие множества в каждом ряду расположены в лексикографическом порядке входящих в них номеров гридов. Например, второй снизу ряд, содержащий множества размера 8, включает следующие множества (слева направо): $M_1 = \{1, \underline{2}, 3, 5, 6, 8, 10, 12\}$, $M_2 = \{1, 3, \underline{4}, 5, 6, 8, 10, 12\}$,

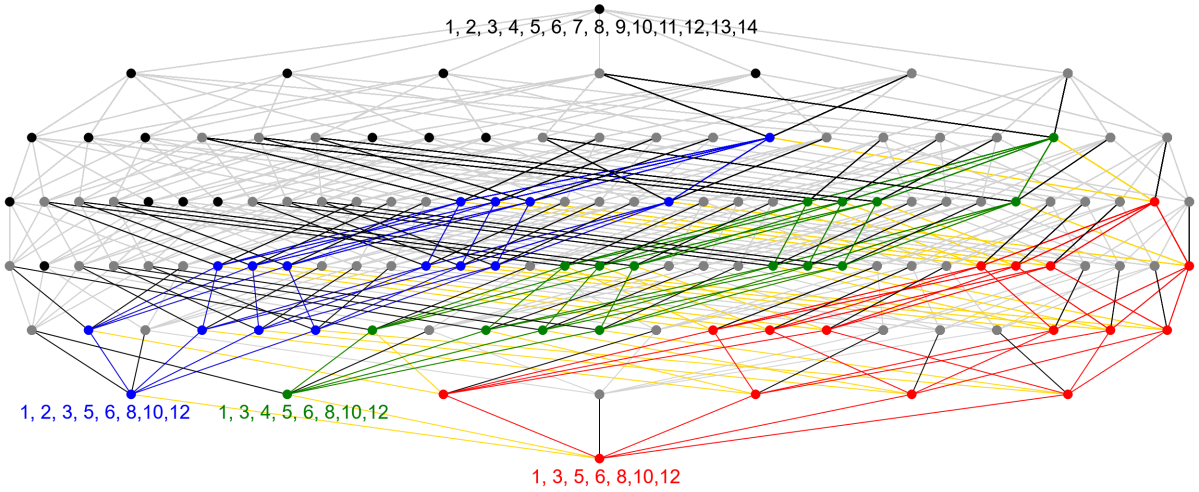


Рис. 13. Полурешетка A

$M_3 = \{1, 3, 5, 6, \underline{7}, 8, 10, 12\}$, $M_4 = \{1, 3, 5, 6, 8, \underline{9}, 10, 12\}$, $M_5 = \{1, 3, 5, 6, 8, 10, \underline{11}, 12\}$, $M_6 = \{1, 3, 5, 6, 8, 10, 12, \underline{13}\}$, $M_7 = \{1, 3, 5, 6, 8, 10, 12, \underline{14}\}$.

В каждом из этих множеств выделен подчеркиванием дополнительный грид, добавленный к минимальному покрывающему множеству M_0).

Вершины, соответствующие попарно эквивалентным покрывающим автоматам, которые не эквивалентны исходному автомату Ватерлоо, выделены одинаковым цветом: красным, синим или зеленым. Таким образом, имеются три набора автоматов, эквивалентных между собой и не эквивалентных автомату Ватерлоо:

– набор N_1 (начинается с покрывающего автомата, соответствующего множеству M_0),
– набор N_2 (начинается с автомата, соответствующего множеству M_1),
– набор N_3 (начинается с автомата, соответствующего множеству M_2).

Каждый из них содержит по 16 элементов, образующих 4-мерные гиперкубы. Ребра, соединяющие неэквивалентные автоматы из этих трех наборов, изображены желтым цветом. Набор покрывающих автоматов, эквивалентных автомату Ватерлоо, будем обозначать N_0.

Ребра черного цвета соответствуют границе гиперкуба, разделяющей автоматы, эквивалентные и не эквивалентные автомату Ватерлоо. Черным цветом изображены также вершины, соответствующие покрывающим автоматам, эквивалентным автомату Ватерлоо, которые не связаны ребрами с неэквивалентными им покрывающими автоматами.

Полученные дополнительные соотношения между элементами полурешетки A можно отобразить более наглядным образом, если представить каждый из наборов N_1, N_2, N_3 попарно эквивалентных покрывающих автоматов (не эквивалентных автомату Ватерлоо) в виде 4-мерного гиперкуба, а остальные покрывающие автоматы (эквивалентные автомату Ватерлоо) — в виде пяти 4-мерных гиперкубов, после чего естественным образом объединить полученные гиперкубы в один 7-мерный гиперкуб (рис. 14).

Каждый из восьми полученных 4-мерных гиперкубов характеризуется особой комбинацией трех дополнительных гридов с номерами 2, 4 и 9. А именно, если в базовое покрывающее множество $M_0 = \{1, 3, 5, 6, 8, 10, 12\}$, дополненное гридами из дополнительного *упорядоченного*[1] множества $D_0 = (2, 4, 7, 9, 11, 13, 14)$, не входит ни один из гридов 2, 4, 9,

[1]Порядок элементов множества D_0 фиксируется, так как это позволит легко определять его различные подмножества, как будет описано далее.

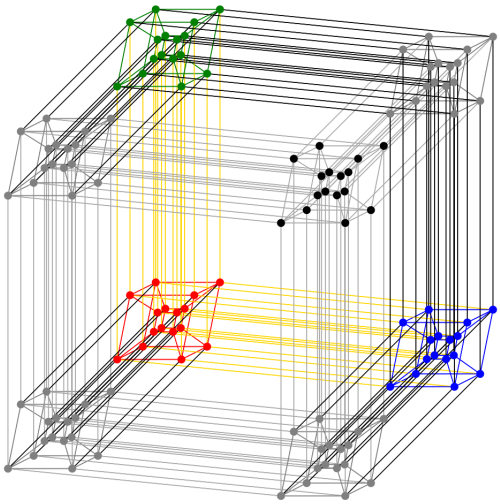


Рис. 14. Второй вариант графического представления полурешетки A

то будет получен набор N_1 (4-мерный гиперкуб красного цвета), если входит только грид 2 или только грид 4, то будут получены соответственно наборы N_2 (4-мерный гиперкуб синего цвета) и N_3 (4-мерный гиперкуб зеленого цвета). Наличие грида 9 или одновременное наличие гридов 2 и 4 гарантирует, что полученный покрывающий автомат является эквивалентным исходному автомату Ватерлоо и, таким образом, входит в N_0.

Отмеченные соотношения представлены в табл. 2. Комбинации гридов задаются строкой из 7 символов, которая определяет наличие или отсутствие каждого из гридов, входящих в D_0: если некоторый грид из D_0 входит во все элементы набора, то на его позиции в строке указывается 1, если грид не входит, то указывается 0, если грид может как входить, так и не входить в набор, то на его позиции указывается звездочка. Например, обозначение `01*0***` для набора N_2 означает, что в этот набор входят покрывающие множества гридов, включающие грид 4 и не включающие гриды 2 и 9 (гриды 7, 11, 13, 14 могут как входить, так и не входить в покрывающее множество).

Таблица 2. Подмножества полурешетки A

Набор, в который входят элементы подмножества	Описание подмножества (наличие или отсутствие гридов 2, 4, 7, 9, 11, 13, 14)	Размер подмножества	Эквивалентны ли элементы множества автомату W
N_1 (весь набор)	`00*0***`	16	нет
N_2 (весь набор)	`01*0***`	16	нет
N_3 (весь набор)	`10*0***`	16	нет
N_0	`11*0***`	16	да
N_0	`***1***`	64	да

Таким образом, *необходимым и достаточным условием того, что покрывающий автомат из полурешетки A эквивалентен автомату Ватерлоо, является наличие в соответствующем покрывающем множестве грида 9 или одновременно гридов 2 и 4.*

4. Исследование других полурешеток для автомата Ватерлоо

Обратимся к остальным полурешеткам, приведенным в табл. 1. Их можно разбить на две группы. В первую группу входят полурешетки B, C, D, E, для которых базовый покрывающий автомат эквивалентен автомату W. Поэтому и все прочие элементы данных полурешеток эквивалентны автомату W, и дополнительного исследования этих полурешеток не требуется.

В отличие от полурешеток B, C, D, E, полурешетки F, G, H содержат покрывающие автоматы, не эквивалентные автомату W. Дополнительный численный анализ показывает, что эти автоматы можно разбить на 5 наборов: N_4, N_5, N_6, N_7, N_8, в каждый из которых входят автоматы, эквивалентные между собой и не эквивалентные автоматам из других наборов (а также автоматам из наборов N_1, N_2, N_3, описанных в предыдущем разделе). При этом наборы N_4 и N_5 полностью содержатся в полурешетке F, набор N_6 полностью содержится в полурешетке G, а наборы N_7 и N_8 полностью содержатся в полурешетке H. Кроме того, полурешетка G включает половину элементов, входящих в каждый из наборов N_4, N_5, N_7 и N_8.

Более подробный анализ полурешеток F, G, H и связанных с ними наборов покрывающих автоматов, не эквивалентных автомату W, удобно провести совместно, так как наборы N_4, N_5, N_7 и N_8 входят одновременно в несколько полурешеток.

Еще раз приведем базовые покрывающие множества гридов для полурешеток F, G и H:
F: $\{1, 2, 4, 5, 6, 7, 9, 10, 12\}$,
G: $\{1, 2, 4, 5, 6, 7, 9, 11, 12\}$,
H: $\{1, 2, 4, 5, 6, 8, 9, 11, 12\}$.

Поскольку все эти множества содержат гриды из множества $M' = \{1, 2, 4, 5, 6, 9, 12\}$, данные гриды можно исключить из дальнейшего рассмотрения и ограничиться анализом гридов, входящих в *упорядоченное* множество $D_1 = (3, 7, 8, 10, 11, 13, 14)$. Для различных подмножеств множества D_1 будем использовать обозначения, аналогичные тем, которые ранее использовались для подмножеств множества D_0 (см. табл. 2): подмножество определяется строкой из 7 символов; если некоторый грид из D_1 входит во все элементы подмножества, то на его позиции в строке указывается 1, если грид не входит, то указывается 0, если грид может как входить, так и не входить в элементы подмножества, то на его позиции указывается звездочка. С учетом таких обозначений полурешетка F может быть описана следующим образом: `*1*1***`. Это означает, что, помимо гридов из множества M', во все ее элементы входят гриды 7 и 10, а прочие гриды из D_1 могут как входить, так и не входить. Полурешетка G описывается строкой `*1**1**`, а полурешетка H — строкой `**1*1**`. Описание наборов N_4, N_5, N_6, N_7, N_8 приведено в табл. 3.

Анализ таблицы 3 показывает, что *необходимым и достаточным условием того, что покрывающий автомат из полурешеток F, G, H эквивалентен автомату Ватерлоо, является наличие в соответствующем покрывающем множестве грида 3 или одновременно гридов 8 и 10.*

5. Критерий эквивалентности покрывающих автоматов автомату Ватерлоо

При исследовании полурешетки A в разделе 3 и полурешеток F, G, H в разделе 4 была выявлена особая роль гридов 3 и 9, а также комбинаций гридов 2, 4 и 8, 10. Напомним, что наличие в покрывающем множестве грида 9 или одновременно гридов 2 и 4 гарантирует,

Таблица 3. Наборы N_4, N_5, N_6, N_7, N_8

Набор	Описание (наличие или отсутствие гридов 3, 7, 8, 10, 11, 13, 14)	Размер	В какие полурешетки входит
N_4	`0101**0`	4	F (все элементы набора), G (элементы с гридом 11)
N_5	`0101**1`	4	F (все элементы набора), G (элементы с гридом 11)
N_6	`01001**`	4	G (все элементы набора)
N_7	`0*1010*`	4	H (все элементы набора), G (элементы с гридом 7)
N_8	`0*1011*`	4	H (все элементы набора), G (элементы с гридом 7)

что покрывающий автомат, входящий в полурешетку A, будет эквивалентен автомату W, а для полурешеток F, G, H аналогичный факт справедлив при наличии в покрывающем множестве грида 3 или одновременно гридов 8 и 10.

Эти результаты можно обобщить на весь набор покрывающих автоматов для автомата Ватерлоо, если дополнительно проанализировать наличие этих особых гридов в базовых покрывающих множествах для всех полурешеток. Соответствующие данные приведены в табл. 4. В ней для краткости используется операция «+» для обозначения *одновременного* наличия двух гридов: $2+4$, $8+10$.

Таблица 4. Дополнительные свойства полурешеток

Полурешетка	Базовое покрывающее множество	Особые гриды в базовом множестве	Дополнительные гриды, обеспечивающие эквивалентность автомату Ватерлоо
A	$\{1,3,5,6,8,10,12\}$	3, $8+10$	$2+4$ или 9
B	$\{1,2,4,5,6,8,10,12\}$	$2+4$, $8+10$	–
C	$\{1,3,5,6,7,9,10,12\}$	3, 9	–
D	$\{1,3,5,6,7,9,11,12\}$	3, 9	–
E	$\{1,3,5,6,8,9,11,12\}$	3, 9	–
F	$\{1,2,4,5,6,7,9,10,12\}$	$2+4$, 9	3 или $8+10$
G	$\{1,2,4,5,6,7,9,11,12\}$	$2+4$, 9	3 или $8+10$
H	$\{1,2,4,5,6,8,9,11,12\}$	$2+4$, 9	3 или $8+10$

Из табл. 4 следует, что *для того, чтобы покрывающий автомат, построенный по автомату COM для автомата Ватерлоо, был эквивалентен автомату Ватерлоо, необходимо и достаточно, чтобы в соответствующем покрывающем множестве содержались либо гриды 3 и 9, либо гриды 2, 4, 8 и 10. При нарушении данного условия будут получены покрывающие автоматы, не эквивалентные автомату Ватерлоо. Имеется 8 наборов таких автоматов; в каждый набор входят автоматы, эквивалентные между собой.*

Полученные результаты можно представить в виде табл. 5, в которой по вертикали указываются гриды из множества $\{2, 3, 4\}$, входящие в покрывающее множество, а по горизонтали — гриды из множества множества $\{8, 9, 10\}$, входящие в покрывающее множество. Таблица включает 5 строк и 5 столбцов, так как не существует покрывающих множеств, которые не содержат ни одного грида из множеств $\{2, 3, 4\}$ и $\{8, 9, 10\}$ или содержат только по одному гриду 2 или 4 из множества $\{2, 3, 4\}$ или только по одному гриду 8 или 10 из множества $\{8, 9, 10\}$. В ячейках таблицы указывается, к какому набору попарно эквивалентных покрывающих автоматов принадлежат автоматы с данным набором гридов, а также (в скобках) количество таких автоматов. Напомним, что покрывающие автоматы, эквивалентные автомату Ватерлоо, входят в набор N_0; для большей наглядности ячейки с набором N_0 выделяются дополнительным символом $*$.

Таблица 5. Распределение наборов эквивалентных покрывающих автоматов в зависимости от гридов 2, 3, 4, 8, 9, 10, входящих в покрывающие множества

	9	**8, 9**	**8, 10**	**9, 10**	**8, 9, 10**
3	$*N_0$ (4)	$*N_0$ (8)	N_1 (16)	$*N_0$ (8)	$*N_0$ (16)
2, 3	$*N_0$ (4)	$*N_0$ (8)	N_2 (16)	$*N_0$ (8)	$*N_0$ (16)
2, 4	N_6 (4)	N_7 (4), N_8 (4)	$*N_0$ (16)	N_4 (4), N_5 (4)	$*N_0$ (16)
3, 4	$*N_0$ (4)	$*N_0$ (8)	N_3 (16)	$*N_0$ (8)	$*N_0$ (16)
2, 3, 4	$*N_0$ (4)	$*N_0$ (8)	$*N_0$ (16)	$*N_0$ (8)	$*N_0$ (16)

Заключение

В настоящей работе дано полное описание полурешеток покрывающих автоматов с точки зрения эквивалентности входящих в них автоматов исходному автомату Ватерлоо.

Выделены три класса полурешеток. Первая полурешетка (A) строится на минимальном покрывающем автомате; в нее входят три набора попарно эквивалентных покрывающих автоматов, которые неэквивалентны автомату Ватерлоо. Для данной полурешетки получено графическое представление, которое позволяет наглядно отразить соотношения между ее наборами, состоящими из попарно эквивалентных автоматов. Во второй класс полурешеток входят четыре полурешетки (B, C, D, E), полностью состоящие из покрывающих автоматов, которые эквивалентны автомату Ватерлоо. В третий класс входят три полурешетки (F, G, H), содержащие пять наборов попарно эквивалентных покрывающих автоматов, которые неэквивалентны автомату Ватерлоо, причем четыре из пяти наборов входят одновременно в две полурешетки (что отличает их от наборов, входящих в полурешетку A).

В ходе исследования различных полурешеток выделены гриды, отвечающие за то, в какой из наборов попарно эквивалентных покрывающих автоматов войдет рассматриваемый автомат. На этой основе сформулирован критерий эквивалентности покрывающего автомата автомату Ватерлоо в терминах свойств подмножества гридов, определяющего покрывающий автомат.

Рассмотренная тема предполагает ряд вариантов развития, некоторые из которых перечислены ниже. Во-первых, рассмотрение возможных аналогов полурешеток для автомата Kiel [16]. (несмотря на то что всеми свойствами автомата Ватерлоо автомат Kiel не обладает, исследование последнего представляется достаточно интересным). Во-вторых, поиск

«минимального» автомата, обладающего теми же свойствами, что и автомат Ватерлоо, т. е. автомата, для которого существует неэквивалентный покрывающий автомат; при этом имеются по крайней мере две возможности для определения такой минимальности: либо по числу состояний автомата, либо по произведению количеств состояний двух канонических автоматов (для заданного регулярного языка и для его зеркального образа). В-третьих, описание эвристик для человеко-машинных систем, предназначенных для автоматического поиска Ватерлоо-подобных автоматов.

Литература

1. Гинзбург С. Математическая теория контекстно-свободных языков. М.: Мир, 1973. 301 с.
2. Брауэр В. Введение в теорию конечных автоматов. М.: Радио и связь, 1987. 390 с.
3. Мельников Б. Регулярные языки и недетерминированные конечные автоматы. М.: Изд-во Российского государственного социального университета, 2018. 179 с.
4. Долгов В., Мельников Б., Мельникова А. Циклы графа переходов базисного конечного автомата и связанные вопросы // Вестник Воронежского государственного университета. Серия: Физика. Математика. 2016. № 4. С. 95–111.
5. Долгов В., Мельников Б. Построение универсального конечного автомата. II. Примеры работы алгоритмов // Вестник Воронежского государственного университета. Серия: Физика. Математика. 2014. № 1. С. 78–85.
6. Melnikov B., Melnikova A. Edge-minimization of non-deterministic finite automata // The Korean Journal of Computational and Applied Mathematics (Journal of Computational and Applied Mathematics). 2001. Vol. 8, no. 3. P. 469–479.
7. Jiang T., Ravikumar B. Minimal NFA problems are hard // SIAM Journal on Computing. 1993. Vol. 22, no. 6. P. 1117–1141. DOI: 10.1137/0222067.
8. Мельников Б.Ф. Полурешетки подмножеств потенциальных корней в задачах теории формальных языков. Часть I. Извлечение корня из языка // International Journal of Open Information Technologies. 2022. Т. 10, № 4. С. 1–9.
9. Мельников Б.Ф. Полурешетки подмножеств потенциальных корней в задачах теории формальных языков. Часть II. Построение инверсного морфизма // International Journal of Open Information Technologies. 2022. Т. 10, № 5. С. 1–8.
10. Мельников Б.Ф. Полурешетки подмножеств потенциальных корней в задачах теории формальных языков. Часть III. Условие существования решетки // International Journal of Open Information Technologies. 2022. Т. 10, № 7. С. 1–9.
11. Зяблицева Л.В., Корабельщикова С.Ю., Абрамян М.Э. Графы полурешеток, матричные представления полугрупп идемпотентов и полурешетки автомата Ватерлоо // International Journal of Open Information Technologies. 2023. Т. 11, № 7. С. 69–76.
12. Abramyan M.E., Melnikov B.F. A Program Study of the Union of Semilattices on the Set of Subsets of Grids of Waterloo Language // Journal of Applied Mathematics and Physics. 2023. Vol. 11. P. 1459–1470. DOI: 10.4236/jamp.2023.115095.
13. Kameda T., Weiner P. On the state minimization of nondeterministic finite automata // IEEE Transactions on Computers. 1970. Vol. C-19, no. 7. P. 617–627. DOI: 10.1109/t-c.1970.222994.
14. Абрамян М.Э. О вычислении веса подзадач при вершинной минимизации недетерминированных конечных автоматов методом ветвей и границ // Известия высших учебных заведений. Поволжский регион. Физико-математические науки. 2021. № 2 (58). С. 46–52. DOI: 10.21685/2072-3040-2021-2-4.

15. Melnikov B. The complete finite automaton // International Journal of Open Information Technologies. 2017. Vol. 5, no. 10. P. 9–17.

16. Polak L. Minimalizations of NFA using the universal automaton // International Journal of Foundation of Computer Science. 2005. Vol. 16, no. 5. P. 999–1010. DOI: 10.1142/s0129054105003431.

Абрамян Михаил Эдуардович, к.ф.-м.н., доцент, факультет вычислительной информатики и кибернетики, Университет МГУ–ППИ в Шэньчжэне (Шэньчжэнь, Китайская Народная Республика); Институт математики, механики и компьютерных наук им. И.И. Воровича, Южный федеральный университет (Ростов-на-Дону, Российская Федерация)

DOI: 10.14529/cmse240101

A PROGRAM STUDY OF SEMILATTICES CONNECTED WITH THE WATERLOO AUTOMATON

© 2024 M.E. Abramyan[1,2]

[1]*Shenzhen MSU – BIT University*

(International University Park Road 1, Shenzhen, Guangdong Province, 518172 PRC),

[2]*Southern Federal University*

(Str. Bolshaya Sadovaya 105/42, Rostov-on-Don, 344006 Russia)

E-mail: m-abramyan@yandex.ru

Received: 17.08.2023

The paper is devoted to the study of semilattices containing covering automata for the Waterloo automaton. In the initial sections of the paper, the process of constructing covering automata on the basis of subsets of the grids of the original automaton is described (each grid represents some set of arcs associated with the original automaton), and also the properties of semilattices formed by covering automata are considered. The main result of the paper is a complete description of the obtained semilattices from the point of view of equivalence of the covering automata included in them to the original Waterloo automaton. Three classes of semilattices are distinguished, each of which has special properties. For the semilattice constructed on the basis of a minimal covering automaton, a graphical representation is obtained, which allows us to visualize the relations between its sets consisting of pairwise equivalent automata. In addition, a criterion of equivalence of the covering automaton to the Waterloo automaton is formulated in terms of the properties of the subset of grids defining the covering automaton. The study was carried out using the NFALib library for analysis of nondeterministic finite automata, implemented by the author in the C# language. The relevance of the study of the Waterloo automaton is connected with its role in the study of the problem of vertex minimization of nondeterministic finite automata and the development of real-time heuristic algorithms used for its solution.

Keywords: nondeterministic finite automata, universal automaton, grid, covering automaton, equivalent transformation algorithms, the Waterloo automaton.

FOR CITATION

Abramyan M.E. A Program Study of Semilattices Connected with the Waterloo Automaton. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2024. Vol. 13, no. 1. P. 5–21. (in Russian) DOI: 10.14529/cmse240101.

References

1. Ginsburg S. The mathematical theory of context-free languages. Moscow: Mir Publ., 1973. 301 p. (in Russian)
2. Brauer W. Introduction in the Finite Automata Theory. Moscow: Radio I Svyaz Publ., 1987. 390 p. (in Russian)
3. Melnikov B. Regular languages and nondeterministic finite automata: a monograph. Moscow: RGSU Publ., 2018. 179 p. (in Russian)
4. Dolgov V., Melnikov B., Melnikova A. The loops of the basis finite automaton and the connected questions. Bulletin of the Voronezh State University. Series: Physics. Mathematics. 2016. No. 4. P. 95–111. (in Russian)
5. Dolgov V., Malnikov B. Construction of universal finite automaton. II. Examples of algorithms functioning. Bulletin of the Voronezh State University. Series: Physics. Mathematics. 2014. No. 1. P. 78–85. (in Russian)
6. Melnikov B., Melnikova A. Edge-minimization of non-deterministic finite automata. The Korean Journal of Computational and Applied Mathematics (Journal of Computational and Applied Mathematics). 2001. Vol. 8, no. 3. P. 469–479. DOI: 10.1007/bf02941980.
7. Jiang T., Ravikumar B. Minimal NFA problems are hard. SIAM Journal on Computing. 1993. Vol. 22, no. 6. P. 1117–1141. DOI: 10.1137/0222067.
8. Melnikov B.F. Semi-lattices of the subsets of potential roots in the problems of the formal languages theory. Part I. Extracting the root from the language. International Journal of Open Information Technologies. 2022. Vol. 10, no. 4. P. 1–9. (in Russian)
9. Melnikov B.F. Semi-lattices of the subsets of potential roots in the problems of the formal languages theory. Part II. Constructing an inverse morphism. International Journal of Open Information Technologies. 2022. Vol. 10, no. 5. P. 1–8. (in Russian)
10. Melnikov B.F. Semi-lattices of the subsets of potential roots in the problems of the formal languages theory. Part III. The condition for the existence of a lattice. International Journal of Open Information Technologies. 2022. Vol. 10, no. 7. P. 1–9. (in Russian)
11. Zyablitceva L.V., Korabelshchikova S.Yu., Abramyan M.E. Semilattice graphs, matrix representations of idempotent semigroups and Waterloo automaton semilattices. International Journal of Open Information Technologies. 2023. Vol. 11, no. 7. P. 69–76. (in Russian)
12. Abramyan M.E., Melnikov B.F. A Program Study of the Union of Semilattices on the Set of Subsets of Grids of Waterloo Language. Journal of Applied Mathematics and Physics. 2023. Vol. 11. P. 1459–1470. DOI: 10.4236/jamp.2023.115095.
13. Kameda T., Weiner P. On the state minimization of nondeterministic finite automata. IEEE Transactions on Computers. 1970. Vol. C-19, no. 7. P. 617–627. DOI: 10.1109/t-c.1970.222994.
14. Abramyan M.E. Computing the weight of subtasks in state minimization of nondeterministic finite automata by the branch and bound method. University proceedings. Volga region. Physical and mathematical sciences. 2021. No. 2 (58). P. 46–52. (in Russian). DOI: 10.21685/2072-3040-2021-2-4.
15. Melnikov B. The complete finite automaton. International Journal of Open Information Technologies. 2017. Vol. 5, no. 10. P. 9–17.
16. Polak L. Minimalizations of NFA using the universal automaton. International Journal of Foundation of Computer Science. 2005. Vol. 16, no. 5. P. 999–1010. DOI: 10.1142/s0129054105003431.

УДК 004.021, 519.116 DOI: 10.14529/cmse240102

РАСПРЕДЕЛЕНИЕ КВАДРАТОВ И ПРОВЕРКА ГИПОТЕЗ В НЕЧЕТНЫХ РАЗБИЕНИЯХ ЧИСЕЛ

© 2024 А.А. Самойлов

Южно-Уральский государственный университет
(454080 Челябинск, пр. им. В.И. Ленина, д. 76)
E-mail: samoilovaa@susu.ru

Поступила в редакцию: 10.03.2023

В статье рассматриваются разбиения натурального числа n, части которого различны, нечетны и их произведение не является квадратом. Такие разбиения применимы для определения ранга группы центральных единиц целочисленного группового кольца знакопеременной группы. Количество разбиений растет экспоненциально, следовательно, задача перебора является вычислительно затратной. В статье предложен параллельный алгоритм в общей памяти для нахождения количества разбиений числа n с дополнительными условиями. Алгоритм основан на концепции распараллеливания по данным и использовании вложенного параллелизма. Выделяется множество длин K разбиения числа n, элементы которого обрабатываются параллельно. Во время обработки длины k разбиения числа n выделяется множество уровней L, рассмотрение которого также выполняется параллельно. Приемлемые значения ускорения и параллельной эффективности предложенного алгоритма получаются при использовании двух нитей на параллельный регион по длинам и двух — по уровням. Таким образом, ускорение при разных n превышает 2.1, а параллельная эффективность не опускается ниже 50%. Полученные результаты использованы для проверки гипотез Каргаполова и анализа распределения значений нечетных разбиений на некоторых диапазонах. Предложен алгоритм поиска оптимального коэффициента c. С помощью этого алгоритма получена асимптотическая формула количества разбиения числа n, в котором части различны и нечетны, а их произведение является квадратом. Эта формула основана на экспериментальных данных и сформулирована как гипотеза.

