МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПРИКЛАДНАЯ ДИСКРЕТНАЯ МАТЕМАТИКА

Научный журнал

2013 №1(19)

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

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

РЕДАКЦИОННАЯ КОЛЛЕГИЯ ЖУРНАЛА
«ПРИКЛАДНАЯ ДИСКРЕТНАЯ МАТЕМАТИКА»

Агибалов Г. П., д-р техн. наук, проф. (председатель); Девянин П. Н., д-р техн. наук, проф. (зам. председателя); Парватов Н. Г., д-р физ.-мат. наук, доц. (зам. председателя); Черемушкин А. В., д-р физ.-мат. наук, чл.-корр. Академии криптографии РФ (зам. председателя); Панкратова И. А., канд. физ.-мат. наук, доц. (отв. секретарь); Алексеев В. Б., д-р физ.-мат. наук, проф.; Бандман О. Л., д-р техн. наук, проф.; Глухов М. М., д-р физ.-мат. наук, академик Академии криптографии РФ; Евдокимов А. А., канд. физ.-мат. наук, проф.; Закревский А. Д., д-р техн. наук, проф., чл.-корр. НАН Беларуси; Колесникова С. И., д-р техн. наук; Костюк Ю. Л., д-р техн. наук, проф.; Логачев О. А., канд. физ.-мат. наук, доц.; Салий В. Н., канд. физ.-мат. наук, проф.; Сафонов К. В., д-р физ.-мат. наук, проф.; Фомичев В. М., д-р физ.-мат. наук, проф.; Чеботарев А. Н., д-р техн. наук, проф.; Шойтов А. М., д-р физ.-мат. наук, чл.-корр. Академии криптографии РФ; Шоломов Л. А., д-р физ.-мат. наук, проф.

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

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

Периодичность выхода журнала: 4 номера в год.

Редактор *Н. И. Шидловская*
Верстка *И. А. Панкратовой*

Подписано к печати 11.03.2013.
Формат $60 \times 84\frac{1}{8}$. Усл. п. л. 13,4. Уч.-изд. л. 15. Тираж 300 экз.

Издательство ТГУ. 634029, Томск, ул. Никитина, 4
Отпечатано в типографии ТПУ.

СОДЕРЖАНИЕ

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ПРИКЛАДНОЙ ДИСКРЕТНОЙ МАТЕМАТИКИ

Биляк И. Б. Оценки числа появлений элементов на отрезках линейных рекуррентных последовательностей 5
Коломеец Н. А. О верхней оценке нелинейности некоторого класса булевых функций с максимальной алгебраической иммунностью 14
Шоломов Л. А. Двоичные представления недоопределённых данных и дизъюнктивные коды 17

МАТЕМАТИЧЕСКИЕ ОСНОВЫ КОМПЬЮТЕРНОЙ БЕЗОПАСНОСТИ

Семенова Н. А. Представление системы семантически осмысленного ролевого управления доступом в виде цветной сети Петри 34
Смольянинов В. Ю. Правила преобразования состояний СУБД ДП-модели 50

ПРИКЛАДНАЯ ТЕОРИЯ ГРАФОВ

Бадеха И. А. Исследование кликовых покрытий рёбер графа 69
Ураков А. Р., Тимеряев Т. В. Алгоритм поиска кратчайших путей для разреженных графов большой размерности 84
Цициашвили Г. Ш., Осипова М. А., Лосев А. С. Асимптотика вероятности связности графа с низконадёжными рёбрами 93

ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ В ДИСКРЕТНОЙ МАТЕМАТИКЕ

Колоколов А. А., Адельшин А. В., Ягофарова Д. И. Исследование задач дискретной оптимизации с логическими ограничениями на основе метода регулярных разбиений 99
Кузнецов А. А., Кузнецова А. С. Быстрое умножение элементов в конечных двупорождённых группах периода пять 110
Мурин Д. М. О верхней границе плотности инъективных векторов 117
СВЕДЕНИЯ ОБ АВТОРАХ 125
АННОТАЦИИ СТАТЕЙ НА АНГЛИЙСКОМ ЯЗЫКЕ 127

CONTENTS

THEORETICAL BACKGROUNDS OF APPLIED DISCRETE MATHEMATICS

Bilyak I. B. Estimates for the number of element appearances in segments of linear recurrent sequences 5

Kolomeec N. A. An upper bound for the nonlinearity of some Boolean functions with maximal possible algebraic immunity 14

Sholomov L. A. Binary representations of underdetermined data and superimposed codes 17

MATHEMATICAL BACKGROUNDS OF COMPUTER SECURITY

Semenova N. A. Representation of semantic role based access control system by a colored Petri net 34

Smolyaninov V. Y. Transformation rules for states in the DBMS DP-model 50

APPLIED GRAPH THEORY

Badekha I. A. Edge clique coverings of graphs 69

Urakov A. R., Timeryaev T. V. All-pairs shortest paths algorithm for high-dimensional sparse graphs 84

Tsitsiashvili G. Sh., Osipova M. A., Losev A. C. Asymptotics for connectivity probability of graph with low reliable arcs 93

COMPUTATIONAL METHODS IN DISCRETE MATHEMATICS

Kolokolov A. A., Adelshin A. V., Yagofarova D. I. Study of discrete optimization problems with logical constraints based on regular partitions 99

Kuznetsov A. A., Kuznetsova A. S. Fast multiplication in finite two-generated groups of exponent five 110

Murin D. M. On the upper bound for the density of any injective vector 117

BRIEF INFORMATION ABOUT THE AUTHORS 125

PAPER ABSTRACTS 127

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ПРИКЛАДНОЙ ДИСКРЕТНОЙ МАТЕМАТИКИ

DOI 10.17223/20710410/19/1

УДК 621.391.1:004.7

ОЦЕНКИ ЧИСЛА ПОЯВЛЕНИЙ ЭЛЕМЕНТОВ НА ОТРЕЗКАХ ЛИНЕЙНЫХ РЕКУРРЕНТНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ

И. Б. Биляк

г. Москва, Россия

E-mail: bil-ib@mail.ru

Рассмотрен некоторый класс тригонометрических сумм от линейных рекуррентных последовательностей. Эти суммы исследуются с использованием метода В. М. Сидельникова. Получены оценки числа появлений элементов на отрезках линейных рекуррент, которые в некоторых случаях уточняют ранее известные результаты.

Ключевые слова: *тригонометрические суммы, линейные рекуррентные последовательности, число появлений элементов.*

Введение

Изучение числа появлений элементов в линейных рекуррентных последовательностях (ЛРП) над кольцами является одной из важных математических задач. Интерес к этой задаче связан прежде всего с построением на основе ЛРП генераторов псевдослучайных чисел, использующих различные способы усложнения аналитического строения линейных рекуррент (см., например, [1]).

Пусть $\mathrm{GF}(q)$ — конечное поле из q элементов, $f(x) = x^m - a_{m-1}x^{m-1} - \ldots - a_1x - a_0$ — реверсивный ($a_0 \neq 0$) неприводимый многочлен степени m над этим полем. Линейной рекуррентной последовательностью над полем $\mathrm{GF}(q)$ с характеристическим многочленом $f(x)$ будем называть последовательность $u = u(0)u(1)u(2)\ldots$ элементов этого поля, удовлетворяющую соотношению

$$u(i+m) = a_0u(i) + a_1u(i+1) + \ldots + a_{m-1}u(i+m-1), \quad i \geqslant 0.$$

Каждая такая ненулевая ЛРП u является чисто периодической последовательностью, при этом её период $T(u)$ равен периоду $T(f)$ многочлена $f(x)$ и делит $q^m - 1$ (см., например, [2]).

Рассмотрим линейные рекуррентные последовательности $u_1, u_2, \ldots, u_r$ с характеристическим многочленом $f(x)$. Назовём эти последовательности линейно независимыми над полем $\mathrm{GF}(q)$, если для всех ненулевых векторов $\bar{c} = (c_1, c_2, \ldots, c_r) \in \mathrm{GF}(q)^r$ последовательность $c_1u_1 + c_2u_2 + \ldots + c_ru_r$ является ненулевой. Обозначим через $N_l(\overline{z}, u_1, \ldots, u_r)$ количество целых чисел $i \in \{0, 1, \ldots, l-1\}$, удовлетворяющих условиям $u_1(i) = z_1$, $u_2(i) = z_2$, …, $u_r(i) = z_r$, где $\overline{z} = (z_1, z_2, \ldots, z_r) \in \mathrm{GF}(q)^r$. Таким образом, величина $N_l(\overline{z}, u_1, \ldots, u_r)$ равна количеству появлений r-граммы $\overline{z}$ на отрезке длины l последовательности векторов, элементы которой имеют вид $(u_1(i), u_2(i), \ldots, u_r(i))$ для всех $i \geqslant 0$.

Всюду в дальнейшем будем считать, что $u_1, u_2, \ldots, u_r$ — линейно независимая система ЛРП над полем $\mathrm{GF}(q)$ с реверсивным неприводимым характеристическим многочленом $f(x)$. В работах В. И. Нечаева [3] и И. Е. Шпарлинского [4] доказано, что для произвольной r-граммы $\overline{z} \in \mathrm{GF}(q)^r$ при всех l, не превосходящих период $T = T(f)$ многочлена $f(x)$, справедлива следующая оценка:

$$\left| N_l(\overline{z}, u_1, \ldots, u_r) - \frac{l}{q^r} \right| \leqslant \frac{q^r - 1}{q^r} q^{\frac{m}{2}} \ln T. \tag{1}$$

Величина l/q^r представляет собой «естественное» среднее значение для количества появлений r-граммы на отрезке длины l в последовательности векторов. Таким образом, неравенство (1) можно рассматривать как верхнюю оценку модуля отклонения количества появлений r-граммы на отрезке ЛРП векторов от средней величины. Из этого неравенства следуют верхняя и нижняя оценки числа $N_l(\overline{z}, u_1, \ldots, u_r)$. При условии $l \geqslant q^{m/2+r} \ln T$ нижняя оценка рассматриваемой величины больше нуля, а верхняя — меньше l, т. е. оценка (1) нетривиальна. Впервые она была получена Н. М. Коробовым в [5] для простого поля и случая, когда ЛРП $u_1, u_2, \ldots, u_r$ — сдвиги одной последовательности. В этой работе впервые был применен аппарат тригонометрических сумм для получения оценок количества появлений элементов в ЛРП.

В. М. Сидельниковым в работе [6] в случае, когда q — простое число, доказано неравенство

$$\left| N_l(\overline{z}, u_1, \ldots, u_r) - \frac{l}{q^r} \right| \leqslant \frac{q^r - 1}{q^r} \left(3(q^m l - l^2)\right)^{\frac{1}{3}}. \tag{2}$$

Оценка (2) является нетривиальной и улучшает оценку (1) при

$$\sqrt{3} q^{(m+3r)/2} \leqslant l \leqslant \frac{1}{24}((m + 3r) \ln q)^3 q^{m/2}.$$

В данной работе доказано, что в условиях, сформулированных для результата (1), и при дополнительном условии $l \leqslant \min\{T/2, T/(T, q-1)\}$ справедлива следующая оценка:

$$\left| N_l(\overline{z}, u_1, \ldots, u_r) - \frac{l}{q^r} \right| \leqslant \frac{q^r - 1}{q^r(q-1)} \left(\frac{3q}{2} \left((q-1)q^m l - l^2\right) \right)^{\frac{1}{3}}. \tag{3}$$

Оценка (3) является нетривиальной и улучшает оценку (1) при значениях l, удовлетворяющих условию

$$\frac{\sqrt{3} q^{\frac{m+3r}{2}}}{\sqrt{h(q)}} \leqslant l \leqslant \frac{h(q)}{3} q^{\frac{m}{2}} \ln^3 T,$$

где $h(q) = 2(q-1)^2/q$. Правые части неравенств (2) и (3) совпадают, если $q = 2$. Оценка (3) улучшает также оценку (2) при $q \geqslant 3$ и

$$\frac{\sqrt{3} q^{\frac{m+3r}{2}}}{\sqrt{h(q)}} \leqslant l \leqslant \frac{10}{13} q^m.$$

Кроме того, при $q \geqslant 3$ правая часть неравенства (3) меньше правой части неравенства (2) в $\sqrt[3]{2(q-1)/q}$ раз.

1. Оценка тригонометрической суммы

Для изучения количества появлений элементов на отрезках ЛРП докажем вспомогательные результаты о тригонометрических суммах. Обозначим через χ аддитивный характер поля $\mathrm{GF}(q)$, определённый на элементах поля равенством $\chi(x) = \exp\{\mathrm{tr}_p^q(x)2\pi i/p\}$, где p — характеристика поля $\mathrm{GF}(q)$; $\mathrm{tr}_p^q(x)$ — функция следа из поля $\mathrm{GF}(q)$ в поле $\mathrm{GF}(p)$; i — мнимая единица. Пусть $L(f(x))$ — множество всех ЛРП над полем $\mathrm{GF}(q)$ с характеристическим многочленом $f(x)$, а $L(f(x))^*$ — множество всех ненулевых ЛРП из множества $L(f(x))$. Период каждой ЛРП из множества $L(f(x))^*$ равен периоду T многочлена $f(x)$ и порядку корня α многочлена $f(x)$ в мультипликативной группе поля $\mathrm{GF}(q^m)$ (см., например, [2]). Определим величину $\nu_l(z,u)$ следующим равенством:

$$\nu_l(z,u) = \sum_{c\in\mathrm{GF}(q)^*} \chi(-cz) \sum_{i=0}^{l-1} \chi(cu(i)),$$

где $\mathrm{GF}(q)^*$ — мультипликативная группа поля $\mathrm{GF}(q)$, а z — произвольный элемент этого поля. Сформулируем и докажем некоторые свойства этой величины.

Лемма 1. Пусть $f(x)$ — реверсивный неприводимый многочлен степени m над полем $\mathrm{GF}(q)$, T — его период, а l — натуральное число, не превосходящее величину $T/(T, q-1)$. Тогда для любого элемента z поля $\mathrm{GF}(q)$ справедливо соотношение

$$\sum_{u\in L(f(x))^*} |\nu_l(z,u)|^2 = \begin{cases} (q-1)q^m l - l^2, & \text{если } z \neq 0, \\ (q-1)(q^m l - (q-1)l^2), & \text{если } z = 0. \end{cases} \tag{4}$$

Доказательство. Левую часть (4) можно представить в виде

$$\sum_{u\in L(f(x))^*} \sum_{b,c\in\mathrm{GF}(q)^*} \chi(-cz)\overline{\chi(-bz)} \sum_{i,j=0}^{l-1} \chi(cu(i))\overline{\chi(bu(j))}, \tag{5}$$

где черта обозначает комплексное сопряжение. Очевидно, что $\overline{\chi(x)} = \chi^{-1}(x) = \chi(-x)$. Воспользуемся этим в (5) и получим

$$\sum_{u\in L(f(x))^*} |\nu_l(z,u)|^2 = \sum_{u\in L(f(x))^*} \sum_{b,c\in\mathrm{GF}(q)^*} \chi(-cz+bz) \textstyle\sum_{i,j=0}^{l-1} \chi(cu(i)-bu(j)). \tag{6}$$

Для нулевой последовательности u из определения величины $\nu_l(z,u)$ следует, что при $z \neq 0$ она принимает значение $-l$, а при $z = 0$ — значение $(q-1)l$. Таким образом, получим

$$\sum_{u\in L(f(x))^*} |\nu_l(z,u)|^2 = \begin{cases} \displaystyle\sum_{u\in L(f(x))} |\nu_l(z,u)|^2 - l^2, & \text{если } z \neq 0, \\ \displaystyle\sum_{u\in L(f(x))} |\nu_l(z,u)|^2 - (q-1)^2 l^2, & \text{если } z = 0. \end{cases} \tag{7}$$

Для любой ЛРП из множества $L(f(x))$ найдётся единственный элемент γ из поля $\mathrm{GF}(q^m)$, такой, что

$$u(i) = \mathrm{tr}_q^{q^m}(\gamma\alpha^i) \text{ для всех } i \geqslant 0,$$

где $\mathrm{tr}_q^{q^m}(x)$ — функция следа из поля $\mathrm{GF}(q^m)$ в поле $\mathrm{GF}(q)$ (см. [2, Теорема 8.24]). Далее, с использованием этого представления знаков ЛРП, из выражения (6) получим

$$\sum_{u\in L(f(x))} |\nu_l(z,u)|^2 = \sum_{b,c\in\mathrm{GF}(q)^*} \chi(-cz+bz) \sum_{i,j=0}^{l-1} \sum_{\gamma\in\mathrm{GF}(q^m)} \chi_m(\gamma(c\alpha^i - b\alpha^j)). \tag{8}$$

где χ_m — аддитивный характер поля $\mathrm{GF}(q^m)$, определённый на элементах этого поля равенством $\chi_m(x) = \chi(\mathrm{tr}_q^{q^m}(x))$. Воспользуемся соотношением ортогональности для характеров и получим

$$\sum_{\gamma\in\mathrm{GF}(q^m)} \chi_m(\gamma(c\alpha^i - b\alpha^j)) = \begin{cases} q^m, & \text{если } c\alpha^i = b\alpha^j, \\ 0, & \text{если } c\alpha^i \neq b\alpha^j. \end{cases} \tag{9}$$

Если для некоторых элементов b и c поля $\mathrm{GF}(q)$ выполняется соотношение $c\alpha^i = b\alpha^j$, то элемент α^{i-j} является элементом поля $\mathrm{GF}(q)$. Далее получим $\alpha^{(i-j)(q-1)} = e$ и, следовательно, $T(f) = \operatorname{ord}\alpha | (j-i)(q-1)$, $T(f)/(T(f), q-1) | (j-i)$. Элементы i и j всегда меньше $T(f)/(T(f), q-1)$, значит, соотношение $c\alpha^i = b\alpha^j$ выполняется тогда и только тогда, когда $i = j$ и $b = c$, откуда, используя (8) и (9), получим

$$\sum_{u\in L(f(x))} |\nu_l(z,u)|^2 = \sum_{c\in\mathrm{GF}(q)^*} \sum_{i=0}^{l-1} q^m = (q-1)q^m l. \tag{10}$$

Для завершения доказательства остаётся подставить равенство (10) в равенство (7). ■

Лемма 2. Пусть u — ЛРП периода T над полем $\mathrm{GF}(q)$, j — целое число из отрезка $\overline{-T, T}$, z — произвольный элемент поля $\mathrm{GF}(q)$. Тогда

$$|\nu_l(z, x^T u) - \nu_l(z, x^{T+j} u)| \leqslant q|j|.$$

Доказательство. Пусть $\chi(x) = \exp\{\mathrm{tr}_p^q(x) 2\pi i/p\}$ для всех элементов x поля $\mathrm{GF}(q)$. Воспользуемся соотношением ортогональности для характеров

$$\sum_{i=0}^{l-1} \sum_{c\in\mathrm{GF}(q)} \chi(c(u(i) - z)) = qN_l(z,u),$$

где $N_l(z,u)$ — число появлений z среди элементов $u(0), u(1), \ldots, u(l-1)$. Так как $|N_l(z, x^T u) - N_l(z, x^{T+j} u)| \leqslant |j|$ для всех $j \in \overline{-T, T}$, то $|\nu_l(z, x^T u) - \nu_l(z, x^{T+j} u)| = q|N_l(z, x^T u) - N_l(z, x^{T+j} u)| \leqslant q|j|$. ■

Лемма 3 [7]. Пусть q — натуральное число, тогда для всех действительных чисел $\nu \geqslant 0$ справедливо неравенство

$$\nu^2 + 2\sum_{j=1}^{[\nu/q]} (\nu - qj)^2 \geqslant \frac{2\nu^3}{3q}.$$

Теорема 1. Пусть $f(x)$ — реверсивный неприводимый многочлен степени m над полем $\mathrm{GF}(q)$, T — его период, u — ненулевая ЛРП над полем $\mathrm{GF}(q)$ с характеристическим многочленом $f(x)$, z — элемент поля $\mathrm{GF}(q)$, а l — натуральное число. Тогда при $l \leqslant \min\{T/2, T/(T, q-1)\}$ справедлива оценка

$$|\nu_l(z,u)| \leqslant \left(\frac{3q}{2}\left((q-1)q^m l - l^2\right)\right)^{\frac{1}{3}}.$$

Доказательство. Рассмотрим в множестве $L(f(x))^*$ ЛРП u_0, для которой величина $\nu = |\nu_l(z, u_0)|$ равна максимально возможному значению. Пусть j — произвольное целое число из интервала $\overline{-[\nu/q], [\nu/q]}$. Через u_j обозначим ЛРП, определённую равенством

$$u_j(i) = u_0(i + T + j) \text{ для всех } i \geqslant 0.$$

Покажем, что среди u_j, где $j \in \overline{-[\nu/q], [\nu/q]}$, нет одинаковых. Если найдутся целые числа j_1 и j_2, такие, что $u_{j_1} = u_{j_2}$, то для корня α многочлена $f(x)$ будет справедливо соотношение $\alpha^{j_1 - j_2} = e$. Это равенство может выполняться лишь при условии, что T делит $j_1 - j_2$. Как показано при доказательстве леммы 2, справедливо соотношение $\nu = |qN_l(z, u_0) - l|$, поэтому $\nu < ql$ и в условиях теоремы справедливы неравенства

$$|j_1 - j_2| \leqslant 2[\nu/q] < 2l \leqslant T.$$

Значит, $u_{j_1} = u_{j_2}$ тогда и только тогда, когда $j_1 = j_2$. Теперь с использованием лемм 1–3 имеем

$$(q-1)q^m l - l^2 \geqslant \sum_{j=-[\nu/q]}^{[\nu/q]} |\nu_l(z, u_j)|^2 \geqslant \nu^2 + 2\sum_{j=1}^{[\nu/q]} (\nu - qj)^2 \geqslant \frac{2\nu^3}{3q}.$$

Окончательно получим

$$|\nu_l(z, u)| \leqslant \nu \leqslant \left(\frac{3q}{2}\left((q-1)q^m l - l^2\right)\right)^{\frac{1}{3}}.$$

Теорема доказана. ■

Заметим, что полученная оценка справедлива для произвольного $z \in \mathrm{GF}(q)$. Далее сформулируем результат, доказанный О. В. Камловским для случая $z = 0$.

Теорема 2 [7]. Пусть $f(x)$ — реверсивный неприводимый многочлен степени m над полем $\mathrm{GF}(q)$, T — его период, u — ненулевая ЛРП над полем $\mathrm{GF}(q)$ с характеристическим многочленом $f(x)$, а l — натуральное число. Тогда при условии $l \leqslant t/2$, где $t = T/(T, q-1)$, справедлива оценка

$$|\nu_l(0, u)| \leqslant \left(\frac{3q}{2}\left(q^m l - (q-1)l^2\right)\right)^{\frac{1}{3}}.$$

2. Оценка числа появлений r-грамм в ЛРП векторов

Сформулируем и докажем основной результат работы.

Теорема 3. Пусть $f(x)$ — реверсивный неприводимый многочлен степени m над полем $\mathrm{GF}(q)$, T — его период, $u_1, u_2, \ldots, u_r$ — линейно независимая система ЛРП над полем $\mathrm{GF}(q)$ с характеристическим многочленом $f(x)$, l — натуральное число, не превосходящее значения $\min\{T/2, T/(T, q-1)\}$, $\overline{z}$ — произвольный элемент множества $\mathrm{GF}(q)^r$. Тогда справедлива следующая оценка:

$$\left|N_l(\overline{z}, u_1, \ldots, u_r) - \frac{l}{q^r}\right| \leqslant \frac{q^r - 1}{q^r(q-1)}\left(\frac{3q}{2}\left((q-1)q^m l - l^2\right)\right)^{\frac{1}{3}}.$$

Доказательство. Пусть $\chi(x) = \exp\{\mathrm{tr}_p^q(x)2\pi i/p\}$ — аддитивный характер поля $\mathrm{GF}(q)$, где p — характеристика поля. Используя соотношения ортогональности для характеров, представим $N_l(\overline{z}, u_1, \ldots, u_r)$ в виде

$$N_l(\overline{z}, u_1, \ldots, u_r) = \sum_{i=0}^{l-1}\prod_{j=1}^{r}\frac{1}{q}\sum_{c_j \in \mathrm{GF}(q)}\chi\left(c_j(u_j(i) - z_j)\right).$$

Тогда справедливо соотношение

$$N_l(\overline{z}, u_1, \ldots, u_r) = \frac{1}{q^r}\sum_{i=0}^{l-1}\sum_{(c_1, c_2, \ldots, c_r) \in \mathrm{GF}(q)^r}\chi\left(\sum_{j=1}^{r} c_j(u_j(i) - z_j)\right). \tag{11}$$

1) Пусть $\overline{z} = \overline{0}$. Тогда воспользуемся результатом теоремы 2:

$$\left| N_l(\overline{0}, u_1, \ldots, u_r) - \frac{l}{q^r} \right| \leqslant \frac{q^r - 1}{q^r(q-1)} \left(\frac{3q}{2} \left(q^m l - (q-1)l^2 \right) \right)^{\frac{1}{3}}.$$

2) Пусть далее $\overline{z} \neq \overline{0}$. Для каждого вектора $\bar{c} = (c_1, c_2, \ldots, c_r)$ через $u_{\bar{c}}$ обозначим такую последовательность, элементы которой определены равенствами

$$u_{\bar{c}}(i) = \sum_{j=1}^{r} c_j u_j(i) \quad \text{для всех} \quad i \geqslant 0,$$

и обозначим

$$z_{\bar{c}} = \sum_{j=1}^{r} c_j z_j.$$

Последовательность $u_{\bar{c}}$ — ЛРП с характеристическим многочленом $f(x)$, причём из условия линейной независимости системы u_1, u_2, …, u_r следует, что для всех $\bar{c} \neq \bar{0}$ ЛРП $u_{\bar{c}}$ является ненулевой. В равенстве (11) выделим слагаемое, соответствующее нулевому набору $(c_1, c_2, \ldots, c_r)$, тогда получим соотношение

$$N_l(\overline{z}, u_1, \ldots, u_r) = \frac{l}{q^r} + \frac{1}{q^r} \sum_{i=0}^{l-1} \sum_{\bar{c} \in \mathrm{GF}(q)^r \setminus \{\bar{0}\}} \chi(u_{\bar{c}}(i) - z_{\bar{c}}).$$

Для любого $a \in \mathrm{GF}(q)^*$справедливо соотношение $au_{\bar{c}} = u_{a\bar{c}}$, $az_{\bar{c}} = z_{a\bar{c}}$, следовательно, получим

$$N_l(\overline{z}, u_1, \ldots, u_r) - \frac{l}{q^r} = \frac{1}{q^r} \frac{1}{q-1} \sum_{a \in \mathrm{GF}(q)^*} \sum_{i=0}^{l-1} \sum_{\bar{c} \in \mathrm{GF}(q)^r \setminus \{\bar{0}\}} \chi(au_{\bar{c}}(i) - az_{\bar{c}}),$$

а значит,

$$N_l(\overline{z}, u_1, \ldots, u_r) - \frac{l}{q^r} = \frac{1}{q^r(q-1)} \sum_{\bar{c} \in \mathrm{GF}(q)^r \setminus \{\bar{0}\}} \nu_l(z_{\bar{c}}, u_{\bar{c}}). \tag{12}$$

Из соотношения (12) следует неравенство

$$\left| N_l(\overline{z}, u_1, \ldots, u_r) - \frac{l}{q^r} \right| \leqslant \frac{q^r - 1}{q^r(q-1)} \max_{\bar{c} \neq \bar{0}} |\nu_l(z_{\bar{c}}, u_{\bar{c}})|.$$

Используя теоремы 1 и 2, получим

$$\left| N_l(\overline{z}, u_1, \ldots, u_r) - \frac{l}{q^r} \right| \leqslant \frac{q^r - 1}{q^r(q-1)} \left(\frac{3q}{2} \left(q^m l(q-1) - l^2 \right) \right)^{\frac{1}{3}}. \tag{13}$$

В силу того, что при $\overline{z} = \overline{0}$ оценка является более точной, можем применить оценку (13) для любого z. ■

Если ограничиться случаем $\bar{z} \neq \bar{0}$, то при некоторых ограничениях на l возможно доказать более сильную оценку, чем в теореме 3.

Теорема 4. Пусть дополнительно, в условиях теоремы 3, $l \leqslant \dfrac{T}{2(T, q-1)}$ и $\overline{z} \neq \overline{0}$. Тогда справедлива следующая оценка:

$$\left| N_l(\overline{z}, u_1, \ldots, u_r) - \frac{l}{q^r} \right| \leqslant \frac{q^r - q^{r-1}}{q^r(q-1)} \left(\frac{3q}{2} \left((q-1)q^m l - l^2 \right) \right)^{\frac{1}{3}} + \\ + \frac{q^{r-1} - 1}{q^r(q-1)} \left(\frac{3q}{2} \left(q^m l - (q-1)l^2 \right) \right)^{\frac{1}{3}}.$$

Доказательство. Разобьем правую часть равенства (12) на две суммы:

$$N_l(\overline{z},u_1,\ldots,u_r)-\frac{l}{q^r}=\frac{1}{q^r(q-1)}\sum_{\overline{c}\in\{\mathrm{GF}(q)^r\setminus\overline{0}:z_{\overline{c}}=0\}}\nu_l(z_{\overline{c}},u_{\overline{c}})+\frac{1}{q^r(q-1)}\sum_{\overline{c}\in\{\mathrm{GF}(q)^r\setminus\overline{0}:z_{\overline{c}}\neq0\}}\nu_l(z_{\overline{c}},u_{\overline{c}}).$$

Очевидно, что

$$\left|\{\overline{c}\in\mathrm{GF}(q)^r\setminus\overline{0}:z_{\overline{c}}\neq0\}\right|=q^r-q^{r-1},$$
$$\left|\{\overline{c}\in\mathrm{GF}(q)^r\setminus\overline{0}:z_{\overline{c}}=0\}\right|=q^{r-1}-1.$$

Отсюда получим

$$\left|N_l(\overline{z},u_1,\ldots,u_r)-\frac{l}{q^r}\right|\leqslant\frac{q^r-q^{r-1}}{q^r(q-1)}\max_{\overline{c}\neq\overline{0},z\neq0}|\nu_l(z_{\overline{c}},u_{\overline{c}})|+\frac{q^{r-1}-1}{q^r(q-1)}\max_{\overline{c}\neq\overline{0}}|\nu_l(0,u_{\overline{c}})|.$$

Применив теоремы 1 и 2, получим необходимый результат. ■

Сформулируем и докажем утверждение, вытекающее из теоремы 3.