Ключевые слова: разбиение натурального числа, асимптотическая формула, параллельные вычисления, OpenMP.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Самойлов А.А. Распределение квадратов и проверка гипотез в нечетных разбиениях чисел // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2024. Т. 13, № 1. С. 22–37. DOI: 10.14529/cmse240102.

Введение

В этой работе рассматриваются следующие условия на разбиение $\lambda = (a_1, \ldots, a_k)$ натурального числа n.

1. a_i нечетно при $1 \le i \le k$.
2. $a_i \ne a_j$ при $i \ne j$.
3. $n \equiv k \pmod 4$.
4. $\prod\limits_{i=1}^{k} a_i$ *не является квадратом* натурального числа.

Число разбиений, удовлетворяющих первым двум условиям, обозначается $r(n)$, удовлетворяющих первым трем условиям — $r_4(n)$, а разбиение, у которого выполняются первые три условия и последнее нарушается, называется *квадратичным нечетным разбиением* числа n, и число таких разбиений обозначается $qop(n)$. Количество разбиений, удовлетворяющих всем условиям, обозначается $rank(n)$. Можно заметить, что $qop(n) = r_4(n) - rank(n)$.

Одно из практических применений этих четырех условий — это определение ранга группы центральных единиц целочисленного группового кольца знакопеременной группы A_n [1].

Эта проблема рассматривалась авторами из работы [2], которым удалось достигнуть результата для $n \leq 600$ с шагом 100. В работе [3] Каргаполовым достигнут результат для $n \leq 1000$ с шагом 10. Каргаполов выдвинул предположения, связанные с оценкой ранга группы центральных единиц целочисленного группового кольца знакопеременной группы A_n.

Разбиения чисел могут быть применены в криптографии для создания протоколов аутентификации и шифрования. Известно, что число классов сопряженных элементов симметрической группы S_n равно количеству разбиений числа n (см. [4]). Пример криптосистемы на основе симметрических групп можно найти в [5].

Целью исследования являются следующие задачи.

1. Проверить применимость предположений Каргаполова для различных диапазонов натуральных чисел.
2. Вычислить количество и найти оценки квадратичных нечетных разбиений для различных диапазонов натуральных чисел.

Для достижения поставленных задач предложен и реализован алгоритм с шагом 1, который использует стандарт для распараллеливания программ OpenMP.

Статья организована следующим образом. В разделе 1 описаны теоретические сведения о разбиении числа n, количестве и оценке. В разделе 2 приведен алгоритм подсчета количества разбиений с дополнительными условиями и предложен метод поиска оптимального коэффициента c. В разделе 3 представлены реализация алгоритма с использованием стандарта распараллеливания программ OpenMP и вычислительные эксперименты. В разделе 4 приведена проверка гипотез Каргаполова и дано предположение об асимптотической формуле числа квадратичных нечетных разбиений. Заключение содержит краткую сводку результатов, полученных в работе.

1. Разбиение числа, количество и оценка

Определение 1. Разбиение λ натурального числа n — представление этого числа в виде суммы натуральных чисел

$$n = a_1 + a_2 + \cdots + a_k, \tag{1}$$

где, a_i называются частями разбиения.

Разбиения одного числа считаются равными, если они различаются только порядком слагаемых. Вычисления, связанные с разбиениями чисел, являются вычислительно затратными, так как количество разбиений быстро увеличивается с ростом n [6]. Приблизительно число разбиений $p(n)$ можно найти с помощью формулы Харди—Рамануджана.

Теорема 1. (Теорема 6.3 [6], §2.7 [7]). При $n \to \infty$ имеет место асимптотическая формула для числа разбиений

$$p(n) \sim \frac{1}{4n\sqrt{3}} e^{\pi\sqrt{\frac{2n}{3}}}. \tag{2}$$

Один из подходов вычисления количества разбиений с условиями из введения следующий. Генерацию разбиений можно осуществить с помощью алгоритма Липского, который перебирает все разбиения числа n в невозрастающем порядке (алгоритм 1.22. [8]). Этот алгоритм нуждается в изменениях, чтобы части разбиения соответствовали условиям различности, нечетности, и добавлении проверки на квадрат. Для сокращения времени на

вычисление количества разбиений прибегают к параллельной обработке в распределенной памяти. Для этого все множество разбиений делят на равные части, и каждый узел кластера выполняет генерацию разбиений в заданном диапазоне.

В этой статье предложен параллельный алгоритм в общей памяти с использованием вложенного параллелизма для нахождения количества разбиений числа n с дополнительными условиями. Вложенный параллелизм заключается в том, что каждая нить, в которой встретится параллельный регион, породит для его выполнения новую группу нитей.

Лемма 1. (Лемма 2 [2]) Число элементов разбиения k, удовлетворяющих условиям из введения, не превосходит целой части $\sqrt{n}$.

Определим множество длин K разбиения числа n, значения элементов которых не превышают $\sqrt{n}$ (см. лемму 1) и удовлетворяют третьему условию из введения $n \equiv k \pmod 4$, как

$$K = \{k_i : 1 \leq k_i \leq \lfloor \sqrt{n} \rfloor \wedge n \equiv k_i \pmod 4\}. \tag{3}$$

При генерации разбиений, элементы которых нечетны и попарно различны, можно заметить, что все части некоторых разбиений отличаются на 2. Например, для длины $k = 3$ существует следующий набор разбиений: $l_1 = (1, 3, 5), l_2 = (3, 5, 7), l_3 = (5, 7, 9), \ldots$ Назовем такие случаи *уровнями* и определим множество уровней L разбиения числа n длины k как

$$l_j = \{a_i : 1 \leq i \leq k \wedge a_1 = 1 + 2(j-1) \wedge a_i = a_{i-1} + 2\}, \tag{4}$$

$$L = \{l_j : 1 \leq j \leq \infty \wedge \sum_{i=1}^{k} a_i \leq n, \text{ где } a_i \text{ элемент } l_j\}. \tag{5}$$

Из работы [3] стоит отметить способ проверки на квадрат натурального числа — использование разложения на простые множители. Если число является квадратом, то показатели степеней простых чисел в его разложении четны. Для каждого натурального числа n сохраняется битовый вектор четностей показателей степеней простых чисел, где на i-й позиции стоит остаток от деления на 2 показателя степени i-го простого числа в разложении n.

В работе [3] Каргаполовым получена асимптотическая формула, которая позволяет вычислить приближенно $r(n)$, и высказаны предположения о $r_4(n)$ и $rank(n)$.

Теорема 2. (Теорема 3.1 [3]. При $n \to \infty$ имеет место асимптотическая формула

$$r(n) \sim \frac{1}{2\lambda_n^{\frac{3}{2}}\sqrt[4]{24}} e^{\frac{\pi\lambda_n}{\sqrt{6}}}, \text{ где } \lambda_n = \sqrt{n - \frac{1}{24}}. \tag{6}$$

Предположения Каргаполова

Предположение 1. При $n \to \infty$ имеет место асимптотическая формула

$$r_4(n) \sim \frac{r(n)}{2}. \tag{7}$$

Предположение 2. При $n \to \infty$ имеет место асимптотическая формула

$$rank(n) \sim r_4(n) \sim \frac{1}{4\sqrt[4]{24n^3}} e^{\pi\sqrt{\frac{n}{6}}}. \tag{8}$$

Из теорем 1 и 2 можно сказать об экспоненциальном росте числа разбиений и предположить, что при $n \to \infty$ имеет место следующая форма записи асимптотической формулы для количества разбиений

$$f(n) \sim \frac{a}{n^c} e^{b\sqrt{n}} \tag{9}$$

где, a, b, c некоторые коэффициенты. Гипотеза Каргаполова (8) также соответствует этому выражению.

2. Описание методов

2.1. Алгоритм подсчета числа разбиений

Алгоритм подсчета количества разбиений числа n с указными во ведении условиями состоит из следующих шагов.

1. Вычисление битовых векторов четностей показателей степеней простых чисел.
2. Определение множества длин K разбиения числа n.
3. Определение множества уровней L разбиения числа n длины k_i.
4. Рекурсивный подсчет количества разбиений числа n длины k_i уровня l_j.

Шаг 1. В алг. 1 описывается вычисление битовых векторов четностей показателей степеней простых чисел $D_{N,b}$, где N — количество натуральных чисел, b — количество бит. Для получения простых чисел (функция $GenPrimes$) можно использовать алгоритм решета Эратосфена (§7.1 [9]). Для $N = 1000$ достаточно $b = 168$ бит, так как в диапазоне $[1, 1000]$ существуют 168 простых чисел. Далее проверку на целочисленный квадратный корень можно осуществить путем применения операции xor над битовым вектором каждого элемента разбиения. Произведение частей разбиения числа n является квадратом в том случае, если на выходе получился нулевой вектор (§2.1 [3]).

Алгоритм 1 GenDecomposition (in N, b; out $D_{N,b}$)

```
1: D_{N,b} ← ∅; P ←GenPrimes(N)
2: for i ← 2 to N do
3:     t ← i; D_i ← 0 {инициализация нулевого битового вектора}
4:     for j ← 1 to b do
5:         while t mod P_j = 0 do
6:             D_{i,j} ← ¬D_{i,j}; t ← t/P_j
7: return D_{N,b}
```

Шаг 2. В алг. 2 описываются нахождение (строки 1–4) и параллельная обработка (строки 5–7) множества длин, которое определяется выражением (3). Вычислительные эксперименты показали (см. табл. 2), что большими затратами времени на нахождение количества разбиений обладает средний и ближние к среднему элементы множества K. Следовательно, при параллельном рассмотрении этого множества с использованием двух нитей, одной из них достанется элемент, который обладает большим временем выполнения, а другой — два соседних.

Алгоритм 2 LenParall (in n)

```
1: K_m ← ∅; m ← 0
2: for i ← 1 to ⌊√n⌋ do
3:     if n mod 4 = i mod 4 then
4:         K_m ← i; m ← m + 1
5: #pragma omp parallel for num_threads(2) schedule(dynamic, 1) reduction(+:r_4, qop, rank)
6: for i ← 1 to m do
7:     r_4, qop, rank ← r_4, qop, rank + LevelParall(n, K_i)
```

Шаг 3. Подсчет количества уровней (сами уровни определяются выражениями (4)–(5)) и их параллельная обработка описаны в алг. 3. Строки 1–2 соответствуют частному случаю, в котором происходит проверка на четность и квадрат числа n, когда длина разбиения $k = 1$ (описание функции *SpecialCase* опущена для краткости). Строки 4–12 соответствуют подсчету количества уровней разбиения числа n длины k. В строках 4–5 производится нахождение суммы первого уровня. Далее, в строках 6–12 происходит инкремент переменной m до тех пор, пока сумма элементов разбиения, по определению 1, не станет равной n. Строки 13–19 описывают параллельную обработку уровней. В строках 17–18 происходит генерация уровня l_j для разбиения числа n длины k, и вычисляется сумма частей этого разбиения.Вычислительные эксперименты показали (см. табл. 3), что чем меньше номер уровня, тем большими затратами времени на вычисление количества разбиений он обладает.

Алгоритм 3 LevelParall (in n, k; out $r_4, qop, rank$)

```
1: if k = 1 then
2:     return SpecialCase (in n; out r_4, qop, rank)
3: m ← 0; sum ← 0; r_4 ← 0; rank ← 0; qop ← 0
4: for i ← 1 to k do
5:     sum ← sum + 1 + 2(i − 1)
6: while true do
7:     if sum > n then
8:         break
9:     else if sum = n then
10:        m ← m + 1; break
11:    else
12:        m ← m + 1; sum ← sum + 2k
13: #pragma omp parallel for num_threads(2) schedule(dynamic, 1) reduction(+:r_4, qop, rank)
14: for j ← 1 to m do
15:    L_k^k ← 0; sum ← 0
16:    for i ← 1 to k do
17:        L_i^1 ← 1 + 2(j − 1) + 2(i − 1)
18:        sum ← sum + L_i^1
19:    r_4, qop, rank ← r_4, qop, rank + LevelRec(2, sum, n, k, L_k^k)
20: return r_4, qop, rank
```

Шаг 4. На последнем этапе (см. алг. 4) выполняется рекурсивная генерация разбиений числа n длины $k > 1$ от второго элемента к последнему в рамках заданного уровня l. Перед вызовом нового шага рекурсии сохраняется текущее разбиение (строка 4). Для удовлетворения условиям нечетности и различности элементов на каждом j-м шаге после вызова рекурсии прибавляется к частям разбиения, которые стоят на позициях в диапазоне $[j, k]$, число 2 и пересчитывается их сумма (строки 6–8). При достижении $j = k$ прибавляется к последнему элементу разбиения недостающая сумма, чтобы соответствовать определению 1 (строка 10). Далее, выполняется проверка на квадрат числа с помощью показателей степеней простых чисел (строки 13–15).

Алгоритм 4 LevelRec (in j, sum, n, k, L_k^k; out qop, $rank$, r_4)

1: **while** $true$ **do**
2: **if** $sum < n$ **then**
3: **if** $k \neq j$ **then**
4: $L^{j+1} \leftarrow L^j$ {Сохранение текущего разбиения для следующего шага рекурсии}
5: $qop, rank, r_4 \leftarrow qop, rank, r_4 + \text{LevelRec}(j+1, sum, n, k, L_k^k)$
6: $sum \leftarrow sum + 2(k-j-1)$ {Сумма частей разбиения на текущем шаге рекурсии}
7: **for** $i \leftarrow j$ **to** k **do**
8: $L_i^j \leftarrow L_i^j + 2$ {Прибавление числа 2 к частям разбиения начиная с позиции равному шагу рекурсии}
9: **else**
10: $L_k^j \leftarrow L_k^j + (n - sum)$; $sum \leftarrow n$ {На последнем шаге рекурсии прибавляется недостающая сумма к последнему элементу}
11: **else if** $sum = n$ **then**
12: $r_4 \leftarrow r_4 + 1$; $factor \leftarrow 0$
13: **for** $i \leftarrow 1$ **to** k **do**
14: $factor \leftarrow factor \oplus D_{L_i^j}$ {Использование битовых векторов четностей показателей степеней простых чисел для проверки на квадрат}
15: **if** $factor = 0$ **then**
16: $qop \leftarrow qop + 1$ {Произведение элементов разбиения является квадратом}
17: **else**
18: $rank \leftarrow rank + 1$
19: **return** $qop, rank, r_4$
20: **else**
21: **return** $qop, rank, r_4$

2.2. Поиск оптимального коэффициента c

В алг. 5 описывается поиск оптимального коэффициента c. Пусть X_N содержит значения $n \in [1, N]$, Y_N — количество разбиений числа n, и задан замкнутый интервал $C \in [begin, end]$ с шагом ϵ. Возьмем во внимание, что при выполнении аппроксимации на разных диапазонах заданных точек можно получить разные значения параметров. *Поиск оптимального коэффициента c* заключается в нахождении такого значения коэффициента c, при котором отклонение двух других коэффициентов a и b из (9) на меньшем и большем диапазонах аппроксимации минимально. Будем считать большим диапазоном аппроксима-

ции равный $n \in [1, N]$, а меньшим — $n \in [\lfloor N/8 \rfloor, \lfloor N/4 \rfloor]$ (строки 2–3). Последовательно рассматривая C с шагом ϵ и выполняя функцию $CurveFit$ получим значения коэффициентов a_1, b_1 для меньшего и a_2, b_2 для большего диапазонов аппроксимации (строки 4–11). Для подбора коэффициентов по формуле (9) и в качестве реализации функции $CurveFit$ можно использовать нелинейный метод наименьших квадратов (§2.4 [10]). Эта функция на вход принимает коэффициент c и табличные значения X_N, Y_N (строки 6–7). Обозначим разность коэффициентов как $t_1 = |a_2 - a_1|$ и $t_2 = |b_2 - b_1|$. Выбираем то значение c, у которого на замкнутом интервале C значения t_1 и t_2 минимальны (строки 8–10). Будем считать оптимальным значением c то, у которого на меньшем промежутке аппроксимации значения t_1 и t_2 минимальны (строки 12–13). Например, для $rank(n)$ при $C \in [0, 1]$ и $\epsilon = 0.001$ по предложенному алгоритму получен результат $c = 0.731 \sim 3/4$, который близок к значению в гипотезе Каргаполова (8).

Алгоритм 5 OPTIMIZE (IN $begin$, end, ϵ, X_N, Y_N; OUT opt_{t_1}, opt_{t_2}, opt_c)

```
1:  opt_t1 ← ∞; opt_t2 ← ∞; opt_c ← begin
2:  for n ← ⌊N/8⌋ to ⌊N/4⌋ do
3:      A ← X_1..n; B ← Y_1..n {Копируем n элементов}
4:      tmp_t1 ← ∞; tmp_t2 ← ∞; tmp_c ← begin; c ← begin
5:      while c ≤ end do
6:          a_1, b_1 ← CurveFit(A, B, c)
7:          a_2, b_2 ← CurveFit(X, Y, c)
8:          t_1 ← |a_2 − a_1|; t_2 ← |b_2 − b_1|
9:          if tmp_t1 > t_1 and tmp_t2 > t_2 then
10:             tmp_t1 ← t_1; tmp_t2 ← t_2; tmp_c ← c
11:         c ← c + ε
12:     if opt_t1 > tmp_t1 and opt_t2 > tmp_t2 then
13:         opt_t1 ← tmp_t1; opt_t2 ← tmp_t2; opt_c ← tmp_c
14: return opt_t1, opt_t2, opt_c
```

3. Реализация и вычислительные эксперименты

Алгоритм подсчета числа разбиений реализован на языке C при использовании стандарта для распараллеливания программ OpenMP. Поиск оптимального коэффициента c реализован на языке Python. Исходные коды свободно доступны в репозитории GitHub [11].

По предложенному алгоритму возможно выполнить вложенный параллелизм следующим образом.

1. Параллельный регион из двух OpenMP-нитей на обработку множества длин K разбиения числа n.
2. Параллельный регион из двух OpenMP-нитей на обработку множества уровней L разбиения числа n в рамках соответствующей длины k.

На рис. 1 представлена схема, согласно которой программа создаст, при максимальной загрузке, четыре OpenMP-нити. Для поддержки вложенного параллелизма, необходимо изменить переменную окружения, отвечающую за данную опцию, с помощью функции $omp\_set\_nested(1)$.

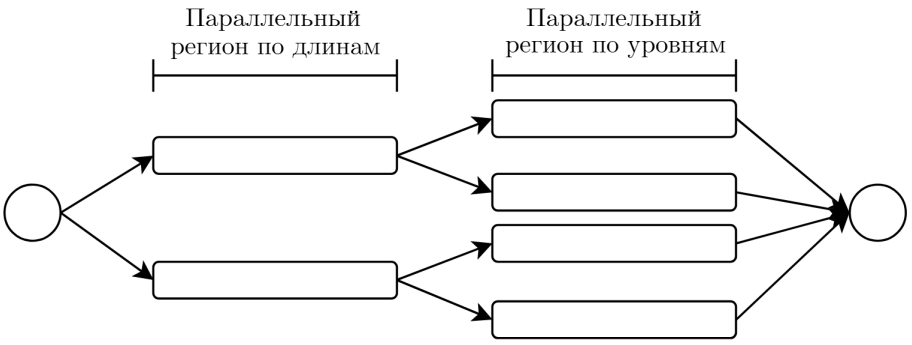


Рис. 1. Схема распараллеливания

Для вычислительных экспериментов использован персональный компьютер, который обладает характеристиками, указанными в табл. 1.

Таблица 1. Характеристики

Параметр	**Значение**
Процессор	AMD Ryzen 9 5950X 3.40 GHz
Число физ./лог. ядер	16/32
Кэш	L1 — 1.0 МБ, L2 — 8.0 МБ, L3 — 64.0 МБ
Операционная система	Windows 10 Pro x64
Оперативная память	32 ГБ, 3200 MHz

Время выполнения алгоритма для элементов k_i из множества длин K отображено в табл. 2 в зависимости от n. Вычислительные эксперименты показали, что количество разбиений числа n длины k_i возрастает до некоторого элемента множества K, а затем уменьшается. В большинстве случаев основное время вычисления занимает элемент с индексом $i = \lceil (|K| + 1)/2 \rceil$.

Таблица 2. Время выполнения k_i, с.

$\boldsymbol{n}$	$\boldsymbol{i = 1}$	$\boldsymbol{i = 2}$	$\boldsymbol{i = 3}$	$\boldsymbol{i = 4}$	$\boldsymbol{i = 5}$	$\boldsymbol{i = 6}$	$\boldsymbol{i = 7}$
400	0.001	0.387	1.662	0.043	0.001	-	-
425	0.001	0.007	1.586	2.882	0.032	-	-
450	0.001	0.051	4.748	4.125	0.019	-	-
475	0.001	0.374	13.104	5.192	0.009	-	-
500	0.001	2.042	30.408	5.646	0.004	-	-
525	0.001	0.016	9.909	62.869	5.353	0.001	-
550	0.001	0.144	34.640	107.794	4.288	0.001	-
575	0.001	1.229	111.656	165.038	2.956	0.001	-
600	0.002	7.842	301.919	214.401	1.698	0.001	-
625	0.001	0.033	44.023	725.841	247.097	0.817	0.001
650	0.001	0.338	176.511	1447.473	245.358	0.322	-
675	0.001	3.326	650.631	2547.758	211.521	0.101	-
700	0.003	24.070	2008.258	3903.361	156.567	0.025	-

Время выполнения алгоритма для элементов l_j из множества уровней L отображено в столбцах 5–9 в табл. 3 в зависимости от n. Во втором столбце указана длина k_i разбиения числа n, где $i = \lceil (|K|+1)/2 \rceil$, в третьем — мощность множества $|L|$, в четвертом — количество $r_4(n)$ для числа n длины k_i. Вычислительные эксперименты показали, что количество разбиений числа n длины k_i уровня l_j уменьшается с ростом j.

Таблица 3. Время выполнения l_j, с.

n	k_i	$\lvert L\rvert$	r_4	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$
400	12	11	$1.3 \cdot 10^8$	0.943	0.430	0.185	0.073	0.026
425	9	20	$1.4 \cdot 10^8$	0.506	0.361	0.252	0.173	0.117
450	10	18	$4.2 \cdot 10^8$	1.770	1.167	0.753	0.474	0.289
475	11	17	$1.1 \cdot 10^9$	5.524	3.352	1.979	1.128	0.619
500	12	15	$2.5 \cdot 10^9$	14.308	7.967	4.275	2.186	1.069
525	13	14	$4.8 \cdot 10^9$	33.040	16.661	8.021	3.660	1.571
550	14	13	$7.9 \cdot 10^9$	61.741	28.069	12.045	4.841	1.800
575	15	12	$1.2 \cdot 10^{10}$	100.423	40.685	15.391	5.377	1.707
600	16	11	$1.5 \cdot 10^{10}$	140.244	50.385	16.635	4.977	1.324
625	13	18	$5.5 \cdot 10^{10}$	334.202	193.965	107.724	58.396	30.497
650	14	17	$1.1 \cdot 10^{11}$	724.543	391.159	199.100	97.121	45.406
675	15	16	$1.8 \cdot 10^{11}$	1377.487	677.709	313.998	137.651	57.033
700	16	14	$2.7 \cdot 10^{11}$	2280.791	1014.834	422.054	164.246	59.489

Таблица 4 содержит затраченное время в секундах на выполнение алгоритма поиска количества разбиения с дополнительными условиями для числа n различной размерности. В таблице th_k — число нитей на параллельный регион по длинам и th_l — число нитей на параллельный регион по уровням.

Таблица 4. Время выполнения алгоритма, с.

n	$th_k=1$				$th_k=2$			
	$th_l=1$	$th_l=2$	$th_l=3$	$th_l=4$	$th_l=1$	$th_l=2$	$th_l=3$	$th_l=4$
400	2.094	1.179	1.123	1.105	1.672	0.958	0.958	0.973
500	38.255	20.773	19.532	19.306	30.770	15.684	14.721	14.636
600	530.554	299.363	269.692	268.925	308.938	156.577	147.153	146.164
700	6091.733	3422.781	3143.981	3137.729	3920.940	2323.431	2300.573	2309.608

Ускорение, получаемое при использовании параллельного алгоритма для p ядер, по сравнению с последовательным вариантом выполнения вычислений, определяется как

$$S_p(n) = \frac{T_1(n)}{T_p(n)}, \tag{10}$$

где $T_1(n)$ — время выполнения последовательного алгоритма и $T_p(n)$ — время выполнения параллельного алгоритма при использовании p ядер.

Эффективность — средняя доля времени выполнения параллельного алгоритма, в течение которого ядра процессора реально используются для решения задачи, определяется

как

$$E_p(n) = \frac{S_p(n)}{p}. \tag{11}$$

Из практики известно, что одна нить использует одно ядро в многоядерном процессоре, тогда число используемых ядер равно $th_k * th_l$. Результаты исследования показаны на рис. 2 при $th_k = 2$. Вычислительные эксперименты показали, что оптимальным вариантом является использование двух нитей на параллельный регион по длинам $th_k = 2$ и двух — по уровням $th_l = 2$. Например, при использовании 6 ядер ускорение увеличивается незначительно, а эффективность становиться меньше 50% при некоторых n.

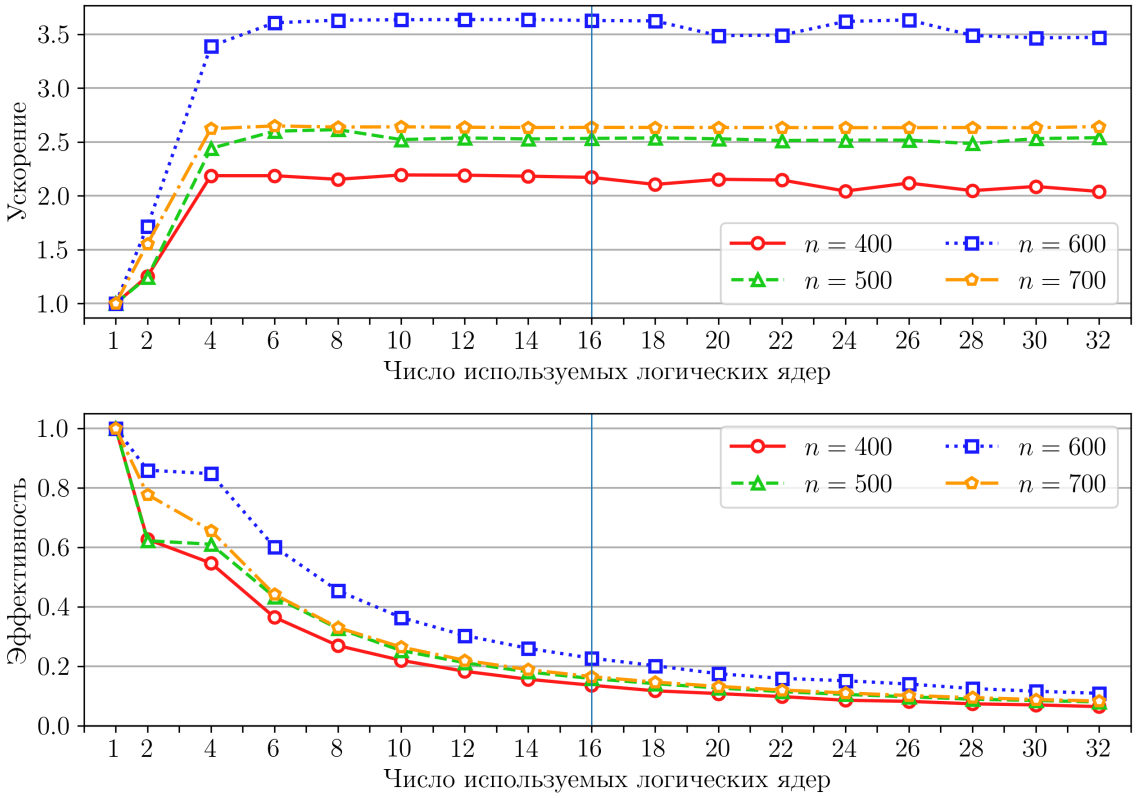


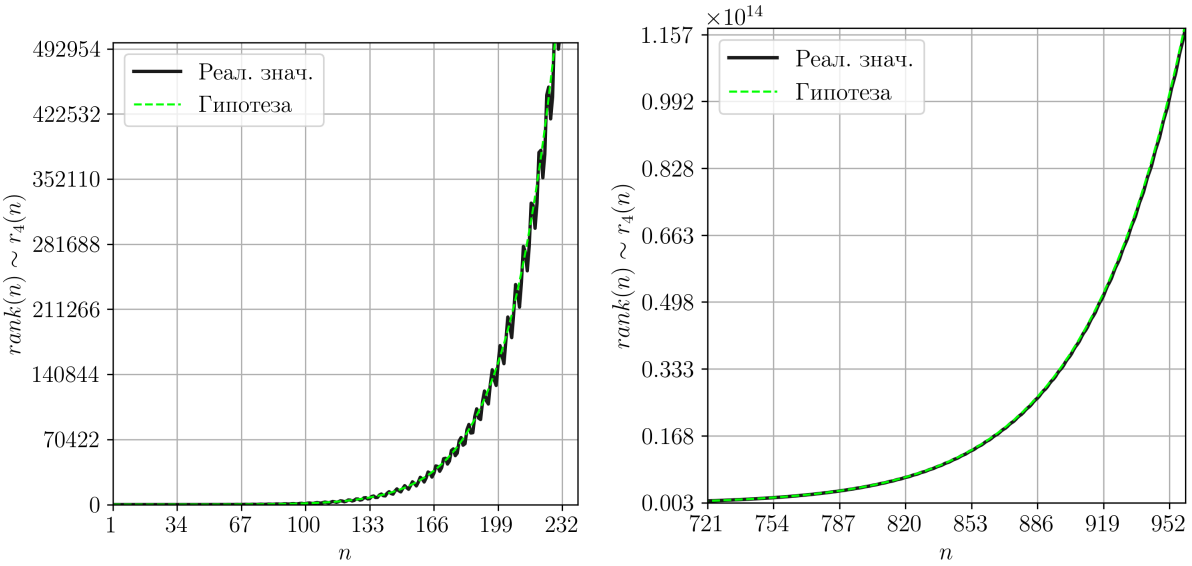
Рис. 2. Ускорение и параллельная эффективность алгоритма при $th_k = 2$

4. Анализ результатов

4.1. Гипотезы Каргаполова

Вычислены значения количества разбиений для трех величин, соответствующим условиям из введения, для $n \leq 960$ с шагом 1. Для проверки применимости предположений Каргаполова (7) и (8), рассмотрим два диапазона значений $n \in [1, 240]$ и $n \in [721, 960]$. Выбор обусловлен тем, что при небольших значениях n присутствуют колебания графика количества разбиений, но с возрастанием n они визуально исчезают.

Расчет по формулам (7) и (8) показал, что обе гипотезы дают приблизительно одинаковые значения. На рис. 3 изображен график функции из второй гипотезы Каргаполова. На рис. 3а присутствуют колебания $rank(n) \sim r_4(n)$, но при увеличении n кривая сглаживается (рис. 3б). Относительная погрешность гипотезы от реальных значений показана на рис. 4 (красной линией обозначено пороговое значение 1%).



а) в диапазоне $n \in [1, 240]$ б) в диапазоне $n \in [721, 960]$

Рис. 3. Предположение Каргаполова

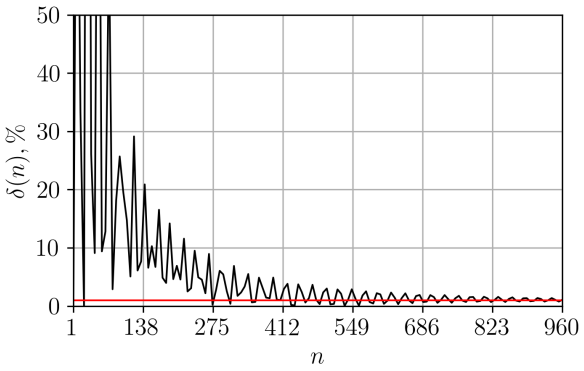


Рис. 4. Относительная погрешность гипотезы

Имеет место следующий вывод по гипотезам Каргаполова, что $rank(n) \sim r_4(n) \sim r(n)/2$ с ростом n (см. табл. 5). Таким образом, можно допустить, что при $n > 960$ гипотезы будут давать точные результаты, но при небольших n имеет место значительное расхождение с реальными значениями.

4.2. Приближенная формула для квадратичных нечетных разбиений

Для получения приближенной формулы воспользуемся методом, описанным в подразделе 2.2. Для $qop(n)$ при $C \in [0, 2]$ и $\epsilon = 0.001$ получаем коэффициент $c = 1.449$. После выполнения аппроксимации по формуле (9) для квадратичных нечетных разбиений, стали известны коэффициенты $a = 0.43$ и $b = 1$. Подставляя результаты в (9) получим при $n \to \infty$

$$qop(n) \sim \frac{0.43}{n^{1.449}} e^{\sqrt{n}}. \tag{12}$$

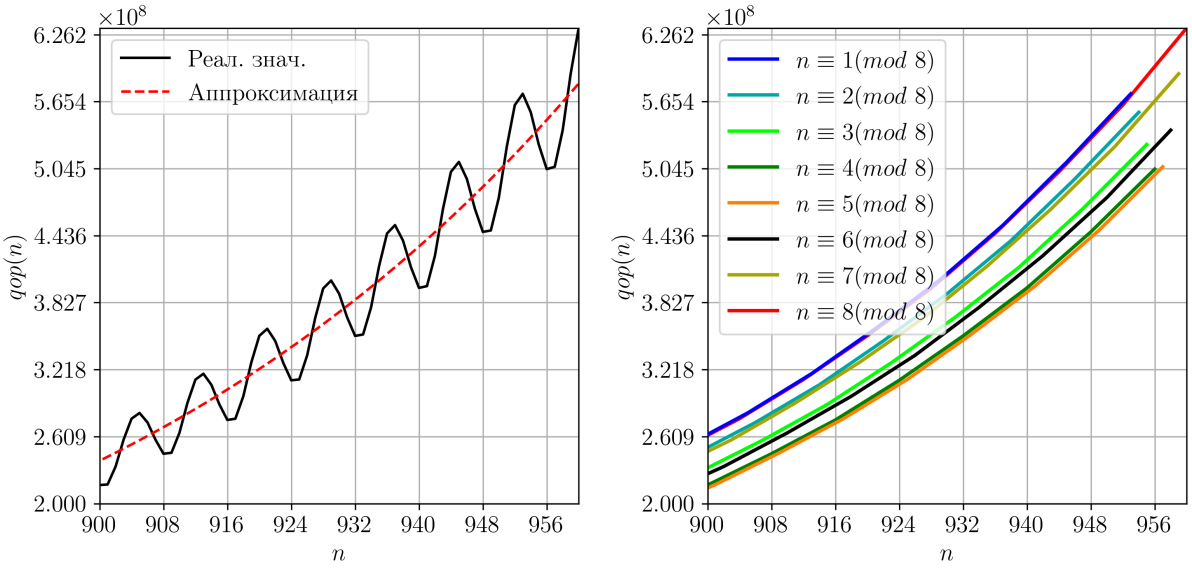
Из графика (см. рис. 5а) функции (12) видно, что простое применение метода не учитывает колебания, которые с ростом n не сглаживаются для $qop(n)$, и имеют период $T = 8$. Разобьем на части в зависимости от сравнения по модулю 8 числа n. Таким образом, на

Таблица 5. Относительная погрешность, %

n	Гипотеза 1	Гипотеза 2
236	2.55	2.39
237	4.15	4.00
238	6.63	6.80
239	9.32	9.49
240	1.69	1.53
...	...	...
956	1.00	1.08
957	0.68	0.76
958	0.92	1.01
959	1.25	1.33
960	1.01	1.09

рис. 5б для $qop(n)$ можно заметить образование четырех пар. Следовательно, для оценки колебаний рассмотрим разложение на 4 части

$$\{8k+1,\ 8k+8\};\ \{8k+2,\ 8k+7\};\ \{8k+3,\ 8k+6\};\ \{8k+4,\ 8k+5\}. \tag{13}$$



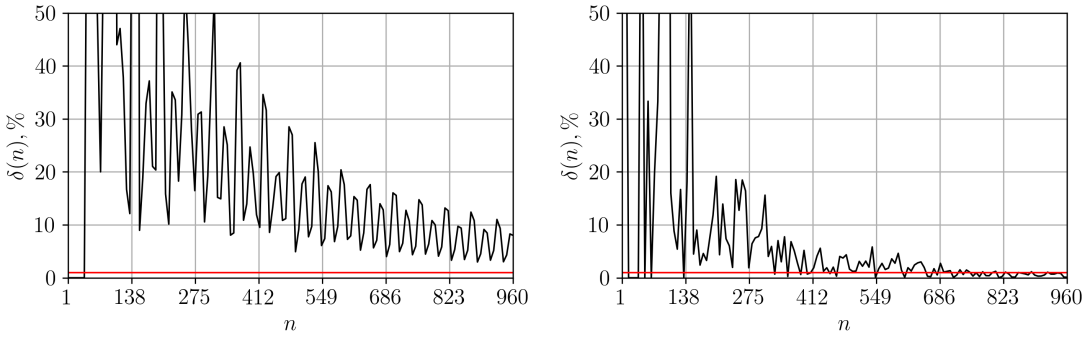
а) без разложения б) разложение на части

Рис. 5. Квадратичные нечетные разбиения

При таком рассмотрении количества квадратичных нечетных разбиений, получены коэффициенты c для каждой части из списка, и найдены a и b при выполнении аппроксимации по формуле (9). Полученные результаты сформулированы как предположение 1. При разложении $qop(n)$ на 4 части относительная погрешность уменьшилась (см. рис. 6 и табл. 6).

Таблица 6. Относительная погрешность $qop(n)$, %

n	Без разложения	Разложение на 4 части
236	49.69	19.43
237	67.48	9.58
238	30.19	17.88
239	18.25	1.96
240	36.85	11.40
...	...	...
956	8.87	0.69
957	10.02	0.38
958	4.75	0.83
959	2.94	0.47
960	7.99	0.10



а) без разложения б) разложение на 4 части

Рис. 6. Относительная погрешность $qop(n)$

Предположение 1. При $n \to \infty$ имеет место асимптотическая формула

$$qop(n) \sim \begin{cases} \frac{3.301}{n^{1.802}} e^{1.015\sqrt{n}}, & \text{при } n \equiv 1 \pmod 8 \text{ или } n \equiv 8 \pmod 8 \\ \frac{1.888}{n^{1.711}} e^{1.011\sqrt{n}}, & \text{при } n \equiv 2 \pmod 8 \text{ или } n \equiv 7 \pmod 8 \\ \frac{0.277}{n^{1.381}} e^{0.998\sqrt{n}}, & \text{при } n \equiv 3 \pmod 8 \text{ или } n \equiv 6 \pmod 8 \\ \frac{0.003}{n^{0.536}} e^{0.952\sqrt{n}}, & \text{при } n \equiv 4 \pmod 8 \text{ или } n \equiv 5 \pmod 8. \end{cases} \tag{14}$$

Заключение

В работе рассмотрены разбиения натурального числа n, части которого различны, нечетны и их произведение не является квадратом. Задача нахождения количества разбиений числа n является вычислительно затратной, так как число разбиений быстро увеличивается с ростом n.

Предложен параллельный алгоритм в общей памяти для нахождения количества разбиений числа n с дополнительными условиями. Алгоритм основан на концепции распараллеливания по данным и использовании вложенного параллелизма. Выделяется множество длин K разбиения числа n, элементы которого обрабатываются параллельно. Во время обработки длины k разбиения числа n выделяется множество уровней L, рассмотрение которого также выполняется параллельно. Приемлемые значения ускорения и параллель-

ной эффективности предложенного алгоритма получаются при использовании двух нитей на параллельный регион по длинам и двух — по уровням. Таким образом, ускорение при разных n превышает 2.1, а параллельная эффективность не опускается ниже 50%.

Рассмотрены гипотезы Каргаполова для оценки количества нечетных разбиений, соответствующие условиям из введения, для $r_4(n)$ и $rank(n)$. Анализ результатов показал, что на диапазоне $n \in [1, 960]$ с возрастанием n разница между реальными и оценочными значениями быстро уменьшается и относительная погрешность также незначительна. Таким образом, можно допустить, что оба предположения Каргаполова верны при $n > 960$.

Предложен алгоритм поиска оптимального коэффициента c. Этот алгоритм позволяет подобрать коэффициенты a и b с учетом того, что на разных диапазонах аппроксимации, могут получится разные значения этих коэффициентов. Для оценки количества квадратичных нечетных разбиений, множество значений $qop(n)$ разбивается на 4 части в зависимости от числа n. Затем использовался предложенный метод поиска оптимального коэффициента c. После подбора коэффициентов a, b и c из полученных результатов сформулировано предположение 1.

В качестве дальнейшего исследования может быть предложено теоретическое доказательство или опровержение предположения 1.

Литература

1. Ferraz R.A. Simple components and central units in group algebras // Journal of Algebra. 2004. Vol. 279, no. 1. P. 191–203. DOI: 10.1016/j.jalgebra.2004.05.005.
2. Алеев Р.Ж., Каргаполов А.В., Соколов В.В. Ранги групп центральных единиц целочисленных групповых колец знакопеременных групп // Фундаментальная и прикладная математика. 2008. Т. 14, № 7. С. 15–21.
3. Каргаполов А.В. Центральные единицы целочисленных групповых колец знакопеременных групп: дис. ... канд. / Каргаполов Андрей Валерьевич. Южно-Уральский Государственный Университет, Россия, 2012.
4. Sagan B.E. The Symmetric Group: Representations, Combinatorial Algorithms, and Symmetric Functions. New York: Springer New York, 2001. XVI, 240 p. DOI: 10.1007/978-1-4757-6804-6.
5. Doliskani J.N., Malekian E., Zakerolhosseini A. A Cryptosystem Based on the Symmetric Group S_n // IJCSNS International Journal of Computer Science and Network Security. 2008. Vol. 8, no. 2. P. 226–234.
6. Эндрюс Г. Теория разбиений. Москва: Наука, 1982. 256 с.
7. Постников А.Г. Введение в аналитическую теорию чисел. Москва: Наука, 1971. 416 с.
8. Липский В. Комбинаторика для программистов. Москва: Мир, 1988. 200 с.
9. Андерсон Д.А. Дискретная математика и комбинаторика. Москва: Вильямс, 2004. 960 с.
10. Kelley C.T. Iterative Methods for Optimization. University City, Philadelphia: Society for Industrial and Applied Mathematics, 1999. 196 p. DOI: 10.1137/1.9781611970920.
11. Самойлов А.А. Исследование разбиений с дополнительными условиями. URL: https://github.com/SashaSamoilov/qop-partitions (дата обращения: 25.09.2023).

Самойлов Александр Андреевич, аспирант, кафедра системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

DOI: 10.14529/cmse240102

DISTRIBUTION OF SQUARES AND HYPOTHESIS VERIFICATION IN ODD INTEGER PARTITIONS

© 2024 A.A. Samoilov

South Ural State University (pr. Lenina 76, Chelyabinsk, 454080 Russia)

E-mail: samoilovaa@susu.ru

Received: 10.03.2023

The article considers partitions of a natural number n whose parts are distinct, odd and their product is not a square. Such partitions are applicable for determining the rank of the group of central units of an integral group ring of an alternating group. The number of partitions grows exponentially, hence the enumeration task is computationally expensive. The article proposes a parallel algorithm in shared memory for finding the number of partitions of the n with additional conditions. The algorithm is based on the concept of data parallelism and the use of nested parallelism. A set of lengths K of the partition of the n is obtained, the elements of which are processed in parallel. During the processing of the length k of the partition of the n, the set of levels L is obtained, the elements of which are processed in parallel as well. Acceptable values of speedup and parallel efficiency of the proposed algorithm are obtained by using two threads per parallel length region and two threads per parallel level region. Thus, the speedup for distinct n increases to 2.1, and the parallel efficiency does not decrease below 50%. The results obtained were used to check Kargapolov's hypotheses and analyze the distribution of the number of odd partition in certain ranges. An algorithm for finding the optimal coefficient c is proposed. Using this algorithm, an asymptotic formula for the number of partitions of the n is obtained, in which the parts are distinct, odd and their product is a square. This formula is based on experimental data and formulated as a conjecture.

Keywords: integer partition, asymptotic formula, parallel computing, OpenMP.

FOR CITATION

Samoilov A.A. Distribution of Squares and Hypothesis Verification in Odd Integer Partitions. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2024. Vol. 13, no. 1. P. 22–37. (in Russian) DOI: 10.14529/cmse240102.

References

1. Ferraz R.A. Simple components and central units in group algebras. Journal of Algebra. 2004. Vol. 279, no. 1. P. 191–203. DOI: 10.1016/j.jalgebra.2004.05.005.
2. Aleev R.Z., Kargapolov A.V., Sokolov V.V. The ranks of central unit groups of integral group rings of alternating groups. Fundamental and Applied Mathematics. 2008. Vol. 14, no. 7. P. 15–21. (in Russian).
3. Kargapolov A.V. Central units of integral group rings of alternating groups: PhD thesis / Kargapolov Andrey Valerievich. South Ural State University, Russia, 2012. (in Russian).
4. Sagan B.E. The Symmetric Group: Representations, Combinatorial Algorithms, and Symmetric Functions. New York: Springer New York, 2001. XVI, 240 p. DOI: 10.1007/978-1-4757-6804-6.

5. Doliskani J.N., Malekian E., Zakerolhosseini A. A Cryptosystem Based on the Symmetric Group S_n. IJCSNS International Journal of Computer Science and Network Security. 2008. Vol. 8, no. 2. P. 226–234.

6. Andrews G. The Theory of Partitions. Moscow: Nauka, 1982. 256 p. (in Russian).

7. Postnikov A.G. Introduction to Analytic Number Theory. Moscow: Nauka, 1971. 416 p. (in Russian).

8. Lipsky V. Combinatorics for Programmers. Moscow: Mir, 1988. 200 p. (in Russian).

9. Anderson J.A. Discrete Mathematics with Combinatorics. Moscow: Williams, 2004. 960 p. (in Russian).

10. Kelley C.T. Iterative Methods for Optimization. University City, Philadelphia: Society for Industrial and Applied Mathematics, 1999. 196 p. DOI: 10.1137/1.9781611970920.

11. Samoilov A.A. Study of partitions with additional conditions. URL: https://github.com/SashaSamoilov/qop-partitions (accessed: 25.09.2023).

УДК 519.766, 519.1, 519.713.1, 519.713.2 DOI: 10.14529/cmse240103

О НЕСУЩЕСТВОВАНИИ ПРОСТОГО ВАРИАНТА ПОЛИНОМИАЛЬНОГО АЛГОРИТМА ИЗВЛЕЧЕНИЯ КОРНЯ ИЗ ЯЗЫКА

© 2024 Б.Ф. Мельников

Совместный университет МГУ – ППИ в Шэньчжэне
(Китай, 518172, провинция Гуандун, Шэньчжэнь, район Лунган,
Даюньсиньчэн, ул. Гоцзидасюеюань, д. 1)
E-mail: bormel@mail.ru

Поступила в редакцию: 07.10.2023

Для стандартной операции конкатенации слов, рассматриваемой как умножение, естественным образом определяется конкатенация языков, а на основе последней операции – степень языка и, при наличии, корень заданной степени. При описании алгоритмов построения языка, являющегося корнем степени M из заданного языка, большое значение имеют так называемые потенциальные корни: это такие слова (не языки), рассматриваемая M-я степень которых входит в заданный язык. Несложно показать, что все потенциальные корни для заданного языка строятся с помощью полиномиального алгоритма. Эта задача, по-видимому, не упрощается при рассмотрении слов и языков над 1-буквенным алфавитом — что и делается в настоящей статье. Табуированная пара потенциальных корней — это такая пара, конкатенация слов которой в язык не входит. В предыдущих публикациях на тему описания алгоритмов извлечения корней из языка возникала гипотеза, что полиномиальный алгоритм извлечения корня из языка может быть описан на основе рассмотрения только множества табуированных пар — путем перебора специально описываемых подмножеств множества потенциальных корней. В настоящей статье показывается, что подобный алгоритм (называемый «простым») невозможен, т. е. если и существует полиномиальный алгоритм извлечения корня из языка, то он (алгоритм) должен использовать некоторую дополнительную информацию.

Ключевые слова: формальные языки, конкатенация языков, степень языка, корень из языка, полиномиальные алгоритмы.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Мельников Б.Ф. О несуществовании простого варианта полиномиального алгоритма извлечения корня из языка // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2024. Т. 13, № 1. С. 38–56. DOI: 10.14529/cmse240103.

Введение

Настоящую статью можно рассматривать как продолжение [1]. В ней (а также в более ранней статье [2]) рассматривалась задача извлечения корня из заданного языка.

При этом корень — как и в других областях алгебры — является обратной операцией к операции возведения в степень и, также (так же) как и в других областях алгебры, определяется, вообще говоря, неоднозначно. При этом, конечно, последняя операция (возведение в степень) стандарнтым образом определяется через операцию умножения, которой в теории формальных языков обычно называют конкатенацию. Какой-либо пример неоднозначности извлечения корня легко конструируется даже для случая 1-буквенного алфавита: квадрат языка

$$B = \{\varepsilon, a, a^2, a^3, a^4\}$$

равен

$$A = B^2 = \{\varepsilon, a, a^2, \ldots, a^7, a^8\},$$

и из языка A извлекается не только корень B, но и

$$B' = \{\varepsilon, a, a^3, a^4\}.$$

Повторим информацию, уже приведенную в [1], более подробно. Несмотря на простоту формулировки задачи, автор не смог найти в литературе (а также в Интернете) какого-либо ее описания, в том числе даже ее постановки. Но, несмотря на это, возникает впечатление, что весь материал [1] и настоящей статьи является очень легким, и вызывает недоумение тот факт, что в известных автору монографиях нет хотя бы формулировок подобных задач.

Вообще, всюду в настоящей статье будет рассматриваться случай 1-буквенного алфавита и 2-й степени для корня; при этом на основе материала статьи [1] создается впечатление, что получаемые при таком упрощении результаты на более сложные ситуации *обобщаются всегда*; но, конечно, строго доказать сформулированное в этом абзаце предположение вряд ли возможно.

Задачу извлечения корня можно рассматривать в нескольких вариантах (см. подробности далее) — но в статье будет рассматриваться задача построения *за полиномиальное время хотя бы одного* подходящего корня. При этом, конечно, всюду в настоящей статье полиномиальность понимается относительно размера входных данных — что в нашей ситуации можно считать суммарной длиной слов заданного языка (в примере выше — A).

Для формулировки результатов очень важно понятие «потенциальный корень» — и, в отличие от корня (являющегося языком), последний объект является словом: потенциальные корни нужной степени для заданного языка A — это те слова, у которых такая степень принадлежит языку A. Множество потенциальных корней строится за полиномиальное время; отметим, что это не очень сложное утверждение — однако тривиальным его назвать нельзя: алгоритмы, решающие эту задачу «в лоб», полиномиальными не являются, а полинмиальность *похожих* алгоритмов была показана, например, в [3].

А *экспоненциальные алгоритмы извлечения корня из языка описываются тривиальным образом*: надо просто перебрать все подмножества множества потенциальных корней. К последнему стоит добавить, что при наличии проверяемого множества потенциальных корней — в литературе алгоритм его выбора часто называется оракулом, [4–8] и др. — саму проверку того, является ли рассматриваемое подмножество корнем нужной степени, можно выполнить за полиномиальное время. Отметим еще, что в отличие от предыдущей проверки, полиномиальность этой проверки очевидна. Однако при этом надо осознавать, что степень полинома не может не зависеть от требуемой в задаче степени для извлечения корня.

Потенциальные корни используются и в другом классе задач — по-видимому, более сложном, чем задачи извлечения корня из языка. Это — задачи, которые условно можно назвать «раскодированием», или «построением инверсного морфизма», см. [3] и ссылки из этой работы. При этом, несмотря на бóльшую сложность задач «раскодирования», описать полиномиальный алгоритм построения инверсного морфизма — при некотором специально сделанном предположении теории формальных языков, — в отличие от рассматриваемой здесь задачи извлечения корня из языка, уже удалось.

Итак, полный перебор всех подмножеств множества потенциальных корней (“brute-force algorithm”) считается невозможным. Но основным результатом статьи [1] была теорема о том, что если известен некоторый корень, то при выполнении специальных несложных условий к этому корню (как к подмножеству) можно добавить еще один потенциальный

корень (как элемент); заранее отметим, что далее в настоящей статье будет приведено более подробное доказательство этой теоремы. Поэтому может возникнуть предположение, что можно осуществить следующую схему алгоритма:

- некоторым образом описать все такие подмножества множества потенциальных корней, к которым заведомо нельзя добавить ни одного нового элемента (еще одного потенциального корня);
- проверить все такие подмножества: не является ли какое-либо из них корнем, —

и все это должно дать решение основной задачи (т. е. построение полиномиального алгоритма извлечения корня). Однако основной предмет настоящей статьи заключается в том, что

> *подобные простые действия не могут привести к желаемому описанию полиномиального алгоритма*,

поскольку существуют примеры (и они ниже описаны), в которых число подмножеств, необходимых для подобной проверки, от размерности задачи зависит экспоненциально.

Но важно отметить, что при этом вовсе

> *не утверждается несуществование полиномиального алгоритма*

извлечения корня: пока говорится лишь о том, что не может сработать подобный простой способ, то есть если и существует полиномиальный алгоритм решения рассматриваемой задачи, то в нем должна использоваться более существенная информация, чем та, которая формулирует возможность добавления нового элемента (потенциального корня) в некоторое подходящее под ответ множество потенциальных корней.

Отметим еще, что предмет настоящей статьи может быть охарактеризован и по-другому: в ней формулируются те утверждения, которые можно получить рассмотрением только таких пар потенциальных корней, оба элемента которых *не могут одновременно входить* в решение задачи.

Приведем содержание статьи по разделам. *Раздел 1* — предварительные сведения и обозначния, частично стандартные, а частично введенные ранее в предыдущих публикациях. Специальные обозначения, связанные с гиперкубами, вынесены в отдельный *подраздел 1.2*; там же приведены и соответствующие примеры.

В разделе 2 приводится дополненное доказательство теоремы о возможном увеличении одной координаты имеющегося корня: в опубликованном в [1] варианте доказательства этой теоремы не были рассмотрены все возможные ситуации.

Продолжение рассмотрения гиперкуба — применительно к N-мерному гиперкубу потенциальных корней — приведено *в разделе 3*. В основном, в этом разделе рассматриваются множество простых путей в этом гиперкубе — между «максимальной» и «минимальной» его вершинами. Уточняется формулировка гипотезы о множестве т. н. важных потенциальных корней.

В разделе 4 показывается возможность применение оптимизационных задач на графах — для решения некоторых подзадач, связанных с задачей извлечения корня из языка; в частности, рассматривается применение задачи 2-SAT.

Основные результаты статьи приведены *в разделе 5*: показана возможность конструирование примеров на тему несуществования простого алгоритма извлечения корня степени 2 — причем примеров для сколь угодно большой (но заранее известной) размерности; как

и всюду в настоящей статье, простыми называются те алгоритмы, которые основаны лишь на информации о табуированных парах потенциальных корней.

В разделе 6 очень кратко рассмотрен частный случай для степени 3; при этом есть основания полагать, что на основе приведенного примера можно построить примеры для степени 3 сколь угодно большой (но, конечно, заранее известной) размерности — примеры, показывающие несуществование простого алгоритма извлечения корня 3-й степени.

В заключении повторены основные результаты статьи и приведены возможные направления дальнейших исследований по рассматриваемой тематике — описанию алгоритмов извлечения корня из заданного языка.

1. Теоретическиие основы

1.1. Общие обозначения

Приведем основные обозначения — частично стандартные, частично введенные ранее в [1], а частично новые.