Утверждение 1. Пусть выполнено условие теоремы 3. Тогда среди элементов

$$u(0),u(1),\ldots,u\left(\left[\sqrt{3}q^{\frac{m+3r}{2}}/\sqrt{h(q)}\right]\right),$$

где $h(q)=2(q-1)^2/q$, появляются все r-граммы.

Доказательство. Для произвольного $\overline{z}\in\mathrm{GF}(q)^r$ справедливо соотношение

$$\frac{l}{q^r}-\frac{q^r-1}{q^r(q-1)}\left(\frac{3q}{2}\left(q^ml(q-1)-l^2\right)\right)^{\frac{1}{3}}\leqslant N_l(\overline{z},u_1,\ldots,u_r).$$

Если левая часть больше нуля, то появятся все r-граммы. Рассмотрим неравенство

$$\frac{l}{q^r}-\frac{q^r-1}{q^r(q-1)}\left(\frac{3q}{2}\left(q^ml(q-1)-l^2\right)\right)^{\frac{1}{3}}>0.$$

Оно равносильно

$$l^3>\frac{(q^r-1)^3}{(q-1)^3}\left(\frac{3q}{2}(q^ml(q-1)-l^2)\right).$$

Это неравенство заведомо выполнено при условии

$$l^3\geqslant\frac{(q^r)^3}{(q-1)^3}\left(\frac{3q}{2}(q^ml(q-1))\right),$$

то есть при

$$l\geqslant\frac{\sqrt{3}q^{\frac{m+3r}{2}}}{\sqrt{h(q)}},$$

где $h(q)=2(q-1)^2/q$. ■

Будем говорить, что оценка $|N_l(\overline{z},u_1,\ldots,u_r)-l/q^r|\leqslant S$ является нетривиальной, если в неравенстве $l/q^r-S\leqslant N_l(\overline{z},u_1,\ldots,u_r)\leqslant-l/q^r+S$ левая часть больше нуля, а правая часть меньше l. Ясно, что это равносильно условию $l/q^r-S>0$.

Следствие 1. Оценка (3) является нетривиальной при $l>\sqrt{3}q^{\frac{m+3r}{2}}/\sqrt{h(q)}$.

Доказательство. Условие нетривиальности доказано в утверждении 1. ■

Сравним оценку (3) и оценку (2). Заметим, что при $q = 2$ оценки совпадают.

Утверждение 2. Пусть $q \geqslant 3$. Тогда оценка (3) улучшает оценку (2) при

$$l \leqslant \frac{10}{13} q^m.$$

Доказательство. Выпишем необходимое условие:

$$\frac{q^r - 1}{(q-1)q^r} \left(\frac{3q}{2} \left(q^m l(q-1) - l^2 \right) \right)^{\frac{1}{3}} \leqslant \frac{q^r - 1}{q^r} \left(3 \left(q^m l - l^2 \right) \right)^{\frac{1}{3}}.$$

Это неравенство равносильно соотношению

$$\frac{1}{(q-1)^3} \left(\frac{q}{2} \left(q^m(q-1) - l \right) \right) \leqslant q^m - l,$$

которое выполняется при условии

$$\left(1 - \frac{q/2}{(q-1)^3} \right) l \leqslant q^m \left(1 - \frac{q}{2(q-1)^2} \right).$$

Выделив l, получим

$$l \leqslant q^m \left(\frac{(q-1)^3 - q(q-1)/2}{(q-1)^3 - q/2} \right) = q^m \left(1 + \frac{q/2 - q(q-1)/2}{(q-1)^3 - q/2} \right).$$

Таким образом,

$$l \leqslant q^m \left(1 - \frac{q}{1 + 2q(q-1)} \right). \tag{14}$$

Возьмём производную функции $g(q) = \left(1 - \frac{q}{1 + 2q(q-1)} \right)$:

$$g'(q) = \frac{2q^2 - 1}{(1 + 2q(q-1))^2} > 0, \quad q \in \mathbb{N}.$$

Это означает, что если $l \leqslant q^m g(3) = \frac{10}{13} q^m$, то для любого $q \geqslant 3$ выполняется неравенство (14). ■

Используя рассуждения, аналогичные доказательству утверждения 2, несложно показать, что при $q \geqslant 3$ и $l \leqslant (q-1)q^{m-1}$ правая часть неравенства (3) меньше правой части неравенства (2) в $\sqrt[3]{2(q-1)/q}$ раз.

Далее сравним полученную оценку (3) с оценкой (1).

Утверждение 3. Оценка (3) улучшает оценку Н. М. Коробова (1) при

$$l \leqslant \frac{h(q)}{3} q^{\frac{m}{2}} \ln^3 T.$$

Доказательство. Выпишем необходимое условие:

$$\frac{q^r-1}{(q-1)q^r}\left(\frac{3q}{2}\left(q^m l(q-1)-l^2\right)\right)^{\frac{1}{3}} < \frac{q^r-1}{q^r}q^{\frac{m}{2}}\ln T.$$

Оно равносильно соотношению

$$\frac{3}{2}\frac{q}{(q-1)^3}(q^m l(q-1)-l^2) < q^{\frac{3m}{2}}\ln^3 T.$$

Это неравенство выполнено, когда $\dfrac{3}{2}\dfrac{q}{(q-1)^2}l \leqslant q^{\frac{m}{2}}\ln^3 T$, что равносильно тому, что $l \leqslant \dfrac{h(q)}{3}q^{\frac{m}{2}}(\ln T(u))^3$. ■

Таким образом, из утверждений 2 и 3 следует, что полученная оценка при некоторых ограничениях на l уточняет известные результаты.

ЛИТЕРАТУРА

1. *Алферов А. П., Зубов А. Ю., Кузьмин А. С., Черемушкин А. В.* Основы криптографии: учеб. пособие. М.: Гелиос АРВ, 2001. 480 с.
2. *Лидл Р., Нидеррайтер Г.* Конечные поля. Т. 1, 2. М.: Мир, 1988. 824 с.
3. *Нечаев В. И.* Распределение знаков в последовательности прямоугольных матриц над конечным полем // Труды матем. института им. В. А. Стеклова. 1997. Т. 218. С. 335–342.
4. *Шпарлинский И. Е.* О распределении значений рекуррентных последовательностей // Проблемы передачи информации. 1989. Т. 25. № 2. С. 46–53.
5. *Коробов Н. М.* Распределение невычетов и первообразных корней в рекуррентных рядах // Докл. Акад. наук СССР. 1953. Т. 88. № 4. С. 603–606.
6. *Сидельников В. М.* Оценки для числа появлений знаков на отрезке рекуррентной последовательности над конечным полем // Дискретная математика. 1991. Т. 3. № 2. С. 87–95.
7. *Камловский О. В.* Оценки частот появления нулей в линейных рекуррентных последовательностях векторов // Чебышевский сборник. 2005. Т. 6. № 1. С. 135–144.

2013 Теоретические основы прикладной дискретной математики №1(19)

DOI 10.17223/20710410/19/2

УДК 519.7

О ВЕРХНЕЙ ОЦЕНКЕ НЕЛИНЕЙНОСТИ НЕКОТОРОГО КЛАССА БУЛЕВЫХ ФУНКЦИЙ С МАКСИМАЛЬНОЙ АЛГЕБРАИЧЕСКОЙ ИММУННОСТЬЮ[1]

Н. А. Коломеец

Институт математики СО РАН, г. Новосибирск, Россия

E-mail: nkolomeec@gmail.com

Доказывается верхняя оценка нелинейности булевых функций от чётного числа переменных, обладающих максимально возможной алгебраической иммунностью и предложенных D. K. Dalai и др. в 2006 г.

Ключевые слова: *булевы функции, нелинейность, алгебраическая иммунность.*

Введение

В работе [1] N. Courtois и W. Meier предложили метод криптоанализа шифров, основанных на фильтрующих генераторах. Этот криптоанализ использует следующие слабости фильтрующей функции: наличие у неё аннигиляторов низкой степени и множителей, существенно уменьшающих степень получающейся в результате произведения функции. В работе [2] сформулировано понятие алгебраической иммунности булевой функции, которое объединило эти слабости в одно свойство. Таким образом, булевы функции с низкой алгебраической иммунностью обладают слабостью к алгебраической атаке на поточные шифры.

В данной работе рассматривается верхняя оценка нелинейности булевых функций от чётного числа переменных, предложенных в [3].

Основная часть

Через $\mathbb{Z}_2^n$ обозначим n-мерный булев куб. Под *расстоянием* между двумя булевыми функциями будем понимать *расстояние Хэмминга* (число векторов, на которых функции различаются). *Вес Хэмминга* вектора x ($\mathrm{wt}(x)$) — количество его ненулевых координат. Будем называть i-м слоем n-мерного булева куба все векторы веса i.

Степень алгебраической нормальной формы булевой функции называется *алгебраической степенью* функции (обозначается $\deg(f)$). Булева функция называется *аффинной*, если её алгебраическая степень не превосходит 1.

Алгебраической иммунностью булевой функции f ($\mathrm{AI}(f)$) называют минимальную положительную алгебраическую степень булевой функции, которая аннулирует f или $f \oplus 1$, т. е.

$$\mathrm{AI}(f) = \min_{g \neq 0} \{\deg(g) : f(x)g(x) \equiv 0 \text{ или } (f(x) \oplus 1)g(x) \equiv 0\}.$$

Известно, что для функции f от n переменных $\mathrm{AI}(f) \leqslant \lceil n/2 \rceil$. Для криптографических приложений наибольший интерес представляют функции с максимально возможной алгебраической иммунностью, т. е. с $\mathrm{AI}(f) = \lceil n/2 \rceil$ (это значение достигается для любого n).

[1]Исследование выполнено при поддержке РФФИ, проект № 12-01-31097.

Рассмотрим нелинейность функций, обладающих максимальной возможной алгебраической иммунностью, а именно функций, построенных с помощью одной из самых простых конструкций для чётного числа переменных, которая предложена в работе [3]:

$$f(x) = \begin{cases} 0, & \mathrm{wt}(x) < n/2, \\ b(x) \in \{0,1\}, & \mathrm{wt}(x) = n/2, \\ 1, & \mathrm{wt}(x) > n/2. \end{cases} \tag{1}$$

Все такие функции обладают алгебраической иммунностью $n/2$.

Напомним, что нелинейностью булевой функции f (обозначается $\mathrm{nl}(f)$) называется расстояние Хэмминга от функции f до класса аффинных функций.

Известна следующая оценка снизу на нелинейность функции от n переменных [4]:

$$\mathrm{nl}(f) \geqslant 2 \sum_{i=0}^{\mathrm{AI}(f)-2} C_{n-1}^{i}.$$

Для булевых функций с максимально возможной алгебраической иммунностью от чётного числа переменных n эта оценка выглядит следующим образом:

$$\mathrm{nl}(f) \geqslant 2^{n-1} - 2C_{n-1}^{n/2}. \tag{2}$$

Среди функций вида (1) существуют такие, для которых в (2) достигается равенство. Эта оценка доказана также в [3] для функций вида (1), уравновешенных на среднем слое.

Докажем верхнюю оценку нелинейности для данного класса функций. Следует упомянуть, что данная оценка приводится (как тезис) в презентации Ch. Li [5].

Теорема 1. Для функций f вида (1) выполняется

$$\mathrm{nl}(f) \leqslant 2^{n-1} - C_{n-1}^{n/2}.$$

Доказательство. Оценим расстояния d_i от функций вида (1) до линейных функций $l_i(x) = x_i$, $i \in \{1, \ldots, n\}$, n — количество переменных. Пусть f — функция вида (1) от n переменных. Рассмотрим расстояния на каждом слое булева куба $\mathbb{Z}_2^n$ отдельно. Пусть d_i^j — расстояние до l_i на j-м слое, т .е.

$$d_i^j = \left|\{x \in \mathbb{Z}_2^n : \mathrm{wt}(x) = j, f(x) \neq l_i(x)\}\right|, j \in \{0, \ldots, n\}.$$

Тогда $d_i = d_i^0 + d_i^1 + \ldots + d_i^n$. Очевидно, что

$$d_i^j = \begin{cases} 0, & j = 0, \\ C_{n-1}^{j-1}, & 0 < j < n/2, \\ d_i^{n/2}, & j = n/2, \\ C_{n-1}^{j}, & n/2 < j < n, \\ 0, & j = n. \end{cases} \tag{3}$$

Поэтому

$$d_i = \sum_{j=1}^{n/2-1} C_{n-1}^{j-1} + \sum_{j=n/2+1}^{n-1} C_{n-1}^{j} + d_i^{n/2} = \sum_{j=0}^{n/2-2} C_{n-1}^{j} + \sum_{n/2+1}^{n-1} C_{n-1}^{j} + d_i^{n/2} =$$

$$= \sum_{j=0}^{n-1} C_{n-1}^{j} - C_{n-1}^{n/2-1} - C_{n-1}^{n/2} + d_i^{n/2} = 2^{n-1} - 2C_{n-1}^{n/2} + d_i^{n/2}.$$

Далее заметим, что для любого вектора x из среднего слоя ровно половина из l_i отличается от f на этом векторе x. Таким образом,

$$\sum_{i=1}^{n} d_i^{n/2} = n/2 \cdot C_n^{n/2} = n \cdot C_{n-1}^{n/2}.$$

Следовательно, существует i', такой, что

$$d_{i'}^{n/2} \leqslant C_{n-1}^{n/2},$$

т. е.

$$d_{i'} \leqslant 2^{n-1} - 2C_{n-1}^{n/2} + C_{n-1}^{n/2} = 2^{n-1} - C_{n-1}^{n/2}.$$

Теорема доказана. ■

В работе [3] доказано, что для функции

$$f(x) = \begin{cases} 0, & \mathrm{wt}(x) \leqslant n/2, \\ 1, & \mathrm{wt}(x) > n/2 \end{cases}$$

от n переменных выполняется

$$\mathrm{nl}(f) = 2^{n-1} - C_{n-1}^{n/2}.$$

Таким образом, оценка из теоремы 1 достигается.

Следствие 1. Для любой функций f вида (1) верно

$$2^{n-1} - 2C_{n-1}^{n/2} \leqslant \mathrm{nl}(f) \leqslant 2^{n-1} - C_{n-1}^{n/2},$$

при этом обе оценки достигаются.

Напомним, что максимально возможная нелинейность булевой функции от чётного числа переменных равна $2^{n-1} - 2^{n/2-1}$, т. е. нелинейность рассматриваемых функций весьма далека от максимально возможной.

ЛИТЕРАТУРА

1. *Courtois N. and Meier W.* Algebraic attacks on stream ciphers with liner feedback // LNCS. 2003. V. 2656. P. 345–359.
2. *Meier W., Pasalic E., and Carlet C.* Algebraic attacks and decomposition of Boolean functions // LNCS. 2004. V. 3027. P. 474–491.
3. *Dalai D.K., Maitra S., and Sarkar S.* Basic theory in construction of Boolean functions with maximum possible annihilator immunity // Designs, Codes and Cryptography. 2006. V. 40. Iss. 1. P. 41–58.
4. *Лобанов М. С.* Точное соотношение между нелинейностью и алгебраической иммунностью // Дискретная математика. 2006. Вып. 18. № 3. С. 152–159.
5. *Li Ch.* A survey on construction of Boolean function with optimum algebraic immunity (AI) // `http://www.frisc.no/wp-content/uploads/2011/10/Li-A-survey-on-constructions-of-BFs-with-optimum-AI.pdf`

2013 Теоретические основы прикладной дискретной математики №1(19)

DOI 10.17223/20710410/19/3

УДК 621.391: 519.728

ДВОИЧНЫЕ ПРЕДСТАВЛЕНИЯ НЕДООПРЕДЕЛЁННЫХ ДАННЫХ И ДИЗЪЮНКТИВНЫЕ КОДЫ[1]

Л. А. Шоломов

Институт системного анализа РАН, г. Москва, Россия

E-mail: sholomov@isa.ru

Рассматриваются достаточно компактные представления недоопределённых данных, позволяющие полностью восстановить исходные данные (а не только их доопределения). Их построение основано на введённых и изученных в данной работе специальных матрицах, названных селективными. Они обобщают широко применяемые в информатике дизъюнктивные матрицы. Исследованы свойства селективных матриц и получены оценки длины представления данных в функции от некоторых параметров. Рассмотрены сложностные вопросы, связанные с построением представлений.

Ключевые слова: *недоопределённые данные, сжатие, двоичное представление, базис системы множеств, длина представления, дизъюнктивная матрица, дизъюнктивный код, свободное от покрытий семейство, полиномиальный алгоритм.*

Введение

Для полностью определённых данных термины «сжатие» и «архивация» обычно понимаются одинаково. Они означают экономное кодирование данных, обеспечивающее их полное восстановление. Для недоопределённых данных эти понятия будем различать.

В качестве недоопределённых данных мы рассматриваем последовательности недоопределённых символов. Каждому такому символу соответствует некоторое множество основных (полностью определённых) символов, одним из которых он может быть замещён (доопределён). Имеются две естественные постановки задачи кодирования недоопределённых данных.

В первой постановке кодирование должно обеспечить восстановление какого-либо доопределения исходных данных (но не их самих). Она соответствует случаю, когда допустимо любое доопределение данных. При второй постановке от кодирования требуется, чтобы оно позволяло полностью восстановить исходные недоопределённые данные. К такой постановке приводит задача их хранения. Термин «сжатие» будем связывать с первой постановкой, термин «архивация» — со второй.

До сих пор мы имели дело с первой постановкой (например, в [1]). На её основе введена энтропия недоопределённых данных как характеристика степени сжатия, определены другие информационные показатели. Данная работа инициирована второй постановкой; часть её результатов имеется в [2].

Рассмотрим побуквенное представление недоопределённых последовательностей. Символы основного алфавита кодируются наборами из 0 и 1, а недоопределённые сим-

[1]Работа выполнена при поддержке ОНИТ РАН по проекту «Теоретические основы эффективного использования недоопределённой информации» программы «Интеллектуальные информационные технологии, системный анализ и автоматизация».

волы представляются наборами из 0, 1 и $*$, где $*$ — неопределённый символ. Ставится условие, чтобы отношение доопределимости между основными и недоопределёнными символами переносилось на представляющие их наборы. Это позволяет решать задачи, связанные со сжатием и доопределением данных, работая непосредственно с представлениями.

Если Λ и $\tilde{\Lambda}$ — матрицы, столбцами которых являются соответственно представления основных и недоопределённых символов, то при мощности m основного алфавита матрица Λ состоит из m столбцов, а число столбцов матрицы $\tilde{\Lambda}$ может достигать 2^m-1. Это делает процедуры представления недоопределённых последовательностей, применяющие матрицу $\tilde{\Lambda}$, неэффективными. В работе предложен эффективный способ, использующий лишь матрицу Λ. В результате этого задача экономного представления недоопределённых последовательностей свелась к построению допустимых матриц кодирования Λ с возможно малым числом строк.

Работа посвящена характеризации и изучению свойств матриц кодирования Λ, их построению и оценке параметров. Показано, что эти матрицы обладают некоторым свойством, названным селективностью, и их частным случаем являются дизъюнктивные матрицы. Последние введены в работе [3] как средство описания дизъюнктивных (superimposed) кодов. Эти коды широко применяются в различных задачах прикладной математики и информатики (см., например, [3–7] и цитированные там источники). Учитывая направленность журнала ПДМ, отметим применение дизъюнктивных кодов в криптографии для задачи распределения ключей [5, 8–9]. Дизъюнктивные коды интенсивно изучаются в течение многих лет, и некоторые из полученных для них результатов используются в данной работе.

1. Двоичное представление

Задан алфавит $A_0 = \{a_0, a_1, \ldots, a_{m-1}\}$ *основных символов*. Пусть $M = \{0, 1, \ldots, m-1\}$ и каждому непустому $T \subseteq M$ поставлен в соответствие символ a_T. Символ a_T называется *недоопределённым* и его *доопределением* считается всякий основной символ a_i, $i \in T$. Символ a_M, доопределимый любым основным символом, называется *неопределённым* и обозначается $*$. Под *доопределением последовательности* $a_{T_1} \ldots a_{T_N}$ недоопределённых символов понимается любая последовательность $a_{i_1} \ldots a_{i_N}$ основных символов, полученная из исходной заменой всех символов какими-либо доопределениями. Пусть выделена система $\mathcal{T}$ некоторых непустых подмножеств T множества M и с ней связан *недоопределённый алфавит* $A = A_{\mathcal{T}} = \{a_T : T \in \mathcal{T}\}$.

Задавшись натуральным числом s, припишем каждому $a_i \in A_0$ некоторый набор $\lambda_i = (\lambda_i(1), \lambda_i(2), \ldots, \lambda_i(s)) \in \{0,1\}^s$ — *код символа* a_i, а каждому $a_T \in A$ — некоторый набор $\lambda_T = (\lambda_T(1), \lambda_T(2), \ldots, \lambda_T(s)) \in \{0, 1, *\}^s$. Обозначим через Λ матрицу, столбцами которой являются наборы (точнее, транспонированные наборы) λ_i, $i \in M$, а через $\tilde{\Lambda}$ — матрицу со столбцами λ_T, $T \in \mathcal{T}$. Скажем, что пара $(\Lambda, \tilde{\Lambda})$ задает *двоичное представление алфавита* A, если для любых $T \in \mathcal{T}$ и $i \in M$

$$\lambda_i \textit{ доопределяет } \lambda_T \Leftrightarrow i \in T.$$

(Фактически используется расширенное двоичное представление — к 0 и 1 добавляется символ $*$, но удобно говорить о двоичном представлении.)

Утверждение 1. Всякий алфавит A двоично представим, т. е. обладает представлением указанного вида.

Такое представление можно получить, назначив $s = m$ и взяв в качестве λ_i, $i \in M$, набор $(\lambda_i(v), v = 1, \ldots, m)$, в котором компонента $\lambda_i(v)$ равна 1 для $v - 1 = i$ и 0 для

$v-1 \neq i$, а в качестве λ_T, $T \in \mathcal{T}$, — набор $(\lambda_T(v),\ v = 1, \ldots, m)$, в котором компонента $\lambda_T(v)$ равна $*$ для $v-1 \in T$ и 0 для $v-1 \notin T$. ■

Двоичное представление алфавита $A_{\mathcal{T}}$ допускает интерпретацию в терминах вложения системы $\mathcal{T}$ подмножеств множества M в булев куб $\{0,1\}^s$ (подходящей размерности s). При вложении каждому $i \in M$ сопоставляется некоторая точка λ_i куба, каждому $T \in \mathcal{T}$ — некоторый подкуб λ_T, и этот подкуб содержит те и только те точки λ_i, которые соответствуют значениям $i \in T$. Параметр s будем называть *размерностью представления* (вложения). Наименьшее s, при котором для алфавита A имеется двоичное представление размерности s, обозначим $s(A)$ и назовем *размерностью алфавита* A.

Будем рассматривать побуквенное представление недоопределённых последовательностей, при котором последовательности $a_{T_1}a_{T_2}\ldots a_{T_N}$ соответствует представление $\lambda_{T_1}\lambda_{T_2}\ldots\lambda_{T_N}$. Его построение и восстановление по нему исходной недоопределённой последовательности использует матрицу $\tilde{\Lambda}$, число столбцов которой может достигать $2^m - 1$. Это делает процедуру неэффективной (неполиномиальной). Покажем, что задачи представления и восстановления недоопределённых последовательностей можно эффективно решать, пользуясь лишь матрицей Λ, имеющей m столбцов.

Дальше, говоря о множестве столбцов T матрицы Λ, будем иметь в виду столбцы λ_i с индексами $i \in T$. Недоопределённый набор $\dot{\lambda}_T \in \{0,1,*\}^s$ назовём *T-селектором* для матрицы Λ, если все столбцы множества T являются его доопределениями и никакой другой столбец матрицы его не доопределяет. Матрицу Λ будем называть *$\mathcal{T}$-селективной*, если при каждом $T \in \mathcal{T}$ для нее существует T-селектор. В случае, когда система $\mathcal{T}$ образована всеми t-элементными подмножествами множества M, $\mathcal{T}$-селективную матрицу будем называть *t-селективной*.

Утверждение 2. Двоичное представление алфавита $A_{\mathcal{T}}$, использующее матрицу кодирования Λ, существует тогда и только тогда, когда она $\mathcal{T}$-селективна.

Действительно, если Λ $\mathcal{T}$-селективна, то в качестве столбцов матрицы $\tilde{\Lambda}$ в двоичном представлении могут быть взяты T-селекторы $\dot{\lambda}_T$, и наоборот, если представление $(\Lambda, \tilde{\Lambda})$ существует, то роль T-селекторов $\dot{\lambda}_T$ могут играть столбцы матрицы $\tilde{\Lambda}$. ■

Число строк матрицы Λ будем обозначать $s(\Lambda)$. По матрице кодирования Λ с $s(\Lambda) = s$ построим наборы $\hat{\lambda}_T \in \{0,1,*\}^s$, $T \in \mathcal{T}$, положив компоненту $\hat{\lambda}_T(v)$, $v = 1, \ldots, s$, набора равной $\tau \in \{0,1\}$, если $\lambda_i(v) = \tau$ для всех $i \in T$, и равной $*$, если найдутся $i, j \in T$, для которых $\lambda_i(v) \neq \lambda_j(v)$.

Утверждение 3. Если матрица Λ $\mathcal{T}$-селективна, то при каждом $T \in \mathcal{T}$ набор $\hat{\lambda}_T$ является T-селектором.

Доказательство. Для всех $i \in T$ столбцы λ_i доопределяют $\hat{\lambda}_T$ по построению. Рассмотрим $j \notin T$. Возьмем произвольный T-селектор $\dot{\lambda}_T$. Он не доопределим до λ_j, т. е. при некоторых v и τ имеет место $\dot{\lambda}_T(v) = \tau$, $\lambda_j(v) = \bar{\tau}$. Все столбцы λ_i, $i \in T$, доопределяют $\dot{\lambda}_T$, а потому имеют $\dot{\lambda}_i(v) = \tau$. Отсюда $\hat{\lambda}_T(v) = \tau \neq \lambda_j(v)$ и, следовательно, λ_j не доопределяет $\hat{\lambda}_T$. ■

Отметим, что если столбцы λ_i матрицы Λ интерпретировать как точки куба $\{0,1\}^s$, то набор $\hat{\lambda}_T$ соответствует минимальному (по включению) подкубу, содержащему все точки λ_i, $i \in T$.

Следствие 1. Если алфавит A двоично представим с матрицей кодирования Λ, то одним из его двоичных представлений является $(\Lambda, \hat{\Lambda})$, где $\hat{\Lambda}$ — матрица, образованная столбцами $\hat{\lambda}_T$, $T \in \mathcal{T}$.

Представление $(\Lambda, \hat{\Lambda})$, будем называть *каноническим*. С использованием матрицы Λ нахождение канонического представления $\hat{\lambda}_T$ символа a_T и восстановление символа a_T по каноническому представлению $\hat{\lambda}_T$ выполнимы с линейной относительно площади ms матрицы сложностью. Дальше будем рассматривать канонические представления. В этом случае задача нахождения двоичного представления алфавита $A_{\mathcal{T}}$ сводится к задаче построения $\mathcal{T}$-селективной матрицы.

2. Сведение к задаче о конъюнктивном базисе

Скажем, что система $\mathcal{Z}$ подмножеств множества M образует *конъюнктивный базис* системы $\mathcal{T}$, если каждое множество $T \in \mathcal{T}$ может быть получено как пересечение некоторых множеств из $\mathcal{Z}$, и образует *обобщённый конъюнктивный базис*, если каждое $T \in \mathcal{T}$ может быть получено как пересечение каких-либо множеств из $\mathcal{Z}$ либо их дополнений (до M). При этом считается, что пересечение пустой совокупности множеств даёт M, так что множество M конъюнктивно порождаемо любой системой $\mathcal{Z}$.

Строки матрицы Λ будем обозначать через $\lambda(v)$, $v = 1, \ldots, s$. Строке $\lambda(v)$ сопоставим множество $Z_v \subseteq M$, $Z_v = \{i : \lambda_i(v) = 1\}$, характеристическим набором которого строка является. Положим $\mathcal{Z} = \{Z_1, \ldots, Z_s\}$.

Утверждение 4. Матрица кодирования $\mathcal{T}$-селективна тогда и только тогда, когда соответствующая ей система множеств $\mathcal{Z}$ образует обобщённый конъюнктивный базис системы $\mathcal{T}$.

Доказательство. Для $Z \subseteq M$ и $\tau \in \{0, 1\}$ под Z^τ будем понимать Z при $\tau = 1$ и $\bar{Z} = M \setminus Z$ при $\tau = 0$.

1. Пусть матрица кодирования Λ является $\mathcal{T}$-селективной, $\mathcal{Z}$ — соответствующая ей система множеств, $\hat{\Lambda}$ — матрица канонического представления, построенная по Λ. Рассмотрим произвольное $T \in \mathcal{T}$. Положим $V_T = \{v : \hat{\lambda}_T(v) \neq *\}$ и убедимся, что

$$T = \bigcap_{v \in V_T} Z^{\hat{\lambda}_T(v)}. \tag{1}$$

Если множество V_T пусто, то столбец $\hat{\lambda}_T$ образован элементами $*$ и его доопределяют все столбцы матрицы Λ. В этом случае $T = M$ и равенство (1) выполнено в силу соглашения относительно пересечения пустой системы множеств.

Дальше считаем $V_T \neq \varnothing$. Для любого $i \in T$ столбец λ_i доопределяет $\hat{\lambda}_T$ и, следовательно, при каждом $v \in V_T$ выполнено $\lambda_i(v) = \hat{\lambda}_T(v)$. Это означает, что i содержится в каждом множестве $Z_v^{\hat{\lambda}_T(v)}$, $v \in V_T$, а потому и в их пересечении. С учётом произвольности $i \in T$ заключаем, что T включено в пересечение из правой части (1). Обратное включение вытекает из того, что для всякого $j \notin T$ столбец λ_j не является доопределением столбца $\hat{\lambda}_T$, т. е. при некотором $v \in V_T$ имеет место $\lambda_j(v) \neq \hat{\lambda}_T(v)$, а потому $j \notin Z_v^{\hat{\lambda}_T(v)}$ и j не содержится в пересечении из (1). Равенство (1) показывает, что система $\mathcal{Z}$ образует обобщённый конъюнктивный базис для $\mathcal{T}$.