Смысл слова «умножение» для всех рассматриваемых объектов — обычный для теории формальных языков: это просто результат произведения (приписывания, конкатенации) двух слов (двух языков). Для иллюстрации рассмотрим такой пример для 1-буквенного алфавита — основного объекта настоящей статьи: если

$$C = \left\{ a^i \,\middle|\, i \in I \right\} \quad \text{и} \quad D = \left\{ a^j \,\middle|\, j \in J \right\}$$

(I и J — некоторые конечные подмножества целых неотрицательных чисел), то

$$C \cdot D = \left\{ a^k \,\middle|\, (\exists\, i \in I,\, j \in J)\,(k = i + j) \right\}.$$

Слово «умножение» будем в дальнейшем применять без кавычек: в нашем случае оно обладает не только обычными полугрупповыми свойствами (которые практически всегда необходимы при рассмотрении моделей теории формальных языков), но и, дополнительно к ним, коммутативностью.

Среди конечных подмножеств множества целых неотрицательных чисел, рассматривавшихся в предыдущем абзаце, особую роль будут играть множества вида

$$\{K, K+1, \ldots, L-1, L\} \quad \text{для некоторых} \quad K, L \in \mathbb{N}, \quad L \geqslant K. \tag{1}$$

Множество вида (1) будем обозначать записью $\overline{K, L}$.

На основе умножения (произведения) естественным образом определяется степень языка. Далее нам необходимо ввести описание операции извлечения корня — однако сначала зафиксируем несколько соглашений о терминах и о применяемых константах, которых будем всегда придерживаться при описании задач и рассматриваемых к ним примеров.

- Первое соглашение. Всюду далее, если не сказано иного, решается задача поиска языка X, являющегося корнем уравнения

$$X^2 = A, \tag{2}$$

где язык A заранее задан. Отметим, что в [1] рассматривалось уравнение $X^M = A$ для различных $M \geqslant 2$, а в настоящей статье в разделе 6 кратко рассматривается случай $M = 3$.

- При этом *потенциальный корень* — это некоторое слово $u \in \Sigma^*$, такое что $u^2 \in A$. В отличие от них (т. е. слов) — множество (язык) X будем называть корнем (без прилагательного «потенциальный»), если выполнено условие (2). Как уже отмечалось, найти все потенциальные корни можно за полиномиальное время с помощью несложных алгоритмов.
- Второе соглашение: N — количество потенциальных корней, будем их нумеровать от 1 от N; можно сказать, что здесь используется «обычная индексация Паскаля».
- Третье соглашение. Если, как в приведенном во введении примере, «задача формируется» возведением во 2-ю степень некоторого языка (пусть это язык B; считаем что при решении получаем язык X), то представление этого языка (как множества, B или X) выполняем перечислением разрядов, от 0 до некоторого заданного n; здесь, наоборот, используется «индексация Си-подобных алгоритмических языков».
- Таким образом, размерность каждого из языков B и X равна $n+1$ — и можно рассматривать только такие ситуации, когда и старший, и младший разряды равны 1. При этом $2{\cdot}n+1$ будет размерностью *исходного* языка A (т. е. для его задания нужно некоторым способом определить разряды от 0-го до $2n$-го, причем также можно считать, что старший и младший разряды равны 1).

Для того, что выше названо «третьим соглашением», во всех случаях (т. е. как для A, так и для $B\,/\,X$) будут применяться 4 варианта обозначения множества слов:

- во-первых, «самый обычный»: например,

$$\left\{\, \varepsilon{=}a^0, a{=}a^1, aaa{=}a^3, aaaaaa{=}a^6 \,\right\}. \tag{3}$$

А остальные три варианта записи (см. далее, они описаны в виде примеров) будут применяться как для подмножеств множества потенциальных корней, так и для нескольких элементов множества

$$\left\{\, \varepsilon, a, aa, \ldots a^{K-1}, a^K \,\right\}$$

(при этом обычно будет либо $K = n$, либо $K = 2n$). Выше уже было сказано, что множество потенциальных корней считается зафиксированным, причем их число равно N. Будем также считать, что зафиксирована и их нумерация. Итак, приведем примеры других вариантов записи — для конкретного языка (3):

- [0 1 3 6] — множество используемых степеней записываем в квадратных скобках в порядке возрастания;
- 1001011 — двоичное число, в котором разряды предыдущего списка отмечены 1; разряды двоичного числа нумеруем начиная с 0 справа налево, каждые 10 разрядов (при наличии) отделяем небольшим пробелом;
- 75 — число, принадлежащее $\mathbb{N}_0$, представляющее собой десятичную запись предыдущего пункта.

Как уже отмечалось, те же самые обозначения будут использоваться и для $K = N$, и при этом рассматриваются подмножества множества потенциальных корней; однако в этом случае разряды двоичного числа удобнее записывать слева направо и нумеровать начиная с 1. Например, пусть всего 5 элементов, а подмножество — $\{2, 5\}$:

- [2 5], иногда просто 2 5, даже иногда 25 — неоднозначности это не вызовет; пустое множество обозначается обычно, т. е. $\varnothing$;
- 01001; пустым множеством здесь является последовательность нулей;
- 9; пустое множество — 0.

Отметим, что что недоразумений (неоднозначности прочтения) в статье возникнуть не должно.

1.2. Обозначения и примеры, связанные с гиперкубами

В нескольких приведенных выше варантах записи уже были использованы подмножества; при этом подмножества множества потенциальных корней (напомним, что ранее сделано предположение о том, что число их равно N) будем рассматривать как вершины N-мерного гиперкуба. Определим несколько связанных с подобным гиперкубом вспомогательных понятий.

- Определение максимальной вершины гиперкуба естественное — это $(1, 1, \ldots, 1, 1)$.
- Для дальнейшего — в случае рассмотрения задачи извлечения корня 2-й степени — часто будут использованы *пары* потенциальных корней; в частности, т. н. табуированные пары (о них подробности далее). Для примера — если:
  - принимается «третье соглашение» (напомним, что для него возможен только случай 1-буквенного алфавита)
  - и при этом рассматривается множество [0 1 2 3 6] (оно действительно будет подробно рассмотрено в примерах в дальнейшей части статьи),
  - а пара — $\{2, 6\}$ (конечно, порядок элементов пары несущественен),

  то возможны еще и такие обозначения этой пары:
  - $\overbrace{2, 6}$ (это просто их по значениям); это то же самое, что и $\overbrace{6, 2}$;
  - $\overbrace{\#3, \#5}$ (это их по номерам, считаем номера начиная с 1).
- Для некоторой табуированной пары табуированная гиперплоскость — это такая N−2-мерная гиперплоскость N-мерного гиперкуба, для которой обе координаты элементов пары равны 1.
- Обобщением табуированной пары и табуированной гиперплоскости для степеней извлекаемого корня 3 более (продолжаем работать с 1-буквенным алфавитом) являются гиперплоскости соответствующих размерностей — но мы, однако, в настоящей статье подробно эти вопросы рассматривать не будем, это нужно только для примера, кратко рассматривающегося в разделе 6. Однако уже здесь стоит отметить, что в случае больши́х размерностей (пусть k) нужно рассматривать гиперплоскости не только размерности N−k, но и бо́льших размерностей: среди элементов подмножества множества потенциальных корней, необходимого для формирования одного элемента исходного языка, могут быть совпадающие потенциальные корни.
- Примеры обозначний для степени 3 —

$$\overbrace{2, 5, 6} \quad \text{и} \quad \overbrace{2, 6}.$$

В первом случае элементы, которых *не должно быть* в исходном языке — это

[6 9 10 12 13 14 16 17 18],

(поскольку каждый из элементов исходной табуированной тройки может входить в формируемую сумму более одного раза); а во втором случае —

[6 10 14 18].

- Отметим, что максимальная вершина этого гиперкуба входит в пересечение любого (ненулевого) количества любых табуированных гиперплоскостей — независимо от их размерностей (т. е. от размерности рассматриваемой задачи, иными словами — от степени извлекаемого корня).
- На рисунках будем изображать гиперкуб «в обычном виде» в тех случаях, когда он приводится полностью; иначе (в частности, при больших размерностях) будем необходимую его часть изображать как подграф графа булеана.
- Для некоторой точки гиперкуба $(b_1 b_2 \dots b_N)$ обозначение $(b_1 b_2 \dots b_N)_{+k}$ означает вершину, полученной из предыдущей путем замены k-й координаты на 1.

Перейдем к уточнению непосредственной постановки решаемой задачи. Как отмечалось в [1] (и ранее в [9]), рассматривая для некоторого заданного конечного языка задачу извлечения корня заданной степени — либо максимально возможной степени — фактически имеется в виду не с одна задача, а целая *группа задач*, поскольку имеется несколько вариантов для требуемого ответа:

- найти *любой* (корень из языка);
- найти *все*
- найти *минимальный* (по некоторой метрике),
- и т. п.

Однако ниже всегда будет иметься в виду только задача построения *любого* корня.

Повторим рисунок из предыдущей статьи (ранее он был опубликован как [1, рис. 9]) — с некоторыми новыми комментариями.

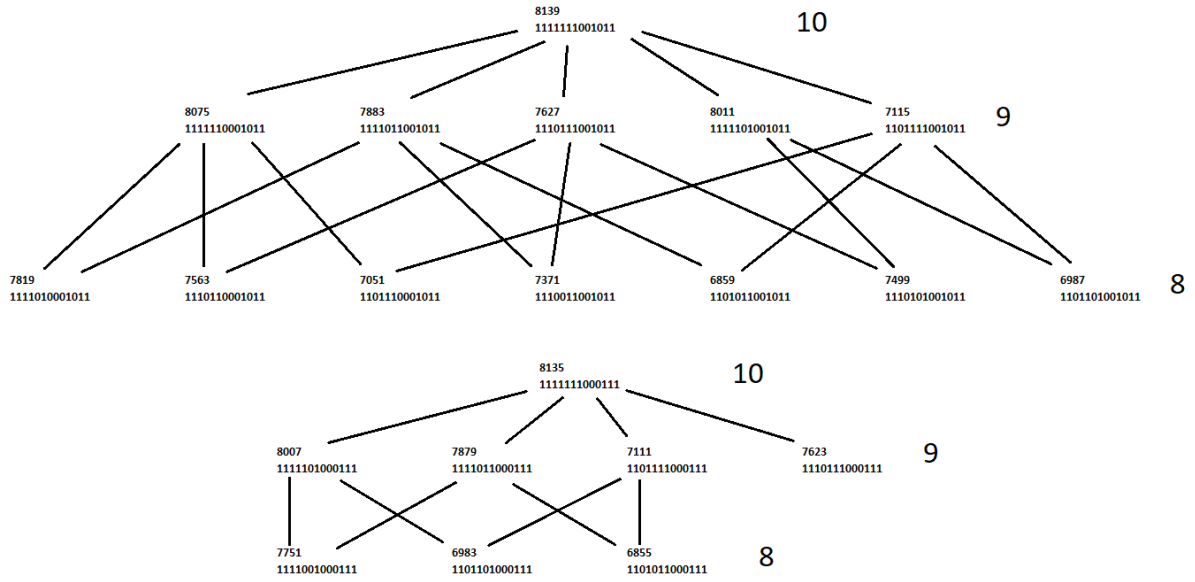


Рис. 1. Подграф графа булеана размерности 13

Как было сказано в [1], на рисунке показан подграф графа булеана размерности 13 со всеми квадратными корнями квадрата слова

$$1101011000111 \quad [0\ 1\ 2\ 6\ 7\ 9\ 11\ 12] \quad (6855).$$

Справа от элементов показаны уровни булеана — количества потенциальных корней, входящих (в качестве элементов) в рассматриваемый корень из языка. Сразу отметим, что всего

существует $C_{13}^{8} = 1287$ вариантов подмножеств из 8 потенциальных корней, но лишь 10 из них являются квадратными корнями из заданного языка.

Также повторим в новых обозначениях пример с несколькими возможными корнями: у языка [0 1 2 3 4 5 6 7 8] имеется не только очевидный корень [0 1 2 3 4], но и корень [0 1 3 4]. Понятно, что конструировать подобные примеры совсем несложно — гораздо сложнее обратная операция, о ней в настоящей статье и идет речь.

В заключение раздела отметим, что существуют примеры, когда (квадратные) корни есть — но ближайший к максимальной вершине корень находится от нее на сколь угодно большом (но заданном заранее) расстоянии. В [1] такой пример был приведен для расстояния 4 — но при этом был приведен с опечаткой, одна из форм записи рассматриваемого языка (квадрат которого и поступает на вход задачи) — это

$$[0\ 1\ 4\ 6\ 8]$$

(в [1] была опечатка: ошибочно было указано [0 2 4 7 8], но при этом остальные возможные варианты записи рассматриваемого языка были приведены верно).

Более важно, что на основе такого примера для любого заданного $L \in \mathbb{N}$ можно построить пример языка, когда ближайший к максимальной вершине корень находится от нее на расстоянии L — в то время как общая размерность исходной задачи не превышает $2 \cdot L$. Эти примеры очень простые, поэтому не будем приводить их общего описания (т. е. описания для произвольного заданного L), а рассмотрим только минимальный пример для следующего значения, $L = 5$ (как и ранее в [1], минимальность понимается для исходного значения, взятого для возведения в квадрат — причем именно этот квадрат подается на вход рассматриваемой задачи). Итак,

- язык, взятый для формирования примера, — 1363; в других вариантах записи его же получается

$$[0\ 1\ 4\ 6\ 8\ 10] \quad \text{или} \quad 10101010011;$$

тогда:

- его квадрат (считаем именно его исходным языком для этой задачи) —

$$10101010111\,1111110111 \qquad (\text{т. е. } 1400823);$$

- потенциальные корни — [0 1 2 3 4 5 6 7 8 9 10] (т. е. все возможные значения в заранее известном промежутке, здесь — от 0 от 10), в другой записи — 2047.

Перебором на компьютере можно убедиться, что на расстояниях от максимальной вершины от 0 до 4 корней нет; ближайший корень — [0 1 4 6 8 10], находящийся на расстоянии 5.

Вообще, приведенный ваше способ дает возможность конструировать такие примеры, в которых ближайший корень находится на любом заданном заранее расстоянии от максимальной вершины гиперкуба. При этом если расстояние должно быть равно L, то размерность примера (т. е. значение максимального потенциального корня) равна, как несложно убедиться, $2 \cdot L$.

2. О возможном увеличении координаты имеющегося корня

В этом разделе приводится дополненное доказательство теоремы о возможном увеличении одной координаты имеющегося корня. Вариант теоремы был опубликован в [1] — однако приведенное там доказательство нельзя назвать полным. Конкретно: с формальной

точки зрения возможна такая ситуация, когда выбирается одно значение i_1 (i в обозначениях приведенного в [1] доказательства) — в то время как условие $\overbrace{k, i_2} \in \mathcal{T}(A)$ выполняется не для i_1, а для некоторого $i_2 \neq i_1$. Понятно, что конкретных примеров в принципе привести невозможно — поскольку доказательство теоремы ведется от противного, т. е. показывается именно невозможность подобных ситуаций.

Теорема 1. Пусть

$$B = (b_1 b_2 \dots b_N) \in 2^N \quad -$$

некоторый корень уравнения (2), т. е. $M = 2$. Пусть при этом

$$b_k = 0 \quad \text{для некоторого} \quad k \in \overline{1, N}.$$

Пусть также для каждого $i \in \overline{1, N} \setminus \{k\}$, такого что $b_i = 1$, выполнено условие

$$\overbrace{k, i} \notin \mathcal{T}(A).$$

Тогда

$$B_{+k} = (b_1 b_2 \dots b_N)_{+k}$$

также является корнем уравнения (2).

Доказательство. Очевидно, что $B^2 \subseteq (B_{+k})^2$. Поэтому из выполнения неравенства $(B_{+k})^2 \neq A$ следует обязательное выполнение условия $(B_{+k})^2 \supset A$ и существование по крайней мере одного разряда $i \in \overline{1, N} \setminus \{k\}$, для которого выполнены следующие условия:

- $b_i = 1$,
- и при этом при возведении в квадрат у произведения появлялся бы *новый* разряд, равный 1 («новый» — отсутствовавший в B, т. е. принимающий в B значение, равное 0).

Отметим, что единственность такого значения i не утверждается, поэтому множество, состоящее из всех таких i, будет обозначаться как I.

А поскольку к B был добавлен только один разряд, а именно k-й, то последний факт возможен только при условии

$$\left(\exists\, i \in I\right) \left(\overbrace{k, i} \in \mathcal{T}(A)\right),$$

что противоречит условию теоремы. □

3. О путях в N-мерном гиперкубе потенциальных корней

В этом разделе рассматриваются вопросы, связанные с возможными путями в N-мерном гиперкубе потенциальных корней. Как и ранее, будем рассматривать 1-буквенный алфавит — хотя, как и многие вопросы настоящей статьи, материал этого раздела может быть обобщен на случай произвольного конечного алфавита. Отметим, что материал этого раздела связан с возможным описанием *эвристических* алгоритмов задачи извлечения корня из языка.

Итак, у нас есть N-мерный гиперкуб, каждая его точка (а их всего 2^N) — это некоторое подмножество множества потенциальных корней. Придадим *самим точкам* следующие значения:

- значение 1 — это корень (в примере на рис. 1 их всего 21);
- значение 2 устанавливаем в том случае, когда квадрат этого языка включает в себя требуемый (исходный) язык как *собственное подмножество*; отметим, что таковой (имеющей значение 2) должна являться *максимальная* точка гиперкуба, т. е. $[1^N] = (1, 1, \ldots, 1)$ — в противном случае рассматриваемая задача неинтересна:
  - если это значение равно 1, то искомый корень известен;
  - если это значение равно 0 (см. ниже), то и все значения вершин гиперкуба равны 0 и корней нет;
  - значения, равного ∅ (также см. ниже это обозначние), у максимальной точки гиперкуба быть не может — это несложно доказывается а основе введенных определений.
- значение 0 устанавливаем в том случае, когда квадрат этого языка является *собственным подмножеством* требуемого (исходного) языка аналогично предыдущему пункту, таковой (имеющей значение 0) должна являться *минимальная* точка гиперкуба, т. е. $[0^N] = (0, 0, \ldots, 0)$;
- значение ∅ — оставшийся вариант: ни одно из рассматриваемых двух множеств (квадрат языка, соответствующего точке гиперкуба, а также исходный язык) не является подмножеством другого; отметим, что вариант совпадения этих множеств уже был рассмотрен: в этом случае значение равно 1, то есть это один из возможных корней.

Будем рассматривать все простые пути по ребрам гиперкуба из точки $[1^N]$ в точку $[0^N]$; отметим по этому поводу, что всего имеется $N!$ таких путей — поэтому вряд ли для какой-либо задачи представляют интерес варианты алгоритма, связанного с их полным перебором (brute force method).

У любого такого пути:

- сначала несколько значений, установленных для вершин, равны 2;
- затем, *возможно*, равны либо 1 либо ∅ (одновременное включение в путь как 1, так и ∅, невозможно — этот факт можно несложно доказать аналогично сказанному выше;
- в конце пути несколько значений равны 0.

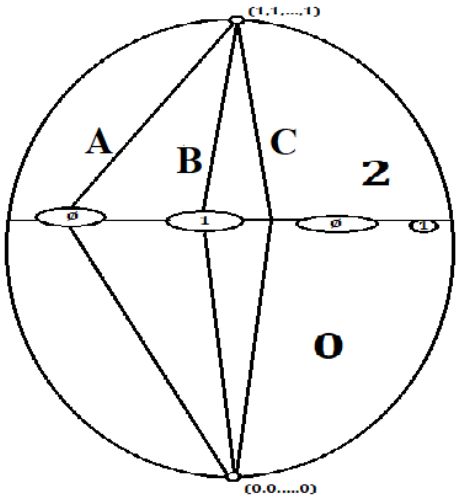


Рис. 2. Схема различных путей в графе булеана

На приведенном рис. 2:

- при наличии вершин со значениями ∅ путь помечен A;

- при наличии вершин со значениями 1 путь помечен B;
- при отсутствии вершин как со значениями ∅, так и со значениями 1, путь помечен C.

Далее будем рассматривать только пути вида C (если таковые имеются). Переход из значения 2 в значение 0, как и любое другое единичное движение, осуществляется удалением из рассматриваемого множества какого-либо из имеющихся в нем потенциальных корней; такой потенциальный корень (для которого существует хоть один подобный путь по вершинам гиперкуба) назовем *важным*. Заметим, что задача, заключающаяся в поиске всех важных потенциальных корней, конечно, может быть решена простым переборным алгоритмом (brute force method) — но в настоящей статье формулируется задача, заключающаяся в описании быстрых *эвристических* алгоритмов для нее.

В [1] была кратко сформулирована гипотеза, заключающаяся в том, что множество всех важных потенциальных корней (рассматриваемое как язык) — это один из корней из исходного языка. Пример, опровергающий эту гипотезу, может быть получен на основе модификации примера, приведенного на рис. 1. Действительно, как уже было отмечено, всего существует 1287 вариантов подмножеств из 8 потенциальных корней, но лишь 10 из них являются квадратными корнями из заданного языка. И можно считать, что горизонтальная линия на рис. 2, прерываемая несколькими овалами — подмножествами со значениями 1 и ∅, это и есть множество вершин гиперкуба, каждая из которых имеет ровно 8 таких координат, каждая из которых равна 1.

4. О применении оптимизационных задач на графах

Рассматривавшийся выше в разделе 1.2 пример с 13 потенциальными корнями (в этом примере — все возможные значения в заранее известном промежутке от 0 от 12), по-видимому, малоинтересен:

- в нем имеется только одно невозможное значение возможной суммы потенциальных корней (а именно, значение 5);
- при этом, в отличие от материала следующего раздела, непонятно, как обобщить этот пример на произвольную размерность; отметим, что при возможном таком обобщении нам интерсны:
  - во-первых, существование решения (в нашем примере оно есть),
  - и, во-вторых, линейная зависимость числа табуированных пар от размерности задачи.

В связи с этим рассмотрим сначала тривиальные определения, потом — построения, связанные с применением задачи 2-SAT, и уже после этого более сложный пример.

Итак, для очевидным образом определяемого *графа табуированных пар*, в котором:

- множество вершин V является множеством потенциальных корней,
- а множество ребер E соответствует всем табуированным парам.

Несложно убедиться, что такой граф является дополнением к графу, рассматривавшемуся в [1, разд. III], один из примеров приведен на [1, рис. 3].

Рассмотрим тривиальный пример графа табуированных пар — соответствующий языку, приведенному в разделе 1.2, см. рис. 3. Однако важно отметить, что тривиальным пример является с точки зрения материала настоящего раздела, а также с точки зрения теории графов вообще, — но, как уже понятно, не с точки зрения материала раздела 1.2.

Далее в N-мерном гиперкубе рассматриваются «нетабуированные» точки булеана — т. е. такие точки, в координаты которых не входит ни одна табуированная пара; стоит

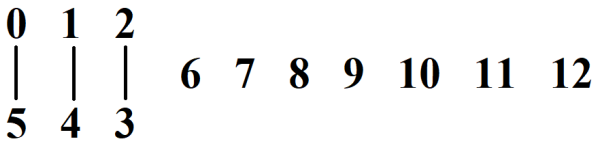


Рис. 3. Очень простой пример графа табуированных пар

отметить, что для каждой табуированной пары размерность получаемой «табуированной гиперплоскости» равна $N-2$. На основе результатов теоремы 1 можно утверждать, что для поиска решения исходной задачи можно искать «нетабуированные» точки булеана, лежащие ближе всего к максимальной вершине. Решение — если хотя бы одно имеется — обязательно будет среди тех «нетабуированных» точек, для которых не существует соседней «нетабуированной», лежащей при этом ближе к максимальной вершине.

Однако простой подсчет даже при закреплении каких-то конкретных значений элементов табуированной пары дает общее число элементов табуированной плоскости 2^{N-2}; интуитивно понятно — даже без проведения точных подсчетов возможного числа соседних элементов — что подобное знчение вряд ли может дать полиномиальный алгоритм. И, по-видимому, такой подсчет вообще не может являться доказательством несуществованя простого (полиномиального) алгоритма, заключающегося именно в переборе точек, лежащих рядом (т. е. на расстоянии 1) с точками, соответствующими табуированным плоскостям. А следует такое несуществование из примеров, приведенных в разделе 5.

Дальнейшие построения связаны с формированием на основе графа табуированных пар задач дискретной оптимизации: задачи построения независимых множеств вершин графа (в том числе максимальных множеств), а также, в упрощенном случае, задачи 2-SAT — см. [7] и мн. др. При формировании соответствующих задач (2-SAT в первую очередь) отметим, что все входящие в получающуюся 2-КНФ переменные *не* содержат отрицаний. Поэтому возможный переборный алгоритм, сформированный на основе такой 2-SAT, представляет собой:

- во-первых, поиск всех максимальных независимых подмножеств множества вершин V;
- и, во-вторых, рассмотрение всех точек N-мерного гиперкуба, соответствующих уменьшению какой-либо координаты какой-либо вершины одного из этих подмножеств.

Отметим, немного забегая вперед, что примеры следующего раздела показывают, что количество соответствующих вершин (необходимых для рассмотрения) может быть экспоненциально — относительно размера исходной задачи.

Рассмотрим более интересный пример. В качестве *исходного* языка выберем

$$[0\ 1\ 3\ 6\ 10];$$

квадрат этого языка (*входной* язык рассматриваемой задачи) —

$$[0\ 1\ 2\ 3\ 4\ 6\ 7\ 9\ 10\ 11\ 12\ 13\ 16\ 20],$$

множество потенциальных корней —

$$[0\ 1\ 2\ 3\ 5\ 6\ 8\ 10].$$

Несложно получаемое множество табуированных пар сразу изобразим на графе, см. рис. 4.

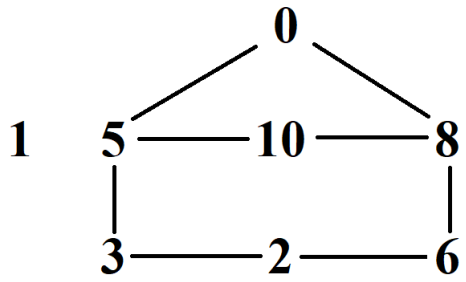


Рис. 4. Более сложный пример графа табуированных пар

Из предыдущего материала следует, что существуют две «максимальные» вершины N-мерного куба (т. е. такие вершины, увеличение любой равной 0 координаты которых приведет к появлеию хотя бы одной табуированной пары); они являются решениям как задачи построения максимального независимого множества вершин графа, так и задачи 2-SAT. Записи этих вершин приведем подробные (выше в настоящей статье такой способ записи не использовался), обозначая координату, соответствующую потенциальному корню q, записью x_q.

Итак, координаты двух вершин следующие:

$$x_0 = 1,\ x_1 = 1,\ x_2 = 0,\ x_3 = 1,\ x_5 = 0,\ x_6 = 1,\ x_8 = 0,\ x_{10} = 1$$
$$\text{и} \quad x_0 = 0,\ x_1 = 1,\ x_2 = 1,\ x_3 = 0,\ x_5 = 1,\ x_6 = 0,\ x_8 = 1,\ x_{10} = 0.$$

Таким образом, в рассматриваемом примере для поиска «максимальных» корней достаточно проверить только 2 точки 8-мерного гиперкуба; первая из них в точности соответствует исходному языку, а вторая при проверке дает ошибку: квадрат построенного языка не равен языку исходному.

Важно, что в рассмотренном примере специально не отмечена разница между двумя оптимизационными задачами на графах, т. е. между 2-SAT и существенно более сложной задачей построения независимых множеств вершин графа. Причина такова. Основная цель получения *необходимых* условий для задачи извлечения корня заключается в том, чтобы *на примерах* показать невозможность краткого решения основной задачи, т. е. невозможность построения полиномиального алгоритма, использующего лишь информацию о табуированных парах потенциальных корней. А показать это можно и на очень простых графах — которые, однако, соответствуют экспоненциально большому числу вершин гиперкуба, которые необходимы для перебора; см. следующий раздел.

А в завершение этого раздела приведем очевидную *формальную* запись задачи 2-SAT — которая выше фактически была решена; для входящих в задачу переменных будем пользоваться теми же обозначениями.

$$(x_0 \,|\, x_5)\ \&\ (x_5 \,|\, x_3)\ \&\ (x_3 \,|\, x_2)\ \&\ (x_2 \,|\, x_6)\ \&\ (x_6 \,|\, x_8)\ \&\ (x_8 \,|\, x_0)\ \&\ (x_5 \,|\, x_{10})\ \&\ (x_{10} \,|\, x_8)\,.$$

Понятно, что реальные примеры решения задачи построения независимых множеств вершин графа существенно сложнее, чем приведенные в этом разделе. Однако нам нет необходимости рассматривать такие более сложные примеры: нам достаточно существование таких примеров, в которых при рассмотрении только решения 2-SAT требуется перебор вершин N-мерного гиперкуба, число которых от размерности задачи зависит экспоненциально; это сделано в следующем разделе.

5. Примеры про несуществование простого алгоритма: сколь угодно большая размерность для степени 2

В этом разделе приведены основные результаты статьи: показана возможность конструирование примеров на тему несуществования простого алгоритма извлечения корня степени 2 — причем примеров для сколь угодно большой (но заранее известной) размерности; как и всюду в настоящей статье, простыми здесь считаются алгоритмы, основанные *только* на информации о табуированных парах потенциальных корней.

Для осуществления описания подобных примеров возникают следующие вопросы. Первый, более простой (но, видимо, неочевидный) — про существование описания задачи, для которой построенное число «максимальных» точек от размерности этой задачи зависит *экспоненциально*. Второй, более сложный, — когда дополнительно к этому сформулированная таким образом задача *должна иметь решение* (т. е. для заданной задачи корень нужной степени должен извлекаться). Описанный в этом разделе пример (на самом деле — целое семейство примеров) дает положительный ответ *сразу на оба* эти вопроса.

Итак, сначала рассмотрим следующий пример. Исходный язык такой:

$$[0\ 1\ 3\ 5\ 8\ 9\ 10],$$

его квадрат (заданный, входной язык) —

$$[0\ 1\ 2\ 3\ 4\ 5\ 6\ 8\ 9\ 10\ 11\ 12\ 13\ 14\ 15\ 16\ 17\ 18\ 19\ 20] \quad = \quad \overline{0,6} \cup \overline{8,20}$$

(т. е. все значения от 0 до 20, за исключением 7), язык потенциальных корней —

$$[0\ 1\ 2\ 3\ 4\ 5\ 6\ 9\ 8\ 9\ 10] \quad = \quad \overline{0,10}$$

(т. е. все возможные значения для рассматриваемой размерности входного языка, равной 20), в наших обозначениях $N = 11$. Формулировку задачи можно изобразить в виде таблицы, приведенной на рис. 5.

	0	1	2	3	4	5	6	7	8	9	10
0	1	1	1	1	1	1	1	0	1	1	1
1	1	1	1	1	1	1	0	1	1	1	1
2	1	1	1	1	1	0	1	1	1	1	1
3	1	1	1	1	0	1	1	1	1	1	1
4	1	1	1	0	1	1	1	1	1	1	1
5	1	1	0	1	1	1	1	1	1	1	1
6	1	0	1	1	1	1	1	1	1	1	1
7	0	1	1	1	1	1	1	1	1	1	1
8	1	1	1	1	1	1	1	1	1	1	1
9	1	1	1	1	1	1	1	1	1	1	1
10	1	1	1	1	1	1	1	1	1	1	1

Рис. 5. Возможная формулировка задачи для $M = 2$, $n = 10$

А решение задачи — фактически приведенное выше в качестве исходного языка — можно изобразить в виде таблицы, приведенной на рис. 6; в ней (в ее клетках) имеются все числа от 0 до 20, кроме 7.

	0	1	3	5	8	9	10
0	0	1	3	5	8	9	10
1	1	2	4	6	9	10	11
3	3	4	6	8	11	12	13
5	5	6	8	10	13	14	15
8	8	9	11	13	16	17	18
9	9	10	12	14	17	18	19
10	10	11	13	15	18	19	20

Рис. 6. Решение приведенной задачи для $M = 2$, $n = 10$

В рассмотренном примере граф табуированных пар содержит ребра

$$\{\overbrace{0,7},\ \overbrace{1,6},\ \overbrace{2,5},\ \overbrace{3,4}\},$$

которые удобно обозначать так же, как и сами пары. Итак, если размерность входной задачи считать равной 20 (что естественно), то число этих ребер равно $4 = \frac{20}{4} - 1$. Поэтому число «максимальных» точек N-мерного гиперкуба (как уже было отмечено, в рассматриваемом примере $N = 11$), необходимых для проверки наличия корня, равно $2^4 = 16$:

- для 3 координат 11-мерного гиперкуба, не входящих в табуированные пары, нужно установить значения 1 (иначе будет нарушено требование максимальности для формируемой точки гиперкуба),
- а для остальных 4 пар координат (отметим, что каждая координата входит в единственную табуированную пару) нужно рассмотреть по 2 возможности (пары значений (0,1) и (1,0)), которые можно устанавливать независимо от остальных значений пар.

Покажем, как для ранеее использованного обозначения n (в примере выше было $n = 10$) получать необходимые примеры *решаемых* задач при задании больших значений n. Для простоты будем считать n четным, причем $n \geqslant 10$; пусть $n = 2q$. Входными языками примеров будут

$$\overline{0, n{-}4} \ \cup\ \overline{n{-}2, 2n}.$$

Для некоторого конкретного $n = 2q$ граф табуированных пар содержит ребра

$$\{\overbrace{0, n{-}3},\ \overbrace{1, n{-}2},\ \ldots,\ \overbrace{q{-}3, q},\ \overbrace{q{-}2, q{-}1}\},$$

причем только их; всего таких ребер $q - 3$. Поэтому в получаемых примерах для любого выбранного четного $n \geqslant 10$ из-за независимости рассмотрения пар существует

$$2^{\frac{n}{2}-3} \tag{4}$$

вариантов значений точек $n{+}1$-мерного гиперкуба, необходимых для переборного поиска решения — в случае, когда такой поиск основан только на информации о табуированных парах.

6. Пример для степени 3

Совершенно аналогично [1, разд. VIII], здесь частный случай для степени 3 рассматривается очень кратко; однако есть основания полагать, что:

- во-первых, на основе приведенного далее примера можно построить примеры для степени 3 сколь угодно большой (но, конечно, заранее известной) размерности — примеры, показывающие несуществование простого алгоритма извлечения корня 3-й степени [1];
- во-вторых, примеры обобщаются и на произвольную степень $M > 3$.

Итак, пусть исходное множество степеней буквы таково:

$$[0\ 1\ 4\ 7\ 10\ 16\ 17\ 18]\,.$$

На его основе приведем разложения чисел от 0 до 54 в виде суммы трех чисел, входящих в исходное множество (если вариантов несколько, то приводим только один):

$0 = 0+0+0$	$1 = 0+0+1$	$2 = 0+1+1$	$3 = 1+1+1$	$4 = 0+0+4$
$5 = 0+1+4$	$6 = 1+1+4$	$7 = 0+0+7$	$8 = 0+1+7$	$9 = 1+1+7$
$10 = 0+0+10$	$11 = 0+1+10$	$12 = 1+1+10$	13 ...	$14 = 0+4+10$
$15 = 1+4+10$	$16 = 0+0+16$	$17 = 0+0+17$	$18 = 0+0+18$	$19 = 0+1+18$
$20 = 0+10+10$	$21 = 1+10+10$	$22 = 0+4+18$	$23 = 1+4+18$	$24 = 4+10+10$
$25 = 0+7+18$	$26 = 1+7+18$	$27 = 0+10+17$	$28 = 0+10+18$	$29 = 1+10+18$
$30 = 10+10+10$	$31 = 7+7+17$	$32 = 7+7+18$	$33 = 0+16+17$	$34 = 0+17+17$
$35 = 0+17+18$	$36 = 10+10+16$	$37 = 10+10+17$	$38 = 10+10+18$	$39 = 1+10+18$
$40 = 4+18+18$	$41 = 7+17+17$	$42 = 7+17+18$	$43 = 7+18+18$	$44 = 10+17+17$
$45 = 10+17+18$	$46 = 10+18+18$	47 ...	$48 = 16+16+16$	$49 = 16+16+17$
$50 = 16+16+18$	$51 = 17+17+17$	$52 = 17+17+18$	$53 = 17+18+18$	$54 = 18+18+18$

Таким образом в рассмотренном примере получаются 2 «пропущенных» значения для *входного* языка — что дает основания предполагать возможность описать на основе этого примера примеров сколь угодно большой размерности для этой же степени 3. Однако оценок, аналогичных (4), мы в настоящей статье делать не будем.

Заключение

Итак, в настоящей статье заканчивается рассмотрение тех вопросов, связанных с извлечением корня 2-й степени из языка над 1-буквеным алфавитом, которые можно получить на основе одних лишь табуированных пар потенциальных корней. В частности, мы показали, что на основе информации лишь о потенциальных корнях — и рассмотрении специальных вершин гиперкуба потенциальных корней — нельзя построить полиномиальный алгоритм извлечения корня из языка. Однако при этом не утверждается несуществование такого алгоритма: возможно, такие алгоритмы и существуют, но они долждны получаться на основе более полной информации об исходной задаче.

В качестве одного из направлений развития темы, рассмотренной в настоящей статье, укажем связь между рассмотренными выше вершинами гиперкуба потенциальных корней (0, 1, 2 и $\varnothing$) — и практически такими же обозначениями, применявшимися в нескольких относительно недавних статьях Г. Г. Рябова [2] для задач, связанных с ДНФ и проблемами их минимизации; упомянем статьи [10–12]. На вершинах гиперкуба потенциальных корней,

[1] Табуированные тройки — аналоги табуированных пар для степени 3 — также были кратко рассмотрены в заключении процитированной статьи.

[2] К сожалению, исследование этой тематики, по-видимому, никто не продолжает.

как и на т. н. кубантах в этих статьях, может быть построена специальная *алгебра*, которую можно использовать в дальнейших построениях и задачах — как чисто теоретических, так и вычислительных.

Работа частично поддержана грантом научной программы китайских университетов "Higher Education Stability Support Program" (раздел "Shenzhen 2022 — Science, Technology and Innovation Commission of Shenzhen Municipality").

Литература

1. Мельников Б.Ф., Мельникова А.А. О задачах извлечения корня из заданного конечного языка // International Journal of Open Information Technologies. 2023. Vol. 11, no. 5. P. 1–14.
2. Melnikov B.F., Korabelshchikova S.Yu., Dolgov V.N. On the task of extracting the root from the language // International Journal of Open Information Technologies. 2019. Vol. 7, no. 3. P. 1–6.
3. Мельников Б.Ф. Полиномиальный алгоритм построения оптимального инверсного морфизма // International Journal of Open Information Technologies. 2023. Vol. 11, no. 6. P. 1–10.
4. Stockmeyer L.J. The polynomial-time hierarchy // Theoretical Computer Science. 1976. Vol. 3, no. 1. P. 1–22.
5. Chandra A.K., Kozen D., Stockmeyer L.J. Alternation // Journal of the ACM. 1981. Vol. 28, no. 1. P. 114–133.
6. Immerman N. TDescriptive Complexity. Berlin: Springer, 1999. 284 p.
7. Hromkovič J. Theoretical Computer Science: Introduction to Automata, Computability, Complexity, Algorithmics, Randomization, Communication, and Cryptography. Berlin: Springer, 2003. 323 p.
8. Hromkovič J. Algorithmics for Hard Problems: Introduction to Combinatorial Optimization, Randomization, Approximation, and Heuristics. Berlin: Springer, 2004. 547 p.
9. Мельников Б.Ф. Полурешетки подмножеств потенциальных корней в задачах теории формальных языков. Часть I. Извлечение корня из языка // International Journal of Open Information Technologies. 2022. Vol. 10, no. 4. P. 1–9.
10. Рябов Г.Г. О четверичном кодировании кубических структур // Вычислительные методы и программирование. 2009. Т. 10, № 3. С. 340–347.
11. Рябов Г.Г. Хаусдорфова метрика на гранях N-мерного куба // Фундаментальная и прикладная математика. 2010. Т. 16, № 1. С. 151–155.
12. Рябов Г.Г. Полиморфизм символьных троичных матриц и генетическое пространство кратчайших K-путей в N-кубе // International Journal of Open Information Technologies. 2015. Vol. 3, no. 7. P. 1–11.

Мельников Борис Феликсович, д.ф.-м.н., профессор, факультет Вычислительной математики и кибернетики, Совместный университет МГУ–ППИ в Шэньчжэне (Шэньчжэнь, Китай)

DOI: 10.14529/cmse240103

ON THE NON-EXISTENCE OF A SIMPLE VERSION OF THE POLYNOMIAL ALGORITHM FOR EXTRACTING THE ROOT FROM THE LANGUAGE

© 2024 B.F. Melnikov

Shenzhen MSU – BIT University (International University Park Road 1, Dayun New Town, Longgang District, Shenzhen, Guangdong Province, 518172 China)

E-mail: bormel@mail.ru

Received: 07.10.2023

For the usual operation of concatenation of words considered as multiplication, the concatenation of languages is obviously determined, and on the basis of the last operation, the degree of the language and the root of a given degree (if available) is determined. When describing algorithms for constructing a language that is a root of degree M from a given language, so called potential roots are of great importance: these are the words (not the languages) whose considered M-th degree is included in a given language. It is easily to show that all potential roots for a given language are constructed using a polynomial algorithm. This task, apparently, is not simplified when considering words and languages over the 1-letter alphabet, which is done in this paper. So called taboo pair of potential roots is a pair whose word concatenation is not included in the language. In previous publications on the topic of describing algorithms for extracting roots from a language, the hypothesis arose that a polynomial algorithm for extracting a root from a language can be described on the basis of considering the set of taboo pairs only, by iterating over specially described subsets of the set of potential roots. This paper shows, that such an algorithm (called "simple") is impossible, i.e., if there is a polynomial algorithm for extracting the root from the language, then this algorithm must use some additional information.

Keywords: formal languages, concatenation of languages, degree of the language, root of the language, polynomial algorithms.

FOR CITATION

Melnikov B.F. On the Non-existence of a Simple Version of the Polynomial Algorithm for Extracting the Root from the Language. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2024. Vol. 13, no. 1. P. 38–56. (in Russian) DOI: 10.14529/cmse240103.

References

1. Melnikov B.F., Melnikova A.A. On the problems of extracting the root from a given finite language. International Journal of Open Information Technologies. 2023. Vol. 11, no. 5. P. 1–14. (in Russian)

2. Melnikov B.F., Korabelshchikova S.Yu., Dolgov V.N. On the task of extracting the root from the language. International Journal of Open Information Technologies. 2019. Vol. 7, no. 3. P. 1–6.

3. Melnikov B.F. A polynomial algorithm for constructing the optimal inverse morphism. International Journal of Open Information Technologies. 2023. Vol. 11, no. 6. P. 1–10. (in Russian)

4. Stockmeyer L.J. The polynomial-time hierarchy. Theoretical Computer Science. 1976. Vol. 3, no. 1. P. 1–22.

5. Chandra A.K., Kozen D., Stockmeyer L.J. Alternation. Journal of the ACM. 1981. Vol. 28, no. 1. P. 114–133.

6. Immerman N. TDescriptive Complexity. Berlin: Springer, 1999. 284 p.

7. Hromkovič J. Theoretical Computer Science: Introduction to Automata, Computability, Complexity, Algorithmics, Randomization, Communication, and Cryptography. Berlin: Springer, 2003. 323 p.

8. Hromkovič J. Algorithmics for Hard Problems: Introduction to Combinatorial Optimization, Randomization, Approximation, and Heuristics. Berlin: Springer, 2004. 547 p.

9. Melnikov B.F. Semi-lattices of the subsets of potential roots in the problems of the formal languages theory. Part I. Extracting the root from the language. International Journal of Open Information Technologies. 2022. Vol. 10, no. 4. P. 1–9. (in Russian)

10. Ryabov G.G. On the quaternary coding of cubic structures. Computational methods and programming. 2009. Vol. 10, no. 3. P. 340–347. (in Russian)

11. Ryabov G.G. Hausdorff metric on the faces of an N-dimensional cube. Fundamental and Applied Mathematics. 2010. Vol. 16, no. 1. P. 151–155. (in Russian)

12. Ryabov G.G. Polymorphism of symbolic ternary matrices and genetic space of shortest K-paths in an N-cube. International Journal of Open Information Technologies. 2015. Vol. 3, no. 7. P. 1–11. (in Russian)

УДК 519.7 **DOI: 10.14529/cmse240104**

ПРЕОБРАЗОВАНИЕ ЛАПЛАСА—СТИЛТЬЕСА ФУНКЦИИ РАСПРЕДЕЛЕНИЯ ПИКОВОГО ВОЗРАСТА ИНФОРМАЦИИ В МНОГОКАНАЛЬНОЙ ГРУППЕ ПЕРЕДАЧИ

© 2024 С.И. Матюшенко

Российский университет дружбы народов
(117198 Москва, ул. Миклухо-Маклая, д. 6)
E-mail: matyushenko-si@rudn.ru

Поступила в редакцию: 12.12.2023

Данная статья продолжает цикл работ автора, посвященных проблеме возраста информации (Age of Information, AoI) — метрики, используемой в информационных системах для мониторинга и управления удаленными источниками информации со стороны центра управления. Теоретический анализ систем передачи информации требует количественной оценки «свежести» информации, доставляемой в центр управления. В данной работе рассматривается модель двухузловой группы передачи, состоящей из источника информации (узла-отправителя), центра управления (узла-получателя) и нескольких каналов связи между ними. Предполагается, что пропускные способности каналов могут быть различными. При этом, сетевой протокол требует, чтобы информация, поступающая в узел-получатель считывалась в той же последовательности, в какой она была передана из узла-отправителя. В результате пакеты, нарушившие установленный порядок, задерживаются в узле-отправителе на время, требуемое для восстановления порядка. В данной работе процесс передачи информации моделируется с помощью многоканальной системы массового обслуживания с ограниченным накопителем, пуассоновским потоком заявок, экспоненциальным обслуживанием и переупорядочиванием заявок. При этом заявки моделируют пакеты передаваемой информации, накопитель системы — очередь пакетов на передачу, обслуживание заявок на приборах различной интенсивности — процесс передачи пакетов по каналам связи. Данная модель для оценки возраста информации использовалась впервые. В результате проведенного исследования получены выражения для преобразования Лапласа—Стилтьеса стационарной функции распределения и начальных моментов максимального значения возраста информации, называемого пиковым возрастом. Проведено численное исследование показателей производительности системы, включающее анализ пикового возраста информации при различных загрузках системы. Корректность аналитических результатов подтверждена результатами имитационного моделирования.

Ключевые слова: возраст информации, пиковый возраст информации, многоканальная система массового обслуживания, переупорядочивание заявок.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Матюшенко С.И. Преобразование Лапласа—Стилтьеса функции распределения пикового возраста информации в многоканальной группе передачи // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2024. Т. 13, № 1. С. 57–73. DOI: 10.14529/cmse240104.

Введение

Одним из важных сценариев использования приложений сверхнадежной связи является обмен информацией между удаленными системами и центром управления, который обеспечивает контроль и повышает их безопасность за счет своевременного принятия верных управленческих решений [1]. Задачи своевременной доставки информации возникают в различных сферах человеческой деятельности: в энергетических системах, в индустриальном интернете вещей, в сфере автономного транспорта, в системах видеонаблюдения

и т.д. [2–4]. В 2011 году для количественной оценки свежести информации, поступающей в центр управления, была предложена метрика Age of Information (AoI) [5], представляющая собой функцию времени между генерациями сообщений в узле-отправителе (УО) и доставкой их в узел-получатель (УП).

Наиболее удобным аппаратом для исследования проблемы возраста информации является аппарат систем и сетей массового обслуживания. Обзор работ, в которых анализ возраста информации предлагается проводить с использованием этого аппарата, можно найти, например, в [6].

В данной работе будем рассматривать системы передачи данных, в которых информация от источника к получателю передается одновременно по нескольким каналам связи [7–11]. В таких системах из-за возможного различия в пропускных способностях каналов, либо из-за случайности объемов информации, заключенной в каждом пакете, порядок, в котором были переданы пакеты, может быть нарушен. В этом случае для адекватного прочтения информации в узле-получателе производится переупорядочивание полученных пакетов. Процесс восстановления порядка требует определенных временных затрат, которые необходимо учитывать при оценке возраста информации.

Цель данной работы состоит в оценке возраста информации, передаваемой от периферийного источника к центру обработки и принятия решений, включающей в себя затраты времени на переупорядочивание пакетов. Для решения данной задачи в качестве математической модели будем использовать многоканальную систему массового обслуживания с переупорядочиванием заявок [12, 13]. Заметим, что задача оценки возраста информации в системе с переупорядочиванием заявок рассматривается впервые.

Работа состоит из шести основных разделов. В первом разделе приводится техническое описание рассматриваемой системы и строится ее математическая модель в виде системы массового обслуживания (СМО). Стохастическое поведение СМО описывается с помощью однородного марковского процесса. В разделах 2–5 на основе стационарного распределения состояний марковского процесса определяются выражения для функций распределений следующих случайных величин: времени ожидания обслуживания (раздел 2); задержки переупорядочивания (раздел 3); времени обслуживания (раздел 4); пикового возраста информации (раздел 5). В разделе 6 приводятся результаты численного исследования основных показателей производительности рассматриваемой системы при различных значениях исходных параметров. В заключении подводятся итоги и намечаются планы дальнейшего исследования.

1. Описание модели

Рассмотрим группу передачи информации, состоящую из узла-отправителя, узла получателя и m каналов связи между ними (рис. 1).

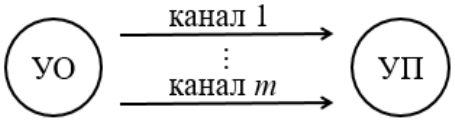


Рис. 1. m-канальная группа передачи

Пропускные способности каналов могут быть различными. Если в момент отправки из УО очередного пакета информации свободны несколько каналов, то для передачи выбирается канал с наивысшей пропускной способностью.

Каждому пакету в момент отправки из УО присваивается порядковый номер. Предполагается, что информация в УП будет воспринята адекватно, если она будет считана в том же порядке, в котором была отправлена из УО. Поэтому, если какой-то пакет нарушит установленный порядок, то он будет задержан в УП до момента поступления пакетов, передача которых началась раньше передачи данного пакета.

Процесс передачи информации из УО в УП будем моделировать с помощью многоканальной системы массового обслуживания (СМО) с m обслуживающими приборами, $2 \leqslant m < \infty$, общим накопителем ограниченной емкости и буфером переупорядочивания (БП). При этом заявка будет моделировать пакет информации, накопитель — очередь пакетов на передачу, обслуживание на приборе j — передачу пакета по j-му каналу связи, буфер переупорядочивания — место в УП, занимаемое пакетами, задержанными для восстановления порядка.

Далее будем полагать, что на систему поступает пуассоновский поток заявок с параметром λ. Заявки имеют случайную длину и при этом предполагается, что все длины заявок независимы в совокупности и имеют общую функцию распределения (ФР) $G(x) = 1 - \mathrm{e}^{-\gamma x}$, $x \geqslant 0$. Времена обслуживания на приборе j независимы между собой, а также не зависят от времени обслуживания на других приборах и распределены по экспоненциальному закону с параметром μ_j, $j = \overline{1, m}$.

Емкость накопителя системы характеризуется двумя параметрами: максимальным числом r мест для ожидания, $r < \infty$, и числом v, $v > 0$, ограничивающим суммарный объем заявок в очереди. Заявка, поступающая в систему, когда в ней находиться $m+r$ заявок или же когда суммарный объем ожидающих в очереди заявок и данной заявки превышает v, теряется и в дальнейшем не влияет на функционирование системы.

Далее, без ограничения общности примем, что $\mu_1 \geqslant ... \geqslant \mu_m$ и будем предполагать, что заявка, имеющая возможность выбора прибора, выбирает прибор с наименьшим порядковым номером. Выбор заявки из очереди на обслуживание происходит в порядке их прибытия в систему, т.е. согласно дисциплине FCFS — первый пришел, первый обслужился.

Предполагается, что всем заявкам в момент поступления в систему присваивается порядковый номер. При этом, если в момент окончания обслуживания заявки с номером n (n-заявки) продолжается обслуживание заявки с номером меньшим n, то заявка n помещается в буфер переупорядочивания (БП) (рис. 2). В противном случае n-заявка покидает систему и вслед за ней из БП уходят все заявки с номерами, отличающимися друг от друга на единицу, начиная с номера $n + 1$ (если таковые в этом буфере имеются). Данное предположение позволяет моделировать механизм сохранения порядка на выходе из системы, в соответствии с которым заявки поступали в нее.

В соответствии с обозначениями Кенделла рассматриваемую СМО будем кодировать как $M/M/m/(r, v)/res$, где (r, v) означает, что емкость накопителя ограничивается двумя параметрами: числом мест для ожидания r и ограничением на суммарный объем заявок в очереди v, а res (сокращение от resequence) означает, что в системе предусмотрен механизм переупорядочивания заявок.

Как уже было замечено выше, данная система уже рассматривалась в работе [12, 13]. В [12] функционирование системы было описано однородным марковским процессом $X(t)$,

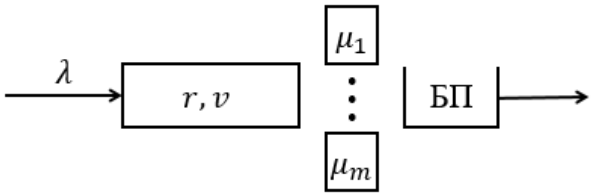


Рис. 2. m-канальная экспоненциальная СМО конечной емкости с буфером переупорядочивания

$t \geqslant 0$, над множеством состояний

$$\mathscr{X}^m = \bigcup_{k=0}^{m+r} \mathscr{X}_k^m, \tag{1}$$

$$\mathscr{X}_k^m = \begin{cases} \Big\{(k, i_1, ..., i_m),\ i_j = \overline{0,k},\ \sum_{j=1}^m u(i_j) = k, \ \text{ при этом,} \\ \text{если } i_j i_s > 0, \text{то } i_j \neq i_s,\ j,s = \overline{1,m}\Big\}, k = \overline{0, m-1}, \\ \Big\{(k, i_1, ..., i_m),\ i_j = \overline{1,m},\ i_j \neq i_s,\ j,s = \overline{1,m}\Big\}, k = \overline{m, m+r}. \end{cases}$$

где $u(x) = \begin{cases} 1,\ x > 0 \\ 0,\ x \leqslant 0 \end{cases}$ — функция Хевисайда.

Здесь для некоторого момента времени t: $X(t) = (k, i_1, ..., i_m)$, если в системе находится k заявок, $k = \overline{0, m+r}$, $i_j = 0$ означает, что прибор j пуст, в противном случае i_j есть номер заявки, обслуживаемой прибором j, $j = \overline{1,m}$.

В [12] был разработан алгоритм для определения стационарных вероятностей $p(k, i_1, ..., i_m)$ состояний $(k, i_1, ..., i_m) \in \mathscr{X}^m$ МП $X(t)$. Цель данного исследования состоит в определении стационарной ФР пикового возраста информации, передаваемой заявками СМО $M/M/m/(r,v)/res$. Очевидно, что в этой системе пиковый возраст Z_{n-1} заявки $n-1$ определяется выражением:

$$Z_{n-1} = G_n + W_n + Q_n + R_n, \tag{2}$$

где G_n — время генерации; W_n — время ожидания обслуживания; Q_n — длительность обслуживания; R_n — время переупорядочивания заявки n. Поэтому задача состоит в том, чтобы найти ФР для каждой из перечисленных выше случайных величин (с.в.).

2. Стационарная функция распределения времени ожидания обслуживания

Рассмотрим множество S различных векторов длины m, координаты которых могут принимать лишь два значения: ноль или единица. Очевидно, что множество S представимо в виде:

$$S = \bigcup_{k=0}^{m} S_k,$$

где S_k — множество, состоящее из всех возможных векторов из S, у которых ровно k координат равны единице.

Далее, обозначим через $(\mathbf{s}_k)$ макросостояние рассматриваемой СМО, $\mathbf{s}_k \in S_k$, означающее, что в системе k заявок, $k = \overline{0,m}$, и при этом прибор j занят обслуживанием, если j-я

координата s_k^j вектора $\mathbf{s}_k$ равна единице, в противном случае прибор j свободен, $j = \overline{1,m}$. Кроме этого, обозначим через (k) макросостояние, означающее, что в системе k заявок, $k = \overline{m+1, m+r}$.