2. Пусть задан обобщённый конъюнктивный базис $\mathcal{Z} = \{Z_1, \ldots, Z_s\}$ системы $\mathcal{T}$. Для $i \in M$ и $v = 1, \ldots, s$ положим $\lambda_i(v) = 1$, если $i \in Z_v$, и $\lambda_i(v) = 0$, если $i \notin Z_v$. Образуем матрицу $\Lambda = \|\lambda_i(v)\|$. Всякое множество $T \in \mathcal{T}$ представимо в виде пересечения некоторых множеств Z_v или их дополнений. Фиксируем одно из таких представлений и обозначим через V_T совокупность индексов v участвующих в нём множеств Z_v. Для $v \in V_T$ положим $\lambda_T(v)$ равным 1 или 0 в зависимости от того, в какой из форм — Z_v или $\bar{Z}_v$ — входит в пересечение это множество. Если множество Z_v системы $\mathcal{Z}$ не

участвует в рассматриваемом представлении, полагаем $\lambda_T(v) = *$. В принятых обозначениях представление записывается в виде

$$T = \bigcap_{v \in V_T} Z^{\lambda_T(v)}. \tag{2}$$

Образуем матрицу $\tilde{\Lambda} = \|\lambda_T(v)\|$. Пара $(\Lambda, \tilde{\Lambda})$ двоично представляет алфавит A, ибо столбец λ_i доопределяет λ_T тогда и только тогда, когда для любого $v \in V_T$ выполнено $\lambda_i(v) = \lambda_T(v)$, что эквивалентно соотношениям $i \in Z^{\lambda_T(v)}$, а это в силу (2) имеет место тогда и только тогда, когда $i \in T$. ■

Следствие 2. Двоичное представление алфавита $A = A_{\mathcal{T}}$, имеющее размерность s, существует тогда и только тогда, когда у системы $\mathcal{T}$ имеется обобщённый конъюнктивный базис мощности s.

Проверка, образует ли система $\mathcal{Z}$ обобщённый конъюнктивный базис для $\mathcal{T}$, выполнима с полиномиальной относительно мощностей систем $\mathcal{Z}$, $\mathcal{T}$ и множества M сложностью. Это следует из того, что множество $T \in \mathcal{T}$ представимо в виде пересечения некоторых множеств из $\mathcal{Z}$ или их дополнений тогда и только тогда, когда T совпадает с пересечением всех содержащих его множеств Z_v^τ, $Z_v \in \mathcal{Z}$, $\tau \in \{0, 1\}$.

Обозначим через $[\mathcal{T}]$ замыкание системы множеств $\mathcal{T}$ относительно операции пересечения. В $[\mathcal{T}]$, в частности, содержится множество M как пересечение пустой системы множеств. Из того, что обобщённый конъюнктивный базис системы $\mathcal{T}$ является таковым и для $[\mathcal{T}]$, вытекает следующий факт.

Следствие 3. $\mathcal{T}$-селективная матрица $[\mathcal{T}]$-селективна.

Это позволяет в задачах построения и упрощения $\mathcal{T}$-селективных матриц не принимать в расчет множества системы $\mathcal{T}$, являющиеся пересечениями других.

Задача о размерности алфавита состоит в том, чтобы по алфавиту A (эквивалентно, по системе $\mathcal{T}$) и числу s узнать, существует ли для A двоичное представление размерности s.

Утверждение 5. Задача о размерности алфавита NP-полна.

Доказательство. Согласно следствию 2, достаточно установить NP-полноту задачи об обобщённом конъюнктивном базисе. Переход от множеств системы $\mathcal{T}$ к их дополнениям превращает её в задачу об обобщённом дизъюнктивном базисе (определения дизъюнктивного и обобщённого дизъюнктивного базисов аналогичны соответствующим определениям конъюнктивного и обобщённого конъюнктивного базисов с тем отличием, что роль пересечения играет объединение). Укажем сведение к ней NP-полной задачи о дизъюнктивном базисе (задача MP7 из [11]).

Пусть требуется решить задачу о (дизъюнктивном) базисе системы $\mathcal{T}$ некоторых подмножеств T множества $M = \{0, 1, \ldots, m-1\}$. Можно ограничиться случаем $M \notin \mathcal{T}$, ибо общий случай сводится к этому очевидным образом. Образуем множество $M' = \{0, 1, \ldots, m-1, m\}$ и рассмотрим систему $\mathcal{T}'$, полученную из $\mathcal{T}$ добавлением одноэлементного множества $\{m\}$. Докажем, что система $\mathcal{T}$ обладает базисом мощности s (т. е. состоящим из s множеств) тогда и только тогда, когда у системы $\mathcal{T}'$ имеется обобщённый базис мощности $s+1$.

Если система $\mathcal{Z}$ мощности s является базисом для $\mathcal{T}$, то система $\{\mathcal{Z}, \{m\}\}$ имеет мощность $s+1$ и образует базис (и обобщённый базис) для $\mathcal{T}'$. Обратно, пусть у системы $\mathcal{T}'$ имеется обобщённый базис $\mathcal{Z}'$ мощности $s+1$. Чтобы покрыть множество $\{m\} \in \mathcal{T}'$, он должен содержать множество $\{m\}$ или его дополнение $M' \setminus \{m\} = M$.

Ни одно из них не может участвовать в покрытии множеств системы $\mathcal{T}$. Удалив их из $\mathcal{Z}'$, получим обобщённый базис $\mathcal{Z}''$ системы $\mathcal{T}$, состоящий из не более чем s множеств. Для всякого $Z'' \in \mathcal{Z}''$ лишь одно из множеств Z'' и $\bar{Z}'' = M' \setminus Z''$ не содержит элемента m и может участвовать в покрытии множеств системы $\mathcal{T}$. Совокупность таких множеств образует дизъюнктивный базис системы $\mathcal{T}$. Если он состоит из менее чем s множеств, произвольно дополним его до мощности s. ■

В связи с трудностью нахождения значения $s(A)$ представляют интерес оценки этой величины. Обозначим через $s(m,n)$ максимальное значение $s(A)$ по всем алфавитам A с $|A_0| = m$, $|A| = n$, где $|\cdot|$ означает мощность множества.

Утверждение 6. Имеет место равенство

$$s(m,n) = \min\{m,n\}.$$

Доказательство. Для алфавита $A = A_{\mathcal{T}}$ с $|A_0| = m$ и $|A| = n$ верхнюю оценку $s(A) \leqslant m$ получим, если в качестве конъюнктивного базиса системы $\mathcal{T}$ возьмем совокупность всех $(m-1)$-элементных подмножеств множества M, а верхнюю оценку $s(A) \leqslant n$ обеспечим, если в качестве конъюнктивного базиса возьмем саму систему $\mathcal{T}$.

В случае $m \leqslant n$ нижняя оценка $s(A) \geqslant m$ справедлива для алфавита $A = A_{\mathcal{T}}$, у которого система $\mathcal{T}$ состоит из всех $(m-1)$-элементных подмножеств множества M и $n-m$ любых других подмножеств, а при $n < m$ оценка $s(A) \geqslant n$ имеет место для алфавита $A = A_{\mathcal{T}}$, у которого в качестве $\mathcal{T}$ взята система, образованная множествами $M \setminus \{i\}$, $i = 0, 1, \ldots n-1$. ■

Мощность $|A|$ алфавита недоопределённых символов может достигать $2^{|A_0|} - 1$. В содержательных задачах обычно $|A_0| \leqslant |A|$, поэтому в качестве основного будем рассматривать случай

$$m \leqslant n. \tag{3}$$

При условии (3) утверждение 6 приобретает вид

$$s(m,n) = m. \tag{4}$$

3. Алфавит с ограниченным числом доопределений

Представляет интерес рассмотрение содержательных случаев, когда размерность алфавита существенно ниже гарантируемой утверждением 6. Недоопределённые данные, с которыми имеют дело в приложениях, обычно помимо неопределённого символа $*$ используют лишь символы, имеющие небольшое число доопределений. Так, например, если в качестве недоопределённого символа выступает нечётко написанная буква, то скорее всего она похожа лишь на небольшое число реальных букв.

Обозначим через $s(m,n;t)$ максимальную из размерностей $s(A)$ алфавитов A, для которых $|A_0| = m$, $|A| = n$ и каждый символ $a_T \in A$ имеет не более t доопределений либо является неопределённым.

Утверждение 7. Справедливы оценки

$$s(m,n;2) \leqslant 4\ln(mn) + 1; \tag{5}$$

$$s(m,n;3) \leqslant 8\ln(mn) + 1; \tag{6}$$

$$s(m,n;t) \leqslant e(t+1)\ln(mn) + 1. \tag{7}$$

Доказательство. Рассмотрим произвольный алфавит A с параметрами m, n и t. Пусть $A = A_{\mathcal{T}}$. Поскольку любое двоичное представление реализует символ $* = a_M$ автоматически, можно считать, что $M \notin \mathcal{T}$ и для любого $T \in \mathcal{T}$ выполнено $|T| \leqslant t$. Согласно утверждению 4, достаточно оценить сверху мощность обобщённого конъюнктивного базиса для системы $\mathcal{T}$.

Пары (T, j), $j \in M \setminus T$, будем называть *фрагментами*. Скажем, что множество Z *реализует* фрагмент (T, j) *прямо*, если $T \subseteq Z$, $j \notin Z$, и реализует *инверсно*, если $T \cap Z = \varnothing$, $j \in Z$. Множество *реализует* фрагмент, если оно его реализует прямо или инверсно. Ясно, что система $\mathcal{Z}$ образует обобщённый конъюнктивный базис для $\mathcal{T}$ тогда и только тогда, когда множествами системы $\mathcal{Z}$ реализуются все фрагменты (T, j), $T \in \mathcal{T}$, $j \in M \setminus T$. Для построения и оценки мощности системы с указанным свойством используем вероятностный метод.

Рассмотрим систему s случайных подмножеств множества M, в каждое из которых элементы $i \in M$ включаются независимо с одинаковой вероятностью p, которая будет выбрана позже. Вероятность того, что случайное множество реализует заданный фрагмент (T, j), составляет $p^{|T|}(1-p) + (1-p)^{|T|}p$, а вероятность, что ни одно из s множеств системы его не реализует, равна $\bigl(1 - (p^{|T|}(1-p) + (1-p)^{|T|}p)\bigr)^s$. Эта величина не превосходит $\bigl(1 - (p^t(1-p) + (1-p)^t p)\bigr)^s$. Поскольку число фрагментов меньше $|\mathcal{T}| \cdot |M| = nm$, вероятность отсутствия реализации хотя бы у одного из них меньше

$$mn\bigl(1 - (p^t(1-p) + (1-p)^t p)\bigr)^s.$$

Отсюда и из соотношения $\ln(1-x) \leqslant -x$ следует, что при любом

$$s \geqslant \frac{\ln(mn)}{p^t(1-p) + (1-p)^t p}$$

эта вероятность меньше 1 и, следовательно, существует обобщённый конъюнктивный базис такой мощности. Это дает

$$s_t(m, n) \leqslant \frac{\ln(mn)}{\psi_t(p)} + 1, \tag{8}$$

где $\psi_t(p) = p^t(1-p) + (1-p)^t p$.

Выберем значение p. Для $t = 2$ и 3 функция $\psi_t(p)$ достигает максимума при $p = 1/2$. Подстановка этого значения в (8) приводит к оценкам (5) и (6). При произвольном t с учетом того, что $\psi_t(p) \geqslant p^t(1-p)$ и величина $p^t(1-p)$ при $p = t/(t+1)$ принимает значение $t^t/(t+1)^{t+1}$, имеем

$$\psi_t\Bigl(\frac{t}{t+1}\Bigr) \geqslant \frac{t^t}{(t+1)^{t+1}} \geqslant \frac{1}{e(t+1)}.$$

Подставляя эту оценку в (8), приходим к (7). ■

В основном случае (3) соотношение (7) приводит к оценке

$$s(m, n; t) \leqslant 2e(t+1)\ln n + 1. \tag{9}$$

Справедлива тривиальная нижняя оценка

$$s(m, n; t) \geqslant \log_3 n,$$

вытекающая из мощностного соотношения $3^{s(m,n;t)} \geqslant |\mathcal{T}| = n$. Она отличается от верхней оценки (9) по порядку в t раз. Дальше этот разрыв будет сокращен до логарифма t (утверждение 14).

4. Строго двоичные представления

В двоичных представлениях рассматриваемого вида недоопределённым последовательностям в алфавите A соответствуют недоопределённые двоичные последовательности, фактически использующие трёхбуквенный алфавит $\{0, 1, *\}$. Обычно, когда речь идет о сжатии данных, характеристики сжатия связываются с параметрами двоичного кодирования. Данный пункт посвящён представлениям в двухбуквенном алфавите.

Двоичное представление $(\Lambda, \tilde{\Lambda})$ алфавита A назовем *строго двоичным*, если $\tilde{\Lambda}$ — матрица в двухбуквенном алфавите. Этим алфавитом может быть $\{0, *\}$ либо $\{1, *\}$. Оба случая равноценны, но удобнее иметь дело с $\{0, *\}$, и дальше будем рассматривать этот случай. Построенное в утверждении 1 двоичное представление является строгим, поэтому всякий недоопределённый алфавит строго двоично представим. Если интерпретировать строгое двоичное представление алфавита $A_{\mathcal{T}}$ как вложение системы множеств $\mathcal{T}$ в булев куб (с. 18), то подкубы, содержащие множества $T \in \mathcal{T}$, должны обладать дополнительным свойством — проходить через вершину $(0,\ldots,0)$.

Опишем вид матриц кодирования Λ, возникающих при строго двоичном представлении. Под *дизъюнкцией* $\lambda \vee \lambda'$ *двоичных наборов* $\lambda = (\lambda(1), \ldots, \lambda(s))$ и $\lambda' = (\lambda'(1), \ldots, \lambda'(s))$ понимается набор с компонентами $\lambda(i) \vee \lambda'(i)$, $i = 1, \ldots, s$. Скажем, что набор λ' *покрывает* λ, если $\lambda \vee \lambda' = \lambda'$, и что некоторое множество наборов покрывает λ, если λ покрывается их дизъюнкцией. Матрицу Λ со столбцами λ_i, $i \in M$, назовем $\mathcal{T}$*-дизъюнктивной*, если для любого $T \in \mathcal{T}$ множество столбцов T не покрывает ни одного столбца, не входящего в это множество. В случае, когда система $\mathcal{T}$ состоит из всех t-элементных подмножеств множества M, $\mathcal{T}$-дизъюнктивную матрицу называют t*-дизъюнктивной*. Множество наборов, соответствующих столбцам t-дизъюнктивной матрицы, образует t*-дизъюнктивный код*. Дизъюнктивные (superimposed) коды, введённые в работе [3], находят широкое применение в информатике; $\mathcal{T}$-дизъюнктивные матрицы общего вида встречались в [12].

Утверждение 8. Строго двоичное представление алфавита $A_{\mathcal{T}}$, использующее матрицу кодирования Λ, существует тогда и только тогда, когда она $\mathcal{T}$-дизъюнктивна.

Доказательство. Если строго двоичное представление с матрицей кодирования Λ существует, то у Λ при каждом $T \in \mathcal{T}$ имеется некоторый T-селектор $\dot{\lambda}_T$ в алфавите $\{0, *\}$. Столбцы множества T доопределяют его, и их единицы находятся в позициях, где $\dot{\lambda}_T$ принимает значение $*$. Любой другой столбец содержит 1 в некотором нулевом разряде T-селектора $\dot{\lambda}_T$ и потому не покрывается столбцами множества T. В силу произвольности $T \in \mathcal{T}$ это означает $\mathcal{T}$-дизъюнктивность матрицы.

Обратно, если матрица $\mathcal{T}$-дизъюнктивна, то T-селектор в алфавите $\{0, *\}$ для $T \in \mathcal{T}$ можно образовать, взяв дизъюнкцию столбцов из множества T и заменив в ней все элементы 1 на $*$. Очевидно, что столбцы λ_i, $i \in T$, доопределяют этот набор, а любой столбец λ_j, $j \notin T$, не доопределяет, ибо имеет 1 в некотором разряде, где дизъюнкция равна 0. Наличие для каждого $T \in \mathcal{T}$ T-селектора в алфавите $\{0, *\}$ обеспечивает строго двоичную представимость. ■

Из этого утверждения, в частности, следует, что $\mathcal{T}$-дизъюнктивные матрицы $\mathcal{T}$-селективны. Применяя в качестве столбцов λ_T матрицы $\tilde{\Lambda}$ при строго двоичном представлении T-селекторы из второй части доказательства, можно, пользуясь лишь матрицей Λ, эффективно строить строгое представление недоопределённой последовательности и эффективно восстанавливать по нему исходную последовательность. Заменив в этом представлении символы $*$ на 1, получим последовательность в алфавите $\{0, 1\}$,

которую можно рассматривать как код исходной последовательности. При этом кодом символа a_T будет дизъюнкция столбцов λ_i, $i \in T$. Заметим, что по той же матрице Λ можно построить каноническое двоичное представление, но оно, как правило, не будет строгим.

Никакой столбец t-дизъюнктивной матрицы не покрывается t другими, поэтому такие матрицы также называют *свободными от t-покрытий* [13]; $\mathcal{T}$-дизъюнктивные матрицы естественно называть *свободными от $\mathcal{T}$-покрытий*. Представим в аналогичных терминах понятие $\mathcal{T}$-селективной матрицы. Под *инверсией двоичного набора* будем понимать набор, полученный из него заменой всех компонент их отрицаниями. Будем говорить, что некоторое множество наборов *инверсно покрывает* набор λ, если множество инверсий этих наборов покрывает инверсию набора λ. Скажем, что множество наборов *дважды покрывает* набор λ, если оно покрывает и инверсно покрывает набор λ. Матрицу Λ назовем *свободной от двойных $\mathcal{T}$-покрытий*, если для любого $T \in \mathcal{T}$ множество столбцов λ_i, $i \in \mathcal{T}$, не покрывает дважды ни одного столбца, не входящего в это множество.

Утверждение 9. Матрица Λ $\mathcal{T}$-селективна тогда и только тогда, когда она свободна от двойных $\mathcal{T}$-покрытий.

Доказательство. Рассмотрим матрицу Λ и построенную по ней матрицу $\hat{\Lambda}$ канонического представления. Убедимся, что столбец λ_j доопределяет столбец $\hat{\lambda}_T$ тогда и только тогда, когда совокупность столбцов λ_i, $i \in T$, дважды покрывает λ_j. Пусть λ_j доопределяет $\hat{\lambda}_T$. Если для некоторой позиции v имеет место $\lambda_j(v) = \tau$, то $\hat{\lambda}_T(v) \in \{*, \tau\}$ и по построению $\hat{\Lambda}$ найдется $i \in T$, для которого $\lambda_i(v) = \tau$. Применив это рассуждение ко всем позициям v с $\tau = 1$, заключаем, что множество столбцов λ_i, $i \in T$, покрывает λ_j. Аналогичное рассмотрение позиций с $\tau = 0$ показывает, что множество инверсий этих столбцов покрывает инверсию столбца λ_j. Это означает, что покрытие двойное. Доказательство в другую сторону проводится обращением этих рассуждений.

Матрица Λ $\mathcal{T}$-селективна тогда и только тогда, когда алфавит $A_{\mathcal{T}}$ представим парой $(\Lambda, \hat{\Lambda})$ (утверждение 3). Это означает, что для $T \in \mathcal{T}$ ни один из столбцов λ_j, $j \notin T$, не доопределяет столбец $\hat{\lambda}_T$ и по доказанному выше не покрывается дважды множеством столбцов T, т. е. матрица Λ свободна от двойных $\mathcal{T}$-покрытий. ■

Используя это утверждение, продемонстрируем возможность улучшения параметров при переходе от дизъюнктивных матриц к селективным.

Утверждение 10. В результате дописывания к произвольной 2-дизъюнктивной матрице столбца, составленного из единиц, получается 2-селективная матрица.

Доказательство. Обозначим через Λ' матрицу, полученную из 2-дизъюнктивной матрицы Λ дописыванием единичного столбца $\tilde{1}$. Рассмотрим три разных столбца λ_i, λ_j, λ_k матрицы Λ'. Если среди столбцов λ_i, λ_j нет $\tilde{1}$, то они не покрывают λ_k в силу 2-дизъюнктивности Λ либо того, что $\lambda_k = \tilde{1}$. Если же, например, $\lambda_i = \tilde{1}$, то $\lambda_i \wedge \lambda_j = \lambda_j$ не покрывается столбцом λ_k, а потому столбцы λ_i, λ_j не покрывают инверсно λ_k. ■

Полученная матрица Λ' не дизъюнктивна, поскольку в дизъюнктивных матрицах отсутствуют поглощения столбцов.

Для $\mathcal{T}$-дизъюнктивных матриц справедлив аналог утверждения 4. Как и раньше, строкам $\lambda(v)$, $v = 1, \ldots, s$, матрицы Λ сопоставим множества $Z_v = \{i : \lambda_i(v) = 1\}$. Положим $\mathcal{Z}' = \{\bar{Z}_1, \ldots, \bar{Z}_s\}$, где $\bar{Z}_v = M \setminus Z_v$.

Утверждение 11. Матрица $\mathcal{T}$-дизъюнктивна тогда и только тогда, когда соответствующая ей система множеств $\mathcal{Z}'$ образует конъюнктивный базис системы $\mathcal{T}$.

Доказательство опускаем. Это свойство может быть использовано для построения и упрощения $\mathcal{T}$-дизъюнктивных матриц для конкретных систем $\mathcal{T}$.

Следующее утверждение указывает способ преобразования $\mathcal{T}$-селективной матрицы в $\mathcal{T}$-дизъюнктивную. Пусть $\bar{\Lambda}$ означает инверсию матрицы Λ — результат замены в ней всех элементов их отрицаниями.

Утверждение 12. Если матрица Λ $\mathcal{T}$-селективна, то матрица $\begin{bmatrix}\Lambda\\ \bar{\Lambda}\end{bmatrix}$, полученная дописыванием к строкам матрицы Λ строк её инверсии, $\mathcal{T}$-дизъюнктивна.

Этот факт следует из утверждений 4 и 12, поскольку указанное преобразование матриц соответствует переходу от обобщённого конъюнктивного базиса к конъюнктивному базису за счет добавления дополнений множеств, входящих в обобщённый базис. ■

Укажем некоторые различия в свойствах $\mathcal{T}$-селективных и $\mathcal{T}$-дизъюнктивных матриц, легко проверяемые непосредственно. В $\mathcal{T}$-селективных матрицах допустима инверсия строк, а свойство $\mathcal{T}$-дизъюнктивности при инверсиях может нарушаться. Если $\vee_T$ означает дизъюнкцию столбцов множества T, то в $\mathcal{T}$-дизъюнктивных матрицах наборы $\vee_T$ для разных $T \in \mathcal{T}$ различны. Для $\mathcal{T}$-селективных матриц они могут совпадать, но различными являются пары $(\vee_T, \wedge_T)$, где $\wedge_T$ — конъюнкция столбцов множества T. Двойственным к понятию $\mathcal{T}$-дизъюнктивности является $\mathcal{T}$-конъюнктивность (для любых $T \in \mathcal{T}$ и $j \notin T$ конъюнкция столбцов множества T не покрывается столбцом λ_j). Класс $\mathcal{T}$-селективных матриц включает оба эти класса матриц и не исчерпывается ими.

Обозначим через $s_0(A)$ размерность алфавита A для строго двоичных представлений, определяемую аналогично $s(A)$. Очевидно, $s(A) \leqslant s_0(A)$. Следующий пример показывает, что неравенство может быть строгим.

Пример 1. Пусть основным алфавитом является $A_0 = \{a_0, a_1, a_2, a_3\}$, а алфавит A образован всеми символами, имеющими два доопределения (эквивалентно, не более двух — следствие 3). В силу утверждения 10 матрица

λ_0	λ_1	λ_2	λ_3
1	1	0	0
1	0	1	0
1	0	0	1

,

образованная из диагональной матрицы дописыванием столбца из единиц, 2-селективна. Она обеспечивает для алфавита A размерность 3. Из приведённой в [4] нижней оценки Л. А. Бассалыго следует, что при использовании 2-дизъюнктивных матриц наименьшей достижимой размерностью здесь является 4.

На основе $s_0(A)$ введём функции $s_0(m, n)$ и $s_0(m, n; t)$, аналогичные тем, которые были определены применительно к $s(A)$. Для них справедливы аналоги утверждений 5, 6 и формулы (4). Оценка функции $s_0(m, n; t)$ проводится, как в утверждении 7, но вместо $\psi_t(p)$ в формуле (8) возникает функция $\psi_t^0(p) = p^t(1-p)$, принимающая наибольшее значение $t^t/(t+1)^{t+1}$ при $p = t/(t+1)$. Аналог оценки (7) сохраняет тот же вид, поскольку в процессе установления (7) вместо $\psi_t(p)$ фактически использовалась функция $\psi_t^0(p)$, а константы в аналогах (5) и (6) возрастают. Подсчёт показывает, что имеет место следующий факт.

Утверждение 13. Справедливы оценки

$$\begin{aligned} s_0(m,n;2) &\leqslant 6{,}75\ln(mn)+1,\\ s_0(m,n;3) &\leqslant 9{,}48\ln(mn)+1,\\ s_0(m,n;t) &\leqslant e(t+1)\ln(mn)+1. \end{aligned}$$

Тривиальной нижней оценкой является $s_0(m,n;t) \geqslant \log n$ (она будет улучшена). Всюду под $\log n$ понимается $\log_2 n$.

5. Нижние оценки

В конце п. 3 и 4 приведены тривиальные (мощностные) нижние оценки функций $s(m,n;t)$ и $s_0(m,n;t)$. Далее получим нетривиальные нижние оценки на основе установленного в [14] соотношения

$$m \leqslant t + \binom{s}{\left\lceil \frac{2(s-t)}{t(t+1)} \right\rceil}, \tag{10}$$

связывающего параметры t-дизъюнктивной матрицы. Здесь m — число столбцов, s — число строк. Подробнее об оценках для t-дизъюнктивных матриц и их авторстве будет сказано далее.

Начнём со случая строго двоичных представлений.

Утверждение 14. При выполнении условия

$$t = \mathrm{o}\left(\frac{\log n}{\log\log n}\right) \tag{11}$$

справедлива оценка

$$s_0(m,n;t) \gtrsim \frac{(t+1)\log n}{2(2\log t + c)}, \quad c = \log\frac{3e}{4} < 1{,}027. \tag{12}$$

Доказательство. Поскольку n не превосходит числа подмножеств множества M, имеющих мощность не выше t, справедливо неравенство

$$n \leqslant \sum_{u\leqslant t}\binom{m}{u}. \tag{13}$$

Пусть m' — максимальное значение, удовлетворяющее условию $n \geqslant \sum\limits_{u\leqslant t}\binom{m'}{u}$. Очевидно, что $m' \leqslant m$. В силу максимальности m' выполнено

$$n \leqslant \sum_{u\leqslant t}\binom{m'+1}{u} \leqslant (m'+1)^t.$$

Отсюда с учётом (11) получаем

$$\log(m'+1) \geqslant \frac{\log n}{t} \gg \log\log n \tag{14}$$

(соотношение $u \gg v$ означает $v = \mathrm{o}(u)$), а потому

$$m' \gg \log n \gg t. \tag{15}$$

Положим $M' = \{0, 1, \ldots, m' - 1\}$ и возьмём в качестве $\mathcal{T}$ некоторую систему, которая состоит из n подмножеств множества M мощности не выше t и включает все t-элементные подмножества множества M'. Рассмотрим произвольную $\mathcal{T}$-дизъюнктивную матрицу Λ, первые m' столбцов которой соответствуют элементам множества M'. Эти столбцы образуют t-дизъюнктивную матрицу и для неё справедливо соотношение (10). В силу тривиальной нижней оценки $s \geqslant \log n$ и условия (11) выполнено $s \gg t$. Прологарифмировав (10), получаем с учётом неравенства $\log \binom{u}{v} \leqslant \log \frac{eu}{v}$ и соотношений (14), (15) и $s \gg t$

$$\frac{\log n}{t} \lesssim \log m' \lesssim \left(\frac{2s}{t(t+1)} + 1\right) \log \frac{et(t+1)}{2}.$$

Отсюда

$$\frac{2s}{t(t+1)} + 1 \gtrsim \frac{\log n}{t \log \dfrac{et(t+1)}{2}}. \tag{16}$$

Так как в силу (11) выполнено

$$\frac{\log n}{t \log \dfrac{et(t+1)}{2}} \gg \frac{\log n}{\left(\dfrac{\log n}{\log \log n}\right) \log \log n} = 1,$$

из (16) находим

$$s \gtrsim \frac{t(t+1)\log n}{2t \log \dfrac{et(t+1)}{2}} = \frac{(t+1)\log n}{2\left(2\log t + \log \dfrac{(t+1)e}{2t}\right)} \geqslant \frac{(t+1)\log n}{2\left(2\log t + \log \dfrac{3e}{4}\right)}.$$

Отсюда следует (12). ■

При растущем t оценка (12) отличается от верхней оценки функции $s_0(m, n; t)$ по порядку в $\log t$ раз.

Рассмотрим теперь функцию $s(m, n; t)$, относящуюся к двоичным представлениям общего вида. Из утверждения 12 вытекает, что $s(A) \geqslant s_0(A)/2$, а потому $s(m, n; t) \geqslant \geqslant s_0(m, n; t)/2$ и для $s(m, n; t)$ справедлива нижняя оценка из (12), уменьшенная в 2 раза. Следующее утверждение содержит оценку, которая лучше этой при всех $t \geqslant 2$.

Утверждение 15. При выполнении условия (11) справедлива оценка

$$s(m, n; t) \gtrsim \frac{(t-1)\log n}{2(2\log(t-1) + c)}, \tag{17}$$

где c — константа из (12).

Доказательство.