Заметим, что $(\mathbf{s}_k)$ и (k) являются макросостояниями по отношению к состояниям МП $X(t)$. Они учитывают лишь количество заявок в системе и то, занят ли прибор или нет, но не учитывают порядок, в котором занимались приборы.

Очевидно, что стационарное распределение вероятностей $p(\mathbf{s}_k)$, $\mathbf{s}_k \in S_k$, $k = \overline{0,m}$ и $p(k)$, $k = \overline{m+1, m+r}$, несложно подсчитать, суммируя вероятности соответствующих микросостояний системы и определяемые в соотвествии с [12].

Далее обозначим через $\pi_A^-(\mathbf{s}_k)$ и $\pi_A^-(k)$ стационарные вероятности макросостояний $(\mathbf{s}_k)$, $\mathbf{s}_k \in S_k$, $k = \overline{0,m}$ и (k), $k = \overline{m+1, m+r}$, в момент $t-0$ поступления в СМО очередной заявки. Для определения вероятностей $\pi_A^-(\mathbf{s}_k)$ и $\pi_A^-(k)$ воспользуемся результатами работы [14], согласно которым:

$$\pi_A^-(\mathbf{s}_k) = p(\mathbf{s}_k), \quad \mathbf{s}_k \in S_k, \quad k = \overline{0,m}, \tag{3}$$

$$\pi_A^-(k) = p(k), \quad k = \overline{m+1, m+r}. \tag{4}$$

Кроме этого, введем следующие обозначения:

- $W(t)$ — ФР времени ожидания обслуживания для заявки, принятой в систему в стационарном режиме ее работы;
- $W_k(t)$ — условная стационарная ФР времени ожидания обслуживания для заявки, принятой в систему, при условии, что в момент $t-0$ ее поступления в системе уже имелось k заявок;
- $w(s)$ — преобразование Лапласа—Стилтьеса (ПЛС) ФР $W(t)$
- $w_k(s)$ — ПЛС ФР $W_k(t)$.

Теорема 1. ФР времени ожидания обслуживания для заявок, принятых в СМО $M/M/m/(r,v)/res$ в стационарном режиме ее работы определяется выражениями:

$$W(t) = \frac{1}{1-\pi}\left[\sum_{k=0}^{m}\sum_{\mathbf{s}_k \in S_k} p(\mathbf{s}_k)W_k(t) + \sum_{k=m+1}^{m+r-1} p(k)W_k(t)\right], \tag{5}$$

$$W_k(t) = \begin{cases} 1, & k = \overline{0, m-1}, \\ f_k F_\xi^{*(k-m+1)}(t), & k = \overline{m, m+r-1}, \end{cases} \tag{6}$$

где $F_\xi(t)$ — ФР с.в. $\xi = \min\{\xi_1, ..., \xi_m\}$, ξ_j — с.в. — длительность обслуживания на приборе j, $j = \overline{0,m}$, запись $F_\xi^{*(k-m+1)}(t)$ означает $(k-m+1)$-кратную свертку ФР $F_\xi(t)$, π — вероятность потери заявки.

Доказательство. Заявка, поступающая в систему, и застающая хотя бы один прибор свободным, сразу направляется на обслуживание и время ожидания для нее равно нулю. Если же в момент $t-0$ поступления очередной заявки в очереди имеется $k-m$ заявок, $k = \overline{m, m+r-1}$, то заявка с вероятностью f_k присоединяется к очереди, где f_k согласс-

но [12], определяется по формуле:

$$f_k = \begin{cases} 1, & k = \overline{0, m-1} \\ G_{k-m+1}(v)\,/G_{k-m}(v)\,, & k = \overline{m, m+r-1} \\ 0, & k = m+k \end{cases}, \tag{7}$$

$G_0(v) = 1$, $G_s(v) = 1 - \mathrm{e}^{-\gamma v} \sum_{j=0}^{s-1} \dfrac{(\gamma v)^j}{j!}$, $s = \overline{1, r}$.

В этом случае время ожидания для нее будет складываться из минимума времен дообслуживания заявок, находящихся в момент $t - 0$ на приборах и времени обслуживания любых $k - m$ заявок, пришедших в систему раньше данной заявки. Учитывая, что времена обслуживания заявок на приборе j, $j = \overline{1, m}$, независимы между собой, а также не зависят от времени обслуживания на других приборах, приходим к выражению (6).

Нетрудно видеть, что с.в. ξ распределена по экспоненциальному закону с параметром $\mu = \sum_{j=1}^{m} \mu_j$, следовательно выражение (6) можно записать в следующем виде:

$$W_k(t) = \begin{cases} 1, & k = \overline{0, m-1}, \\ f_k \left[1 - \mathrm{e}^{-\mu t} \sum_{j=0}^{k-m} \frac{(\mu t)^j}{j!}\right], & k = \overline{m, m+r-1}, \end{cases} \tag{8}$$

Далее применяя формулы полной и условной вероятностей, а также учитывая формулы (3)–(4), получаем (5). Тем самым, теорема доказана.

Заметим, что выражение для вероятности потерь было получено в [12] и имеет следующий вид:

$$\pi = \sum_{k=0}^{m} \sum_{\mathbf{s}_k \in S_k} p(\mathbf{s}_k)(1 - f_k) + \sum_{k=m+1}^{m+r} p(k)(1 - f_k). \tag{9}$$

□

Из теоремы 1 вытекает два следствия.

Следствие 1. Начальные моменты v_ϑ порядка ϑ, $\vartheta = 1, 2, ...$ времени ожидания обслуживания для заявок, принятых в СМО $M/M/m/(r, v)/res$ в стационарном режиме ее работы определяется выражением:

$$v_\vartheta = \frac{1}{1-\pi} \left[f_m p(\mathbf{s}_m) \frac{\vartheta!}{\mu^\vartheta} + \sum_{k=m+1}^{m+r-1} f_k p(k) \frac{(k-m+1) \cdot ... \cdot (k-m+\vartheta)}{\mu^\vartheta} \right]. \tag{10}$$

В частности, для $\vartheta = 1$ получим:

$$v_1 = \frac{1}{1-\pi} \left[f_m p(\mathbf{s}_m) \frac{1}{\mu} + \sum_{k=m+1}^{m+r-1} f_k p(k) \frac{k-m+1}{\mu} \right]. \tag{11}$$

Доказательство следствия 1 вытекает непосредственно из определения начальных моментов порядка ϑ.

Следствие 2. ПЛС ФР времени ожидания обслуживания для заявок, принятых в систему $M/M/m/(r, v)/res$ в стационарном режиме ее работы определяется выражениями:

$$w(s) = \frac{1}{1-\pi} \left[\sum_{k=0}^{m} \sum_{\mathbf{s}_k \in S_k} p(\mathbf{s}_k) w_k(s) + \sum_{k=m+1}^{m+r-1} p(k) w_k(s) \right] \tag{12}$$

$$w_k(s) = \begin{cases} 1, & k = \overline{0, m-1}, \\ f_k \left(\frac{\mu}{s+\mu}\right)^{k-m+1}, & k = \overline{m, m+r-1}. \end{cases} \tag{13}$$

Доказательство. Дифференцируя (8) по t, а затем домножая на e^{-st} и интегрируя по t на интервале $(0; \infty)$, получаем выражение для ПЛС $w_k(s)$:

$$w_k(s) = \begin{cases} 1, & k = \overline{0, m-1}, \\ f_k \left(\frac{\mu}{s+\mu}\right)^{k-m+1}, & k = \overline{m, m+r-1}, \end{cases} \tag{14}$$

Далее, применяя формулы полной и условной вероятностей, получаем (12). □

3. Стационарная функция распределения задержки переупорядочивания

Теперь обозначим через $\pi^-_{D,j}(k, i_1, ..., i_m)$ стационарную вероятность состояния $(k, i_1, ..., i_m)$ в момент $t - 0$ окончания обслуживания заявки на приборе j, а через $\lambda_{D,j}$ — интенсивность выхода заявок, обслуженных прибором j, $j = \overline{1, m}$. Нетрудно видеть, что

$$\lambda_{D,j} = \sum_{(k,i_1,...,i_m) \in \mathscr{X}^m} p(k, i_1, ..., i_m) u(i_j) \mu_j, \quad j = \overline{1, m}. \tag{15}$$

На основании результатов работы [14] получаем:

$$\pi_{D,j}(k, i_1, ..., i_m) = \frac{1}{\lambda_{D,j}} p(k, i_1, ..., i_m) u(i_j) \mu_j. \tag{16}$$

Далее введем обозначения:

- $R(t)$ — стационарная ФР времени пребывания заявки в буфере переупорядочивания, которое в дальнейшем будем называть задержкой переупорядочивания;
- $R_{(k,i_1,\ldots,i_m),j}(t)$ — условная стационарная ФР задержки переупорядочивания заявки, обслуженной на приборе j, при условии, что в момент $t - 0$ окончания обслуживания этой заявки системы находилась в состоянии $(k, i_1, \ldots, i_m)$;
- S_ν — начальный момент порядка ν задержки переупорядочивания, $\nu = 1, 2, ...$;
- $S_{(k,i_1,\ldots,i_m),j,\nu}$ — условный начальный момент порядка ν, $\nu = 1, 2, ...$ задержки переупорядочивания заявки, обслуженной на приборе j, при условии, что в момент $t - 0$ окончания ее обслуживания заявка находилась в состоянии $(k, i_1, ..., i_m)$;
- $r(s)$ — ПЛС ФР $R(t)$;
- $r_{(k,i_1,\ldots,i_m),j}(s)$ — ПЛС ФР $R_{(k,i_1,\ldots,i_m),j}(t)$.

Теорема 2. ФР задержки переупорядочивания в СМО $M/M/m/(r, v)/res$ в стационарном режиме ее работы определяется выражением:

$$R(t) = \frac{1}{\lambda_D} \sum_{(k,i_1,...,i_m) \in \mathscr{X}^m} p(k, i_1, ..., i_m) \sum_{j=1}^{m} u(i_j) \mu_j \prod_{s_n \in Y_{(k,i_1,\ldots,i_m)}} \left(1 - \mathrm{e}^{-\mu_{s_n} t}\right), \tag{17}$$

где $\lambda_D = \sum_{j=1}^m \lambda_{D,j}$.

Доказательство. Ясно, что если непосредственно перед окончанием обслуживания заявки на приборе j система находилась в состоянии $(k, i_1, ..., i_m)$, то задержка переупорядочивания этой заявки будет равна максимуму из времени дообслуживания заявок, пришедших в СМО раньше данной заявки, т.е. заявок, обслуживаемых на приборах с номерами из множества

$$Y_{(k,i_1,...,i_m),j} = \{s_n : 0 < i_{s_n} < i_j\}, \quad j = \overline{1,m}, \quad (k, i_1, ..., i_m) \in \mathscr{X}^m.$$

Так как обслуживание заявки на приборе j распределено по экспоненциальному закону с параметром μ_j, $j = \overline{1,m}$, то

$$R_{(k,i_1,...,i_m),j}(t) = \prod_{s_n \in Y_{(k,i_1,...,i_m),j}} \left(1 - \mathrm{e}^{-\mu_{s_n} t}\right), \quad j = \overline{1,m}, \quad (k, i_1, ..., i_m) \in \mathscr{X}^m. \tag{18}$$

При этом, $R_{(k,i_1,...,i_m),j}(t) = 1$, если $Y_{(k,i_1,...,i_m),j}(t) = \varnothing$, т.к. заявка в этом случае не будет задержана.

Далее, применяя формулу полной вероятности и учитывая (16), а также тот факт, что вероятность выхода заявки с прибора j равна $\lambda_{D,j}/\lambda_D$, получаем формулу (17). Таким образом, теорема 2 доказана. □

Из теоремы 2 вытекает два следствия.

Следствие 3. Начальный момент S_ϑ порядка ϑ задержки переупорядочивания в СМО $M/M/m/(r,v)/res$ в стационарном режиме ее работы определяется выражениями:

$$S_\vartheta = \frac{1}{\lambda_D} \sum_{(k,i_1,...,i_m) \in \mathscr{X}^m} p(k, i_1, ..., i_m) \sum_{j=1}^{m} u(i_j) \mu_j S_{(k,i_1,...,i_m),j,\vartheta}, \quad \vartheta = 1, 2, ... \tag{19}$$

$$S_{(k,i_1,...,i_m),j,\vartheta} = \vartheta! \left[\sum_{n_1=1}^{l} \frac{1}{\mu_{s_{n_1}}^\vartheta} - \sum_{n_1,n_2=1, n_1 \neq n_2}^{l} \frac{1}{(\mu_{s_{n_1}} + \mu_{s_{n_2}})^\vartheta} + ... + \right.$$
$$\left. + (-1)^{l+1} \frac{1}{(\mu_{s_{n_1}} + ... + \mu_{s_{n_l}})^\vartheta} \right], \tag{20}$$

где $s_{n_i} \in Y_{(k,i_1,...,i_m),j}$, $i = \overline{1,l}$, $l = |Y_{(k,i_1,...,i_m),j}|$.

Доказательство.

$$S_{(k,i_1,...,i_m),j,\vartheta} = \int_0^\infty t^\vartheta dR_{(k,i_1,...,i_m),j}(t), \quad j = \overline{1,m}, \quad (k, i_1, ..., i_m) \in \mathscr{X}^m. \tag{21}$$

Из (21) с учетом (18) путем несложных вычислений приходим к (20). И, наконец, применяя формулы полной и условной вероятности и, учитывая (16) и тот факт, что вероятность выхода заявки с прибора j равна $\lambda_{D,j}/\lambda_D$, получаем (19).

□

Следствие 4. ПЛС ФР задержки переупорядочивания в СМО $M/M/m/(r,v)/res$ в стационарном режиме ее работы определяется выражением:

$$r(s) = \frac{1}{\lambda_D} \sum_{(k,i_1,...,i_m) \in \mathscr{X}^m} p(k, i_1, ..., i_m) \sum_{j=1}^{m} u(i_j) \mu_j r_{(k,i_1,...,i_m),j}(s), \tag{22}$$

$$r_{(k,i_1,\ldots,i_m),j}(s) = \sum_{n_1=1}^{l} \frac{\mu_{s_{n_1}}}{\mu_{s_{n_1}} + s} - \sum_{n_1,n_2=1,n_1\neq n_2}^{l} \frac{\mu_{s_{n_1}} + \mu_{s_{n_2}}}{\mu_{s_{n_1}} + \mu_{s_{n_2}} + s} + \ldots + \\ + (-1)^l \frac{\mu_{s_{n_1}} + \ldots + \mu_{s_{n_l}}}{(\mu_{s_{n_1}} + \ldots + \mu_{s_{n_l}} + s)}. \quad (23)$$

Доказательство. Используя определение ПЛС ФР, из формулы (18) путем несложных вычислений приходим к формуле (23). Применяя формулы полной и условной вероятностей, а также равенства (16) и тот факт, что вероятность выхода заявки с прибора j равна $\lambda_{D,j}/\lambda_D$, получаем выражение (22). □

4. Стационарная функция распределения времени обслуживания

Теперь определим ФР $Q(t)$ времени обслуживания заявки в системе в стационарном режиме ее работы. Для этого рассмотрим условные стационарные ФР $Q_{\mathbf{s}_k}(t)$, $\mathbf{s}_k \in S_k$, $k = \overline{0,m}$ и $Q_k(t)$, $k = \overline{m+1, m+r-1}$, времени обслуживания заявки в системе при условии, что в момент $t-0$ поступления данной заявки в систему, система находилась в состоянии $(\mathbf{s}_k)$ либо (k) соответственно.

Очевидно, что при наличии свободных приборов заявка, поступающая в систему, сразу направляется на прибор с минимальным порядковым номером. Следовательно,

$$Q_{\mathbf{s}_k - \mathbf{1}_j}(t) = \delta\left(\sum_{i=1}^{j} s_k^i, j\right)\left(1 - \mathrm{e}^{-\mu_j t}\right), \quad j = \overline{1,m}, \quad (24)$$

где $\mathbf{1}_j$ — вектор длины m с j-й координатой, равной единице и остальными координатами равными нулю; $\delta(x,y)$ — дельта-функция, принимающая значение 1 при $x = y$ и 0 в противном случае; s_k^i — i-я координата $\mathbf{s}_k$.

Если же в момент $t-0$ поступления очередной заявки в очереди уже имеется $k-m$ заявок, $k = \overline{m, m+r-1}$, то данная заявка с вероятностью f_k присоединяется к очереди и через некоторое время перемещается на первое место в ней. Далее с вероятностью μ_j/μ раньше других освободится прибор j и далее заявка с указанной вероятностью поступит на этот прибор. Следовательно,

$$Q_{\mathbf{s}_m}(t) = \sum_{j=1}^{m} f_m \mu_j/\mu \left(1 - \mathrm{e}^{-\mu_j t}\right), \quad (25)$$

$$Q_k(t) = \sum_{j=1}^{m} f_k \mu_j/\mu \left(1 - \mathrm{e}^{-\mu_j t}\right). \quad (26)$$

Далее, применяя формулы условной и полной вероятности, а также учитывая соотношения (3) и (4), приходим к следующему результату

Теорема 3. ФР времени обслуживания заявок, принятых в СМО $M/M/m/(r,v)/res$ в стационарном режиме ее работы, определяется выражением:

$$Q(t) = \frac{1}{1-\pi}\left[\sum_{k=1}^{m}\sum_{\mathbf{s}_k\in S_k}\sum_{j=1}^{m} p(\mathbf{s}_k - \mathbf{1}_j) Q_{\mathbf{s}_k - \mathbf{1}_j}(t) + p(\mathbf{s}_m) Q_{\mathbf{s}_m}(t) + \sum_{k=m+1}^{m+r-1} p(k) Q_k(t)\right], \quad (27)$$

где $Q_{\mathbf{s}_k}(t)$, $\mathbf{s}_k \in S_k$, $k = \overline{0,m}$, $Q_k(t)$, $k = \overline{m+1, m+r-1}$ определяются по формулам (24)–(26).

Из теоремы 3 вытекает очевидное следствие.

Следствие 5. Начальные моменты q_ϑ порядка ϑ, $\vartheta = 1, 2, ...$, времени обслуживания заявок, принятых в СМО $M/M/m/(r,v)/res$ в стационарном режиме ее работы, определяется выражением:

$$q_\vartheta = \frac{1}{1-\pi}\left[\sum_{k=1}^{m}\sum_{\mathbf{s}_k\in S_k}\sum_{j=1}^{m} p(\mathbf{s}_k - \mathbf{1}_k)\delta\left(\sum_{i=1}^{j} s_k^j, j\right)\frac{\vartheta!}{\mu_j^\vartheta} + \right.$$
$$\left. + \left(\sum_{k=m+1}^{m+r-1} p(k)f_k + p(\mathbf{s}_m)f_m\right)\sum_{j=1}^{m}\frac{\vartheta!}{\mu_j\mu_j^{\vartheta-1}}\right]. \quad (28)$$

В частности для $\vartheta = 1$ получаем

$$q_1 = \frac{1}{1-\pi}\left[\sum_{k=1}^{m}\sum_{\mathbf{s}_k\in S_k}\sum_{j=1}^{m} p(\mathbf{s}_k - \mathbf{1}_k)\delta\left(\sum_{i=1}^{j} s_k^j, j\right)\frac{1}{\mu_j} + \left(\sum_{k=m+1}^{m+r-1} p(k)f_k + p(\mathbf{s}_m)f_m\right)\frac{m}{\mu}\right]. \quad (29)$$

Обозначим через $q(s)$ — ПЛС ФР $Q(t)$, а через $q_{\mathbf{s}_k}(s)$ и $q_k(s)$ — ПЛС условных ФР $Q_{\mathbf{s}_k}(t)$ и $Q_k(t)$ соответственно.

Из (24)–(26) очевидным образом следует

$$q_{\mathbf{s}_k-\mathbf{1}_j}(s) = \delta\left(\sum_{i=1}^{j} s_k^i, j\right)\frac{\mu_j}{\mu_j + s}, \quad (30)$$

$$q_{\mathbf{s}_m}(s) = \sum_{j=1}^{m} f_m\frac{\mu_j}{\mu}\frac{\mu_j}{\mu_j + s}, \quad (31)$$

$$q_k(s) = \sum_{j=1}^{m} f_k\frac{\mu_j}{\mu}\frac{\mu_j}{\mu_j + s}, \quad (32)$$

что позволяет нам сформулировать еще одно очевидное следствие из теоремы 3.

Следствие 6. ПЛС ФР времени обслуживания для заявок, принятых в систему $M/M/m/(r,v)/res$ в стационарном режиме ее работы, определяется выражением:

$$q(s) = \frac{1}{1-\pi}\left[\sum_{k=1}^{m}\sum_{\mathbf{s}_k\in S_k}\sum_{j=1}^{m} p(\mathbf{s}_k - \mathbf{1}_k)q_{\mathbf{s}_k-\mathbf{1}_j}(s) + p(\mathbf{s}_m)q_{\mathbf{s}_m}(s) + \sum_{k=m+1}^{m+r-1} p(k)q_k(s)\right], \quad (33)$$

где $q_{\mathbf{s}_k-\mathbf{1}_j}(s)$, $q_{\mathbf{s}_m}(s)$ и $q_k(s)$ вычисляются по формулам (30)–(32).

5. Стационарное распределение пикового возраста информации

Возвращаясь к формуле (2) и учитывая ее справедливость для любого n, а также принимая во внимание факт независимости слагаемых в правой части этой формулы, можем

утверждать, что ФР пикового возраста информации определяется выражением:

$$Z(t) = G(t) * W(t) * Q(t) * R(t), \tag{34}$$

где $G(t)$ – ФР длительности генерации заявок, принятых в систему, т.е.

$$G(t) = \frac{1}{1-\pi}(1 - \mathrm{e}^{-\lambda t}), \quad t \geqslant 0, \tag{35}$$

а ФР $W(t)$, $R(t)$ и $Q(t)$ определяются выражениями (5), (17) и (27).

Однако применение формулы (34) может быть связано с определенными вычислительными трудностями. Поэтому основной результат мы сформулируем в терминах ПЛС, учитывая, что ПЛС ФР $G(t)$ определяется выражением:

$$g(s) = \frac{1}{1-\pi}\frac{\lambda}{\lambda + s}. \tag{36}$$

Теорема 4. ПЛС пикового возраста информации, передаваемой заявками системы $M/M/m/(r,v)/res$, определяется выражением:

$$z(s) = g(s)w(s)q(s)r(s), \tag{37}$$

где $w(s)$, $r(s)$, $q(s)$ и $g(s)$ вычисляются по формулам (12), (22), (33) и (36) соответственно.

Из теоремы 4 вытекает очевидное следствие.

Следствие 7. Начальные моменты z_ϑ порядка ϑ возраста информации, передаваемой заявками системы $M/M/m/(r,v)/res$, определяются выражением:

$$z_\vartheta = g_\vartheta + v_\vartheta + \delta_\vartheta + q_\vartheta, \quad \vartheta = 1, 2, ...,$$

где $g_\vartheta = \frac{\vartheta!}{(1-\pi)\lambda^\vartheta}$, а v_ϑ, δ_ϑ и q_ϑ вычисляются по формулам (10), (19) и (28).

6. Результаты численного исследования

В работе было проведено численное исследование средних значений показателей, характеризующих процесс передачи информации с помощью заявок СМО $M/M/m/(r,v)/res$ в зависимости от интенсивности входящего потока λ. Один из примеров такого исследования представлен ниже:

- количество приборов $m = 3$;
- емкость накопителя $r = 5$;
- интенсивность обслуживания на приборах: $\mu_1 = 3$, $\mu_2 = 2$ и $\mu_3 = 1$;
- длина заявки задавалась рядом распределения:

значение	1	2	3
вероятность	0.33	0.33	0.34

Результаты проведенного исследования отражены в табл. 1 и на рис. 3–6.

Так на рис. 3 отражена зависимость среднего пикового возраста информации от интенсивности входящего потока λ, а, следовательно, и от величины загрузки $\rho = \lambda/(\mu_1 + \mu_2 + \mu_3)$. Как видно из этого рисунка минимальные значения среднего пикового возраста соответствуют значениям λ из интервала от 1.5 до 2.5, что, в свою очередь, соответствует значениям ρ из интервала $0.25 \div 0.42$. Можно считать, что данная загрузка

Таблица 1. Среднее значение пикового возраста z, задержки переупорядочивания δ, времени ожидания v, времени обслуживания q при различных значениях λ и фиксированных значениям μ

λ	ρ	z	δ	v	q
0.25	0.04	4.376	0.013	0.000	0.349
0.50	0.08	2.403	0.030	0.000	0.366
0.75	0.13	1.775	0.054	0.001	0.382
1.00	0.17	1.488	0.083	0.002	0.399
1.50	0.25	1.253	0.151	0.007	0.426
2.00	0.33	1.199	0.230	0.019	0.449
2.50	0.42	1.209	0.305	0.036	0.465
3.00	0.50	1.258	0.381	0.062	0.478
3.50	0.58	1.319	0.446	0.093	0.489
4.00	0.67	1.390	0.505	0.133	0.493
4.50	0.75	1.466	0.562	0.173	0.498
5.00	0.83	1.533	0.597	0.219	0.500
5.50	0.92	1.601	0.633	0.263	0.500
6.00	1.00	1.667	0.662	0.308	0.500
6.50	1.08	1.713	0.677	0.347	0.500
7.00	1.17	1.761	0.691	0.385	0.500
7.50	1.25	1.810	0.710	0.419	0.500
8.00	1.33	1.850	0.721	0.450	0.500
8.50	1.42	1.899	0.743	0.479	0.500
9.00	1.50	1.919	0.745	0.506	0.500

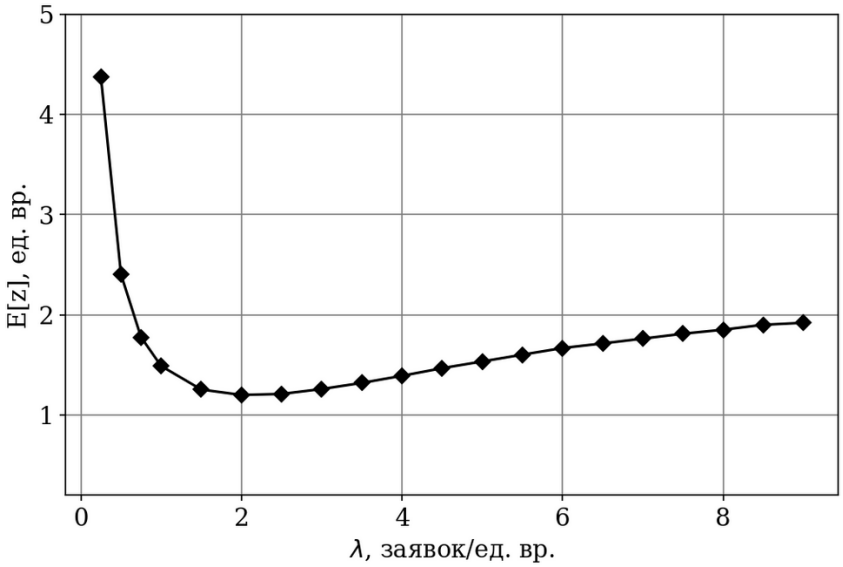


Рис. 3. График зависимости среднего пикового возраста информации от интенсивности входящего потока λ

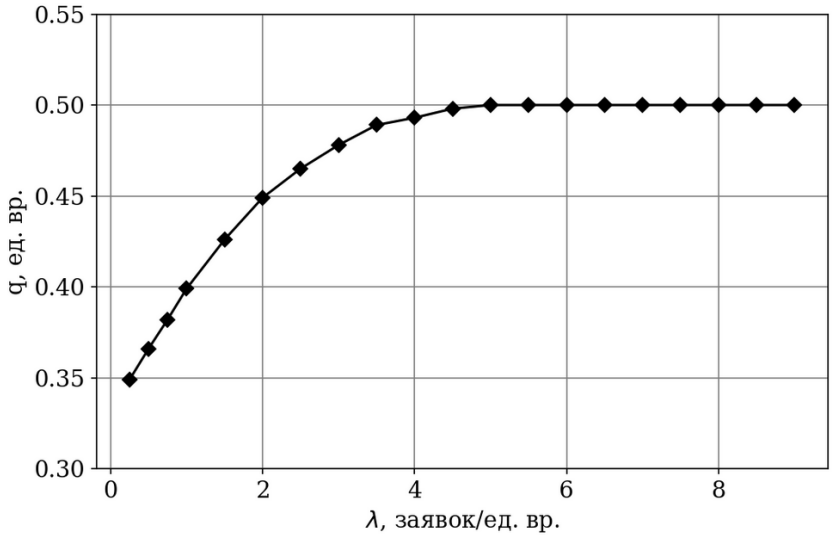


Рис. 4. График зависимости средней длительности обслуживания от интенсивности входящего потока λ

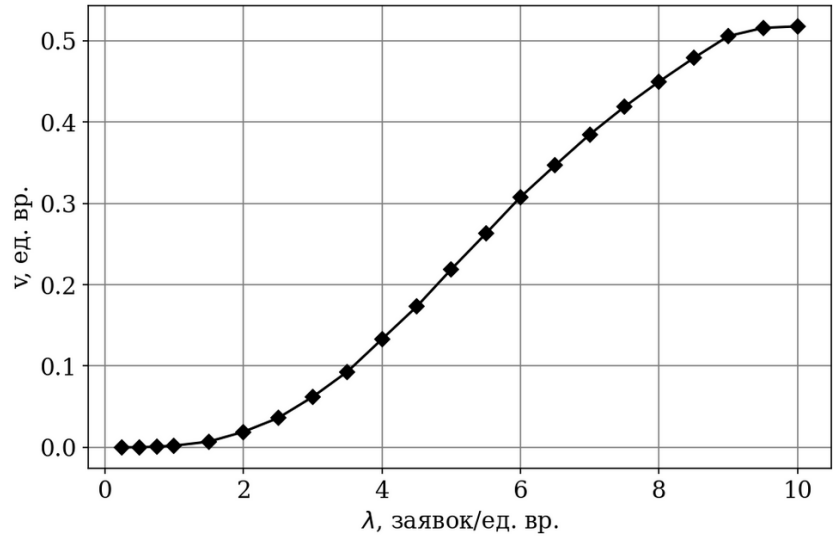


Рис. 5. График зависимости среднего времени ожидания от интенсивности входящего потока λ

является оптимальной с точки зрения частоты и скорости обновления информации. При этом мы наблюдаем значительный рост среднего пикового возраста при малых значениях интенсивности λ. Это связано с тем, что генерация пакетов происходит крайне редко. При больших значениях λ значительного роста пикового возраста не наблюдается. Связано это с тем, что в условиях большой загрузки система с ограниченным накопителем почти всегда полностью занята, что приводит к определенной стабилизации в ее работе.