1. Рассмотрим сначала случай, когда $(m - 1, n, t - 1)$ — допустимая тройка, т. е. существует алфавит с такими параметрами. Покажем, что в этом случае

$$s(m, n; t) \geqslant s_0(m - 1, n; t - 1). \tag{18}$$

Пусть A — произвольный алфавит с параметрами $(m-1, n, t-1)$ и ему соответствуют основной алфавит $A_0 = \{a_0, \ldots, a_{m-2}\}$ и система множеств $\mathcal{T}$. Обозначим через A_0'

основной алфавит, полученный из A_0 присоединением нового символа a_{m-1}. Путём добавления к каждому множеству T системы $\mathcal{T}$ элемента $m-1$ образуем множества T' и систему всех таких множеств обозначим через $\mathcal{T}'$. Алфавит $A' = A_{\mathcal{T}'}$ имеет параметры (m, n, t).

Рассмотрим $\mathcal{T}'$-селективную матрицу, на которой достигается $s(A')$. Путем инверсирования её строк добьемся того, чтобы столбец, соответствующий элементу $m-1$, общему для всех множеств T', стал нулевым; полученную матрицу обозначим Λ'. Все T'-селекторы матрицы Λ' являются наборами в алфавите $\{0, *\}$ и потому она T'-дизъюнктивна. Матрица Λ, образованная из Λ' удалением нулевого столбца, T-дизъюнктивна. В результате получаем

$$s_0(A) \leqslant s(\Lambda) = s(\Lambda') = s(A') \leqslant s(m, n; t).$$

Учитывая произвольность алфавита A с параметрами $(m-1, n, t-1)$, приходим к неравенству (18), которое совместно с (12) обеспечивает в рассматриваемом случае оценку

$$s(n, m; t) \gtrsim \frac{t \log n}{2(2 \log(t-1) + c)}. \tag{19}$$

2. Пусть тройка $(m-1, n, t-1)$ недопустима. Положим $n' = \sum\limits_{u \leqslant t-1} \binom{m-1}{u}$.

Тройка $(m-1, n', t-1)$ допустима и справедливо неравенство $n > n'$. Параметр n удовлетворяет условию (13). Правая часть (13) в силу соотношения $\binom{m}{u} < m\binom{m-1}{u-1}$ не превосходит tmn'. Поэтому $n \leqslant tmn'$. Логарифмируя это неравенство и учитывая (11), получаем

$$\log n \lesssim \log m + \log n'. \tag{20}$$

Из (13) вытекает, что $n \leqslant m^t$, а потому $\log n \leqslant t \log m$. Подстановка этого соотношения в (20) даёт $t \log n' \gtrsim (t-1) \log n$. С учётом этого из оценки (19), применённой к допустимой тройке $(m-1, n', t-1)$, вытекает цепочка соотношений

$$s(m, n; t) \geqslant s(m, n'; t) \gtrsim \frac{t \log n'}{2(2 \log(t-1) + c)} \gtrsim \frac{(t-1) \log n}{2(2 \log(t-1) + c)},$$

приводящая к требуемому результату. ■

При растущем t разрыв между верхней и нижней оценками (9) и (17) функции $s(m, n; t)$ имеет порядок $\log t$.

6. Применение результатов о дизъюнктивных кодах

Естественным способом задания недоопределённого символа a_T является указание характеристического набора (длины m) для множества T. В данной работе этому способу соответствует задание набором длины m в алфавите $\{0, *\}$ (см. утверждение 1). Из (4) видно, что значение m не понижается и при использовании представлений в алфавите $\{0, 1, *\}$. Если недоопределённый алфавит A имеет параметры (m, n, t), то для двоичного представления его символа достаточно $s(m, n; t)$ символов 0, 1 и $*$, а для строго двоичного представления достаточно $s_0(m, n; t)$ символов 0 и $*$. Оценки этих величин в п. 3, 4 и 5 выражены через параметры n и t. Чтобы их сравнивать с длиной m естественного представления, исключим в функциях $s(m, n; t)$ и $s_0(m, n; t)$ параметр n и вместо них будем рассматривать

$$s(m; t) = \max_n s(m, n; t), \quad s_0(m; t) = \max_n s_0(m, n; t),$$

где максимумы берутся по всем допустимым значениям n. Очевидно, что значения $s(m;t)$ и $s_0(m;t)$ достигаются на алфавите A, состоящем из всех символов, допускающих не более t доопределений, и символа $*$. В силу следствия 3 и его аналога для строго двоичных представлений можно в качестве A рассматривать алфавит, образованный всеми символами, имеющими ровно t доопределений. Это означает, что величины $s(m;t)$ и $s_0(m;t)$ совпадают с минимальным числом строк t-селективной и t-дизъюнктивной матриц. В качестве следствия утверждений 7 и 13 получаем следующий факт.

Утверждение 16. Справедливы оценки

$$\begin{aligned} &s(m;2) \leqslant 12\ln m, \quad s_0(m;2) \leqslant 20{,}25\ln m,\\ &s(m;3) \leqslant 32\ln m, \quad s_0(m;3) \leqslant 37{,}92\ln m,\\ &s(m;t) \leqslant s_0(m;t) \leqslant e(t+1)^2\Big(\ln\frac{m}{t}+2\Big). \end{aligned}$$

Доказательство. Будем рассматривать алфавит, образованный всеми символами, имеющими ровно t доопределений. Для него $n = \binom{m}{t}$. Подстановка в (5), (6) и (7) верхних оценок для n, равных соответственно $m^2/2$, $m^3/6$ и $(me/t)^t$, приводит к указанным верхним оценкам для $s(m;2)$, $s(m;3)$ и $s(m;t)$. Аналогично из утверждения 13 получаются верхние оценки для $s_0(m;2)$, $s_0(m;3)$ и $s_0(m;t)$. ■

Отметим, что оценки, асимптотически совпадающие с оценками утверждений 7, 13 и 16, могут быть получены также с использованием жадного алгоритма (в другой терминологии — градиентной процедуры). Применительно к оценке для $s(m;t)$ см. [2].

В случае t-дизъюнктивных матриц метод случайного кодирования для верхней оценки величины $s_0(m;t)$ применялся многими авторами (например, в [4, 5, 13, 15]), а жадный алгоритм — в работе [16]. Все полученные оценки имеют одинаковый порядок $\mathrm{O}(t^2\log m)$, но лучшими являются оценки работы [4]. Оценка для $s_0(m;t)$ из утверждения 16 асимптотически совпадает с оценкой этой величины из [4], но способ её получения путём оценки мощности конъюнктивного базиса отличен от использованного там.

Впервые немощностная (и к данному моменту наилучшая) нижняя оценка для $s_0(m;t)$ установлена в [4]. Она относится к случаю $t = \mathrm{const}$, $m \to \infty$ и может быть представлена в виде

$$s_0(m;t) \gtrsim \frac{t^2\log m}{2\log t + c}, \tag{21}$$

где c — константа из (12).

Эта оценка доказывается достаточно сложно индукцией по t. Позднее в работе [17] было найдено чисто комбинаторное доказательство в 4 раза более слабой оценки. Затем в заметке [14] было представлено очень простое комбинаторное доказательство оценки, которая хуже (21) в 2 раза. Она вытекает из установленного в [14] соотношения (10). Мы будем основываться на последней работе, поскольку её результат удаётся распространить на случай растущего t. Имеет место следующий факт.

Утверждение 17. При условии

$$t = \mathrm{o}(\log m) \tag{22}$$

справедливы оценки

$$s_0(m;t) \gtrsim \frac{t(t+1)\log m}{2(2\log t + c)}; \tag{23}$$

$$s(m;t) \gtrsim \frac{(t-1)t\log m}{2(2\log t + c)}, \tag{24}$$

где c — константа из (12).

Доказательство. Оценка (23) легко получается из соотношения (10) с использованием неравенства $\binom{u}{v} \leqslant (ue/v)^v$ и с учётом (22). Оценка (24) вытекает из неё и соотношения $s(m;t) \geqslant s_0(m-1;t-1)$, доказываемого подобно (18). ■

Метод случайного кодирования не дает конструкции t-селективной (либо t-дизъюнктивной) матрицы, а трудоёмкость жадного алгоритма экспоненциальна. Возникает задача нахождения эффективных и конструктивных алгоритмов, обеспечивающих «достаточно хорошие» верхние оценки величины $s_0(m;t)$. Детерминированный алгоритм считается эффективным, если его трудоёмкость ограничена полиномом от размера исходных данных, и считается конструктивным, если из него извлекается явное описание объекта.

В работе [3] представлены некоторые конструкции дизъюнктивных кодов. Один из предложенных там подходов основан на использовании q-ичных кодов с большим кодовым расстоянием. Реализация этого подхода в [5] с применением кодов Рида — Соломона позволила для дизъюнктивных матриц получить достаточно хорошую эффективную оценку величины m при заданных t и s (в [3] имеется указание на использование кодов Рида — Соломона, но оценки не приведены). Применительно к задаче двоичного представления недоопределённых данных из этой оценки вытекает следующий результат.

Утверждение 18. Существует эффективный и конструктивный метод, обеспечивающий оценку

$$s(m;t) \leqslant s_0(m;t) \lesssim \left(\frac{2(t+1)\log m}{\log(t\log m)}\right)^2. \tag{25}$$

Отношение оценки (25) к оценке, полученной методом случайного кодирования, имеет порядок $\dfrac{\log m}{(\log t + \log\log m)^2}$. При больших значениях t оценка (25) может оказаться даже лучше её. Так, например, при $t = m^c$, $c = \mathrm{const} < 1/2$, оценка (25) приобретает вид $s(m,t) \lesssim 4t^2/c^2$. Для сравнения укажем нижнюю оценку $s(m,t) \gtrsim t^2/4$, вытекающую из нижней оценки Бассалыго [4] для $s_0(m;t)$ и соотношения $s(m;t) \geqslant$ $\geqslant s_0(m;t)/2$.

В приложениях больший интерес представляют малые значения t. Малыми будем считать величины t, удовлетворяющие условию (22). При его выполнении оценка (25) хуже полученной методом случайного кодирования по порядку в $\dfrac{\log m}{(\log\log m)^2}$ раз. В работе [5] приведена ещё одна эффективная конструкция, позволившая при малых t улучшить оценку. Она использует некоторое семейство алгебро-геометрических кодов Гоппы. Из полученной в [5] оценки m в функции от t и s вытекает следующий результат для двоичных представлений.

Утверждение 19. Существует эффективный и конструктивный метод, обеспечивающий оценку

$$s(m;t) \leqslant s_0(m;t) = \mathrm{O}\left(\frac{t^3 \log m}{\log t}\right).$$

Эта оценка отличается от полученной методом случайного кодирования по порядку в $\dfrac{t}{\log t}$ раз и при условии (22) лучше оценки (25). Разрыв между детерминированными эффективными оценками и полученными методом случайного кодирования устранён в работе [6]. Из неё вытекает следующий факт.

Утверждение 20. Существует эффективный метод, обеспечивающий оценку

$$s(m;t) \leqslant s_0(m;t) = \mathrm{O}\left(t^2 \log m\right).$$

Однако этот результат нельзя считать вполне удовлетворительным, поскольку метод работы [6] эффективен (полиномиален), но не конструктивен — использует дерандомизацию метода случайного кодирования.

В [18] представлен эффективный и конструктивный метод построения t-дизъюнктивных кодов, который при некоторых значениях параметров приводит к лучшим результатам, чем метод случайного кодирования.

Введённые и исследованные в данной работе селективные матрицы являются обобщениями широко применяемых в информатике дизъюнктивных матриц. Основное назначение последних — выделение из больших множеств малых подмножеств. Эту задачу можно эффективно решать с помощью селективных матриц: в дизъюнктивных матрицах выделяются столбцы, покрываемые заданным набором, а в селективных — столбцы, доопределяющие заданный недоопределённый набор. Поскольку класс селективных матриц шире, от их применения можно ожидать некоторого выигрыша.

ЛИТЕРАТУРА

1. *Шоломов Л. А.* Элементы теории недоопределённой информации // Прикладная дискретная математика. Приложение. 2009. № 2. С. 18–42.
2. *Шоломов Л. А.* Двоичное представление недоопределённых данных // Доклады Академии наук. 2013. Т. 448. № 3. С. 275–278.
3. *Kautz W. H. and Singleton R. C.* Nonrandom binary superimposed codes // IEEE Trans. Inform. Theory. 1964. V. 10. No. 4. P. 363–377.
4. *Дьячков А. Г., Рыков В. В.* Границы длины дизъюнктивных кодов // Проблемы передачи информации. 1982. Т. 18. Вып. 3. С. 7–13.
5. *Kumar R., Rajagopalan S., and Sahai A.* Coding construction for blacklisting problems without computational assumptions // LNCS. 1999. V. 1666. P. 609–623.
6. *Porat E. and Rotshchild A.* Explicit non-adaptive combinatorial group testing schemes // Automata, Languages and Programming. LNCS. 2008. V. 5125. P. 748–759.
7. *Chuzhoy J. and Khanna S.* An $\mathrm{O}(k^3 \log n)$-approximation algorithm for vertex-connectivity survivable network design //Proc. 50th Annual IEEE Symp. Foundation Computer Science (FOCS). Washington: IEEE Computer Society, 2009. P. 437–441.
8. *Dyer M., Fenner T., Frieze A., and Thomason A.* On key storage in secure networks // J. Cryptology. 1995. V. 8. P. 189–200.
9. *Mitchell C. J. and Piper F. C.* Key storage in secure networks // Discr. Appl. Math. 1988. V. 21. P. 215–228.
10. *Quinn K. A. S.* Bounds for key distribution patterns // J. Cryptology. 1999. V. 12. P. 227–239.

11. *Гэри М., Джонсон Д.* Вычислительные машины и труднорешаемые задачи. М.: Мир, 1982. 416 с.
12. *Коспанов Э. Ш.* О кодировании (0,1)-матриц конъюнкциями // Дискретный анализ. Сборник трудов. Вып. 27. Новосибирск: Институт математики СО АН, 1975. С. 17–22.
13. *Erdos P., Frankl P., and Furedi Z.* Families of finite sets in which no set is covered by the union of r others // Israel J. Math. 1985. V. 51, No. 1–2. P. 79–89.
14. *Furedi Z.* On r-cover-free families // J. Combinator. Theory. Ser. A. 1996. V. 73. P. 172–173.
15. *Cheng V., Du D.-Z., and Lin G.* On the upper bounds of the minimum number of rows of disjunct matrices // Optimizat. Lett. 2009. V. 3. Iss. 2. P. 297–302.
16. *Hwang F. K. and Sos V. T.* Non-adaptive hypergeometric group testing // Studia Scientiarum Mathecarum Hungarica. 1987. V. 22. P. 257–263.
17. *Ruszinko M.* On the upper bound of the size of the r-cover-free families // J. Combinator. Theory. Ser. A. 1994. V. 66. P. 302–310.
18. *D'yachkov A. G., Macula A. J., and Rykov V. V.* New constructions of superimposed codes // IEEE Trans. Inform. Theory. 2000. V. 46. No. 1. P. 284–290.

МАТЕМАТИЧЕСКИЕ ОСНОВЫ КОМПЬЮТЕРНОЙ БЕЗОПАСНОСТИ

DOI 10.17223/20710410/19/4

УДК 004.056.52

ПРЕДСТАВЛЕНИЕ СИСТЕМЫ СЕМАНТИЧЕСКИ ОСМЫСЛЕННОГО РОЛЕВОГО УПРАВЛЕНИЯ ДОСТУПОМ В ВИДЕ ЦВЕТНОЙ СЕТИ ПЕТРИ

Н. А. Семенова

Московский государственный институт электроники и математики (ТУ), г. Москва, Россия

E-mail: natasha_sem@inbox.ru

Рассматривается механизм внесения изменений в систему семантически осмысленного ролевого управления доступом в рамках СК-РУД модели одновременно несколькими администраторами. С использованием математического аппарата сетей Петри описываются процессы перехода системы между состояниями и обосновывается условие безопасности переходов.

Ключевые слова: *ролевое управление доступом, автоматизация управления ролями, сети Петри.*

Введение

Система семантически осмысленного ролевого управления доступом основана на модели ролевого управления доступом с семантическим контекстом СК-РУД [1] и позволяет автоматизировать назначение и отзыв ролей пользователям при наступлении в системе ряда определённых событий. Классическая модель ролевого управления доступом *RBAC* [2], расширением которой является модель СК-РУД, содержит описание процессов внесения изменений в конфигурацию в предположении, что в каждый момент времени в системе действует не более одного субъекта с административными привилегиями. В системах ролевого управления доступом в компьютерных системах с десятками тысяч пользователей и тысячами привилегий администрирование, как правило, осуществляется несколькими пользователями. Данный подход получил название «распределённого администрирования» системы. Применение классического подхода для описания функционирования таких систем может привести к конфликтам между действиями различных администраторов. Примерами таких конфликтов могут быть ситуации, когда два администратора одновременно вносят изменения в учётную запись пользователя или заявка на назначение роли зарегистрирована и назначена одному из администраторов раньше, чем другому администратору назначена заявка на отзыв роли, взаимоисключающей первую роль. Исходя из этого, целесообразно рассмотреть вопрос формального описания процессов распределённого администрирования в системах ролевого управления доступом и определения условий безопасного перехода системы между состояниями.

1. Основные понятия и определения

В данной работе используются следующие обозначения для элементов СК-РУД модели:

P — множество привилегий или полномочий;

A — множество атрибутов учетных записей пользователей;

$V = \{(v_{ij})\}$, где i — номер атрибута из A; $j \in [1, N_i]$; N_i — число возможных различных значений атрибута $a_i \in A$;

$values : A \to 2^V$ — функция, задающая для каждого атрибута множество его допустимых значений;

U — множество атрибут-пользователей (множество векторов вида $(ux_1, \ldots, ux_{na})$, где $ux_i \in values(a_i)$);

R — множество ролей системы;

$UA : U \to 2^R$ — функция, задающая для каждого пользователя множество ролей, на которые он может быть авторизован;

S — множество сессий пользователей системы;

$PA : R \to 2^P$ — функция, задающая для каждой роли множество привилегий. При этом для каждой привилегии $p \in P$ существует роль $r \in R$, такая, что $p \in PA(r)$;

$user : S \to U$ — функция, задающая для каждой сессии пользователя, от имени которого она авторизована;

$roles : S \to 2^R$ — функция, задающая для пользователя множество ролей, на которые он авторизован в данной сессии. При этом в каждый момент времени для каждой сессии $s \in S$ выполняется условие $roles(s) \in UA(user(s))$;

CR — множество предварительных условий назначения ролей;

AP — множество административных привилегий;

$AR \subseteq 2^{AP}$ — множество административных ролей;

$APA : AR \to 2^{AP}$ — функция, задающая для каждой административной роли множество административных привилегий; при этом для каждой привилегии $p \in AP$ существует роль $r \in AR$, такая что $p \in APA(r)$;

$can_assign : AR \to CR \times 2^R$ — функция, определяющая для каждой административной роли множество ролей, которые могут быть назначены пользователю с использованием данной административной роли при выполнении заданных предварительных условий CR;

$can_revoke : AR \to 2^R$ — функция, определяющая для каждой административной роли множество ролей, которые могут быть отозваны у пользователя с использованием данной административной роли.

Определение 1. Две роли $r_1, r_2 \in R$ называются статически взаимоисключающими, если они не могут быть назначены пользователю одновременно: для $s \in S$ выполняется неравенство $\mid \{r_1, r_2\} \cap roles(s) \mid \leqslant 1$.

Определение 2. Две роли $r_1, r_2 \in R$ называются динамически взаимоисключающими, если они не могут быть активированы сессией пользователя одновременно: для $u \in U$ выполняется неравенство $\mid \{r_1, r_2\} \cap UA(u) \mid \leqslant 1$.

Определение 3. Иерархией ролей «по предусловию» RH [1] будем называть заданное на множестве ролей R отношение строгого порядка «$>$». При этом по определению (для пользователя $u \in U$) если $(r_1, r_2) \in RH$, $r_1 > r_2$ и $c_{r_1}(u) = \texttt{true}$, то $c_{r_2}(u) = \texttt{true}$ и $c_{r_1}(u) = c_{r_1}(z_1(u), \ldots, c_{r_2}(u), \ldots, z_k(u))$. Роль r_1 будем называть предком r_2 «по предусловию».

Определение 4. Пусть $t_i : U \to \{\texttt{false}, \texttt{true}\}$ — функция, такая, что $t_i(u)$=`true` тогда и только тогда, когда $t_i \in values(a_i)$ и для $u \in U$ выполняется условие $ux_i = t_i$, где $a_i \in A$, $1 \leqslant i \leqslant |A|$.

Пусть $ca_r : U \to \{\texttt{false}, \texttt{true}\}$ — функция, такая, что $ca_r\,(u) = ca_r\,(t_1\,(u), \ldots, t_{na}\,(u))$, $u \in U$, $r \in R$, где $ca_r(y_1, \ldots, y_{na})$ — булева функция от na переменных, $na \in \{1, \ldots, |A|\}$. Тогда $CA = \{ca_{r1}, \ldots, ca_{rt}\}$ — множество функций, определяющих, при каких ограничениях на атрибуты учетной записи пользователя ему может быть назначена некоторая роль $r_i \in R$.

Дадим основные понятия и определения теории цветных сетей Петри (ЦСП), которые потребуются для дальнейшего описания результатов исследования [3]:

Q_c — множество цветов;

TK — множество токенов. По определению токен — это кортеж вида $tk = (x_1, \ldots, x_n)$, где x_i — переменная некоторого заданного цвета. Цвета токенов могут быть простыми или составными. Множество допустимых значений для токенов составных цветов — это декартово произведение множеств допустимых значений для входящих в его состав простых цветов;

$C_{tk} : TK \to Q_c$ — функция окрашивания, задающая для каждого токена его цвет;

Pl — множество мест (элементов $pl \in Pl$, каждый из которых содержит один или несколько токенов определенного цвета);

$C_{pl} : Pl \to Q_c$ — функция, определяющая множество допустимых цветов для данного места.

Множество цветов токенов, которые могут содержаться на месте pl, называется типом места pl. Множество токенов на месте pl называется содержимым места и обозначается $tk(pl)$.

Определение 5. Пусть Pl — конечное множество мест. Пусть $N_j(pl_i)$ — количество токенов каждого из цветов на месте $pl_i \in Pl$ в момент времени j, где $1 \leqslant i \leqslant |Pl|$. Пусть в момент времени $j+1$ существует такое $1 \leqslant k \leqslant |Pl|$, что $N_{j+1}(pl_k) \neq N_j(pl_k)$. Тогда будем говорить, что количество токенов на месте pl_k изменилось в результате активации перехода. Множество переходов будем обозначать T.

Определение 6. Дуга — это пара вида (pl_i, t_j), где $pl_i \in Pl$, $t_i \in T$. Множество всех дуг будем обозначать Ac.

Каждый переход инцидентен некоторой дуге, задающей правила его активации в зависимости от токенов, расположенных на местах, инцидентных той же дуге.

Дуги могут принадлежать к одному из следующих трех типов:

— NA — нормальные дуги;
— RA — ингибиторные дуги;
— IA — только чтение.

Дугу типа i будем обозначать ac_i.

Будем различать два направления дуг:

— входящие дуги, соединяющие место $pl \in Pl$ и переход $t \in T$;
— исходящие дуги, соединяющие переход $t \in T$ и место $pl \in Pl$.

Входящие дуги могут принадлежать к любому из трёх перечисленных типов и связаны с функциями чтения или удаления токена с места pl. Исходящие дуги всегда имеют тип NA и связаны с функциями добавления токенов на место pl.

Определение 7. Для каждой дуги $ac \in Ac$ определим значение количественной функции E, задающей для каждых места $pl \in Pl$ и перехода $t \in T$, инцидентных

дуге ac, количество токенов заданных цветов для возможности активации перехода t, которое либо должно содержать место pl, когда ac — входящая дуга, либо размещается на месте pl, когда ac — исходящая дуга:

$$E(ac) = m_1 \cdot tk_1 + m_2 \cdot tk_2 + \cdots + m_n \cdot tk_n,$$

где m_i — целые неотрицательные числа; tk_i — токены цвета $i \in C_{pl}(pl)$; $n = |C_{pl}(pl)|$.

Пример 1. Количественная функция $E(ac) = 3 \cdot \langle tk_1 \rangle + 2 \cdot \langle tk_2 \rangle$ задаёт для дуги ac условие активации перехода, инцидентного данной дуге: место $pl \in Pl$, инцидентное дуге, должно содержать три токена цвета tk_1 и два токена цвета tk_2.

Примечание. Если коэффициент m_i количественной функции равен 1, то вместо $1 \cdot tk$ будем писать tk.

2. Представление системы семантически осмысленного ролевого управления доступом в виде ЦСП

2.1. Цвета токенов и мест

В работе используются следующие цвета для обозначения элементов СК-РУД модели:

- $USER\langle u \rangle$ — идентификатор пользователя. Допустимыми значениями для токенов данного цвета являются все пользователи $u \in U$;
- $ROLE\langle r \rangle$ — идентификатор роли. Допустимыми значениями для токенов данного цвета являются все роли $r \in R$;
- $ADMROLE\langle ar \rangle$ — идентификатор административной роли. Допустимые значения: $ar \in AR$;
- $SESSION\langle s \rangle$ — идентификатор сессии. Допустимые значения: $s \in S$;
- $ACOND\langle ca_r \rangle$ — идентификатор атрибут-условия назначения роли. Допустимые значения: $ca_r \in CA$;
- $COMMAND\langle command \rangle$ — идентификатор действия. Допустимые значения: $command \in \{assign_role, revoke_role, auto_assign_role, auto_revoke_role, take_role, remove_role, auto_recalculate\}$;
- $ROLE_ACOND\langle ca_r, r \rangle$ — сопоставление роли и правила ее автоматического назначения. Множество допустимых значений — декартово произведение $ACOND \times \times ROLE$;
- $USER_ROLE\langle u, r \rangle$ — назначение роли пользователю. Множество допустимых значений — $USER \times ROLE$;
- $UAS\langle u, r, s \rangle$ — текущая роль пользователя в сессии. Множество допустимых значений — $USER \times ROLE \times SESSION$;
- $USER_ADMROLE\langle u, ar \rangle$ — назначение административной роли. Множество допустимых значений — $USER \times ADMROLE$;
- $CMD\langle command, u_1, u_2, r, s \rangle$ — токен действия, выполняемого сессией s от имени пользователя u_1. Множество допустимых значений — декартово произведение $COMMAND \times USER \times USER \times ROLE \times SESSION$. Если $u_1 = u_2$, то сессия выполняет операцию над пользователем, от имени которого она активирована.

Рассмотрим цвета мест и токенов, которые могут на них содержаться:

- ETG — генератор токенов. Хранит множество действий, разрешённых в текущем состоянии системы. Для выполнения каждого перехода необходимо, чтобы в ETG содержался токен соответствующего действия. Допустимым цветом токенов для данного места является CMD;

— UA — авторизованные роли. Токен $\langle u, r\rangle$ на этом месте означает, что пользователь u авторизован на роль r. Допустимый цвет токенов $USER_ROLE$;
— AUA — авторизованные административные роли. Токен $\langle u, ar\rangle$ на этом месте означает, что пользователь u авторизован на административную роль ar. Допустимый цвет токенов $USER_ADMROLE$;
— UAS — активация роли сессией пользователем. Каждый токен $\langle u, r, s\rangle$ на этом месте означает, что сессия s активирована от имени пользователя u, обладающего ролью r. Допустимый цвет токенов UAS;
— RCA — правила назначения ролей. Токен $\langle ca_r, r\rangle$ означает, что роли r соответствует условие назначения ca_r. Допустимый цвет токенов $ROLE_ACOND$.

2.2. Количественные и ограничительные функции

Количественная и ограничительная функции дуг используются для представления системы ролевого управления доступом в виде ЦСП, так как данные функции представляют возможность моделировать ограничения взаимного исключения ролей и иерархические связи между ролями, задавая пред- и постусловия переходов системы между состояниями.

Рассмотрим, каким образом происходит активация перехода, связанного с дугой заданного типа и направления.

Для возможности активации перехода t по входящей дуге $ac_{NA} = (pl, t)$ каждое место pl, соединённое с переходом t дугой ac_{NA}, должно содержать токенов заданных цветов не меньше, чем указано количественной функцией дуги $E(ac)$, т. е. не менее m_i токенов цвета tk_i. Переход в момент времени j возможен, если для всех мест pl, инцидентных дуге ac, и для всех цветов i выполняется неравенство $N_j(pl, i) \geqslant m_i$, где $N_j(pl, i)$ — количество токенов цвета i на месте pl в момент времени j, а коэффициенты m_i задаются количественной функцией дуги $E(ac)$. В результате активации перехода токены удаляются из места pl: для всех $1 \leqslant i \leqslant n$ выполняется $N_{j+1}(pl, i) = N_j(pl, i) - m_i$.

Пример 2. Необходимым условием перехода, в результате которого сессия s от имени администратора $admin$ назначит роль r пользователю u, является наличие токена цвета $CMD\langle assign_role, admin, u, r, s\rangle$ на месте ETG. После активации перехода соответствующий токен удаляется, предотвращая повторное назначение роли. Таким образом, значение количественной функции для входящей нормальной дуги, соединяющей место ETG и переход $assign_role$, равно $\langle CMD\rangle$.

Переход по исходящей дуге ac_{NA} в момент времени j активируется в результате завершения всех входящих переходов, инцидентных данной дуге. В результате активации перехода по исходящей дуге $ac_{NA} = (t, pl)$ соответствующее количество токенов, равное $m_1 \cdot tk_1 + m_2 \cdot tk_2 + \cdots + m_n \cdot tk_n$, добавляется на место pl: $N_{j+1}(pl, i) = N_j(pl, i) + m_i$.

Пример 3. Результатом перехода, в рамках которого сессия от имени администратора $admin$ назначает роль r пользователю u, является помещение токена цвета $USER_ROLE\langle u, r\rangle$ на место UA, означающее, что данный пользователь авторизован на роль r. Таким образом, значение количественной функции для соответствующей исходящей нормальной дуги равно $\langle USER_ROLE\rangle$.

Для возможности активации перехода t по входящей дуге $ac_{RA} = (pl, t)$ в момент времени j каждое место pl, соединённое с переходом t, должно содержать токенов заданного цвета и значения не меньше, чем указано количественной функцией дуги $E(ac_{RA})$. Переход в момент времени j возможен, если для всех мест, инцидентных дуге ac, и для всех цветов i выполняется неравенство $N_j(pl, i) \geqslant m_i$. В результате активации перехода количество токенов на месте pl не изменяется: $N_{j+1}(pl, i) = N_j(pl, i)$.