На рис. 4 представлен график, отражающий зависимость средней длительности обслуживания от интенсивности λ. Как видно из рисунка, при больших значениях загрузки средняя длительность обслуживания стремится к предельному значению 0,5. Связано это с тем, что в условиях большой загрузки все приборы полностью заняты обслуживанием. При этом суммарная интенсивность обслуживания μ в нашем примере равна шести, т.е.

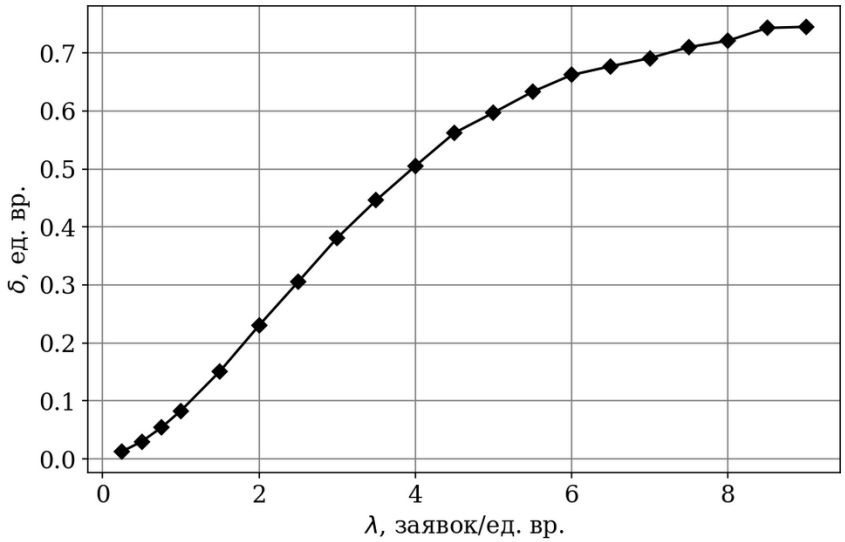


Рис. 6. График зависимости средней задержки переупорядочивания от интенсивности входящего потока λ

средняя интенсивность обслуживания одного прибора равняется двум, а, следовательно, средняя длительность обслуживания на одном приборе равна 0.5.

На рис. 5 и 6 отражена зависимость среднего времени ожидания обслуживания и средней задержки переупорядочивания от λ. На первый взгляд графики демонстрируют разное возрастание этих показателей с ростом λ по сравнению с графиком для средней длительности обслуживания. Это связано с тем, что при малых λ эти показатели фактически имеют нулевые средние значения. Однако с ростом λ и переходом системы в режим полной загрузки эти показатели тоже стабилизируются, что вполне соответствует реальности.

Заключение

Подводя итог проведенному исследованию, заметим, что в данной работе впервые исследована проблема возраста информации в многоканальной группе передачи в совокупности с проблемой переупорядочивания пакетов.

В результате проведенного исследования удалось получить выражения для преобразования Лапласа—Стилтьеса стационарной функции распределения и начальных моментов пикового возраста информации, передаваемой из периферийного источника в центр управления, моделируя процесс передачи с помощью многоканальной экспоненциальной системы массового обслуживания с переупорядочиванием заявок. Данное исследование позволило получить оценки возраста информации, учитывающие затраты времени на восстановление порядка переданных пакетов, установленного в узле-отправителе.

Результаты численного исследования показали, что не исключено существование такого набора исходных параметров рассматриваемой системы, при котором пиковый возраст информации достигает своих минимальных значений. Точное решение данного вопроса может стать предметом дальнейших исследований. Кроме этого, в своих будущих исследованиях автор предполагает обобщить полученные результаты на случай, когда процесс передачи информации моделируется посредством многоканальной системы обслуживания с распределениями длительностей генерации и обслуживания фазового типа. Неограниченный выбор параметров и фаз распределений этого типа позволит строить более точные математиче-

ские модели многоканальных систем передачи данных по сравнению с моделями в основе которых лежат однопараметрические семейства распределений.

Литература

1. Sultan A. Ultra Reliable and Low Latency Communications. 3GPP. 2023. URL: https://www.3gpp.org/technologies/urlcc-2022 (дата обращения: 15.09.2023).
2. Kaul S., Yates R., Gruteser M. Real-time status: How often should one update? // 2012 Proceedings IEEE INFOCOM, Orlando, FL, USA, 25–30 March. 2012. P. 2731–2735. DOI: 10.1109/INFCOM.2012.6195689.
3. Costa M., Codreanu M., Ephremides A. On the Age of Information in Status Update Systems With Packet Management // IEEE Transactions on Information Theory. 2016. Vol. 62, no. 4. P. 1897–1910. DOI: 10.1109/TIT.2016.2533395.
4. Kaul S.K., Yates R.D., Gruteser M. Status updates through queues // 2012 46th Annual Conference on Information Sciences and Systems (CISS), Princeton, NJ, USA, 21–23 March. 2012. P. 1–6. DOI: 10.1109/CISS.2012.6310931.
5. Kaul S., Gruteser M., Rai V., Kenney J. Minimizing age of information in vehicular networks // 2011 8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, Salt Lake City, UT, USA, 27–30 June. 2011. P. 350–358. DOI: 10.1109/SAHCN.2011.5984917.
6. Kosta A., Pappas N., Angelakis V. Angelakis. Age of Information: A New Concept, Metric and Tools // Foundations and Trends in Networking. 2017. Vol. 12, no. 3. P. 162–259.
7. Chiariotti F., Soret B., Popovski P. Peak Age of Information Distribution Bounds for Multi-Connectivity Transmissions // 2021 IEEE 22nd International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), Lucca, Italy, 27–30 September. 2021. P. 321–325. DOI: 10.1109/SPAWC51858.2021.9593271.
8. Liu Q., Zeng H., Chen M. Minimizing AoI With Throughput Requirements in Multi-Path Network Communication // IEEE/ACM Transactions on Networking. 2022. Vol. 30, no. 3. P. 1203–1216. DOI: 10.1109/TNET.2021.3135494.
9. Qian Z., Wu F., Pan J., *et al.* Minimizing Age of Information in Multi-channel Time-sensitive Information Update Systems // IEEE INFOCOM 2020 — IEEE Conference on Computer Communications, Toronto, ON, Canada, 6–9 July. 2020. P. 446–455. DOI: 10.1109/INFOCOM41043.2020.9155420.
10. Beytur H.B., Uysal-Biyikoglu E. Minimizing age of information on multi-flow networks // 2018 26th Signal Processing and Communications Applications Conference (SIU), Izmir, Turkey, 2–5 May. 2018. P. 1–4. DOI: 10.1109/SIU.2018.8404772.
11. Liu Q., Zeng H., Chen M. Minimizing Age-of-Information with Throughput Requirements in Multi-Path Network Communication // Proceedings of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing, New York, NY, USA, July. 2019. P. 41–50. DOI: 10.1145/3323679.3326502.
12. Матюшенко С.И. Анализ многоканальной системы обслуживания с ограниченным накопителем и переупорядочиванием заявок // Вестник ТвГУ. Серия: Прикладная математика. 2010. № 19. С. 55–70.
13. Matyushenko S., Ermolayeva A. On stationary characteristics of a multiserver exponential queuing system with reordering of requests // 2021 13th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), Brno, Czech Republic, 25–27 October. 2021. P. 98–103. DOI: 10.1109/ICUMT54235.2021.9631709.

14. Наумов В.А. О предельных вероятностях полумарковского процесса // Современные задачи в точных науках. М.: Университет дружбы народов, 1975. С. 35–39.

Матюшенко Сергей Иванович, к.ф.-м.н., кафедра теории вероятностей и кибербезопасности, Российский университет дружбы народов (Москва, Российская Федерация, matyushenko-si@rudn.ru)

DOI: 10.14529/cmse240104

THE LAPLACE–STIELTJES TRANSFORMATION OF THE PEAK AGE DISTRIBUTION FUNCTION OF INFORMATION IN A MULTICHANNEL TRANSMISSION GROUP

© 2024 S.I. Matyushenko

Peoples' Friendship University of Russia (RUDN University)
(Miklukho-Maklaya str. 6, Moscow, 117198 Russia)
E-mail: matyushenko-si@rudn.ru

Received: 12.12.2023

This article continues the author's cycle of works devoted to the problem of the Age of Information (AoI), a metric used in information systems for monitoring and managing remote sources of information from the control center. The theoretical analysis of information transmission systems requires a quantitative assessment of the "freshness" of information delivered to the control center. In this paper, we consider a model of a two-node transmission group consisting of an information source (sender node), a control center (recipient node) and several communication channels between them. It is assumed that the channel capacities may be different. At the same time, the network protocol requires that the information received by the recipient node be read in the same sequence as it was transmitted from the sending node. As a result, packets that violate the established order are delayed at the sending node for the time required to restore order. In this paper, the information transfer process is modeled using a multichannel queuing system with a limited storage, Poisson flow of applications, exponential maintenance and reordering of applications. At the same time, applications simulate packets of transmitted information, the system storage is a queue of packets for transmission, the service of applications on devices of varying intensity is the process of transmitting packets over communication channels. This model was used for the first time to estimate the age of information. As a result of the conducted research, expressions for the Laplace–Stieltjes transformation of the stationary distribution function and the initial moments of the maximum value of the information age, called the peak age, were obtained. A numerical study of system performance indicators has been conducted, including an analysis of the peak age of information at various system loads. The correctness of the analytical results is confirmed by the results of simulation modeling.

Keywords: age of information, peak age of information, multi-channel queuing system, reordering of applications.

FOR CITATION

Matyushenko S.I. The Laplace–Stieltjes Transformation of the Distribution Function of the Peak Age of Information in a Multichannel Transmission Group. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2024. Vol. 13, no. 1. P. 57–73. (in Russian) DOI: 10.14529/cmse240104.

References

1. Sultan A. Ultra Reliable and Low Latency Communications. 3GPP. 2023. URL: https://www.3gpp.org/technologies/urlcc-2022 (accessed: 15.09.2023).
2. Kaul S., Yates R., Gruteser M. Real-time status: How often should one update?. 2012 Proceedings IEEE INFOCOM. 2012. P. 2731–2735. DOI: 10.1109/INFCOM.2012.6195689.
3. Costa M., Codreanu M., Ephremides A. On the Age of Information in Status Update Systems With Packet Management. IEEE Transactions on Information Theory. 2016. Vol. 62, no. 4. P. 1897–1910. DOI: 10.1109/TIT.2016.2533395.
4. Kaul S.K., Yates R.D., Gruteser M. Status updates through queues. 2012 46th Annual Conference on Information Sciences and Systems (CISS). 2012. P. 1–6. DOI: 10.1109/CISS.2012.6310931.
5. Kaul S., Gruteser M., Rai V., Kenney J. Minimizing age of information in vehicular networks. 2011 8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks. 2011. P. 350–358. DOI: 10.1109/SAHCN.2011.5984917.
6. Kosta A., Pappas N., Angelakis V. Angelakis. Age of Information: A New Concept, Metric and Tools. Foundations and Trends in Networking. 2017. Vol. 12, no. 3. P. 162–259.
7. Chiariotti F., Soret B., Popovski P. Peak Age of Information Distribution Bounds for Multi-Connectivity Transmissions. 2021 IEEE 22nd International Workshop on Signal Processing Advances in Wireless Communications (SPAWC). 2021. P. 321–325. DOI: 10.1109/SPAWC51858.2021.9593271.
8. Liu Q., Zeng H., Chen M. Minimizing AoI With Throughput Requirements in Multi-Path Network Communication. IEEE/ACM Transactions on Networking. 2022. Vol. 30, no. 3. P. 1203–1216. DOI: 10.1109/TNET.2021.3135494.
9. Qian Z., Wu F., Pan J., *et al.* Minimizing Age of Information in Multi-channel Time-sensitive Information Update Systems. IEEE INFOCOM 2020 — IEEE Conference on Computer Communications. 2020. P. 446–455. DOI: 10.1109/INFOCOM41043.2020.9155420.
10. Beytur H.B., Uysal-Biyikoglu E. Minimizing age of information on multi-flow networks. 2018 26th Signal Processing and Communications Applications Conference (SIU). 2018. P. 1–4. DOI: 10.1109/SIU.2018.8404772.
11. Liu Q., Zeng H., Chen M. Minimizing Age-of-Information with Throughput Requirements in Multi-Path Network Communication. Proceedings of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing. New York, NY, USA: Association for Computing Machinery, 2019. P. 41–50. DOI: 10.1145/3323679.3326502.
12. Matyushenko S.I. Analysis of a multichannel service system with limited storage and reordering of applications. Vestnik TvGU. Seriya: Prikladnaya matematika. 2010. No. 19. P. 55–70. (in Russian).
13. Matyushenko S., Ermolayeva A. On stationary characteristics of a multiserver exponential queuing system with reordering of requests. 2021 13th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT). 2021. P. 98–103. DOI: 10.1109/ICUMT54235.2021.9631709.
14. Naumov V.A. On the limiting probabilities of a semi-Markov process. Sovremennie zadachi v tochnih naukah. M.: Universitet drugbi narodov, 1975. P. 35–39. (in Russian).

УДК 004.891.3 **DOI: 10.14529/cmse240105**

КЛАССИФИКАЦИЯ МУЛЬТИМОДАЛЬНЫХ ДАННЫХ О ЗАБОЛЕВАНИЯХ ЛЕГКИХ НА ОСНОВЕ ПОЗДНЕГО СЛИЯНИЯ МОДАЛЬНОСТЕЙ

© 2024 О.Н. Иванова, С. Кумар, М.Л. Цымблер, Е.В. Иванова

Южно-Уральский государственный университет

(454080 Челябинск, пр. им. В.И. Ленина, д. 76)

E-mail: onivanova@susu.ru, kumars@susu.ru,mzym@susu.ru,elena.ivanova@susu.ru

Поступила в редакцию: 21.09.2023

С развитием аппаратных технологий высококачественные рентгеновские снимки стали доступны для диагностики заболеваний легких с помощью специалистов-радиологов. Однако процесс диагностики занимает много времени и зависит от наличия в медицинском учреждении специалистов соответствующего профиля. В то же время информация о пациенте может включать не только рентгеновские снимки грудной клетки разного качества, а также результаты медицинских анализов, записи и предписания врача, сведения о приеме лекарств и другие. В данном исследовании предложена модель классификации легочных заболеваний на основе мультимодальных данных о клинических исследованиях пациентов и рентгенографических изображений. При подготовке данных использованы различные методы генерации искусственных образцов как для изображений, так и для табличных данных о результатах лабораторных исследований. Предложен метод установления соответствия для сгенерированных образцов между модальностями. Предложенная мультимодальная модель имеет архитектуру позднего слияния. Проведены эксперименты на наборах данных с одной и двумя модальностями. Предложенная модель показала точность на 5.5% выше, чем модели, основанные на одной модальности (91.3% против 86.11% на наборе данных из 1 156 пациентов).

Ключевые слова: мультимодальные данные, заболевания легких, глубокое обучение, позднее слияние.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Иванова О.Н., Кумар С., Цымблер М.Л., Иванова Е.В. Классификация мультимодальных данных о заболеваниях легких на основе позднего слияния модальностей // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2024. Т. 13, № 1. С. 74–86. DOI: 10.14529/cmse240105.

Введение

В классификации ВОЗ число болезней, связанных с пневмонией и гриппом, составляет около 200 [1, 2]. Визуализация — один из возможных подходов к диагностике легочной патологии, в котором рентгенограммы используются для выявления инфекций легких [3], однако такая диагностика требует участия эксперта. Недостаточная квалификация врача или аномалии медицинских рентгенографических изображений, выполненных на различном оборудовании, приводит к потере времени при первичной диагностике. В настоящее время глубокие нейронные сети повсеместно внедряются в медицинскую практику. Во многих работах продемонстрирован положительный эффект применения глубоких нейронных сетей и машинного обучения в решении задач автоматизированного распознавания различных патологий, в том числе заболеваний легких [4, 5]. Одна из первых моделей для обнаружения пораженных легочных узлов была разработана в конце 1980-х годов, но оказалась недостаточной для выявления критических заболеваний легких на ранних стадиях. С развитием искусственного интеллекта исследователи провели ряд исследований, направленных на улучшение диагностики заболеваний легких. Однако медицинская экспертиза очень важна для понимания и диагностики критических заболеваний. Методы глубокого

обучения позволили разработать и обучить модель считывать и понимать изображения легких и диагностировать, поражены они или нет. Однако качество любого прогноза модели глубокого обучения в значительной степени зависит от размера и качества данных. Далее приведены недавние исследования, иллюстрирующие состояние исследований в области классификации и прогнозирования заболеваний легких. В работе [6] предложена 3D глубокая сверточная сеть со стратегиями многомасштабного прогнозирования для обнаружения легочных узлов по сегментированным изображениям. Однако в данном исследовании не удалось классифицировать типы заболеваний, подход к прогнозированию применяется только для небольших легочных узелков. Сверточная нейронная сеть предложена в работе [7] для снижения частоты ложноположительных результатов при классификации набора данных о легочных узелках на наборе данных Luna16. Этот метод позволяет анализировать характер изображений компьютерной томографии только для того, чтобы снизить вероятность неправильного диагноза. Более быстрый метод R-CNN был разработан в работах [7, 8] для обнаружения пораженных легочных узлов, а также снижения частоты ложноположительных результатов. Данная модель показала многообещающие результаты для обнаружения объектов. Слияние архитектуры глубокой сверточной сети и двухпутной сети (DPN, Dual Path Network) было предложено в работе [9] для классификации и извлечения признаков конкреций (минеральных образований). Расположение нескольких патчей с фильтром Франги было использовано в работе [10] для повышения производительности обнаружения легочного узла на рентгеновских снимках легких. Тем не менее эта модель показала чувствительность 94% при коэффициенте ложноположительных результатов 15.1. Применение алгоритмов искусственного интеллекта (ИИ) в классификации легочных заболеваний было предложено в работе [11] с учетом современного уровня классификации рентгенографии грудной клетки и ее анализа. В этой работе авторы создали набор данных, известный как ChestX-ray8, в котором 32 717 рентгеновских снимков принадлежат уникальным пациентам. В работе [12] авторы разработали глубокие сверточные сети для проверки результатов по этим данным о легких и добились многообещающих результатов. Набор данных ChestX-ray8 также адаптирован для использования для многоклассовой классификации заболеваний легких [12]. В работе [12] была предложена модель глубокого обучения для прогнозирования рака легких и пневмонии, предлагающая два метода глубокого обучения. Первоначально они использовали модифицированную модель AlexNet для диагностики рентгенографии грудной клетки. Более того, в модифицированной модели AlexNet был реализован метод опорных векторов (SVM, Support Vector Machine) при решении задачи классификации [13, 14].

В данной работе нами были решены следующие задачи, направленные на получение эффективного инструмента для борьбы с заболеваниями легких, планирования и принятия ответных мер для общественного здравоохранения в Челябинской области и других регионах России:

- провести эксперименты по достижению желаемой точности бинарной классификации наличия заболевания легких на разработанном наборе данных при обработке отдельных модальностей с помощью обучения различных предобученных моделей;
- разработать метод установления соответствия искусственно сгенерированных данных разных модальностей при позднем слиянии;
- разработать архитектуру и реализовать многомодальную модель классификации легочных заболеваний на основе позднего слияния;

- сравнить правильность предложенной многомодальной модели с результатами классификации на одиночных модальностях.

Статья организована следующим образом. В разделе 1 описан подход к слиянию мультимодальных данных. Раздел 2 содержит краткое описание набора данных, использованного в исследовании. В разделе 3 рассмотрены методы, с помощью которых проведено исследование. Результаты исследования и их обсуждение приведены в разделе 4. Заключение подводит итоги исследования.

1. Слияние мультимодальных данных

Слияние является одной из главных задач и вызовов, которые стоят перед исследователем мультимодальных данных. Принятие решений в процессе детектирования медицинских аномалий, классификации заболеваний и прогнозе лечения основывается на изучении различных источников лабораторных исследований — рентгенологических снимков, результатов исследования крови, томографических и многих других типов исследований. Многомодальные модели позволяют объединить данные разной природы различными способами: конкатенация, построение метамодели, обучающейся после и во время добавления новых модальностей, мэппинг и др. Многомодальные данные — это данные, которые представлены в нескольких модальностях: текст, изображения, аудио и видео. Слияние многомодальных данных — это процесс объединения данных из разных модальностей для получения более полной и точной информации. Существует три основных подхода к слиянию многомодальных данных: раннее, позднее и промежуточное. При раннем слиянии входные данные из разнородных источников объединяются перед обработкой сетью глубокого обучения. Это самый простой подход, но он может быть неэффективен, если модальности неоднородны. При позднем слиянии сначала обучаются отдельные модели для каждой модальности. Затем результаты каждой модели объединяются для принятия решения. Этот подход более эффективен, чем раннее слияние, для обработки разнородных модальностей. При промежуточном слиянии данные из разных модальностей объединяются на промежуточных этапах обработки, принимая во внимание скрытые представления (emmbe) не только последних, но и внутренних слоев сетей различных модальностей. Этот подход требует значительных усилий по проработке архитектуры моделей и предназначен для обработки сильно разнородных модальностей.

2. Набор данных

Хорошо подготовленный, четко маркированный набор данных является ключевым фактором успеха высокоточного прогнозирования и классификации заболеваний легких. Набор данных содержит большой набор рентгенограмм и клинических данных об около 65 000 случаев госпитализации. Информационный массив содержит сводную информацию о госпитализациях пациентов, установленных заболеваниях, назначенных курсах лечения и результатах терапевтического воздействия. Рентгенографические исследования, выполненные в рамках одной или нескольких госпитализаций, вместе с параметрами гемограммы, биологической плазмы и тканевой жидкости, включены в категории лабораторной биохимической аналитики набора данных. Всего имеется 1 630 категорий различных лабораторных испытаний. Для экспедитивного изучения нами была взята небольшая часть набора данных из 144 пациентов с диагностированными легочными заболеваниями без сопутствующих заболеваний других категорий и 145 пациентов контрольной группы, у которых были ди-

агностированы заболевания, не связанные с легочными патологиями, а именно нарушения сна. Каждый пациент был помечен, соответственно, как больной или здоровый. Среди более чем 18 000 пациентов с заболеваниями легких по кодам ICD-9 480-508 были отобраны 144 пациента, у которых не были диагностированы другие сопутствующие заболевания, поскольку последние могли повлиять на назначения, курс лечения и рентгеновские снимки. Поскольку каждый пациент, страдавший заболеваниями легких, скорее всего, проходил рентгенологическое обследование несколько раз, нами были рассмотрены только первые снимки, которые были сделаны при поступлении пациента в больницу. Изучение только первых снимков является целесообразным, поскольку во время лечения состояние легких меняется, т.к. они поддавались воздействию назначенных лекарств, а картина симптомов часто становится нечеткой или полностью исчезает. Кроме того, снимки были сделаны на разном оборудовании, в разном качестве, с разным разрешением и в разных плоскостях. Проблемы обработки качества изображений решены нами на этапе предварительной обработки данных, и при первоначальном отборе пациентов были взяты только пациенты с фронтальным положением грудной клетки во время рентгенологического обследования. Пациент мог находиться в положении стоя или лежа, но его грудная клетка была повернута непосредственно к сканеру. У некоторых пациентов было диагностировано несколько заболеваний легких одновременно. Среди пациентов без диагностированных заболеваний легких было отобрано 145 пациентов с заболеваниями, которые с наименьшей степенью вероятности повлияли на типичные изменения медицинских показателей пациентов с заболеваниями легких. Нами рассмотрены пациенты с единичными заболеваниями из группы 327.00–327.59, согласно коду ICD-9-CM, в основном это бессонница и расстройства сна. После отбора пациентов в соответствии с вышеуказанными критериями был проведен анализ клинических показателей. В соответствии с диагностированным заболеванием пациенты проходили различные клинические обследования. Существовало не менее 200 записей результатов по клиническим показателям для каждого пациента. При этом ряд показателей измерялся несколько раз, как в разные периоды госпитализации, так и во время одной госпитализации, чтобы отобразить динамику изменения значений. В связи с этим нами оставлены только те результаты клинических испытаний, которые соответствовали моменту их первого определения. В среднем на каждого пациента приходилось около 65 лабораторных исследований. Сами клинические испытания варьировались от пациента к пациенту в зависимости от особенностей заболеваний и назначенного лечения. Среди других клинических данных в исходном наборе данных есть информация о названиях лекарств и их дозах, назначаемых пациентам. Поскольку они не влияют на процесс бинарной классификации, а в большей степени касаются прогнозирования течения заболеваний, в настоящем исследовании эти данные не были приняты во внимание. Наиболее распространенными диагнозами заболеваний легких, выявленными у более чем 20 пациентов из набора данных, являются пневмония, в том числе инфекционная, хроническая обструкция дыхательных путей, обструктивный хронический бронхит и астма. Среди наиболее распространенных лабораторных тестов, назначенных и проведенных более чем 60 пациентам из набора данных при их первом поступлении, были следующие анализы: гемоглобин, креатинин, гематокрит, уровень глюкозы в крои, кровь, глюкоза и билирубин в моче, аланинаминотрансфераза (АЛТ) и аспаратаминотрансфераза (АСТ).