Пример 4. Если для некоторой роли $r \in R$ существует роль $r' \geqslant r$, то необходимым условием перехода, в результате которого сессия от имени администратора назначит роль r пользователю u, является наличие токенов цвета $USER_ROLE\langle u, r\rangle$ на месте UA для каждой из ролей-предков r. Каждый из токенов означает, что данный пользователь авторизован на соответствующую роль. После завершения перехода токены $USER_ROLE$ остаются на месте UA. Таким образом, значение количественной функции для дуги RA равно $\mid \{r' : r' \geqslant r\} \mid \cdot\langle USER_ROLE\rangle$.

Для входящей дуги ингибиторного типа $ac_{IA} = (pl, t)$ все коэффициенты m_i, за исключением одного m_j, равны нулю. Для возможности активации перехода t по входящей дуге $ac_{IA} = (pl, t)$ типа IA каждое место входа pl, соединённое с переходом t, должно содержать не более m_j токенов заданного цвета j. В результате активации перехода по дуге ac_{IA} количество токенов на месте pl не изменяется: $N_{j+1}(pl, i) = N_j(pl, i)$.

Пример 5. Пусть для роли r существуют статически взаимоисключающие роли. Тогда необходимым условием перехода, в результате которого сессия от имени администратора назначит роль r пользователю u, является отсутствие токенов цвета $USER_ROLE\langle u, r\rangle$ на месте UA, означающее, что данный пользователь не авторизован на роли, взаимоисключающие с r. Таким образом, значение количественной функции для соответствующей входящей ингибиторной дуги равно $\langle USER_ROLE\rangle$.

Введём определение ограничительной функции дуги G.

Определение 8. Множество ограничительных функций дуги G — это множество, включающее в себя ограничения на значения токенов, которые должны находиться на местах, инцидентных входящей дуге, для того чтобы стала возможна инициация перехода, инцидентного дуге. Пусть $v_p : TK \to \{\texttt{true}, \texttt{false}\}$ — одна из следующих функций:

- $assigned_r : U \to \{\texttt{true}, \texttt{false}\}$ — функция, такая, что $assigned_r(u) = \texttt{true}$ тогда и только тогда, когда $r \in UA(u)$;
- $ca_r : U \to \{\texttt{true}, \texttt{false}\}$— функция, такая, что $ca_r(u) = \texttt{true}$ тогда и только тогда, когда атрибуты пользователя $u \in U$ соответствуют атрибут-условию назначения роли $r \in R$;
- $static_conflict_role_r : R \to \{\texttt{true}, \texttt{false}\}$ — функция, такая, что $static_conflict_role_r(r') = \texttt{true}$ тогда и только тогда, когда r' является статически взаимоисключающей для r;
- $dynamic_conflict_role_r : R \to \{\texttt{true}, \texttt{false}\}$ — функция, такая, что $dynamic_conflict_role_r(r') = \texttt{true}$ тогда и только тогда, когда r' является динамически взаимоисключающей для r;
- $can_assign_{ar} : R \to \{\texttt{true}, \texttt{false}\}$ — функция, такая, что $can_assign_{ar}(r) = \texttt{true}$ тогда и только тогда, когда для $r \in R$ и $ar \in AR$ выполняется условие $r \in \in can_assign(ar)$;
- $can_revoke_{ar} : R \to \{\texttt{true}, \texttt{false}\}$ — функция, такая, что $can_revoke_{ar}(r) = \texttt{true}$ тогда и только тогда, когда для $r \in R$ и $ar \in AR$ выполняется условие $r \in \in can_revoke(ar)$;
- $prec_role_r : R \to \{\texttt{true}, \texttt{false}\}$ — функция, такая, что $prec_role_r(r') = \texttt{true}$ тогда и только тогда, когда $r > r'$ в иерархии RH.

Пусть $G_{ac} : Pl \to \{\texttt{true}, \texttt{false}\}$ — функция, такая, что $G_{ac}(pl) = G_{ac}(v_p(tk_1, tk'_1), \ldots, v_p(tk_m, tk'_m))$, $ac \in Ac$, $pl \in Pl$, pl инцидентно ac, $tk_1, \ldots, tk_m$, $tk'_1, \ldots, tk'_m \in TK$, $C_{tk}(tk_1) \subseteq C_{pl}(pl)$, где $1 \leqslant i \leqslant m$ и $G_{ac}(y_1, \ldots, y_m)$ — булева функция от m перемен-

ных. Тогда $\{G_{ac}(pl_1), \ldots, G_{ac}(pl_t)\}$ — множество функций, определяющих, при каких условиях может быть активирован переход, инцидентный соответствующей дуге.

Если с некоторой входящей дугой $ac \in Ac$ связана ограничительная функция G_{ac}, то соответствующий дуге переход t может быть активирован только в том случае, если для всех мест, инцидентных данной дуге, $G_{ac}(pl) = \texttt{true}$. Количественная и ограничительная функции для i-й дуги j-го типа обозначаются как Ei_j и G_i соответственно.

Пример 6. Пусть для некоторой роли $r \in R$ существует $r' \geqslant r$. Тогда необходимым условием перехода, в результате которого сессия от имени администратора $admin$ назначит роль r_j пользователю u, является наличие на месте UA токена цвета $USER_ROLE$, имеющего значение $\langle u, r_i\rangle$, для которого $assigned_{r_i}(u) = \texttt{true}$. После завершения перехода токен $\langle u, r_i\rangle$ остается на месте UA. Таким образом, значение ограничительной функции входящей дуги RA равно $G_{ac} = assigned_{r_i}(u)$.

Пример 7. Пусть $r_i, r_j \in R$ — две статически взаимоисключающие роли. Тогда необходимым условием перехода, в результате которого сессия от имени администратора $admin$ назначит роль r_j пользователю u, является отсутствие на месте UA токена цвета $USER_ROLE$ со значением $\langle u, r_i\rangle$, для которого $assigned_{r_i}(u) = \texttt{true}$. Таким образом, значение ограничительной функции входящей ингибиторной дуги равно $G_a(\neg assigned_{r_i}(u))$, где $\neg$ — оператор логического отрицания.

По определению [3], цветная сеть Петри (ЦСП) — это кортеж $CPN = (Q_c, TK, Pl, T, Ac, C_{tk}, C_{pl}, G, E)$. Состояние ЦСП в каждый момент времени задается размещением токенов на каждом из Pl мест.

Определение 9. Состояние ЦСП M_n называется достижимым из состояния M_0, если существует конечная последовательность переходов $t_0, t_1, \ldots, t_{n-1} \in T$, переводящая ЦСП из состояния M_0 в состояние M_n. Начальное состояние M_0 является достижимым по определению.

Переход сети в новое состояние может быть активирован только в том случае, если хотя бы для одного из переходов $t \in T$ выполнены все предусловия, задаваемые ограничительными и количественными функциями дуг.

В данной работе для описания системы СК-РУД используются следующие функции дуг и ограничений:
$E1_{NA} = CMD\langle assign_role, u, u', r, s\rangle$;
$E2_{RA} = E11_{RA} = E40_{RA} = E43_{RA} = E58_{RA} = USER_ADMROLE\langle u', ra\rangle$;
$E3_{IA} = E19_{IA} = USER_ROLE\langle u, r_c\rangle$;
$E4_{RA} = E20_{RA} = USER_ROLE\langle u, r_{pc}\rangle$;
$E5_{IA} = E6_{NA} = E9_{NA} = E14_{IA} = E15_{IA} = E17_{NA} = E21_{IA} = E22_{NA} = E24_{NA} = = E28_{IA} = E29_{IA} = E31_{NA} = E45_{IA} = E55_{IA} = E63_{IA} = USER\langle u\rangle$;
$E7_{NA} = E8_{RA} = E12_{RA} = E16_{NA} = E23_{NA} = E26_{RA} = E30_{NA} = E33_{RA} = E41_{RA} = = E48_{RA} = E53_{IA} = E59_{RA} = USER_ROLE\langle u, r\rangle$;
$E10_{NA} = E39_{NA} = E57_{RA} = CMD\langle revoke_role, u, u', r, s\rangle$;
$E13_{IA} = E27_{IA} = E42_{RA} = E60_{RA} = USER_ROLE\langle u, r_d\rangle$;
$E18_{NA} = E56_{NA} = CMD\langle auto_assign_role, u, system, r, s\rangle$;
$E25_{NA} = CMD\langle auto_revoke_role, u, system, r, s\rangle$;
$E32_{NA} = CMD\langle take_role, u, u, r, s\rangle$;
$E34_{IA} = UAS\langle u, r_{dc}, s\rangle$;
$E35_{NA} = E37_{NA} = E38_{NA} = E61_{RA} = UAS\langle u, r, s\rangle$;
$E36_{NA} = CMD\langle remove_role, u, u, r, s\rangle$;
$E44_{IA} = E46_{NA} = CMD\langle revoke_role, u, u', rd, s\rangle$;

$E47_{RA} = E51_{RA} = CMD\langle auto_recalculate, u, system, r, s\rangle$;
$E49_{IA} = E50_{NA} = CMD\langle auto_revoke_role, u, system, r, s\rangle$;
$E52_{RA} = ROLE_ACOND\langle ca_r, r\rangle$;
$E54_{IA} = USER_ROLE\langle u, r_{old}\rangle$;
$E62_{IA} = CMD\langle revoke_role, u, u, r, s\rangle + CMD\langle auto_revoke_role, u, system, r, s\rangle$;
$G_2 = \langle u', r_a\rangle : can_assign_{r_a}(r) = \texttt{true}$;
$G_3 = \langle u, r_c\rangle : static_conflict_role_r(r_c) = \texttt{false}$;
$G_4 = \langle u, r_{pc}\rangle : prec_role_r(u) = \texttt{true}$;
$G_{11} = \langle u', r_a\rangle : can_revoke_{r_a}(r) = \texttt{true}$;
$G_{13} = \langle u, r_d\rangle : prec_role_r(r_d) = \texttt{false}$;
$G_{19} = \langle u, r_c\rangle : static_conflict_role_r(r_c) = \texttt{false}$;
$G_{20} = \langle u, r_{pc}\rangle : prec_role_r(u) = \texttt{true}$;
$G_{27} = \langle u, r_d\rangle : prec_role_r(r_d) = \texttt{false}$;
$G_{33} = \langle u, r\rangle : assigned_r(u) = \texttt{true}$;
$G_{34} = \langle u, r_{dc}, s\rangle : dynamic_conflict_role_r(r_{dc}) = \texttt{false}$;
$G_{40} = \langle u', r_a\rangle : can_revoke_{r_a}(r) = \texttt{true}$;
$G_{42} = \langle u, r_d\rangle : prec_role_r(r_d) = \texttt{true}$;
$G_{43} = \langle u', r'_a\rangle : can_revoke_{r'_a}(r_d) = \texttt{true}$;
$G_{48} = \langle u, r\rangle : ca_r(u) = \texttt{false}$;
$G_{52} = \langle ca_r, r\rangle : ca_r(u) = \texttt{true}$;
$G_{53} = \langle u, r\rangle : ca_r(u) = \texttt{true}$;
$G_{54} = \langle u, r_{old}\rangle : ca_{r_{old}}(u) = \texttt{false}$.

3. Переходы сети Петри между состояниями

Рассмотрим переходы следующих типов:

— назначение роли ($assign_role$): сессия s администратора u' назначает пользователю $u \in U$ роль r;
— отзыв роли ($revoke_role$): роль r отзывается у пользователя u сессией s администратора u';
— автоматическое назначение роли ($auto_assign_role$): роль r назначается пользователю u по атрибут-условию ca_r;
— автоматический отзыв ($auto_revoke_role$): роль r отзывается у пользователя u, если он более не удовлетворяет условию ca_r;
— перерасчёт списка ролей пользователя при изменении его атрибутов ($auto_recalculate$): в системе создаётся очередь команд на отзыв и назначение ролей в соответствии с новыми атрибутами учётной записи пользователя;
— активация ($take_role$): активация роли r сессией s от имени пользователя u;
— деактивация ($remove_role$): роль r удаляется из списка активных ролей для заданной сессии s пользователя u.

Переход может быть совершён в любой момент, если он разрешён в данном состоянии действующими ограничениями. Любой из данных переходов переводит систему в новое состояние. Рассмотрим подробнее переходы в системе ролевого управления доступом, представленной в виде ЦСП.

Каждый из рассматриваемых переходов в ЦСП описывается следующим образом: сначала перечисляются входящие дуги с указанием в скобках их типа и связанных количественных и ограничительных функций. Результат перехода описывается с помощью перечисления исходящих дуг с указанием их количественных функций.

3.1. Переход «Назначение роли»

Правила активации перехода:

— (Дуга с количественной функцией $E1_{NA}$.) Существует токен $\langle assign_role, u, u', r, s\rangle$ на месте ETG, означающий, что необходимо выполнить назначение роли r пользователю u сессией от имени администратора u'.
— ($E2_{RA}$). На месте AUA должен существовать токен $\langle u', r_a\rangle$, для которого $can_assign_{r_a}(r) = \texttt{true}$ (ограничительная функция G_2).
— ($E3_{IA}$). Место UA не содержит токенов $\langle u, r_c\rangle$, для которых $static_conflict_role_r(r_c) = \texttt{true}$ (G_3: пользователю не назначены роли, взаимоисключающие с ролью r).
— ($E4_{RA}$). Место UA содержит токены $\langle u, r_{pc}\rangle$, для которых $prec_role_r(r_{pc}) = \texttt{true}$ (G_4: пользователю назначены все роли, необходимые для назначения роли r).
— ($E5_{IA}$). Место UB не содержит токена $\langle u\rangle$ (над пользователем в данный момент не выполняется никаких действий).

Результат перехода: данный переход ($E7_{NA}$) помещает токен $\langle u, r\rangle$ на место UA, что означает авторизацию пользователя u на роль r. На время выполнения перехода на место UB помещается токен $\langle u\rangle$ ($E6_{NA}$), чтобы блокировать одновременное выполнение любых других действий над пользователем u. После активации перехода команда $\langle assign, u, u', r, s\rangle$ удаляется с места ETG ($E1_{NA}$). После выполнения перехода ($E8_{RA}$) токен $\langle u\rangle$ удаляется с места UB ($E9_{NA}$).

3.2. Переход «Отзыв роли»

Правила активации перехода:

— ($E10_{NA}$). Существует токен $\langle revoke_role, u, u', r, s\rangle$ на месте ETG, означающий, что необходимо выполнить отзыв роли r у пользователя u сессией от имени администратора u'.
— ($E11_{RA}$). На месте AUA существует токен $\langle u', r_a\rangle$, для которого $can_revoke_{r_a}(r) = \texttt{true}$ (G_{11}).
— ($E12_{RA}$). Место UA содержит токен $\langle u, r\rangle$ (нельзя отозвать роль, если она не назначена).
— ($E13_{IA}$). Место UA не содержит множество токенов $\{\langle u, r_d\rangle\}$, таких, что $prec_role_r(r_d) = \texttt{true}$ (G_{13}: нельзя отозвать роль, предварительно не отозвав роли, которые от неё зависят).
— ($E14_{IA}$). Место UAS не содержит токена $\langle u, r, s\rangle$ (G_{14}: нельзя отозвать роль, если она активирована сессией пользователя).
— ($E15_{IA}$). Место UB не содержит токена $\langle u\rangle$ (над пользователем в данный момент не выполняется никакой операции).

Результат перехода: данный переход ($E17_{NA}$) удаляет токен $\langle u, r\rangle$ с места UA, что означает отзыв роли r у пользователя u. На время выполнения перехода на место UB помещается токен $\langle u\rangle$ ($E16_{NA}$), чтобы блокировать одновременное выполнение любых других действий над пользователем u. После активации перехода команда $\langle revoke_role, u, u', r, s\rangle$ удаляется с места ETG ($E10_{NA}$). После выполнения перехода токен $\langle u\rangle$ удаляется с места UB.

3.3. Переход «Автоматическое назначение роли»

Правила активации перехода:

— ($E18_{NA}$). Существует токен $\langle auto_assign_role, u, system, r, s\rangle$ на месте ETG, означающий, что необходимо выполнить назначение роли r пользователю u сессией от имени пользователя $system$.

— ($E19_{IA}$). Место UA не содержит токенов $\langle u, r_c\rangle$, для которых $static_conflict_role_r(r_c) = \texttt{true}$ (G_{19}: пользователю не назначены роли, взаимоисключающие с ролью r).

— ($E20_{RA}$). Место UA содержит токены $\langle u, r_{pc}\rangle$, для которых $prec_role_r(r_{pc}) = \texttt{true}$ (G_{20}: пользователю назначены все роли, необходимые для назначения роли r).

— ($E21_{IA}$). Место UB не содержит токена $\langle u\rangle$ (над пользователем в данный момент не выполняется никаких действий).

Результат перехода: данный переход ($E23_{NA}$) помещает токен $\langle u, r\rangle$ на место UA, что означает авторизацию пользователя u на роль r. На время выполнения перехода на место UB помещается токен $\langle u\rangle$ ($E22_{NA}$), чтобы блокировать одновременное выполнение любых других действий над пользователем u. После активации перехода команда $\langle auto_assign_role, u, system, r, s\rangle$ удаляется с места ETG ($E18_{NA}$). После выполнения перехода токен $\langle u\rangle$ удаляется с места UB ($E24_{NA}$).

3.4. Переход «Автоматический отзыв роли»

Правила активации перехода:

— ($E25_{NA}$). Существует токен $\langle auto_revoke_role, u, system, r, s\rangle$ на месте ETG, означающий, что сессии от имени пользователя $system$ необходимо выполнить отзыв роли r у пользователя u.

— ($E26_{RA}$). Место UA содержит токен $\langle u, r\rangle$ (нельзя отозвать роль, если она не назначена).

— ($E27_{IA}$). Место UA не содержит множество токенов $\{\langle u, r_d\rangle\}$, таких, что $prec_role_r(r_d) = \texttt{true}$ (нельзя отозвать роль, предварительно не отозвав роли, которые от неё зависят).

— ($E28_{IA}$). Место UAS не содержит токена $\langle u, r, s\rangle$ (G_{14}: нельзя отозвать роль, если она активирована одной из сессий пользователя).

— ($E29_{IA}$). Место UB не содержит токена $\langle u\rangle$ (над пользователем в данный момент не выполняется никаких действий).

Результат перехода: данный переход ($E31_{NA}$) удаляет токен $\langle u, r\rangle$ с места UA, что означает отзыв роли r у пользователя u. На время выполнения перехода на место UB помещается токен $\langle u\rangle$ ($E30_{NA}$), чтобы блокировать одновременное выполнение любых других действий над пользователем u. После активации перехода команда $\langle auto_revoke_role, u, system, r, s\rangle$ удаляется с места ETG ($E25_{NA}$). После выполнения перехода токен $\langle u\rangle$ удаляется с места UB.

3.5. Переход «Активация роли»

Правила активации перехода:

— ($E32_{NA}$). Существует токен $\langle take_role, u, u, r, s\rangle$ на месте ETG, означающий, что сессия s от имени пользователя u активирует роль r.

— ($E33_{RA}$). Место UA содержит токен $\langle u, r\rangle$ (сессия пользователя может активировать только те роли, на которые он авторизован).

— ($E34_{IA}$). Место UAS не содержит множество токенов $\{\langle u, r_{dc}, s\rangle\}$, таких, что $dynamic_conflict_role_r(r_{dc}) = \texttt{true}$ (G_{34}: сессия пользователя не может активи-

ровать роль r, если для этой сессии в настоящий момент активированы роли, динамически взаимоисключающие с ролью r).

Результат перехода: данный переход ($E35_{NA}$) помещает токен $\langle u, r, s\rangle$ на место UAS, что означает активацию роли r сессией s от имени пользователя u.

3.6. Переход «Деактивация роли»

Правила активации перехода:

— ($E36_{NA}$). Существует токен $\langle remove, u, u, r, s\rangle$ на месте ETG, означающий, что сессия s пользователя u деактивирует роль r.
— ($E37_{NA}$). Место UAS содержит токен $\langle u, r, s\rangle$ (сессия пользователя не может деактивировать роль, которая не активирована для данной сессии).

Результат перехода: данный переход ($E38_{NA}$) удаляет токен $\langle u, r, s\rangle$ с места UAS, что означает деактивацию роли r сессией s от имени пользователя u.

3.7. Переход «Перерасчет зависимых ролей при удалении роли»

Правила активации перехода:

— ($E39_{NA}$). Существует токен $\langle revoke_role, u, u', r, s\rangle$ на месте ETG, означающий, что необходимо выполнить команду отзыва роли r у пользователя u сессией от имени администратора u'.
— ($E40_{RA}$). На месте AUA существует токен $\langle u', r_a\rangle$, для которого $can_revoke_{r_a}(r) = \texttt{true}$ (G_{40}).
— ($E41_{RA}$). Место UA содержит токен $\langle u, r\rangle$ (нельзя отозвать роль, если она не назначена).
— ($E42_{RA}$). Место UA содержит хотя бы один токен $\langle u, r_d\rangle$, такой, что $prec_role_r(r_d) = \texttt{true}$.
— ($E43_{RA}$). На месте AUA существует токен $\langle u', r'_a\rangle$, для которого $can_revoke_{r'_a}(r_d) = \texttt{true}$ (G_{43}).
— ($E44_{IA}$). Не существует токена $\langle revoke_role, u, u', r_d, s\rangle$ на месте ETG.
— ($E45_{IA}$). Место UB не содержит токена $\langle u\rangle$ (над пользователем в данный момент не выполняется никаких действий).

Результат перехода: данный переход ($E46_{NA}$) добавляет токен $\langle revoke, u, u', r_d, s\rangle$ на место ETG, что означает команду на отзыв роли r_d, зависимой от роли r, у пользователя u.

3.8. Переход «Перерасчет ролей для удаления при изменении атрибутов»

Правила активации перехода:

— ($E47_{RA}$). Существует токен $\langle auto_recalculate, u, system, r, s\rangle$ на месте ETG, означающий, что атрибуты учётной записи пользователя u были изменены и требуется перерасчет его ролей.
— ($E48_{RA}$). Место UA содержит хотя бы один токен $\langle u, r\rangle$, такой, что $ca_r(u) = \texttt{false}$.
— ($E49_{IA}$) Не существует токена $\langle auto_revoke_role, u, system, r, s\rangle$ на месте ETG.

Результат перехода: данный переход ($E50_{NA}$) добавляет токен $\langle auto_revoke_role, u, system, r, s\rangle$ на место ETG, что означает команду на автоматический отзыв роли r у пользователя u.

3.9. Переход «Перерасчет ролей для назначения при изменении атрибутов»

Правила активации перехода:

— $(E51_{RA})$. Существует токен $\langle auto_recalculate, u, system, r, s\rangle$ на месте ETG, означающий, что атрибуты учётной записи пользователя u были изменены и требуется перерасчет его ролей.
— $(E52_{RA})$. Место RCA содержит хотя бы один токен $\langle ca_r, r\rangle$, такой, что $ca_r(u) = \texttt{true}$.
— $(E53_{IA})$. Место UA не содержит токен $\langle u, r\rangle$, такой, что $ca_r(u) = \texttt{true}$.
— $(E54_{IA})$. Место UA не содержит ни одного токена $\langle u, r_{old}\rangle$, такого, что $ca_{r_{old}}(u) = \texttt{false}$ (данное условие гарантирует, что расчет назначений ролей начнётся только после окончания отзыва всех старых ролей).
— $(E55_{IA})$. Место UB не содержит токена $\langle u\rangle$ (над пользователем в данный момент не выполняется никаких действий).

Результат перехода: данный переход $(E56_{NA})$ добавляет токен $\langle auto_assign_role, u, system, r, s\rangle$ на место ETG, что означает переход «Автоматическое назначение роли» r пользователю u.

3.10. Переход «Удаление роли из списка активных в результате её отзыва у пользователя»

Правила активации перехода:

— $(E57_{RA})$. Существует токен $\langle revoke_role, u, admin, r, s\rangle$ на месте ETG, означающий, что необходимо выполнить команду отзыва роли r у пользователя u сессией от имени администратора $admin$.
— $(E58_{RA})$. На месте AUA существует токен $\langle admin, r_a\rangle$, для которого $can_revoke_{r_a}(r) = \texttt{true}$ (G_{58}).
— $(E59_{RA})$. Место UA содержит токен $\langle u, r\rangle$ (нельзя отозвать роль, если она не назначена).
— $(E60_{RA})$. Место UA не содержит токенов $\langle u, r_d\rangle$, таких, что $prec_role_r(r_d) = \texttt{true}$.
— $(E61_{RA})$. Место UAS содержит токен $\langle u, r, s\rangle$ (роль в данный момент времени активирована).
— $(E62_{IA})$. Не существует токенов $\langle remove_role, u, u, r, s\rangle$ и $\langle remove_role, u, system, r, s\rangle$ на месте ETG.
— $(E63_{IA})$. Место UB не содержит токена $\langle u\rangle$ (над пользователем в данный момент не выполняется никаких действий).

Результат перехода: данный переход $(E64_{NA})$ добавляет токен $\langle remove_role, u, system, r, s\rangle$ на место ETG, что означает команду на деактивацию роли r, активированной пользователем u в рамках сессии s.

Таким образом, описаны переходы между состояниями в системе семантически осмысленного ролевого управления доступом, заданной в рамках СК-РУД-модели. В отличие от системы, основанной на классической модели $RBAC$, возможность осуществления перехода зависит не только от текущего состояния системы, но и от переходов, выполненных на предыдущих шагах.

4. Доказательство безопасности системы семантически осмысленного ролевого управления доступом, представленной в виде ЦСП

Представление системы ролевого управления доступа в виде ЦСП позволяет осуществить формальное доказательство безопасности системы. Назовём состояние системы безопасным в рамках СК-РУД-модели, если оно не противоречит ограничениям, на-

кладываемым на составляющие её множества и функции, и каждому пользователю $u \in U$ назначены только роли, не противоречащие условиям CR и CA.

Переформулируем определение с учётом перечисленных в п. 1 элементов СК-РУД-модели.

Определение 10. Состояние системы в рамках СК-РУД-модели называется безопасным, если в нем выполняются следующие свойства:

— P_1: у каждого пользователя активны только те роли, на которые он авторизован: в каждый момент времени для каждой сессии $s \in S$ выполняется условие $roles(s) \subseteq \subseteq UA(user(s))$ (условие определения функции $roles$);
— P_2: если $prec_role_r(r') = \texttt{true}$, то роль r может быть назначена пользователю u только в том случае, если он также авторизован на роль r' (определение 3 иерархии «по предусловию» и определение предусловий CR [4]);
— P_3: пользователь может быть авторизован только на те атрибут-роли, атрибут-условиям которых соответствуют значения его атрибутов: для любого пользователя $u \in U$ если $r \in UA(u)$, то $ca_r(u) = \texttt{true}$ (ограничение в определении 4 атрибут-условия);
— P_4: пользователь не может быть одновременно авторизован на две статически взаимоисключающие роли: для любого пользователя $u \in U$ если $r_1, r_2 \in UA(u)$, то $static_conflict_role_{r_1}(r_2) = \texttt{false}$ (определение 1 статически взаимоисключающих ролей);
— P_5: у любой сессии пользователя не могут быть одновременно активированы две динамически взаимоисключающие роли: для сессии любого пользователя $s \in S$ если $r_1, r_2 \in roles(s)$, то $dynamic_conflict_role_{r_1}(r_2) = \texttt{false}$ (определение 2 динамически взаимоисключающих ролей).

Определение 11. Переход системы из состояния в состояние называется безопасным, если он удовлетворяет следующим ограничениям:

— T_1: переход «Активация роли» сессией пользователя может быть осуществлён только для тех ролей, на которые соответствующий пользователь авторизован в настоящий момент времени;
— T_2: переход «Назначение роли» r пользователю u сессией от имени администратора $admin$ может произойти только в том случае, если $can_assign_r(admin) = \texttt{true}$;
— T_3: переход «Назначение роли» r пользователю u может быть осуществлён только в том случае, если $prec_role_r(r_2) = \texttt{true}$ и пользователь в настоящий момент времени авторизован на роль r_2;
— T_4: переход «Назначение роли» r пользователю u может быть осуществлён только в том случае, если он в настоящий момент времени не авторизован на одну или более статически взаимоисключающих ролей;
— T_5: переход «Активация роли» r пользователю u может быть осуществлён только в том случае, если в настоящий момент времени в этой сессии не активны роли, динамически взаимоисключающие с r;
— T_6: переход «Отзыв роли» r у пользователя u сессией от имени администратора $admin$ может быть осуществлён только в том случае, если $can_revoke_r(admin) = = \texttt{true}$;
— T_7: переход «Отзыв роли» r_2 у пользователя u может быть осуществлён только в том случае, если $prec_role_r(r_2) = \texttt{true}$ и пользователь в настоящий момент времени авторизован на роль r.

Утверждение 1. Для системы ролевого управления доступом, начальное состояние M_0 которой является безопасным, все состояния M, достижимые из M_0 в результате конечного числа переходов, удовлетворяют свойству P_1. Для пользователя могут быть активными только те роли, на которые он авторизован.

Доказательство. Входящая дуга A_{33} между UA и переходом $take_role$ даёт возможность выполнить переход $take_role$ для пользователя u только в том случае, если токен $\langle u, r_y\rangle$ находится на месте UA (гарантирует выполнение ограничения T_1). Наличие токена $\langle u, r_y\rangle$ на месте UA означает, что пользователь u авторизован на роль r_y. Следовательно, множество ролей, активированных сессией пользователя, является подмножеством множества ролей, на которые он авторизован. ■

Утверждение 2. Для системы ролевого управления доступом, начальное состояние M_0 которой является безопасным, все состояния M, достижимые из M_0, удовлетворяют свойству P_2.