3. Методы

3.1. Предварительная обработка изображений

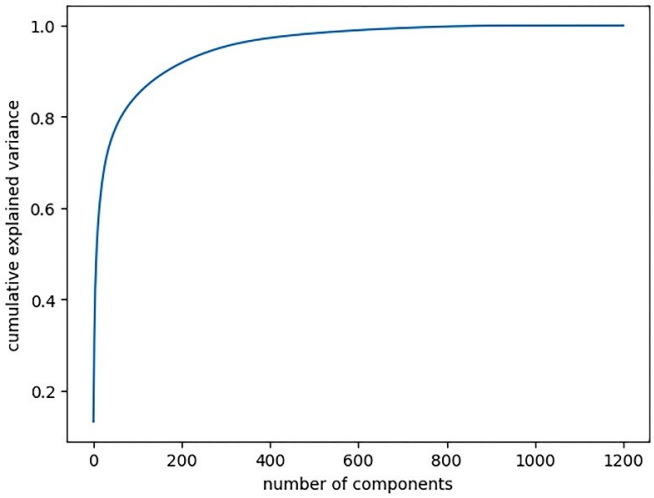


Рис. 1. Объясненное отношение дисперсии к количеству компонентов

Поскольку рентгеновские снимки были выполнены на различном оборудовании, необходимо было преобразовать изображения к единому формату. Все изображения приведены нами к разрешению 64×64 и применена аугментация с помощью следующих методов: горизонтальный поворот, добавление шума (коэффициент 0.2) и преобразование изображений в монохромное без оттенков серого. Применение других методов аугментации представляется нецелесообразными. Большое число методов аугментации, как правило, применяется при отсутствии возможности существенно увеличить мощность набора данных. Мы осознанно сократили имеющийся большой набор данных до небольшого, чтобы показать преимущества многомодального глубокого обучения. При необходимости можно полностью отказаться от аугментации как метода предобработки изображений. На следующем этапе применяется мэппинг для выбора наиболее значимых признаков изображения с помощью методов главных компонент (PCA, Principal Component Analysis) и рекурсивного подавления признаков (RFE, Recursive Feature Elimination). Рисунок 1 отражает отношение выбранных главных компонент изображения к кумулятивному размеру объясненной дисперсии данных. Алгоритм определил 117 определяющих признаков, которые в дальнейшем использовались как порог при запуске метода RFE.

3.2. Предварительная обработка клинических данных

В использованном наборе данных представлены записи о разнообразных лабораторных тестах, назначенных для пациентов с различными заболеваниями. Необходимо было выделить группу лабораторных тестов, которые были назначены и проведены как пациентам с заболеваниями легких, так и пациентам с расстройствами сна. Из 1 630 значений всех возможных медицинских анализов и исследований необходимо было взять приемлемое меньшее количество для решения задачи бинарной классификации. Визуализация значений, выходящих за пределы нормальных диапазонов, представлена на рис. 2 желтым цветом. В верхней строке представлены данные для здоровых пациентов, в нижней строке — для пациентов с легочными заболеваниями.

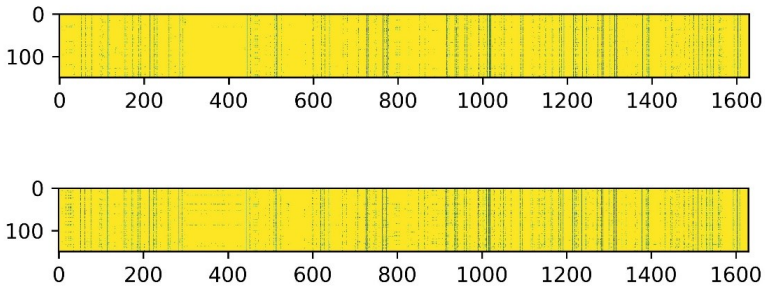


Рис. 2. Клинические анализы: нормальный (желтый) и патологический (зеленый)

Все лабораторные исследования используемого набора данных и, соответственно, их показатели, разделены на четыре группы: 1) назначенные практически всем пациентам (стандартные общие анализы крови и др.); 2) являющиеся специфическими для пациентов с подозрением на заболевание легких; 3) являющиеся специфическими для пациентов с подозрением на заболевания, приводящими к нарушениям сна; 4) те, которые встречаются у пациентов обеих групп эпизодически, всего в одном или нескольких случаях. Соответственно, исследования последней группы были исключены из набора данных. Была произведена операция уплотнения матрицы значений лабораторных исследований и количество параметров снижено до 521.

Для того чтобы расширить набор данных, нами выполнена генерация искусственных образцов (sample) лабораторных исследований. Такой шаг является часто распространенным приемом обработки медицинских клинических данных в связи со сложностью сбора данных. Обычно, для такой генерации табличных данных используются разнообразные методы, например, зашумляющий автоэнкодер, генеративные состязательные сети с использованием метрики Вассерштайна (WGAN, Wasserstein Generative Adversarial Nets), портал MDClone.com и др. В будущих исследованиях мы планируем вернуться к вопросу обработки исходного набора данных из 6 500 записей о пациентах для проверки работы модели на больших данных. В данном исследовании мы решили сделать ограничение по количеству реальных данных и сгенерировать такой же объем данных, какой был сгенерирован для рентгеновских снимков, — 1 156. Нами использован автоэнкодер, который зашумлял исходные данные о значениях клинических исследований пациентов, а затем восстанавливал их, получая похожие результаты. Соответствие между сгенерированными данными клинических испытаний и синтетическими рентгенограммами устанавливалось согласно следующему правилу.

Пусть исходный набор данных описывается следующим образом: $P = \{CXR; EMR\}$, где CXR — рентгенологический снимок грудной клетки; EMR — табличные данные о результатах лабораторных исследований пациента. Тогда каждый пациент описывается как $P_i = CXR_i;\ EMR_i$, где $i \in [1..289]$. Набор сгенерированных с помощью аугментации рентгенологических снимков может быть описан как

$$CXR_i = \{CXR_{i1};\ CXR_{i2};\ CXR_{i3}\}. \quad (1)$$

Набор синтезированных данных о лабораторных исследованиях может быть представлен как

$$EMR_i = \{EMR_{i1};\ EMR_{i2};\ EMR_{i3}\}. \tag{2}$$

Новый образец (пациент) объединяет информацию о снимке и анализах следующим образом:

$$P_{ij} = \{CXR_{ij};\ EMR_{ij}\}, \tag{3}$$

где $i \in [1..3]$, поскольку применены три метода аугментации и выполнены три запуска зашумляющего автоэнкодера. В результате набор синтезированных пациентов может быть описан как

$$P_j = \{P_{i1};\ P_{i2};\ P_{i3}\}. \tag{4}$$

Наконец, расширенный набор данных включил в себя исходных и сгенерированных пациентов: $N = P \cup P_j$. Такой прием может применяться к набору данных малой мощности в любой предметной области, где имеется необходимость в обработке разных модальностей.

3.3. Мультимодальная модель позднего слияния

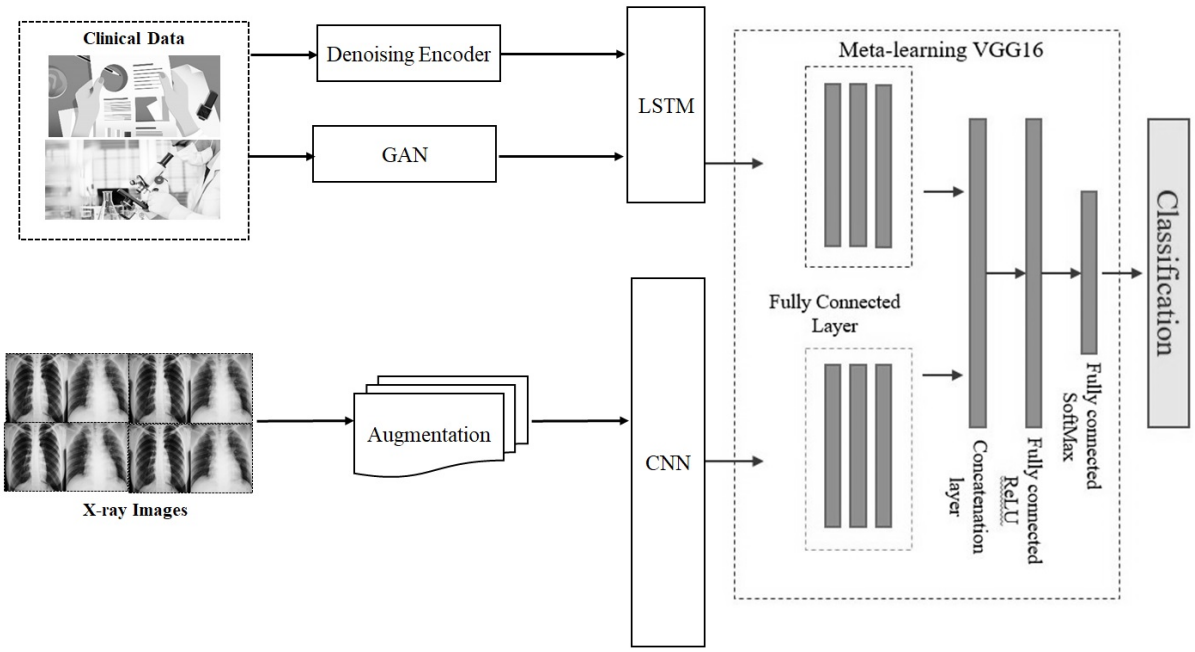


Рис. 3. Модель позднего слияния многомодальных данных для классификации заболеваний легких

Общая схема мультимодальной модели позднего слияния представлена на рис. 3. На первом этапе исследования применены различные методы предобработки графических изображений: метод рекурсивного подавления признаков (RFE), мэппирующий метод главных компонент (PCA) с порогом 0.5 и их комбинация. Изображения, полученные после применения первого из перечисленных методов, оказались наиболее удачным выбором для последующего применения в обучении нейронной сети. Нами использовался метод опорных векторов (SVM) для выравнивания объектов различных модальностей. Завершение обучения по объединенной карте признаков проводилось с учетом соответствия клинических данных рентгеновским изображениям. Для каждой комбинации параметров модель

запускалась не менее 5 раз, чтобы получить средние результаты в условиях случайного выборочного разделения данных на обучающие (60%), валидационные (20%) и тестовые (20%) наборы. Обучение включало раннюю остановку, чтобы предотвратить переобучение. Модель запускалась с различным размером пакета (batch size): от 2 до 10. Меньший размер пакета давал меньший эффект флуктуации точности в конце эпох обучения. Для максимального нивелирования этого эффекта была использована формула (5) адаптивного размера пакета, предложенная нами в работе [15].

$$b_i = b_{i-1} + \frac{N_{i-1} \mod b_i}{\log N_{i-1}}, \tag{5}$$

где b_i — размер пакета в текущую эпоху обучения, b_{i-1} — размер пакета в предыдущую эпоху обучения, N_{i-1} — размер набора данных в предыдущую эпоху обучения. Процесс обучения проводился на 50, 100 и 200 эпохах с ранней остановкой при наличии признаков переобучения. Модель Attention учитывала два последних временных шага в качестве входных характеристик, помогающих понять недавнюю тенденцию в обучении. Для количественной оценки разницы между прогнозируемыми и фактическими значениями использовалась функция потерь со среднеквадратичной ошибкой. Оптимизатор Adam был использован для эффективного обновления весов модели. В качестве параметра градиентного спуска была выбрана скорость обучения, равная 0.01, что обеспечивало превалирование лучшей точности по сравнению со скоростью работы модели. Оценка точности проводилась неоднократно. Эксперименты проводились не менее пяти раз с контролем доверительного интервала в 10% при разделении набора данных на обучающую, валидационную и тестовую выборки. Параметры контролировались во время новых обучающих запусков, с различными значениями генератора случайных состояний, различными значениями пакета и различным количеством эпох.

3.4. Метрики, используемые при оценке работы модели

Для расчета точности используется следующая формула:

$$Accuracy = \frac{\text{True Sick Patients} + \text{True Healthy Patients}}{\text{Number of Patients}}, \tag{6}$$

где True Sick Patients — правильно классифицированные пациенты, имеющие заболевания легких; True Healthy Patients — правильно классифицированные пациенты, не имеющие заболеваний легких; Number of Patients — общее количество пациентов в наборе данных. Модель оптимизировала целевую функцию точности классификации на основе достижения наименьшей среднеквадратичной ошибки в подмножестве валидационных обучающих данных:

$$MSE = \frac{1}{N_{\text{val}}} \sum (y_i - y_{i_{\text{predicted}}})^2, \tag{7}$$

где y_i — i-й пациент из валидационной подвыборки набора данных; $y_{i_{\text{predicted}}}$ — предсказание модели классификации для i-го пациента из валидационной подвыборки набора данных; N_{val} — количество пациентов в валидационной подвыборке набора данных.

4. Результаты и обсуждение

В этом разделе исследована производительность предлагаемой модели при различных параметрах запуска. Сравнение результатов обучения с помощью моделей DenseNet121,

DenseNet169 и ResNet50 на нашем наборе данных показало, что модель DenseNet169 дает наилучшую точность. Модель Attention, по сути являющаяся трансформером с реализованным механизмом самовнимания, предоставляемым фреймворком Keras, показала наилучшие результаты для лабораторных данных. Используя показатель точности, мы сравнили модели с одной модальностью и предложенную модель позднего слияния при обучении на многомодальном наборе данных, который включает информацию об исходных 289 пациентах, а также набор данных с включенными сгенерированными клиническими данными и дополненными рентгеновскими изображениями. Неизменными условиями экспериментов были следующие: фиксированный размер партии, метод генерации искусственных клинических данных — автоэнкодер. Данные представлены в таблице ниже в порядке возрастания точности. В исследуемой задаче классы сбалансированы, поэтому можно не использовать большое количество метрик. Обучение многомодальной модели запускалось не менее пяти раз. В экспериментах с адаптивным размером пакета мы констатировали одновременное повышение точности и увеличение потерь. Это могло означать, что наша модель становилась лучше с точки зрения точности при любом установленном нами пороговом значении. Этот эффект станет предметом дальнейших исследований. В целом, все методы показывают разумную точность. Однако использование многомодальной модели повышает точность классификации как минимум на 2.89% по сравнению с лучшим результатом одномодальной модели на таких же данных.

Таблица 1. Сравнение точности одномодальных и многомодальных методов классификации

Метод	***Точность***	***Потери***
Одна модальность, снимки, 289 пациентов	87.45	4.47
Одна модальность, снимки, 289 пациентов	85.79	5.36
Многомодальная модель позднего слияния, 289 пациентов	90.34	2.17
Одна модальность, снимки, 1156 пациентов	86.11	3.70
Одна модальность, лабораторные анализы, 1 156 пациентов	88.15	4.06
Многомодальная модель позднего слияния, 1 156 пациентов	91.30	1.52

Заключение

В данном исследовании предложена архитектура многомодальной модели классификации легочных заболеваний, основанная на обработке рентгеновских снимков грудной клетки и клинических данных лабораторных исследований больных. Предлагаемая модель является альтернативой стандартному подходу обработки данных одиночной модальности. В ходе исследования имеющийся набор данных был дополнен искусственно сгенерированными данными с рентгенологическими снимками пациентов, зашумленных автоэнкодером, и искусственно сгенерированными табличными данными о результатах медицинских лабораторных анализов. Также предложен алгоритм установления соответствия между искусственно сгенерированными данными различной модальности. Выполнено обучение нескольких моделей на обеих модальностях и выбрана предобученная модель DenseNet169 для обработки рентгенологических снимков легких и модель Attention, являющаяся транформером, для обработки двумерных данных. После обучения выбранных моделей на отдельных модально-

стях использован метод позднего слияния для формирования метамодели мультимодальных данных. Для установления эффективности предложенной мультимодальной модели проведена серия экспериментов на исходном и искусственно расширенном наборах данных, которые показали, что модель имеет более высокую точность классификации в сравнении с традиционными методами.

Исследование выполнено при поддержке регионального гранта Российского научного фонда № 23-21-10009.

Литература

1. 2019 I.-1.V. Chapter X. Diseases of the respiratory system (J00-J99). 2019. URL: https://icd.who.int/browse10/2019/en (дата обращения: 27.10.2019).
2. 1998 I.-9.V. Diseases of the respiratory system (460-519) Chapter Pneumonia and influenza (480–488). 2019. URL: https://www2.gov.bc.ca/assets/gov/health/practitioner-pro/medical-services-plan/diag-codes_respiratory.pdf (дата обращения: 27.10.2019).
3. Sen I., Hossain M.I., Shakib M.F.H., *et al.* In Depth Analysis of Lung Disease Prediction Using Machine Learning Algorithms // Communications in Computer and Information Science. 2020. Vol. 1241. DOI: 10.1007/978-981-15-6318-8_18.
4. Bharati S., Podder P., Mondal M.R.H. Hybrid deep learning for detecting lung diseases from X-ray images // Informatics in Medicine Unlocked. 2020. Vol. 20, no. 100391. DOI: 10.1016/j.imu.2020.100391.
5. Mustafa E., Selim A. Detection of lung disorders using embedded and wrapper feature selection methods // Kahramanmara¸Sutcu Imam Universitesi Muhendislik Bilimleri Dergisi. 2022. Vol. 25, no. 100391. P. 452–460. DOI: 10.17780/ksujes.1138377.
6. Gu Y., Lu X., Yang L., *et al.* Automatic lung nodule detection using a 3D deep convolutional neural network combined with a multi-scale prediction strategy in chest CTs // Computers in Biology and Medicine. 2018. Vol. 103. P. 220–231. DOI: 10.1016/j.compbiomed.2018.10.011.
7. Setio A.A.A., Traverso A., de Bel T., *et al.* Validation, comparison, and combination of algorithms for automatic detection of pulmonary nodules in computed tomography images: The LUNA16 challenge // Medical Image Analysis. 2017. Vol. 42. P. 1–13. DOI: 10.1016/j.media.2017.06.015.
8. Zhu W., Liu C., Fan W., Xie X. DeepLung: Deep 3D Dual Path Nets for Automated Pulmonary Nodule Detection and Classification // 2018 IEEE Winter Conference on Applications of Computer Vision, WACV 2018, Lake Tahoe, NV, USA, March 12-15, 2018. IEEE Computer Society, 2018. P. 673–681. DOI: 10.1109/WACV.2018.00079.
9. Kong W., Hong J., Jia M., *et al.* YOLOv3-DPFIN: A Dual-Path Feature Fusion Neural Network for Robust Real-Time Sonar Target Detection // IEEE Sensors Journal. 2020. Vol. 20, no. 7. P. 3745–3756. DOI: 10.1109/JSEN.2019.2960796.

10. Ronneberger O., Fischer P., Brox T. U-Net: Convolutional Networks for Biomedical Image Segmentation // Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015 - 18th International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III. Vol. 9351 / ed. by N. Navab, J. Hornegger, W.M.W. III, A.F. Frangi. Springer, 2015. P. 234–241. Lecture Notes in Computer Science. DOI: 10.1007/978-3-319-24574-4_28.

11. Kallianos K., Mongan J., Antani S., *et al.* How far have we come? Artificial intelligence for chest radiograph interpretation // Clinical Radiology. 2019. Vol. 74, no. 5. P. 338–345. DOI: 10.1016/j.crad.2018.12.015.

12. Bhandary A., Prabhu G.A., Rajinikanth V., *et al.* Deep-learning framework to detect lung abnormality – A study with chest X-Ray and lung CT scan images // Pattern Recognition Letters. 2020. Vol. 129. P. 271–278. DOI: 10.1016/j.patrec.2019.11.013.

13. Bharati S., Podder P., Paul P.K. Lung Cancer Recognition and Prediction According to Random Forest Ensemble and RUSBoost Algorithm Using LIDC Data // Int. J. Hybrid Intell. Syst. 2019. Vol. 15, no. 2. P. 91–100. DOI: 10.3233/HIS-190263.

14. Behzadi-khormouji H., Rostami H., Salehi S., *et al.* Deep learning, reusable and problem-based architectures for detection of consolidation on chest X-ray images // Computer Methods and Programs in Biomedicine. 2020. Vol. 185. P. 105162. DOI: 10.1016/j.cmpb.2019.105162.

15. Kumar S., Ivanova O., Melyokhin A., Tiwari P. Deep-learning-enabled multimodal data fusion for lung disease classification // Informatics in Medicine Unlocked. 2023. Vol. 42. P. 101367. DOI: 10.1016/j.imu.2023.101367.

Иванова Ольга Николаевна, к.п.н., доцент, кафедра системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

Кумар Сэчин, PhD, в.н.с., лаборатория больших данных и машинного обучения, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

Цымблер Михаил Леонидович, д.ф.-м.н., доцент, кафедра системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

Иванова Елена Владимировна, к.ф.-м.н., доцент, кафедра системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

DOI: 10.14529/cmse240105

CLASSIFICATION OF MULTIMODAL LUNG DISEASE DATA BASED ON LATE FUSION OF MODALITIES

© 2024 O.N. Ivanova, S. Kumar, M.L. Zymbler, E.V. Ivanova

South Ural State University (pr. Lenina 76, Chelyabinsk, 454080 Russia)

E-mail: onivanova@susu.ru, kumars@susu.ru,mzym@susu.ru,elena.ivanova@susu.ru

Received: 21.09.2023

With the development of technology, high-quality X-rays have become available for the diagnosis of lung diseases with the help of radiologists. However, the diagnostic process takes a lot of time and depends on the availability of specialists in a medical institution. At the same time, patient information may include not only chest X-rays of different quality, but also the results of medical tests, doctor's notes and prescriptions, information about taking medications, and others. In this study, we propose a model for the classification of pulmonary diseases based on multimodal data on clinical studies of patients and radiographic images. When preparing the data, we used various methods of generating artificial samples for both images and tabular data on the results of laboratory studies. We have proposed a method for establishing a correspondence for generated samples between modals. The proposed multimodal model has a late fusion architecture. We conducted experiments on datasets with one modality and two modalities. Our model showed accuracy 5.5% higher than models based on single-modality (91.3% vs. 86.11% on a dataset of 1,156 patients).

Keywords: multimodal data, lung diseases, deep learning, late fusion.

FOR CITATION

Ivanova O.N., Kumar S., Zymbler M.L., Ivanova E.V. Classification of Multimodal Lung Disease Data Based on Late Fusion of Modalities. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2024. Vol. 13, no. 1. P. 74–86. (in Russian) DOI: 10.14529/cmse240105.

References

1. 2019 I.-1.V. Chapter X. Diseases of the respiratory system (J00-J99). 2019. URL: `https://icd.who.int/browse10/2019/en` (accessed: 27.10.2019).
2. 1998 I.-9.V. Diseases of the respiratory system (460-519) Chapter Pneumonia and influenza (480–488). 2019. URL: `https://www2.gov.bc.ca/assets/gov/health/practitioner-pro/medical-services-plan/diag-codes_respiratory.pdf` (accessed:1998).
3. Sen I., Hossain M.I., Shakib M.F.H., *et al.* In Depth Analysis of Lung Disease Prediction Using Machine Learning Algorithms. Communications in Computer and Information Science. 2020. Vol. 1241. DOI: `10.1007/978-981-15-6318-8_18`.
4. Bharati S., Podder P., Mondal M.R.H. Hybrid deep learning for detecting lung diseases from X-ray images. Informatics in Medicine Unlocked. 2020. Vol. 20, no. 100391. DOI: `10.1016/j.imu.2020.100391`.
5. Mustafa E., Selim A. Detection of lung disorders using embedded and wrapper feature selection methods. Kahramanmara¸Sutcu Imam Universitesi Muhendislik Bilimleri Dergisi. 2022. Vol. 25, no. 100391. P. 452–460. DOI: `10.17780/ksujes.1138377`.

6. Gu Y., Lu X., Yang L., *et al.* Automatic lung nodule detection using a 3D deep convolutional neural network combined with a multi-scale prediction strategy in chest CTs. Computers in Biology and Medicine. 2018. Vol. 103. P. 220–231. DOI: 10.1016/j.compbiomed.2018.10.011.

7. Setio A.A.A., Traverso A., de Bel T., *et al.* Validation, comparison, and combination of algorithms for automatic detection of pulmonary nodules in computed tomography images: The LUNA16 challenge. Medical Image Analysis. 2017. Vol. 42. P. 1–13. DOI: 10.1016/j.media.2017.06.015.

8. Zhu W., Liu C., Fan W., Xie X. DeepLung: Deep 3D Dual Path Nets for Automated Pulmonary Nodule Detection and Classification. 2018 IEEE Winter Conference on Applications of Computer Vision, WACV 2018, Lake Tahoe, NV, USA, March 12-15, 2018. IEEE Computer Society, 2018. P. 673–681. DOI: 10.1109/WACV.2018.00079.

9. Kong W., Hong J., Jia M., *et al.* YOLOv3-DPFIN: A Dual-Path Feature Fusion Neural Network for Robust Real-Time Sonar Target Detection. IEEE Sensors Journal. 2020. Vol. 20, no. 7. P. 3745–3756. DOI: 10.1109/JSEN.2019.2960796.

10. Ronneberger O., Fischer P., Brox T. U-Net: Convolutional Networks for Biomedical Image Segmentation. Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015 - 18th International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III. Vol. 9351 / ed. by N. Navab, J. Hornegger, W.M.W. III, A.F. Frangi. Springer, 2015. P. 234–241. Lecture Notes in Computer Science. DOI: 10.1007/978-3-319-24574-4_28.

11. Kallianos K., Mongan J., Antani S., *et al.* How far have we come? Artificial intelligence for chest radiograph interpretation. Clinical Radiology. 2019. Vol. 74, no. 5. P. 338–345. DOI: 10.1016/j.crad.2018.12.015.

12. Bhandary A., Prabhu G.A., Rajinikanth V., *et al.* Deep-learning framework to detect lung abnormality – A study with chest X-Ray and lung CT scan images. Pattern Recognition Letters. 2020. Vol. 129. P. 271–278. DOI: 10.1016/j.patrec.2019.11.013.

13. Bharati S., Podder P., Paul P.K. Lung Cancer Recognition and Prediction According to Random Forest Ensemble and RUSBoost Algorithm Using LIDC Data. Int. J. Hybrid Intell. Syst. 2019. Vol. 15, no. 2. P. 91–100. DOI: 10.3233/HIS-190263.

14. Behzadi-khormouji H., Rostami H., Salehi S., *et al.* Deep learning, reusable and problem-based architectures for detection of consolidation on chest X-ray images. Computer Methods and Programs in Biomedicine. 2020. Vol. 185. P. 105162. DOI: 10.1016/j.cmpb.2019.105162.

15. Kumar S., Ivanova O., Melyokhin A., Tiwari P. Deep-learning-enabled multimodal data fusion for lung disease classification. Informatics in Medicine Unlocked. 2023. Vol. 42. P. 101367. DOI: 10.1016/j.imu.2023.101367.

СВЕДЕНИЯ ОБ ИЗДАНИИ

Научный журнал «Вестник ЮУрГУ. Серия «Вычислительная математика и информатика» основан в 2012 году.

Учредитель — Федеральное государственное автономное образовательное учреждение высшего образования «Южно-Уральский государственный университет» (национальный исследовательский университет).

Главный редактор — Л.Б. Соколинский.

Свидетельство о регистрации ПИ ФС77-57377 выдано 24 марта 2014 г. Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций.

Журнал включен в Реферативный журнал и Базы данных ВИНИТИ; индексируется в библиографической базе данных РИНЦ. Журнал размещен в открытом доступе на Всероссийском математическом портале MathNet. Сведения о журнале ежегодно публикуются в международной справочной системе по периодическим и продолжающимся изданиям «Ulrich's Periodicals Directory».

Решением Президиума Высшей аттестационной комиссии Министерства образования и науки Российской Федерации журнал включен в «Перечень рецензируемых научных изданий, в которых должны быть опубликованы основные научные результаты на соискание ученой степени кандидата наук, на соискание ученой степени доктора наук» по научным специальностям и соответствующим им отраслям науки: 1.2.3 – Теоретическая информатика, кибернетика (физико-математические науки), 2.3.5 – Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей (физико-математические науки).

Подписной индекс научного журнала «Вестник ЮУрГУ», серия «Вычислительная математика и информатика»: 10244, каталог «Пресса России». Периодичность выхода — 4 выпуска в год.

Адрес редакции, издателя: 454080, г. Челябинск, проспект Ленина, 76, Издательский центр ЮУрГУ, каб. 32.

ПРАВИЛА ДЛЯ АВТОРОВ

1. Правила подготовки рукописей и пример оформления статей можно загрузить с сайта серии `https://vestnikvmi.susu.ru`. **Статьи, оформленные без соблюдения правил, к рассмотрению не принимаются.**
2. Адрес редакционной коллегии научного журнала «Вестник ЮУрГУ», серия «Вычислительная математика и информатика»:

   Россия 454080, г. Челябинск, пр. им. В.И. Ленина, 76, ЮУрГУ, кафедра СП,
   зам. главного редактора Цымблеру М.Л.
3. Адрес электронной почты редакции: `vestnikvmi@susu.ru`
4. **Плата с авторов за публикацию рукописей не взимается, и гонорары авторам не выплачиваются.**