Доказательство. Рассмотрим ограничение последовательного назначения ролей, согласно которому, если некоторая роль r_x имеет зависимость от назначения подмножества ролей, то роль r_x может быть назначена пользователю только в том случае, если все роли r_y, такие, что $prec_role_{r_x}(r_y) = \texttt{true}$, назначены данному пользователю. Предположим, что ограничение не выполняется для некоторого состояния M, достижимого из M_0. Предположим, что $\{r_1, r_2, \ldots, r_n\}$ — такие роли, что $prec_role_{r_x}(r_i) = \texttt{true}$, $1 \leqslant i \leqslant n$. Согласно предположению, в состоянии M место UA содержит токен $\langle u, r_x\rangle$, но не содержит все токены множества $\{\langle u, r_i\rangle : 1 \leqslant i \leqslant n\}$. Предположим, что M — первое состояние, в котором роль r_x назначена пользователю. Это означает, что состояние M было достигнуто при выполнении перехода $assign_role$ ($auto_assign_role$) из состояния M', в котором токены $\langle u, r_i\rangle$ также отсутствовали на месте UA. Однако дуга A_3 (A_{19}) и соответствующее ограничение G_3 (G_{19}) $prec_role_{r_x}(r_i)$ не позволяет активировать переход $assign_role$ ($auto_assign_role$), гарантируя выполнение ограничения T_3. Следовательно, состояние M не может быть достигнуто. Противоречие. ■

Утверждение 3. Для системы ролевого управления доступом, начальное состояние M_0 которой является безопасным, все состояния M, достижимые из M_0, удовлетворяют свойству P_3.

Доказательство. Рассмотрим ограничение последовательного отзыва ролей, согласно которому, если для ролей из множества $\{r_1, r_2, \ldots, r_n\}$ роль r_x входит в предварительное условие их назначения, то роль r_x может быть отозвана только в том случае, если все роли $\{r_1, r_2, \ldots, r_n\}$, такие, что $prec_role_{r_i}(r_x) = \texttt{true}$, отозваны у пользователя. Предположим противное: ограничение не выполняется для некоторого состояния M, достижимого из M_0. Пусть $\{r_1, r_2, \ldots, r_n\}$ — такие роли, что $prec_role_{r_i}(r_x) = \texttt{true}$, $1 \leqslant i \leqslant n$. Согласно предположению, в состоянии M место UA содержит токен $\langle u, r_x\rangle$, но не содержит всех токенов из множества $\{\langle u, r_i\rangle\}$, где $1 \leqslant i \leqslant n$. Предположим, что M — первое состояние, в котором роль r_i, $1 \leqslant i \leqslant n$, отозвана у пользователя. Это означает, что состояние M было достигнуто при выполнении перехода $revoke_role$ ($auto_revoke_role$) из состояния M', в котором все токены $\langle u, r_i\rangle$ присутствовали на месте UA. Однако дуга A_{13} (A_{27}) и соответствующее ограничение G_{13} (G_{27}) $prec_role_{r_i}(r_x)$ не позволяет активировать переход $revoke_role$ ($auto_revoke_role$), гарантируя выполнение ограничения T_7. Следовательно, состояние M не может быть достигнуто. Противоречие. ■

Утверждение 4. Для системы ролевого управления доступом, начальное состояние M_0 которой является безопасным, все состояния M, достижимые из M_0, удовлетворяют свойству P_4.

Доказательство. Предположим противное: в системе существует пользователь u, для которого ограничение на запрет одновременного назначения статически взаимоисключающих ролей не выполняется, т. е. u может быть одновременно авторизован на роли r_x и r_y, такие, что $static_conflict_role_{r_x}(r_y) = \texttt{true}$. Следовательно, токены $\langle u, r_x\rangle$ и $\langle u, r_y\rangle$ могут одновременно находиться на месте UA. Пусть M_0 — состояние системы непосредственно перед выполнением перехода $assign_role$, добавляющего токен $\langle u, r_y\rangle$ на место UA. Пусть также в состоянии M_0 пользователь u авторизован на роль r_x (токен $\langle u, r_x\rangle$ находится на месте UA, а токен $\langle u, r_y\rangle$ отсутствует на данном месте). Пусть r_y — первая роль, для которой не выполняется ограничение P_4. Ингибиторная дуга A_3 связана с функцией $E3 : \langle u, r_c\rangle$ и ограничением перехода G_3: $static_conflict_role_r(r_c) = \texttt{true}$, отсюда следует, что выполняется условие T_4 и переход $assign_role$ не может быть активирован. Следовательно, свойство P_4 выполняется для любого состояния M, достижимого из M_0. ■

Утверждение 5. Для системы ролевого управления доступом, начальное состояние M_0 которой является безопасным, все состояния M, достижимые из M_0, удовлетворяют свойству P_5.

Доказательство. Предположим противное: в системе существует пользователь u, для которого ограничение на запрет одновременного назначения взаимоисключающих ролей не выполняется, т. е. u может одновременно активировать роли r_x и r_y, такие, что $dynamic_conflict_role_{r_x}(r_y) = \texttt{true}$. Следовательно, токены $\langle u, r_x, s\rangle$ и $\langle u, r_y, s\rangle$ могут одновременно находиться на месте UAS. Пусть M_0 — состояние системы непосредственно перед выполнением перехода $assign_role$, добавляющего токен $\langle u, r_y, s\rangle$ на место UAS. Пусть также в состоянии M_0 сессия от имени пользователя u активирует роль r_x (токен $\langle u, r_x, s\rangle$ находится на месте UAS, а токен $\langle u, r_y, s\rangle$ отсутствует на данном месте). Пусть r_y — первая роль, для которой не выполняется ограничение P_5. Ингибиторная дуга A_{34} связана с функцией E_{34}: $\langle u, r_{dc}, s\rangle$ и ограничением перехода G_{34}: $dynamic_conflict_role_r(r_{dc}) = \texttt{true}$, отсюда следует, что выполняется условие T_5 и переход $take_role$ не может быть активирован. Противоречие. Следовательно, свойство P_5 выполняется для любого состояния M, достижимого из M_0. ■

Теорема 1. Если начальное состояние системы M_0, представленной в виде ЦСП, безопасно, то все состояния M, достижимые из M_0 посредством безопасных переходов, безопасны.

Доказательство. Из утверждений 1–5 следует, что все безопасные переходы между состояниями сохраняют свойства P_1–P_5. Следовательно, по определению 10, любое достижимое состояние является безопасным. ■

Таким образом, теоретически обоснованы достаточные условия безопасности систем, реализующих автоматизированное назначение ролей в рамках СК-РУД-модели.

Заключение

Представление системы семантически осмысленного ролевого управления доступом в рамках модели СК-РУД в виде сети Петри позволяет адаптировать её к условиям функционирования реальных компьютерных сетей предприятий, допускающих распределённое администрирование, а также проводить анализ переходов между состоя-

ниями, состоящих из двух и более операций. Безопасность системы ролевого управления доступом, построенной на основе предложенной модели СК-РУД, подтверждается проведённым анализом переходов между состояниями.

ЛИТЕРАТУРА

1. *Семенова Н. А.* Семантическая ролевая модель управления доступом // Прикладная дискретная математика. 2012. № 2(16). С. 50–64.
2. `http://csrc.nist.gov/rbac/rbacSTD-ACM.pdf` — National Institute of Standards and Technology, Proposed Standard for Role-Based Access Control.
3. *Котов В. Е.* Сети Петри. М.: Наука, 1984. 160 с.
4. *Девянин П. Н.* Модели безопасности компьютерных систем. Управление доступом и информационными потоками: учеб. пособие для вузов. М.: Горячая линия-Телеком, 2011. 320 с.

2013 Математические основы компьютерной безопасности №1(19)
DOI 10.17223/20710410/19/5
УДК 004.94

ПРАВИЛА ПРЕОБРАЗОВАНИЯ СОСТОЯНИЙ СУБД ДП-МОДЕЛИ

В. Ю. Смольянинов

Учебно-методическое объединение по информационной безопасности, г. Москва, Россия

E-mail: VladimirSmall@gmail.com

Рассматривается ДП-модель управления доступом в реляционных системах управления базами данных (кратко СУБД ДП-модель). Для учёта особенностей управления доступом в СУБД в модель включены сущности-процедуры, сущности-триггеры, специфичные для СУБД права доступа, наследование прав доступа, возможность динамического изменения учётной записи пользователя при активации сущностей-процедур и сущностей-триггеров. Обосновывается утверждение о предоставления прав доступа к сущностям-процедурам и контейнерам.

Ключевые слова: *компьютерная безопасность, СУБД ДП-модель, система управления базами данных.*

Введение

В настоящее время при проектировании, разработке и эксплуатации автоматизированных информационных систем особое значение стали приобретать вопросы обеспечения информационной безопасности. Практика показала, что одной из наиболее важных задач является обеспечение корректности управления доступом, а также анализ условий, приводящих к реализации запрещённых информационных потоков.

Для обеспечения возможности поиска, хранения и обработки информации разработчики автоматизированных информационных систем традиционно используют реляционные системы управления базами данных (СУБД). Современные СУБД предоставляют разработчикам прикладных приложений развитые средства управления доступом, которые имеют отличия от механизмов, применяемых в операционных системах (ОС).

В соответствии с «Критериями оценки безопасности информационных технологий» [1] использование формальных математических моделей является обязательным для анализа безопасности управления доступом и информационными потоками в компьютерных системах, начиная с уровня доверия ОУД 5. Существующие формальные модели логического управления доступом [2, 3] ориентированы в основном на применение в ОС и не учитывают некоторых существенных особенностей функционирования автоматизированных информационных систем, построенных с использованием реляционных СУБД, которые могут быть использованы нарушителем для реализации запрещённых доступов и информационных потоков. В частности, ими игнорируется наследование прав доступа к дочерним сущностям, существование прав на предоставление прав доступа, а также возможность автоматического выполнения SQL-кода при реализации доступа к данным и изменения учётной записи пользователя при активации хранимых процедур и триггеров.

Следовательно, теоретический анализ условий, при которых возможно несанкционированное распространение прав доступа, а также возникновение запрещённых информационных потоков в автоматизированных информационных системах, использу-

ющих штатные механизмы управления доступом в СУБД, является важной и актуальной задачей.

Далее описываются элементы СУБД ДП-модели, предназначенной для анализа безопасности управления доступом в СУБД *Microsoft SQL Server*. Построенная модель может быть адаптирована для анализа безопасности других реляционных СУБД.

1. Определения и обозначения

В предлагаемой СУБД ДП-модели, построенной на основе РОСЛ ДП-модели [4], используются следующие обозначения и определения:

O_d — множество сущностей-данных;

O_p — множество сущностей-процедур;

O_t — множество сущностей-триггеров;

$O = O_d \cup O_p \cup O_t$ — множество сущностей-объектов, при этом считается, что множества сущностей-данных, сущностей-процедур и сущностей-триггеров попарно не пересекаются, то есть $O_d \cap O_p = \varnothing$, $O_d \cap O_t = \varnothing$ и $O_p \cap O_t = \varnothing$;

C — множество контейнеров;

$E = O \cup C$ — множество сущностей, при этом считается, что $O \cap C = \varnothing$, то есть контейнеры не являются сущностями-объектами.

Определение 1. Определим $H\colon C \to 2^E$ — функцию иерархии сущностей, сопоставляющую каждому контейнеру $c \in C$ множество $H(c) \subset E$ и удовлетворяющую следующим условиям.

У с л о в и е 1. Пусть $P(e) = \{c \in C : e \in H(c)\}$ — множество родительских контейнеров сущности e. Тогда существует единственный корневой контейнер $c_r \in C$, такой, что $|P(c_r)| = 0$ и для каждой сущности $e \in E \setminus \{c_r\}$ выполняется $|P(e)| = 1$.

У с л о в и е 2. Для каждой сущности $e \in E$ либо $e = c_r$, либо существуют контейнеры $c_1, \ldots, c_n \in C$, где $n \geqslant 1$, такие, что $c_1 = c_r$, $c_{i+1} \in H(c_i)$ для $i = 1, \ldots, n-1$ и $e \in H(c_n)$.

У с л о в и е 3. Пусть $T = \{c \in C : \exists e \in O_d \cup O_t\,(e \in H(c))\}$ — множество таблиц. Тогда для каждой таблицы $t \in T$ по определению выполняется $H(t) \subseteq O_d \cup O_t$.

Определение 2. Если для сущности $e \in E$ существует контейнер $c \in C$, такой, что $e \in H(c)$, то будем говорить, что контейнер c содержит сущность e, а сущность e содержится в контейнере c; также будем говорить, что c является родительским контейнером сущности e, а e — дочерней сущностью контейнера c.

Определение 3. Будем говорить, что сущность $e \in E$ иерархически подчинена контейнеру $c \in C$, если существуют контейнеры $c_1, \ldots, c_n \in C$, где $n \geqslant 1$, такие, что $c_1 = c$, $c_{i+1} \in H(c_i)$ для $i = 1, \ldots, n-1$ и $e \in H(c_n)$.

В случае, если сущность $e \in E$ иерархически подчинена контейнеру $c \in C$, будем использовать обозначение $e < c$. В случае, если сущность $e \in E$ иерархически подчинена контейнеру $c \in C$ или равна ему, будем использовать обозначение $e \leqslant c$.

Замечание 1. Из определений 1 и 3 следует, что для любой сущности $e \in E \setminus \{c_r\}$ существует контейнер $c \in C$, такой, что $e < c$. Кроме того, очевидно, что каждая сущность либо является корневой, либо иерархически подчинена корневой сущности c_r и содержится ровно в одном из её дочерних контейнеров. Будем считать необходимым условием равенства сущностей их принадлежность одному контейнеру.

Замечание 2. Легко показать, что если для некоторой сущности $e \in E$ существуют контейнеры $c_1, \ldots, c_n \in C$, где $n \geqslant 1$, такие, что $c_1 = c_r$, $c_{i+1} \in H(c_i)$ для $i = 1, \ldots, n-1$ и $e \in H(c_n)$, и контейнеры $c'_1, \ldots, c'_{n'} \in C$, где $n' \geqslant 1$, такие, что $c'_1 = c_r$,

$c'_{i+1} \in H(c'_i)$ для $i = 1, \ldots, n' - 1$ и $e \in H(c'_{n'})$, то $n = n'$ и $c'_i = c_i$ для $i = 1, \ldots, n$, то есть для любой некорневой сущности $e \in E$ существует единственная последовательность, начинающаяся с c_r и заканчивающаяся контейнером, содержащим e, такая, что каждый её элемент является родительским для последующего.

Используем также следующие обозначения и определения:

U — множество учётных записей пользователей, при этом считается, что учётные записи пользователей по определению не являются сущностями, то есть $E \cap U = \varnothing$;

S — множество субъект-сессий пользователей, при этом считается, что $S \cap U = \varnothing$ и $S \cap E = \varnothing$;

$user\colon S \to U$ — функция, задающая для каждой субъект-сессии учётную запись пользователя, от имени которой она выполняется;

$assoc_f(s)\colon S \to O_p \cup O_t \cup \{\varnothing\}$ — функция, задающая для каждой субъект-сессии $s \in S$ функционально ассоциированную с ней сущность;

$owner\colon E \to U$ — функция, задающая для каждой сущности учётную запись её владельца, при этом для каждой таблицы $t \in T$ и каждой сущности $e \in H(t)$ по определению выполняется равенство $owner(e) = owner(t)$;

$owners\colon E \to 2^U$ — функция, задающая для каждой сущности учётные записи её иерархических владельцев, при этом для каждой сущности $e \in E$ по определению $u \in owners(e)$ тогда и только тогда, когда либо $u = owner(e)$, либо существует контейнер $c \in P(e)$, такой, что $u \in owners(c)$;

$execute_as\colon O_p \cup O_t \to \{as_caller, as_owner\}$ — функция, задающая для каждой сущности-процедуры и сущности-триггера режим их выполнения субъект-сессиями (as_caller соответствует режиму выполнения сущности от имени учётной записи пользователя, инициировавшей её запуск, а as_owner — от имени учётной записи владельца сущности);

$R_r = \{read_r, write_r, append_r, delete_r, alter_r, execute_r\}$ — множество видов прав доступа к контейнерам и сущностям-процедурам;

O_t^* — множество всех конечных последовательностей сущностей-триггеров.

Определение 4. Определим $triggers\colon T \times \{write_r, append_r, delete_r\} \to O_t^*$ — функцию, задающую для каждой таблицы связанную с ней последовательность сущностей-триггеров и удовлетворяющую следующим условиям.

У с л о в и е 1. Для каждой таблицы $t \in T$ и каждого права доступа $\alpha_r \in \{write_r, append_r, delete_r\}$ выполняется $triggers(t, \alpha_r) \subseteq H(t)$.

У с л о в и е 2. Для каждой сущности-триггера $o_t \in O_t$ существует единственная таблица $t \in T$ и единственное право доступа $\alpha_r \in \{write_r, append_r, delete_r\}$, такое, что $o_t \in triggers(t, \alpha_r)$.

Если существуют таблица $t \in T$, право доступа $\alpha_r \in \{write_r, append_r, delete_r\}$ и сущность-триггер $o_t \in triggers(t, \alpha_r)$, то будем говорить, что сущность-триггер o_t имеет тип α_r и связана с таблицей t.

Введём также следующие обозначения:

$R \subseteq U \times (C \cup O_p) \times R_r$ — множество непосредственно заданных прав доступа учётных записей пользователей к контейнерам и сущностям-процедурам;

$R_{own} = \{(u, e, \alpha_r) : \alpha_r \in R_r, e \in C \cup O_p, u = owner(e)\}$ — множество прав доступа учётных записей пользователей — владельцев контейнеров и сущностей-процедур;

$R_H = \{(u, e, \alpha_r) : u \in U, e \in C \cup O_p, \alpha_r \in R_r, \exists c \in C((u, c, \alpha_r) \in R \cup R_{own} \,\&\, e < c)\}$ — множество иерархических прав доступа учётных записей пользователей к контейнерам и сущностям-процедурам;

$R_e = R \cup R_{own} \cup R_H$ — множество действующих прав доступа учётных записей пользователей к контейнерам и сущностям-процедурам;

$Gr_e = Gr \cup Gr_{own}$ — множество действующих прав на предоставление прав доступа к контейнерам и сущностям-процедурам, при этом множества Gr и Gr_{own} задаются следующим образом:

$Gr \subseteq U \times (C \cup O_p) \times R_r$ — множество непосредственно заданных прав на предоставление прав доступа к контейнерам и сущностям-процедурам, удовлетворяющее условию $Gr \subseteq R_e$;

$Gr_{own} = \{(u, e, \alpha_r) : u \in U, \alpha_r \in R_r, e \in C \cup O_p, u \in owners(e)\}$ — множество прав доступа на предоставление прав доступа иерархическими владельцами контейнеров и сущностей-процедур;

$Gr(e, \alpha_r) = \{u \in U : (u, e, \alpha_r) \in Gr_e\}$, где $e \in C \cup O_p$, $\alpha_r \in R_r$ — множество учётных записей пользователей, которые могут предоставлять право доступа $\alpha_r \in R_r$ к сущности $e \in C \cup O_p$;

OP_{int} — множество внутренних правил преобразования состояний, заданных в табл. 1;

OP_{ext} — множество внешних правил преобразования состояний, заданных в табл. 2;

$OP = OP_{int} \cup OP_{ext}$ — множество правил преобразования состояний;

OP_{ext}^* — множество всех конечных последовательностей внешних правил преобразования состояний;

OP^* — множество всех конечных последовательностей правил преобразования состояний;

$operations\colon O_p \cup O_t \to OP_{ext}^*$ — функция, задающая для каждой сущности-процедуры и сущности-триггера соответствующие им последовательности внешних правил преобразования состояний, реализуемых при выполнении субъектами-сессиями их SQL-кода;

$time \in \mathbb{N}_0$ — счётчик моментов времени, значение которого увеличивается на единицу при реализации каждого правила $op \in OP$;

$tr \in OP^*$ — последовательность всех применённых правил, начиная с нулевого момента времени, непосредственная реализация которых привела к изменению значения счётчика времени;

$G = (U, S, E, R, Gr, H, user, assoc_f, owner, execute_as, triggers, operations, time, tr)$ — состояние системы;

$\Sigma(G^*, OP)$ — система, где G^* — множество всех возможных состояний;

$\Sigma(G^*, OP, G_0)$ — система $\Sigma(G^*, OP)$ с начальным состоянием G_0.

В дальнейшем будем использовать следующее предположение.

Предположение 1. Считается, что в начальном состоянии системы $\Sigma(G^*, OP, G_0)$ отсутствуют субъект-сессии, то есть $S_0 = \varnothing$; кроме того, значение счётчика моментов времени $time_0$ в начальном состоянии равно 0 и последовательность tr_0 не содержит элементов.

Поясним введённые обозначения и определения.

Сущности-данные модели соответствуют записям таблиц СУБД. В отличие от реальных СУБД считается, что у записей таблиц отсутствует внутренняя структура, то есть нет отдельных полей. Причина подобного подхода к рассмотрению записей состоит в том, что в большинстве современных СУБД отсутствуют механизмы контроля доступа не только на уровне полей записей таблиц, но и на уровне отдельных записей таблиц. Следовательно, построение формальных моделей управления доступом

к отдельным полям записей таблиц для современных СУБД не является актуальной задачей.

Таблица 1

Внутренние правила преобразования состояний СУБД ДП-модели

Правило	Исходное состояние G	Результирующее состояние G'
$switch(s, o, u, op)$	$s \in S, u \in U,$ $o \in O_p \cup O_t \cup \{\varnothing\}$, $op \in OP_{ext}$	$G'\{user', assoc'_f, time', tr'\} \sim G',$ $user'\{s \mapsto u\} \sim user$, $assoc'_f\{s \mapsto o\} \sim assoc_f,$ $time' = time + 1$, $tr' = (tr, switch(s, o, u, op))$
$revert(s, u, o)$	$s \in S, u \in U,$ $o \in O_p \cup O_t \cup \{\varnothing\}$	$G'\{user', assoc'_f, time', tr'\} \sim G',$ $user'\{s \mapsto u\} \sim user$, $assoc'_f\{s \mapsto o\} \sim assoc_f,$ $time' = time + 1$, $tr' = (tr, revert(s, u, o))$
$execute(s, o, op)$	$s \in S, o \in O_p \cup O_t,$ $op \in OP_{ext}$, $operations(o) =$ $= (op_1, \ldots, op_k)$, $k \geqslant 0,$ $\forall i \in \overline{1,k}\ op_i \in OP_{ext}$	$G' = G(switch(s, o, u, op), op_1, \ldots, op_k,$ $revert(s, user(s), [s])),$ $u = \begin{cases} owner(o) \text{ при } execute_as(o) = as_owner, \\ user(s) \text{ при } execute_as(o) = as_caller \end{cases}$
do_insert $(s, t_1, o_{d_1}, t_2, o_{d_2})$	$s \in S, t_1, t_2 \in T, u = user(s),$ $o_{d_1} \notin O_d, o_{d_2} \in H(t_2)$	$G'\{E', H', time', tr'\} \sim G,$ $E' = O' \cup C,$ $O' = O \cup \{o_{d_1}\}$, $H'\{t_1 \mapsto H(t_1) \cup \{o_{d_1}\}\} \sim H,$ $time' = time + 1,$ $tr' = (tr, do_insert(s, t_1, o_{d_1}, t_2, o_{d_2}))$
do_update $(s, t_1, o_{d_1}, t_2, o_{d_2})$	$s \in S, t_1, t_2 \in T, u = user(s),$ $o_{d_1} \in H(t_1)$, $o_{d_2} \in H(t_2)$	$G'\{time', tr'\} \sim G$, $time' = time + 1,$ $tr' = (tr, do_update(s, t_1, o_{d_1}, t_2, o_{d_2}))$
$do_delete(s, t, o_d)$	$s \in S, t \in T, u = user(s),$ $o_d \in H(t)$	$G'\{E', H', time', tr'\} \sim G,$ $E' = O' \cup C,$ $O' = O \setminus \{o_d\}$, $H'\{t \mapsto H(t) \setminus \{o_d\}\} \sim H,$ $time' = time + 1$, $tr' = (tr, do_delete(s, t, o_d))$

Таблица 2

Внешние правила преобразования состояний СУБД ДП-модели

Правило	Исходное состояние G	Результирующее состояние G'
1	2	3
$create_session$ (u, s)	$u \in U, s \notin S$	$G'\{S', user', assoc'_f, time', tr'\} \sim G, S' = S \cup \{s\},$ $user'\{s \mapsto u\} \sim user$, $assoc'_f\{s \mapsto \varnothing\} \sim assoc_f,$ $time' = time + 1$, $tr' = (tr, create_session(u, s))$
$grant_right$ $(s, u, e, \alpha_r,$ $with_grant)$	$s \in S$, $u \in U$, $e \in C \cup O_p,$ $\alpha_r \in R_r$, $user(s) \in Gr(e, \alpha_r),$ $with_grant \in \{\texttt{yes}, \texttt{no}\}$	$G'\{R', Gr', time', tr'\} \sim G, R' = R \cup \{(u, e, \alpha_r)\},$ $Gr' = \begin{cases} Gr \cup \{(u, e, \alpha_r)\} \text{ при } with_grant = \texttt{yes}, \\ Gr \text{ при } with_grant = \texttt{no} \end{cases}$ $time' = time + 1$, $tr' = (tr, grant_right(s, u, e,$ $\alpha_r, with_grant))$
$create_container$ (s, c_p, c)	$s \in S, c_p \in C, c \notin C,$ $(user(s), c_p, alter_r) \in R_e$	$G'\{E', H', owner', time', tr'\} \sim G,$ $E' = O \cup C',$ $C' = C \cup \{c\}$, $H'\{c \mapsto \varnothing, c_p \mapsto H(c_p) \cup \{c\}\} \sim H,$ $owner'\{c \mapsto user(s)\} \sim owner$, $time' = time + 1,$ $tr' = (tr, create_container(s, c_p, c))$
$create_procedure$ $(s, c, o_p, md,$ $op_1, \ldots, op_k)$	$s \in S, c \in C \setminus T$, $o_p \notin O_p,$ $md \in \{as_caller, as_owner\},$ $k \geqslant 0$, $\forall i \in \overline{1,k}\ op_i \in OP_{ext},$ $(user(s), c, alter_r) \in R_e$	$G'\{E', H', owner', execute_as', operations',$ $time', tr'\} \sim G,$ $E' = O' \cup C,$ $O' = O \cup \{o_p\},$ $H'\{c \mapsto H(c) \cup \{o_p\}\} \sim H,$ $owner'\{o_p \mapsto user(s)\} \sim owner,$ $execute_as'\{o_p \mapsto md\} \sim execute_as,$ $operations'\{o_p \mapsto (op_1, \ldots, op_k)\} \sim operations,$ $time' = time + 1$, $tr' = (tr, create_procedure(s, c,$ $o_p, md, op_1, \ldots, op_k)$

Окончание табл. 2

1	2	3
$alter_procedure$ $(s, o_p, md,$ $op_1, \ldots, op_k)$	$s \in S$, $o_p \in O_p$, $md \in \{as_caller, as_owner\}$, $k \geqslant 0$, $\forall i \in \overline{1,k}$ $op_i \in OP_{ext}$, $(user(s), o_p, alter_r) \in R_e$	$G'\{execute_as', operations', time', tr'\} \sim G$, $execute_as'\{o_p \mapsto m\} \sim execute_as$, $operations'\{o_p \mapsto (op_1, \ldots, op_k)\} \sim operations$, $time' = time+1$, $tr' = (tr, alter_procedure(s, o_p,$ $md, op_1, \ldots, op_k))$
$create_trigger$ $(s, t, o_t, \alpha_r, md,$ $op_1, \ldots, op_k)$	$s \in S, t \in T, o_t \notin O_t, \alpha_r \in R_r$, $md \in \{as_caller, as_owner\}$, $k \geqslant 0$, $\forall i \in \overline{1,k}$ $op_i \in OP_{ext}$, $(user(s), t, alter_r) \in R_e$	$G'\{E', H', owner', execute_as', triggers',$ $operations', time', tr'\} \sim G$, $E' = O' \cup C$, $O' = O \cup \{o_t\}$, $H'\{t \mapsto H(t) \cup \{o_t\}\} \sim H$, $owner'\{o_t \mapsto user(s)\} \sim owner$, $execute_as'\{o_t \mapsto md\} \sim execute_as$, $triggers'\{(t, \alpha_r) \mapsto (triggers(t, \alpha_r), o_t)\} \sim$ $\sim triggers$, $operations'\{o_t \mapsto (op_1, \ldots, op_k)\} \sim$ $\sim operations$, $time' = time + 1$, $tr' = (tr,$ $create_trigger(s, t, o_t, \alpha_r, md, op_1, \ldots, op_k))$
$alter_trigger$ $(s, t, o_t, md,$ $op_1, \ldots, op_k)$	$s \in S, t \in T, o_t \in O_t$, $md \in \{as_caller, as_owner\}$, $k \geqslant 0$, $\forall i \in \overline{1,k}$ $op_i \in OP_{ext}$, $(user(s), t, alter_r) \in R_e$, $o_t \in H(t)$	$G'\{execute_as', operations', time', tr'\} \sim G$, $execute_as'\{o_t \mapsto md\} \sim execute_as$, $operations'\{o_t \mapsto (op_1, \ldots, op_k)\} \sim operations$, $time' = time + 1$, $tr' = (tr, alter_trigger(s, t, o_t,$ $md, op_1, \ldots, op_k))$
$execute_procedure$ (s, o_p)	$s \in S, o_p \in O_p$, $(user(s), o_p, execute_r) \in R_e$	$G' = G(execute(s, o_p, execute_procedure(s, o_p)))$
$access_insert$ $(s, t_1, o_{d_1}, t_2, o_{d_2})$	$s \in S, t_1, t_2 \in T, u = user(s)$, $(u, t_1, write_r) \in R_e$, $(u, t_2, read_r) \in R_e$, $triggers(t_1, append_r) =$ $= (o_{t,1}, \ldots, o_{t,k})$, $k \geqslant 0$, $\forall i \in \overline{1,k}$ $o_{t,i} \in O_t$	$G' = G(do_insert(s, t_1, o_{d_1}, t_2, o_{d_2}),$ $execute(s, o_{t,1}, access_insert(s, t_1, o_{d_1}, t_2, o_{d_2})),$ $\ldots$ $execute(s, o_{t,k}, access_insert(s, t_1, o_{d_1}, t_2, o_{d_2})))$
$access_update$ $(s, t_1, o_{d_1}, t_2, o_{d_2})$	$s \in S, t_1, t_2 \in T, u = user(s)$, $(u, t_1, write_r) \in R_e$, $(u, t_2, read_r) \in R_e$, $triggers(t_1, write_r) =$ $= (o_{t,1}, \ldots, o_{t,k})$, $k \geqslant 0$, $\forall i \in \overline{1,k}$ $o_{t,i} \in O_t$	$G' = G(do_update(s, t_1, o_{d_1}, t_2, o_{d_2}),$ $execute(s, o_{t,1}, access_update(s, t_1, o_{d_1}, t_2, o_{d_2})),$ $\ldots$ $execute(s, o_{t,k}, access_update(s, t_1, o_{d_1}, t_2, o_{d_2})))$
$access_delete$ (s, t, o_d)	$s \in S, u = user(s)$, $(u, t, delete_r) \in R_e$, $triggers(t, delete_r) =$ $= (o_{t,1}, \ldots, o_{t,k})$, $k \geqslant 0$, $\forall i \in \overline{1,k}$ $o_{t,i} \in O_t$	$G' = G(do_delete(s, t, o_d),$ $execute(s, o_{t,1}, access_delete(s, t, o_d)),$ $\ldots$ $execute(s, o_{t,k}, access_delete(s, t, o_d)))$

Множество контейнеров реальных реляционных СУБД включает такие сущности как экземпляр СУБД, экземпляры баз данных, схемы и таблицы. При этом иерархия сущностей на этом множестве контейнеров задаётся неявно. Как правило, с экземпляром СУБД связаны экземпляры баз данных, которые, в свою очередь, содержат схемы данных, являющиеся контейнерами для хранимых процедур, а также таблиц вместе с содержащимися в них записями и триггерами.

Будем считать, что корневой контейнер c_r, у которого отсутствуют родительские контейнеры, соответствует экземпляру СУБД. В отличие от реальных СУБД, модель не накладывает ограничений на число уровней вложенности в иерархии.

Помимо корневого контейнера c_r, в рамках модели выделяется особый вид контейнеров — таблицы. Согласно условию 3 определения 1, для каждой таблицы $t \in T$ выполняется $H(t) \subseteq O_d \cup O_t$, из чего следует, что таблицы не могут иметь дочерних контейнеров и содержат только сущности-данные и сущности-триггеры.

Обоснуем следующее утверждение.

Утверждение 1. Если $|C| > 1$, то $c_r \notin T$.

Доказательство. Предположим противное, то есть $c_r \in T$.

Из условия утверждения следует, что существует $c \in C$, $c \neq c_r$. Тогда по условию 2 определения 1 существуют контейнеры $c_1, \ldots, c_n$, где $n \geqslant 1$, такие, что $c_1 = c_r$, $c_{i+1} \in H(c_i)$ для $i = 1, \ldots, n-1$ и $c \in H(c_n)$. Это означает, что существует контейнер $c' \in C$, такой, что $c' \in H(c_r)$, причём $c' \notin O_d \cup O_t$, так как $O \cap C = \varnothing$.

По предположению корневой контейнер c_r является таблицей, следовательно, по условию 3 определения 1 выполняется $H(c_r) \subseteq O_d \cup O_t$.

Отсюда одновременно имеет место $c' \in H(c_r) \subseteq O_d \cup O_t$ и $c' \notin O_d \cup O_t$. Пришли к противоречию, следовательно, корневой контейнер c_r не является таблицей. ■

В замечании 1 отмечается, что для каждой сущности за исключением корневой существует единственный контейнер, в котором она содержится. Кроме того, в соответствии с условием 3 определения 1 в множество таблиц включаются все контейнеры, содержащие хотя бы одну сущность-данные или сущность-триггер. Следовательно, для каждой сущности-данные и сущности-триггера $e \in O_d \cup O_t$ существует ровно одна таблица $t \in T$, в которой она содержится, то есть $e \in H(t)$.

Сущности-триггеры соответствуют триггерам таблиц в СУБД, SQL-код которых автоматически выполняется при осуществлении операций вставки, удаления и обновления записей таблиц, с которыми они связаны. Каждая сущность-триггер имеет тип, задаваемый функцией $triggers$ и соответствующий определённому виду доступа. Реализация субъект-сессией доступа вида $\alpha_r \in \{write_r, append_r, delete_r\}$ к таблице $t \in T$ автоматически инициирует выполнение ею SQL-кода всех триггеров из последовательности $triggers(t, \alpha_r)$.

Сущности-процедуры соответствуют хранимым процедурам СУБД, выполнение SQL-кода которых может быть инициировано в рамках взаимодействия с системой. В реальных СУБД хранимые процедуры могут находиться только в схемах данных. В рамках модели считается, что сущности-процедуры могут находиться в любом контейнере, не являющемся таблицей.

Считается, что сущности-триггеры и сущности-процедуры содержат SQL-код, выполнение которого приводит к реализации преобразований системы, задаваемых правилами, описываемыми в п. 2. В рамках модели предполагается, что сущности-триггеры и сущности-процедуры не содержат динамически генерируемого SQL-кода и условий, а их выполнение приводит к применению фиксированной последовательности правил, задаваемой соответствующим значением функции $operations$. В реальных системах сущности-процедуры могут принимать формальные параметры. При вызове хранимой процедуры ей передаются фактические значения формальных параметров, что приводит к выполнению некоторой фиксированной последовательности операторов. В рамках модели считается, что каждой хранимой процедуре, принимающей входные параметры, соответствует множество сущностей-процедур. При этом для каждой допустимой комбинации значений фактических параметров задаётся отдельная сущность-процедура, содержащая SQL-код, являющийся результатом их подстановки в операторы хранимой процедуры. Если внутри хранимой процедуры имеются операторы вызова других хранимых процедур, принимающих параметры, то каждый такой вызов заменяется оператором активации соответствующей сущности-процедуры, полученной подстановкой фактических параметров.

Субъект-сессии соответствуют сессиям пользователей с СУБД, которые, в отличие от процессов ОС, не являются объектами доступа и потому не считаются сущностями. Субъект-сессии инициируют выполнение SQL-кода, что приводит к реализации заданных в нём преобразований состояний системы.

SQL-код может либо относиться к одной из сущностей-процедур или сущностей-триггеров, либо не относиться ни к одной из них. Считается, что SQL-код содержит инструкции, выполнение которых непосредственно приводит к применению внешних правил из множества OP_{ext}. При этом предполагается, что SQL-код не может содержать инструкции, выполнение которых непосредственно приводит к применению внутренних правил из множества OP_{int}. Выполнение внутренних правил инициируется опосредованно как составная часть преобразований системы, задаваемых внешними правилами. Например, выполнение SQL-кода сущностей-триггеров осуществляется только при применении внешних правил, связанных со вставкой, удалением и обновлением сущностей-данных таблиц.

Традиционно для сущностей, функционально-ассоциированных с субъект-сессией s, используется обозначение $[s]$. Необходимость во введении нового обозначения $assoc_f(s)$ обусловлена тем, что в отличие от других моделей полагается, что в каждый момент времени множество функционально-ассоциированных с субъект-сессией s сущностей $assoc_f(s)$ либо содержит ровно одну выполняемую им в текущий момент времени сущность-триггер или сущность-процедуру, либо не содержит элементов. Последнее означает, что субъект-сессия выполняет SQL-код, который не относится ни к одной сущности-процедуре или сущности-триггеру. В последнем случае считается, что выполнение такого SQL-кода эквивалентно непосредственному применению субъект-сессией правил преобразования состояний из множества OP_{ext}. В дальнейшем будем полагать, что по определению выполняется равенство $[s] = assoc_f(s)$.

Субъект-сессии применяют правила преобразования состояний системы от имени учётных записей пользователей из множества U. В реальных СУБД для создания новой сессии требуется указать в строке подключения данные, аутентифицирующие подключающегося пользователя. Изначально субъект-сессия начинает применять внешние правила от имени учётной записи пользователя, инициировавшей её создание. Однако при переходе системы из состояния в состояние допускается изменение значений функции $user$, что связано с наличием в реальных СУБД возможности выполнения кода хранимых процедур и триггеров от имени учётной записи пользователя, не являющегося инициатором их выполнения. Отметим, что возможность изменения значений функции $user$ является существенным отличием от моделей ролевого управления доступом и ДП-моделей, в которых предполагается неизменность её значений.

Функция $owner$ введена в модель в связи с тем, что в современных СУБД владельцы сущностей-контейнеров неявно получают все возможные права доступа как к самому контейнеру, так и ко всем его иерархически подчиненным сущностям. Из определения функции $owner$ следует, что владельцем сущностей-данных и сущностей-триггеров является владелец содержащей их таблицы. Это отражает тот факт, что в реальных СУБД у записей таблиц и триггеров не существует непосредственно задаваемой учётной записи владельца.

Предположение 2. Для любой сущности e учётная запись пользователя владельца $owner(e)$ после создания сущности e не изменяется.

Функция $owners$ задаёт для каждой сущности учётные записи всех её иерархических владельцев, которые обладают к ней всеми возможными правами доступа. При этом справедливо следующее

Утверждение 2. Для каждой сущности $e \in E$ выполняется равенство

$$owners(e) = \{u \in U : \exists c \in C\ (e < c\ \&\ u = owner(c))\} \cup \{owner(e)\}.$$

Доказательство. Обозначим

$$owners'(e) = \{u \in U : \exists c \in C\ (e < c\ \&\ u = owner(c))\} \cup \{owner(e)\}.$$

Покажем, что для каждой $e \in E$ верно равенство $owners'(e) = owners(e)$. Возможны два случая: $e = c_r$ и $e \neq c_r$.

Если $e = c_r$, то по определению 1 выполняется $|P(c_r)| = 0$, а значит, из определения 3 следует, что не существует контейнера $c \in C$, такого, что $c_r < c$. Значит, в данном случае $owners'(e) = \{owner(e)\}$. Кроме того, из способа задания функции $owners$ видно, что в случае $e = c_r$ выполняется равенство $owners(e) = \{owner(e)\}$. Следовательно, в случае $e = c_r$ верно равенство $owners'(e) = owners(e)$.

Если $e \neq c_r$, то, согласно замечанию 2, существует единственная последовательность контейнеров $c_1, \ldots, c_n \in C$, где $n \geqslant 1$, таких, что $c_1 = c_r$, $c_{i+1} \in H(c_i)$ для $i = 1, \ldots, n-1$ и $e \in H(c_n)$.

Покажем, что не существует контейнера $c \in C$, такого, что $e < c$ и $c \neq c_j$, $j = 1, \ldots, n$. Предположим противное. Так как $c_1 = c_r$ и по предположению $c \neq c_1$, то, согласно замечанию 2, существует единственная последовательность контейнеров $c'_1, \ldots, c'_k \in C$, где $k \geqslant 1$, таких, что $c'_1 = c_r$, $c'_{i+1} \in H(c'_i)$ для $i = 1, \ldots, k-1$ и $c \in H(c'_k)$. Кроме того, так как $e < c$, то по определению 3 существуют контейнеры $c'_{k+1}, \ldots, c'_{n'} \in C$, где $n' \geqslant k+1$, такие, что $c'_{k+1} = c$, $c'_{i+1} \in H(c'_i)$ для $i = k+1, \ldots, n'-1$ и $e \in H(c'_{n'})$. Следовательно, существует последовательность контейнеров $c'_1, \ldots, c'_{n'}$, таких, что $c'_1 = c_r$, $c'_{i+1} \in H(c'_i)$ для $i = 1, \ldots, n'-1$ и $e \in H(c'_{n'})$. Так как по замечанию 2 существует лишь единственная последовательность контейнеров, удовлетворяющая данным свойствам, то $n' = n$ и $c'_i = c_i$, $i = 1, \ldots, n$. Следовательно, существует $j = k+1$, $j \in \{1, \ldots, n\}$ и $c = c'_j = c_j$, а по предположению $c \neq c_j$ для всех $j = 1, \ldots, n$. Пришли к противоречию.

Отсюда имеем, что $\{c \in C : e < c\} = \{c_1, \ldots, c_n\}$. Следовательно, верно равенство

$$owners'(e) = \Big(\bigcup_{i=1}^{n} owner(c_i) \Big) \bigcup \{owner(e)\}. \tag{1}$$

Доказательство для случая $e \neq c_r$ проведем индукцией по n — количеству элементов в последовательности контейнеров $c_1, \ldots, c_n \in C$, где $n \geqslant 1$, таких, что $c_1 = c_r$, $c_{i+1} \in H(c_i)$ для $i = 1, \ldots, n-1$ и $e \in H(c_n)$. При $n = 1$ из (1) следует выполнение равенства

$$owners'(e) = owner(c_1) \cup owner(e). \tag{2}$$

Пусть $u \in owners(e)$. Из способа задания функции $owners$ следует, что возможны два случая: либо $u = owner(e)$, либо существует контейнер $c \in P(e)$, такой, что $u \in owners(c)$. Если $u = owner(e)$, то из (2) следует, что $u \in owners'(e)$. Рассмотрим второй случай. Пусть существует контейнер $c \in P(e)$, такой, что $u \in owners(c)$. Так как $e \in H(c_1)$, то $c = c_1$, а следовательно, $u \in owners(c_1)$. В этом случае из (2) следует, что $u \in owners'(e)$.

Пусть $u \in owners'(e)$. Из (2) следует, что возможны два случая: $u = owner(e)$ и $u = owner(c_1)$. В случае $u = owner(e)$ из способа задания функции $owners$ следует, что $u \in owners(e)$, а если $u = owner(c_1)$, то $u \in owners(c_1)$. Таким образом, существует контейнер $c_1 \in P(e)$, такой, что $u \in owners(c_1)$, а следовательно, $u \in owners(e)$.

Итак, в случае $n = 1$ выполняется равенство $owners'(e) = owners(e)$.

Пусть $n > 1$ и равенство $owners'(e) = owners(e)$ выполняется для всех $k < n$. Заметим, что из (1) следует $owners'(e) = owners'(c_n) \cup \{owner(e)\}$. Число элементов в последовательности $c_1, \ldots, c_{n-1}$ меньше, чем n, и, следовательно, по предположению индукции $owners'(c_n) = owners(c_n)$. Таким образом, верно равенство

$$owners'(e) = owners(c_n) \cup \{owner(e)\}. \tag{3}$$

Пусть $u \in owners(e)$. Из способа задания функции $owners$ следует, что возможны два случая: либо $u = owner(e)$, либо существует контейнер $c \in P(e)$, такой, что $u \in owners(c)$. Если $u = owner(e)$, то из (3) следует $u \in owners'(e)$. Рассмотрим случай $u \neq owner(e)$. Так как $c \in P(e)$ и $e \in H(c_n)$, то $c = c_n$, а следовательно, $u \in owners(c_n)$. Тогда из равенства (3) видно, что $u \in owners'(e)$.

Пусть $u \in owners'(e)$. Из (3) следует, что возможны два случая: $u = owner(e)$ и $u \in owners(c_n)$. Если $u = owner(e)$, то из способа задания функции $owners$ следует $u \in owners(e)$. Рассмотрим случай $u \in owners(c_n)$. Ввиду $e \in H(c_n)$ имеем $c_n \in P(e)$. Таким образом, $c_n \in P(e)$ и $u \in owners(c_n)$, откуда, по способу задания функции $owners$, верно $u \in owners(e)$. Следовательно, для случая $k = n$ индуктивный шаг доказан и $owners'(e) = owners(e)$. ■

Таким образом, для каждой сущности $e \in E$ множество $owners(e)$ состоит из учётной записи владельца сущности e, а также учётных записей владельцев контейнеров, которым иерархически подчинена сущность e. Заметим, что значения функции $owners$ однозначно определяются функцией иерархии сущностей H и функцией задания владельцев сущностей $owner$, и поэтому она не рассматривается в качестве самостоятельного элемента состояния системы.

Значение функции $execute_as$ задаёт режим выполнения сущностей-процедур и сущностей-триггеров, определяющий учётную запись пользователя, от имени которой субъект-сессии будут выполнять их SQL-код. Если некоторая субъект-сессия $s \in S$ инициировала выполнение сущности-процедуры или сущности-триггера $e \in O_p \cup O_t$ и $execute_as(e) = as_caller$, то код сущности e будет выполняться от имени учётной записи $user(s)$, в противном случае, если $execute_as(e) = as_owner$, то код сущности e будет выполняться субъект-сессией s от имени учётной записи владельца сущности e, то есть от имени учётной записи пользователя $owner(e)$. Отметим, что режим выполнения as_owner соответствует механизму повышения полномочий $SUID$ в ОС семейства UNIX, а режим as_caller — механизму имперсонации (олицетворения) в ОС семейства Windows.

Замечание 3. Присутствующий в реальных СУБД режим выполнения сущностей-процедур и сущностей-триггеров от имени фиксированной учётной записи в рамках модели не рассматривается.

Виды прав доступа из множества R_r соответствуют по смыслу и назначению традиционным для СУБД правам доступа $SELECT$, $UPDATE$, $INSERT$, $DELETE$, $ALTER$ и $EXECUTE$.

Большинство современных СУБД реализуют, как минимум, дискреционное управление доступом учётных записей пользователей к контейнерам и сущностям-процеду-

рам. При этом для упрощения процедуры администрирования прав доступа в СУБД поддерживается управление правами учётных записей пользователей с учётом иерархических отношений между сущностями, а также выполняется неявное предоставление всех возможных прав доступа учётным записям владельцев сущностей.

При определении возможности применения правил преобразования состояний в модели учитываются права доступа, содержащиеся в множестве действующих прав доступа учётных записей пользователей R_e. Полагается, что субъект-сессии s, функционирующей от имени учётной записи пользователя $user(s) = u$, разрешено реализовать доступ вида α_r к сущности e в случае, если $(u, e, \alpha_r) \in R_e$. Множество действующих прав доступа R_e включает множество непосредственно заданных прав доступа R, множество прав доступа учётных записей владельцев R_{own}, а также множество иерархических прав доступа R_H. Из способа задания множества R_{own} следует, что учётная запись владельца сущности обладает к ней всеми правами доступа. Кроме того, из способа задания множества R_H следует, что учётная запись пользователя, имеющего непосредственно заданное право доступа α_r к контейнеру c либо являющегося его владельцем, также обладает этим правом доступа и к любому его иерархически подчинённому контейнеру и сущности-процедуре. Таким образом, верно следующее замечание.

Замечание 4. Если $u \in U$, $e, e' \in C \cup O_p$, $\alpha_r \in R_r$, $e \leqslant e'$ и $(u, e', \alpha_r) \in R_e$, то $(u, e, \alpha_r) \in R_e$.

Заметим, что множества прав доступа R_{own}, R_H и R_e однозначно определяются множеством прав доступа R, функцией иерархии сущностей H и функцией задания владельцев сущностей $owner$ и поэтому не рассматриваются в качестве самостоятельных элементов состояния системы.

Считается, что наличие у учётной записи прав доступа $read_r$, $write_r$, $append_r$ или $delete_r$ к таблице позволяет выполнять над иерархически подчинёнными ей сущностями-данными операции выборки, обновления, вставки и удаления соответственно. Наличие права доступа $delete_r$ к контейнеру позволяет удалять его дочерние контейнеры и сущности-процедуры. Обладание правом доступа $alter_r$ к сущности-процедуре позволяет задавать её SQL-код и режим выполнения. Право доступа $alter_r$ к таблице позволяет создавать в ней сущности-триггеры, а также изменять SQL-код связанных с ней сущностей-триггеров и режим их выполнения. Наличие права доступа $execute_r$ на сущность-процедуру позволяет выполнять её SQL-код.

При введении множества непосредственно заданных прав доступа R для простоты игнорируется, что права доступа $read_r$, $write_r$ и $append_r$ могут применяться только к контейнерам, а право доступа $execute_r$ неприменимо к таблицам. Отметим, что способ задания множества R учитывает, что большинство современных СУБД предоставляют возможность устанавливать права доступа к контейнерам и сущностям-процедурам, не позволяя определять их на уровне отдельных сущностей-данных и сущностей-триггеров.

Как правило, в ОС, реализующих механизмы дискреционного управления доступом, управлять предоставлением прав к сущности может только её владелец и, возможно, пользователь, обладающий привилегиями администратора. Указанные пользователи могут дать произвольное право доступа к сущности любому другому пользователю системы. В современных СУБД присутствуют штатные механизмы, позволяющие делегировать предоставление прав доступа к сущности заданному для неё кругу учётных записей пользователей. При этом СУБД позволяют ограничить для каждой

учётной записи из этого круга набор прав доступа к сущности, которые она может предоставлять.

Для учёта данной особенности в модель вводится множество действующих прав на предоставление прав доступа к контейнерам и сущностям-процедурам Gr_e. Полагается, что если $(u, e, \alpha_r) \in Gr_e$, то субъект-сессия s, функционирующая от имени учётной записи пользователя $user(s) = u$, может дать любой другой учётной записи пользователя право доступа α_r к сущности e. Помимо множества непосредственно заданных прав на предоставление прав Gr, в Gr_e включаются права иерархических владельцев сущностей, заданные множеством Gr_{own}. При этом субъект-сессия, функционирующая от имени одного из иерархических владельцев сущности, может дать любые права доступа к ней. Заметим, что множества Gr_e и Gr_{own} однозначно определяются множеством Gr, функцией иерархии сущностей H и функцией задания владельцев сущностей $owner$ и поэтому не рассматриваются в качестве самостоятельных элементов состояния системы.

В современных СУБД для того, чтобы пользователь имел возможность предоставить право доступа к сущности, требуется, чтобы он сам обладал к ней этим правом доступа. Обоснуем, что в рамках модели необходимым условием наличия у учётной записи пользователя $u \in U$ права на предоставление права доступа α_r к контейнеру или сущности-процедуре $e \in C \cup O_p$ является наличие у u права доступа α_r к e.

Утверждение 3. $Gr_e \subseteq R_e$.

Доказательство. По способу задания $Gr_e = Gr \cup Gr_{own}$ и $Gr \subseteq R_e$. Покажем, что $Gr_{own} \subseteq R_e$. Пусть $e \in C \cup O_p$, $\alpha_r \in R_r$, $(u, e, \alpha_r) \in Gr_{own}$, где $u \in owners(e)$. В случае $u = owner(e)$, согласно способу задания множества R_{own}, выполняется $(u, e, \alpha_r) \in R_{own} \subseteq R_e$. Если же $u \neq owner(e)$, то, согласно утверждению 2, существует контейнер $c \in C$, такой, что $e < c$ и $u = owner(c)$. Так как $u = owner(c)$, то по способу задания множества R_{own} выполняется $(u, c, \alpha_r) \in R_{own}$. Таким образом, выполняется $u \in U$, $e \in C \cup O_p$, $\alpha_r \in R_r$ и существует контейнер $c \in C$, такой, что $(u, c, \alpha_r) \in R_{own}$ и $e < c$. Следовательно, согласно способу задания множества R_H, выполняется $(u, e, \alpha_r) \in R_H \subseteq R_e$. ■

2. Правила преобразования состояний

Будем использовать запись вида $G'\{I'_1, \ldots, I'_k\} \sim G$ для обозначения того, что все элементы состояния G' равны соответствующим элементам состояния G, за исключением элементов $I'_1, \ldots, I'_k$ состояния G'. Будем также использовать запись вида $f'\{x_1, \ldots, x_k\} \sim f$ для обозначения того, что значения функции $f'(x)$ равны значениям функции $f(x)$ для всех аргументов x, за исключением $x_1, \ldots, x_k$. Кроме того, будем использовать запись вида $f'\{x_1 \mapsto y_1, \ldots, x_k \mapsto y_k\} \sim f$ для обозначения того, что значения функции $f'(x)$ равны значениям функции $f(x)$ для всех аргументов x, кроме $x_1, \ldots, x_k$, и при этом $f'(x_1) = y_1$, …, $f'(x_k) = y_k$. Очевидно, что если выполняется $G'\{I'_1, \ldots, I'_k\} \sim G$, то также верно и $G\{I_1, \ldots, I_k\} \sim G'$.

Используем обозначение: $G \vdash_{op} G'$ — переход системы $\Sigma(G^*, OP)$ из состояния G в состояние G' с применением правила преобразования состояний $op \in OP$, при этом если условия применения правила op не выполняются, то по определению справедливо равенство $G' = G$.

Запись вида $G' = G(op_1, \ldots, op_k)$ означает, что состояние G' есть результат последовательного применения правил преобразования состояний $op_1, \ldots, op_k$ в начальном состоянии G, то есть существуют состояния $G_1, \ldots, G_k$, такие, что $G \vdash_{op_1} G_1 \vdash_{op_2} \ldots \vdash_{op_k} G_k$ и $G' = G_k$.

В СУБД ДП-модели определены приведённые в табл. 1 и 2 правила преобразования состояний из множества OP, в которых учтены особенности управления доступом в современных СУБД.

Поясним приведённые в табл. 1 условия и результаты применения внутренних правил преобразования системы.

В отличие от внешних правил, внутренние правила применяются субъект-сессиями опосредованно в ходе выполнения SQL-кода как составная часть преобразований системы, задаваемых внешними (см. описание правил $execute_procedure$, $access_insert$, $access_update$ и $access_delete$ в табл. 2) или другими внутренними правилами (см. описание правила $execute$).

Запись вида (tr, op), использованная при определении нового значения tr', означает, что в возможно пустую последовательность tr добавляются данные о применяемом правиле op.

В результате применения внутреннего правила $switch(s, o, u, op)$ в качестве учётной записи, от имени которой функционирует субъект-сессия s, устанавливается u, а o становится функционально-ассоциированной с s сущностью. Параметр $op \in OP_{ext}$ задаёт внешнее правило, в результате применения которого было инициировано выполнения правила $switch(s, o, u, op)$.

Правило $revert(s, u, o)$ предназначено для восстановления состояния субъект-сессии s до активации ею сущности-триггера или сущности-процедуры. В результате применения правила $revert(s, u, o)$ в качестве учётной записи, от имени которой функционирует субъект-сессия s, устанавливается u, а o становится функционально-ассоциированной с s сущностью.

Внутреннее правило $execute(s, o, op)$ задаёт состояние системы после выполнения SQL-кода, содержащегося в сущности-процедуре или сущности-триггере $o \in O_p \cup O_t$ субъект-сессией $s \in S$. Перед выполнением SQL-кода сущности o производится изменение учётной записи пользователя, от имени которой субъект-сессия s будет последовательно применять внешние правила $op_1, \ldots, op_k$. Учётная запись пользователя устанавливается в соответствии с режимом выполнения кода сущности-процедуры или сущности-триггера o, задаваемой значением функции $execute_as(o)$. Кроме того, перед выполнением SQL-кода сущности o в качестве функционально-ассоциированной с субъект-сессией s сущностью устанавливается сущность o. После завершения описанных подготовительных действий производится выполнение SQL-кода сущности o, что приводит к реализации преобразований состояний системы, задаваемых последовательностью правил $operations(o) = (op_1, \ldots, op_k)$. После выполнения кода производится восстановление исходной функционально-ассоциированной сущности с субъект-сессией s, а также учётной записи, от имени которой она выполнялась до применения внутреннего правила $execute(s, o, op)$. Параметр $op \in OP_{ext}$ задаёт внешнее правило, в результате применения которого было инициировано выполнение правила $execute(s, o, op)$.

Применение внутреннего правила $do_insert(s, t_1, o_{d_1}, t_2, o_{d_2})$ приводит к добавлению сущности-данные o_{d_1} в таблицу t_1, при этом содержимое сущности-данные o_{d_2} копируется во вновь создаваемую сущность-данные o_{d_1}.

Применение внутреннего правила $do_update(s, t_1, o_{d_1}, t_2, o_{d_2})$ приводит к замене содержимого сущности-данные o_{d_1} на содержимое сущности-данные o_{d_2}.

Внутреннее правило $do_delete(s, t, o_d)$ позволяет активировавшей его субъект-сессии s выполнить удаление сущности-данные o_d из содержащей её таблицы t.

В табл. 2 приведены внешние правила преобразования состояний из множества OP_{ext}, которые могут быть непосредственно применены субъект-сессиями.

Поясним приведённые в табл. 2 условия и результаты применения внешних правил преобразования состояний.

В результате применения правила $create_session(u, s)$ создаётся новая субъект-сессия s, которая в дальнейшем может выполнять действия от имени учётной записи пользователя $u \in U$.

Предположение 3. Будем считать, что все правила из табл. 1 и 2, за исключением правила $create_session(u, s)$, получают в качестве первого параметра субъект-сессию, инициировавшую выполнение данного правила. В рамках вновь созданной субъект-сессии повторное применение правила $create_session(u, s)$ не допускается.

Определение 5. Будем говорить, что применение правила $op \in OP$ инициировано учётной записью пользователя $u \in U$, если либо $op = create_session(u, s)$, либо op имеет в качестве первого параметра субъект-сессию s, такую, что $u = user(s)$.

Отметим, что в реальных СУБД учётная запись пользователя, от имени которой инициировано выполнение SQL-кода, также не может быть задана явным образом.

Определение 6. Пусть $G \vdash_{op} G'$ — переход системы $\Sigma(G^*, OP)$ из состояния G в состояние G'. Назовём переход $G \vdash_{op} G'$ инициированным извне, если $op \in OP_{ext}$ и либо op имеет вид $create_session(u, s)$, либо правило op получает в качестве первого параметра субъект-сессию s, такую, что $[s] = \varnothing$. Если переход $G \vdash_{op} G'$ инициирован извне, то будем называть правило op инициированным извне.

Замечание 5. Согласно предположению 3, применение правила $create_session(u, s)$ не может быть инициировано ранее созданной субъект-сессией. Следовательно, правило $create_session(u, s)$ может быть инициировано только извне.

Допустим, заданы состояния системы $G_0, \ldots, G_k$, где $k \geqslant 1$, и внешние правила преобразования состояний $op_1, \ldots, op_k \in OP_{ext}$, такие, что $G_0 \vdash_{op_1} G_1 \vdash_{op_2} \ldots \vdash_{op_k} G_k$ и для каждого $i = 1, \ldots, k$ переход $G_{i-1} \vdash_{op_i} G_i$ инициирован извне учётной записью пользователя $u \in U_0$, причем $op_1 = create_session(u, s_1)$. Предположим, что в результате применения указанных правил были созданы субъект-сессии $s_1, \ldots, s_m$, где $m \geqslant 1$. Согласно замечанию 5, субъект-сессии могут быть созданы только в результате применения правил, инициированных извне. Следовательно, для каждого $i \in \{1, \ldots, m\}$ существует $j \in \{1, \ldots, k\}$, такой, что $op_j = create_session(u, s_i)$. Обозначим через $op'_1, \ldots, op_{k'}$, где $k' \geqslant 1$, последовательность правил, полученную из $op_1, \ldots, op_k$ путём исключения всех правил вида $create_session(u, s_i)$, $i \geqslant 2$, и заменой всех случаев использования s_i на s_1. Пусть состояния системы $G_0, \ldots, G_{k'}$ таковы, что $G_0 \vdash_{op'_1} G'_1 \vdash_{op'_2} \ldots \vdash_{op'_{k'}} G'_{k'}$. Заметим, что состояние $G'_{k'}$ удовлетворяет соотношению $G'_{k'}\{S'_{k'}\} \sim G_k, S'_{k'} = S_k \backslash \{s_2, \ldots, s_m\}$. Следовательно, без ограничения общности, будем считать, что выполняется следующее предположение.

Предположение 4. В процессе перехода системы из состояния в состояние для каждой учётной записи пользователя создаётся не более чем одна соответствующая ей субъект-сессия. Будем считать, что в начальном состоянии G_0 системы $\Sigma(G^*, OP)$ для каждой учётной записи пользователя $u \in U$ существует единственная соответствующая ей субъект-сессия $s \in S$.

Замечание 6. Как видно из табл. 2, в результате применения внешних правил производится изменение значения счётчика времени. В большинстве правил значение этого счётчика увеличивается на единицу. Однако в результате выполнения SQL-кода

сущностей-процедур и сущностей-триггеров (при применении правил *access_insert*, *access_update*, *access_delete* и *execute_procedure*) значение счётчика времени увеличивается на величину, большую чем единица.

Определение 7. Пусть заданы состояния системы $G_0, \ldots, G_k$, где $k \geqslant 1$, и правила преобразования состояний $op_1, \ldots, op_k$, такие, что $G_0 \vdash_{op_1} G_1 \vdash_{op_2} \ldots \vdash_{op_k} G_k$ и для каждого $i = 1, \ldots, k$ переход $G_{i-1} \vdash_{op_i} G_i$ инициирован извне и $tr_k = (tr_0, \widehat{op}_1, \ldots, \widehat{op}_m)$, где $m \geqslant 0$. Обозначим через $vestige(G, op_1, \ldots, op_k)$ последовательность правил преобразования состояний $(\widehat{op}_1, \ldots, \widehat{op}_m)$.

Замечание 7. Заметим, что последовательность правил $vestige(G, op_1, \ldots, op_k)$ однозначно определяется состоянием системы G и последовательностью инициированных извне правил $op_1, \ldots, op_k$, поэтому в дальнейшем *vestige* будет рассматриваться как функция $G^* \times OP^*_{ext} \to OP^*$.

Замечание 8. Отметим, что в соответствии со способом задания перехода системы из состояния в состояние последовательность правил преобразования состояний $vestige(G_0, op_1, \ldots, op_k)$ не включает правила, условия применения которых не выполняются.

Пусть заданы состояния системы $G_0, \ldots, G_k$, где $k \geqslant 1$, и правила преобразования состояний $op_1, \ldots, op_k$, такие, что $G_0 \vdash_{op_1} G_1 \vdash_{op_2} \ldots \vdash_{op_k} G_k$ и для каждого $i = 1, \ldots, k$ переход $G_{i-1} \vdash_{op_i} G_i$ инициирован извне и $vestige(G_0, op_1, \ldots, op_k) = (\widehat{op}_1, \ldots, \widehat{op}_m)$. Пусть состояния $\widehat{G}_0, \widehat{G}_1, \ldots, \widehat{G}_m$ таковы, что $\widehat{G}_0 = G_0$ и $\widehat{G}_0 \vdash_{\widehat{op}_1} \widehat{G}_1 \vdash_{\widehat{op}_2} \ldots \vdash_{\widehat{op}_m} \widehat{G}_m$. Заметим, используя приведённые выше обозначения, что функция *vestige* обладает следующими свойствами.

Свойство 1. $G_k = \widehat{G}_m$.

Свойство 2. Для каждого $i = 1, \ldots, m$ выполняется $\widehat{time}_i = \widehat{time}_{i-1} + 1$.

Свойство 3. $vestige(G_0, op_1, \ldots, op_k) = (vestige(G_0, op_1, \ldots, op_{k-1}), vestige(G_{k-1}, op_k)) = (vestige(G_0, op_1), vestige(G_1, op_2), \ldots, vestige(G_{k-1}, op_k))$.

Свойство 4. Если в состоянии G_0 не выполнены условия применения правила $op \in OP_{ext}$, то $vestige(G_0, op) = ()$.

Свойство 5. Если выполнены условия применения правила $op \in OP_{ext}$ в состоянии G_0 и *op* не является правилом вида *access_insert*, *access_update*, *access_delete* и *execute_procedure*, то $vestige(G, op) = (op)$.

Свойство 6. $vestige(G_0, op_1, \ldots, op_k)$ не может включать правила преобразования состояний следующего вида: *access_insert*, *access_update*, *access_delete*, *execute* и *execute_procedure* — и может включать правила следующего вида: *switch*, *revert*, *do_insert*, *do_update*, *do_delete*, *create_session*, *create_procedure*, *alter_procedure*, *create_trigger*, *alter_trigger* или *grant_right*.

Выполнение указанных свойств следует из определения функции *vestige* и из способа задания внутренних и внешних правил преобразования состояний в табл. 1 и 2.

В результате применения правила $grant_right(s, u, e, \alpha_r, with_grant)$ учётной записи пользователя u предоставляется право доступа α_r к сущности e. Необходимым условием является наличие у учётной записи, от имени которой выполняется субъект-сессия s, права давать право доступа α_r к сущности e, то есть $user(s) \in Gr(e, \alpha_r)$. В случае если параметр *with_grant* равен `yes`, то самой учётной записи пользователя u даётся право предоставлять право α_r на сущность e.

Правило $create_container(s, c_p, c)$ позволяет субъект-сессии s создать новый контейнер c, включив его в существующий контейнер c_p, при этом владельцем вновь со-

зданного контейнера становится учётная запись пользователя $user(s)$. Необходимым условием применения данного правила является наличие у учётной записи пользователя $user(s)$ права доступа $alter_r$ к контейнеру c_p.

Замечание 9. В результате применения правил $alter_procedure$, $alter_trigger$ и $create_container$ в множества R и Gr не выполняется добавление прав доступа учётной записи владельца к вновь созданной сущности. Это связано с тем, что при задании учётной записи владельца в соответствии с правилами задания в множества R_e и Gr_e включаются права доступа всех её иерархических владельцев.

В результате применения правила $create_procedure(s, c, o_p, md, op_1, \ldots, op_k)$ в контейнере c, не являющемся таблицей, создаётся новая сущность-процедура o_p, содержащая SQL-код, выполнение которого приводит к реализации правил преобразования состояний $op_1, \ldots, op_k$, при этом владельцем вновь созданной сущности-процедуры o_p становится учётная запись пользователя $user(s)$, а режим выполнения o_p устанавливается равным m. Необходимым условием выполнения данного правила является наличие у учётной записи пользователя $user(s)$ права доступа $alter_r$ к контейнеру c.

В результате применения правила $alter_procedure(s, o_p, md, op_1, \ldots, op_k)$ выполняется изменение режима выполнения существующей сущности-процедуры o_p, а также замена её SQL-кода на новый, выполнение которого приводит к реализации правил преобразования состояний $op_1, \ldots, op_k$. Необходимым условием применения данного правила является наличие у учётной записи пользователя $user(s)$ права доступа $alter_r$ к сущности-процедуре o_p. Отметим, что в отличие от правила $create_procedure$ владелец сущности-процедуры o_p не изменяется.

В результате применения правила $create_trigger(s, t, o_t, \alpha_r, md, op_1, \ldots, op_k)$ в таблице t создаётся новая сущность-триггер o_t, содержащая SQL-код, выполнение которого приводит к реализации правил преобразования состояний $op_1, \ldots, op_k$, при этом владельцем вновь созданной сущности-триггера o_t становится учётная запись пользователя $user(s)$, а режим выполнения o_t задаётся равным m. Созданная сущность-триггер активизируется только при реализации к таблице t права доступа α_r. Если создаваемая сущность-триггер имеет тип $\alpha_r = append_r$, то она активизируется при применении в отношении таблицы t внешнего правила $access_insert$. В случае если o_t имеет тип $\alpha_r = write_r$, то o_t активизируется при применении в отношении таблицы t внешнего правила $access_update$. Если же $\alpha_r = delete_r$, то сущность-триггер o_t активизируется при применении к таблице t внешнего правила $access_delete$. Необходимым условием применения правила $create_trigger$ является наличие у учётной записи пользователя $user(s)$ права доступа $alter_r$ к таблице t.

В результате применения правила $alter_trigger(s, t, o_t, md, op_1, \ldots, op_k)$ в таблице t выполняется модификация режима выполнения существующей сущности-триггера o_t, а также замена её SQL-кода на новый, выполнение которого приводит к реализации правил преобразования состояний $op_1, \ldots, op_k$. Необходимым условием применения данного правила является наличие у учётной записи пользователя $user(s)$ права доступа $alter_r$ к таблице t. Отметим, что в отличие от правила $create_trigger$ не выполняется изменение учётной записи владельца сущности-триггера o_t, а также вида доступа, при реализации которого активизируется сущность-триггер o_t.

Предположение 5. Будем считать, что выполнение SQL-кода сущностей-триггеров и сущностей-процедур не приводит к применению правил $create_container$, $create_procedure$, $alter_procedure$, $create_trigger$ или $alter_trigger$, то есть перечисленные правила могут быть применены только извне. Из данного предпо-

ложения следует, что при применении правил $create_procedure(\ldots, op_1, \ldots, op_k)$, $alter_procedure(\ldots, op_1, \ldots, op_k)$, $create_trigger(\ldots, op_1, \ldots, op_k)$ и $alter_trigger(\ldots, op_1, \ldots, op_k)$ ни одно из правил $op_1, \ldots, op_k$ не должно быть равно $create_procedure$, $alter_procedure$, $create_trigger$ или $alter_trigger$.

Замечание 10. Заметим, что если $o \in O_p \cup O_t$, $operations(o) = (op_1, \ldots, op_k)$, где $k \geqslant 0$, то для каждого $i = 1, \ldots, k$ выполняется неравенство $op_i \neq \neq create_session$. Предположим противное: существует $i \in \{1, \ldots, k\}$, такое, что $op_i = create_session$. Согласно способу задания правил преобразования состояний системы, выполнение правил сущностей-триггеров и сущностей-процедур o осуществляется ранее созданной субъект-сессией $s \in S$. Следовательно, применение правила $op_i = create_session$ инициируется субъект-сессией s, что противоречит предположению 3 о том, что в рамках вновь созданной субъект-сессии повторное применение правила $create_session$ не допускается. Следовательно, правила, связанные с сущностями-процедурами и сущностями-триггерами, не могут включать правила вида $create_session$. Кроме того, при применении правил $create_procedure(s, c, o_p, md, op_1, \ldots, op_k)$, $alter_procedure(s, o_p, md, op_1, \ldots, op_k)$, $create_trigger(s, t, o_t, m, op_1, \ldots, op_k)$ и $alter_trigger(s, t, o_t, md, op_1, \ldots, op_k)$ ни одно из правил $op_1, \ldots, op_k$ не должно иметь вид $create_session$.

Замечание 11. Допустим, что $o \in O_p \cup O_t$, $operations(o) = (op_1, \ldots, op_k)$, где $k \geqslant 0$, $op \in operations(o)$, то есть существует $i \in \{1, \ldots, k\}$, такое, что $op = op_i$. Следовательно, в соответствии с предположением 5 и замечанием 10 правило op не может иметь вид $create_container$, $create_procedure$, $alter_procedure$, $create_trigger$, $alter_trigger$ и $create_session$. Учитывая это, а также то, что $op \in OP_{ext}$ по определению функции $operations$, получаем, что в соответствии с табл. 2 op может иметь вид $grant_right$, $execute_procedure$, $access_insert$, $access_update$ или $access_delete$.

Замечание 12. Из способа задания внешних правил видно, что владельцем созданной сущности (сущности-триггера, сущности-процедуры или контейнера) становится учётная запись пользователя, от имени которой выполнялась субъект-сессия, инициировавшая её создание. В соответствии с предположением 2 в системе отсутствуют внешние и внутренние правила, позволяющие изменять учётную запись владельца сущности.

Правило $execute_procedure(s, o_p)$ позволяет вызывающей субъект-сессии s при наличии права доступа $execute_r$ к сущности-процедуре o_p выполнить её SQL-код. При этом результирующее состояние G' есть результат выполнения внутреннего правила $execute(s, o_p, execute_procedure(s, o_p))$. Отметим, что при выполнении SQL-кода сущности-процедуры может осуществляться активация других сущностей-процедур и сущностей-триггеров.

Замечание 13. Будем считать, что при активации правил преобразования состояний, связанных с сущностями-процедурами и сущностями-триггерами, не возникает циклов, являющихся результатом повторного применения правил ранее активированных сущностей.

Применение правила $access_insert(s, t_1, o_{d_1}, t_2, o_{d_2})$ приводит к добавлению сущности-данные o_{d_1} в таблицу t_1, при этом содержимое сущности-данные o_{d_2} копируется во вновь создаваемую сущность-данные o_{d_1}. При применении правила $access_insert$ выполняется активация всех сущностей-триггеров типа $append_r$, связанных с таблицей t_1. Необходимым условием применения данного правила является наличие у учётной записи пользователя $user(s)$ прав доступа на чтение к таблице t_2, а также права

доступа на запись к целевой таблице t_1, в которую выполняется добавление сущности-данные o_{d_1}.

В результате применения правила $access_update(s, t_1, o_{d_1}, t_2, o_{d_2})$ выполняется обновление сущности-данные o_{d_1} с использованием данных, содержащихся в сущности-данные o_{d_2}. При применении правила $access_update$ выполняется активация сущностей-триггеров типа $write_r$ таблицы, содержащей сущность-данные o_{d_1}. Необходимым условием применения данного правила является наличие у учётной записи пользователя $user(s)$ прав доступа на чтение к таблице t_2, а также права доступа на запись к целевой таблице t_1.

Правило $access_delete(s, t, o_d)$ позволяет активировавшей его субъект-сессии s выполнить удаление сущности-данные o_d из содержащей её таблицы. При применении правила $access_delete$ выполняется активация всех сущностей-триггеров типа $delete_r$, связанных с таблицей t. Необходимым условием применения данного правила является наличие у учётной записи пользователя $user(s)$ прав доступа на удаление к таблице t.

Замечание 14. Заметим, что условия применения правил $access_insert(s, t_1, o_{d_1}, t_2, o_{d_2})$, $access_update(s, t_1, o_{d_1}, t_2, o_{d_2})$ и $access_delete(s, t, o_d)$ ослаблены по сравнению с соответствующими внутренними правилами $do_insert(s, t_1, o_{d_1}, t_2, o_{d_2})$, $do_update(s, t_1, o_{d_1}, t_2, o_{d_2})$ и $do_delete(s, t, o_d)$. Отличие в условиях применения указанных правил состоит в отсутствии требований, связанных с наличием или отсутствием сущностей-данных в соответствующих таблицах. Необходимым условием применения перечисленных правил остаётся только наличие соответствующих прав доступа к таблицам. Отметим, что указанные ослабления условий применения правил $access_insert$, $access_update$ и $access_delete$ потенциально могут привести к появлению дополнительных путей распространения прав доступа в системе. Если в дальнейшем будут определены достаточные условия, при выполнении которых невозможно распространение прав доступа, то эти достаточные условия будут применимы и при наличии дополнительных ограничений на условия применения правил.

Замечание 15. В соответствии со способом задания применение внешних правил (за исключением $access_delete$) не приводит к удалению элементов из множеств, являющихся элементами состояния системы $\Sigma(G^*, OP)$. Учитывая, что в условиях применения правил $grant_right$, $alter_procedure$, $alter_trigger$, $execute_procedure$, $access_insert$, $access_update$ и $access_delete$ нет проверок на отсутствие элементов в состоянии системы, отметим, что верно следующее замечание. Если в состоянии G_0 системы $\Sigma(G^*, OP)$ выполнялись условия применения правила $op \in OP$, являющегося одним из перечисленных выше правил, и существуют правила преобразования состояний $op_1, \ldots, op_k \in OP$, где $k \geqslant 0$, то условия применения правила $op \in OP_{ext}$ будут выполняться и в состоянии $G' = G(op_1, \ldots, op_k)$.

3. Условия передачи прав доступа

При отсутствии ограничений на кооперацию субъект-сессий верно следующее утверждение о передаче прав доступа учётными записям пользователей к контейнерам и сущностям-процедурам.

Утверждение 4. Пусть G_0 — состояние системы $\Sigma(G^*, OP)$, $u \in U_0$, $e \in C_0 \cup O_{p_0}$ и пусть $\alpha_r \in R_r$ — некоторое право доступа. Тогда существуют состояния $G_1, \ldots, G_k$, где $k \geqslant 0$, правила преобразования состояний $op_1, \ldots, op_k$, такие, что $G_0 \vdash_{op_1} G_1 \vdash_{op_2} \ldots \vdash_{op_k} G_k$, $(u, e, \alpha_r) \in R_{e_k}$ и для каждого $i = 1, \ldots, k$ переход $G_{i-1} \vdash_{op_i} G_i$ инициирован извне.

Доказательство. Если $(u, e, \alpha_r) \in R_{e_0}$, то $k = 0$ и условия утверждения выполнены. Пусть $(u, e, \alpha_r) \notin R_{e_0}$. Заметим, что ввиду $Gr_{own_0} \subseteq Gr_0$ и способа задания множеств Gr_{own_0} и $Gr_0(e, \alpha_r)$ выполняется соотношение

$$\{(u, e, \alpha_r) : u \in owners_0(e)\} \subseteq Gr_0(e, \alpha_r). \tag{4}$$

Так как множество $owners_0(e)$ включает, как минимум, учётную запись пользователя $owner(e)$, то из (4) следует, что множество $Gr_0(e, \alpha_r)$ не пусто и существует пользователь $u' \in Gr_0(e, \alpha_r)$. Рассмотрим следующую последовательность внешних правил: $op_1 = create_session(u', s)$, $op_2 = grant_right(s, u, e, \alpha_r, no)$. Тогда существуют состояния G_1 и G_2, такие, что $G_0 \vdash_{op_1} G_1 \vdash_{op_2} G_2$. Заметим, что выполнены условия применения правила op_2, так как $u' \in Gr_0(e, \alpha_r)$, а следовательно, $user_1(s) = u' \in$ $\in Gr_1(e, \alpha_r) = Gr_0(e, \alpha_r)$. Отсюда $(u, e, \alpha_r) \in R_{e_2}$ и все переходы $G_0 \vdash_{op_1} G_1$, $G_1 \vdash_{op_2} G_2$ инициированы извне. ■

Из данного утверждения следует, что при отсутствии ограничений на кооперацию субъект-сессий иерархические владельцы сущности могут предоставить к ней произвольное право доступа любой учётной записи.

Заключение

Таким образом, для обеспечения возможности теоретического анализа безопасности СУБД построена формальная модель, в которую по сравнению с РОСЛ ДП-моделью добавлены несколько новых существенных элементов, позволяющих полнее учесть особенности управления доступом в СУБД. В дальнейшем планируется развитие построенной модели по следующим направлениям: анализ условий получения прав доступа при отсутствии кооперации между субъект-сессиями, определение условий активации сущностей-процедур и сущностей-триггеров от имени заданных учётных записей, а также уточнение правил преобразования состояний для определения условий реализации информационных потоков в случаях наличия или отсутствия кооперации субъект-сессий.

ЛИТЕРАТУРА

1. Руководящий документ. Безопасность информационных технологий. Критерии оценки безопасности информационных технологий: в 3 ч. Введён в действие Приказом Гостехкомиссии России от 19.06.02 г. № 187.
2. *Девянин П. Н.* Модели безопасности компьютерных систем. Управление доступом и информационными потоками: учеб. пособие для вузов. М.: Горячая линия-Телеком, 2011. 320 с.
3. *Колегов Д. Н.* Дискреционная модель безопасности управления доступом и информационными потоками в компьютерных системах с функционально или параметрически ассоциированными сущностями: дис. ... канд. техн. наук. Томск, 2009.
4. *Девянин П. Н.* Ролевая ДП-модель управления доступом и информационными потоками в операционных системах семейства Linux // Прикладная дискретная математика. 2012. № 1(15). С. 69–90.

ПРИКЛАДНАЯ ТЕОРИЯ ГРАФОВ

DOI 10.17223/20710410/19/6

УДК 519.1

ИССЛЕДОВАНИЕ КЛИКОВЫХ ПОКРЫТИЙ РЁБЕР ГРАФА

И. А. Бадеха

Российский государственный социальный университет, г. Москва, Россия

E-mail: ivanbadekha@gmail.com

Исследуются кликовые покрытия рёбер графа. Описаны графы, имеющие единственное кликовое покрытие рёбер, рассмотрена возможность снижения порядка (сжатия) графа с сохранением структуры покрытия, приведены методы, упрощающие процедуру поиска минимального кликового покрытия рёбер, исследована взаимосвязь между изоморфизмами графов и соответствующих сжатых графов.

Ключевые слова: *графы, покрытия графов кликами, кликовое покрытие рёбер, изоморфизм графов, NP-полные задачи.*

Введение

Кликовые покрытия рёбер графа и порождаемое представление графа в виде объединения клик изучается в теории графов достаточно давно. Одно из первых исследований проведено в работе L. Kou, L. Stockmeyer и C. Wong [1], в которой показана взаимосвязь данной задачи с другими задачами (поиск кликового покрытия вершин, проблема конфликта ключей), доказан ряд утверждений о полиномиальных алгоритмах поиска покрытия, приведён простейший эвристический алгоритм поиска покрытия, достигающий результата за полиномиальное время. Обзор вопросов, связанных с покрытиями и разбиениями графов, приводит J. Orlin в работе [2], которая также вышла в 70-е годы. Современные результаты по данной проблеме получены J. Gramm, J. Guo, F. Huffner, R. Niedermeier и отражены в работе [3], в которой авторы предлагают и исследуют метод редукции графа, позволяющий восстановить кликовое покрытие графа рёбрами из покрытия меньшего графа. Обзор имеющихся современных результатов приводит в тезисах к своей диссертации на соискание степени Master of science M. Cavers [4].

Введём основные используемые понятия. Базовые понятия теории графов можно найти в монографии Ф. Харари [5].

Примем следующие обозначения: $G = (V, E)$, где V — множество вершин, $|V| = n$; E — множество рёбер графа G. Для любого графа T через V_T, E_T обозначим соответственно множества его вершин и рёбер.

Кликой графа G называется любой его максимальный полный подграф, то есть подграф, все вершины которого попарно смежны [5]. При этом *максимальным* (по включению) полным подграфом клика называется в том смысле, что никакой другой подграф, содержащий клику и не совпадающий с ней, не является полным. Полный подграф из трёх вершин будем далее называть треугольником. Легко видеть, что для любого полного подграфа существует клика, его содержащая.

Под *окружением вершины* v понимается множество $N(v)$, состоящее из вершин графа G, смежных с вершиной v. Расширим понятие окружения.

Определение 1. *Окружением ребра* e будем называть множество вершин $N(e)$ графа G, смежных одновременно с обоими концами ребра e.

По аналогии с данным определением, окружением множества вершин $M \subseteq V_G$ будем считать множество вершин $N(M)$, смежных одновременно с каждой вершиной из M.

В некоторых случаях будем рассматривать *невыколотое окружение* вершины, т. е. множество, состоящее из самой вершины и всех смежных с ней вершин. Аналогичным образом определяется невыколотое окружение ребра: это множество, состоящее из концов u и v ребра (u, v) и всех вершин его окружения. Невыколотым окружением множества вершин M будем называть множество $H(M) = M \cup N(M)$. Невыколотое окружение ребра e будем обозначать $H(e)$, невыколотое окружение вершины $v - H(v)$. Граф, порождённый множеством вершин M графа G, будем обозначать $\overline{M}_G$.

Определение 2. *Кликовым покрытием рёбер* (КПР) графа G называется множество клик, такое, что каждое ребро графа G входит, как минимум, в одну из клик данного множества.

Определение 3. КПР графа назовём *тупиковым*, если при исключении из него любой клики оно перестаёт покрывать все рёбра графа.

Каждая изолированная вершина в КПР входит в единственную клику, состоящую из одной этой вершины. Так как случай наличия изолированной вершины тривиален, далее не будем уделять ему особого внимания.

Определение 4. *Минимальным КПР* (МКПР) графа G назовем КПР графа G, содержащее наименьшее из всех возможных КПР число клик.

Мощность МКПР графа G иногда называют его кликовой характеристикой. Отметим, что задача поиска МКПР графа G является NP-полной (см., например, [2]).

Далее в п. 1 и 2 рассматриваются методы упрощения поиска МКПР. В п. 3 исследуется класс графов, обладающих единственным КПР.

1. Упрощение задачи построения МКПР: использование собственных рёбер и полных наборов баз

1.1. Собственные рёбра графа

Рассмотрим особый вид рёбер графа G, наличие которых упрощает процедуру поиска МКПР.

Определение 5. Ребро e назовем *собственным ребром клики* K, если оно принадлежит клике K и не принадлежит никакой другой клике графа G.

Определение 6. Клику K назовем *зафиксированной*, если она содержит хотя бы одно собственное ребро либо содержит только одну вершину.

Укажем простой способ определения собственного ребра.

Утверждение 1. Ребро e является собственным ребром графа G тогда и только тогда, когда $\overline{N(e)}_G$ представляет собой клику в графе G.

Доказательство. Пусть ребро $e = (a, b)$ является собственным и клика K — единственная клика, содержащая ребро e. Пусть существует вершина x, такая, что $x \in N(e)$ и $x \notin K$. Тогда вершины a, b, x образуют треугольник, который не содержится внутри клики K. Это значит, что существует клика K', содержащая данное ребро и не совпадающая с кликой K, следовательно, ребро e не является собственным, что противоречит предположению.

Пусть теперь $\overline{N(e)}_G = K$, где K — клика. Так как все вершины клики, в которую входит ребро, принадлежат его окружению, клика K — зафиксированная клика, содержащая собственное ребро e. ■

Если граф имеет зафиксированные клики, то процедура поиска МКПР для него упрощается следующим образом: сначала определяются зафиксированные клики, которые входят в любое КПР и соответственно в МКПР, а затем уже производится поиск оставшихся клик МКПР.

1.2. Использование полных наборов баз

Введём ряд понятий, которые будут использованы в дальнейшем для ускорения поиска МКПР. Будем говорить, что *полный подграф H графа G базируется на ребре e*, если любая клика K, содержащая ребро e, содержит H в качестве своего подграфа.

Под пересечением графов далее будем понимать граф, множество вершин которого получено пересечением множеств вершин пересекаемых графов, а множество рёбер — пересечением множеств их рёбер. Очевидно, что пересечением полных подграфов некоторого графа является также полный подграф этого графа.

Определение 7. *Базой b_e ребра e в графе G* назовём полный подграф графа G, являющийся пересечением всех полных подграфов, базирующихся на ребре e. При этом ребро e назовём порождающим ребром базы b_e.

Докажем утверждение, которое позволяет получить характеризацию базы ребра.

Утверждение 2. Пусть $e = (a, b)$ — ребро графа G, $N(e)_d$ — множество, состоящее из доминирующих вершин окружения ребра e. Подграф графа G, порождённый множеством вершин $\{a, b\} \cup N(e)_d$, является базой ребра e.

Доказательство. Несложно убедиться, что граф $\overline{\{a, b\} \cup N(e)_d}$ является полным подграфом графа G. Для этого необходимо отметить, что любая доминирующая вершина из окружения ребра смежна с каждой доминирующей вершиной окружения и обоими концами ребра, а каждый конец ребра смежен с каждой вершиной своего окружения, в том числе с любой доминирующей вершиной окружения. Пусть ребро e принадлежит клике K. Любая вершина клики K смежна с обоими концами ребра e, а значит, принадлежит невыколотому окружению ребра. Предположим, что вершина d, являющаяся доминирующей вершиной в окружении ребра e, не принадлежит клике K. Тогда вершина d смежна с каждой вершиной окружения и с обоими концами ребра. Значит, подграф, порождённый множеством вершин $V_K \cup \{d\}$, является полным. Так как указанный полный граф содержит подграф K, то последний в этом случае не является максимальным по включению полным графом, то есть кликой. Данное противоречие подтверждает верность доказываемого утверждения. ■

Таким образом, поиск базы данного ребра $e = (a, b)$ может производиться по следующему алгоритму: перебираем вершины графа и определяем количество k вершин, смежных с a и b ($O(n)$ элементарных операций). Для каждой вершины u, смежной с обоими концами ребра, снова перебираем все вершины графа и ведём подсчёт количества $k(u)$ вершин, смежных одновременно с a, b и u; если $k(u) = k$, то вершину u помечаем как доминирующую ($O(n)$ элементарных операций).

Получаем, что трудоёмкость нахождения базы для конкретного ребра составляет $O(n^2)$ элементарных операций.

Определение 8. Множество $B = \{b_1, \ldots, b_k\}$, где каждый элемент b_i является базой какого-то ребра, назовем *полным набором баз графа G*, если каждое ребро графа G входит в состав хотя бы одной из этих баз.

Определение 9. *Кликовым покрытием баз* (сокращенно КПБ) графа G для некоторого полного набора баз B назовем множество клик графа G, такое, что каждая база из набора B входит как минимум в одну из клик этого множества.

Докажем утверждение о взаимосвязи между КПР и КПБ.

Утверждение 3. Любое КПБ графа G является его КПР, и любое КПР является КПБ.

Доказательство. Пусть P — КПР графа G. Если ребро e входит в некоторую клику K покрытия P, то и порождённая этим ребром база b_e также входит в эту клику. Так как все рёбра покрыты кликами из КПР P, а любая база порождается каким-то ребром, то в любом полном наборе баз B все базы покрыты кликами из P. Следовательно, P — КПБ графа G.

Теперь предположим, что P — КПБ графа G. Тогда любая база b_e из некоторого полного набора баз B входит в какую-то клику K из P. Поскольку B — полный набор баз, каждое ребро e графа G входит в некоторую базу b_e из B, а следовательно, и в какую-то клику K из P. Следовательно, P — КПР графа G. ■

Таким образом, для получения КПР достаточно получить КПБ.

Отметим, что для того чтобы найти кликовое покрытие баз из некоторого полного набора, достаточно найти множество клик, покрывающих все порождающие рёбра для баз, входящих в этот полный набор. Действительно, если порождающее ребро e для базы b_e покрыто некоторой кликой K, то, согласно определению базы, b_e также покрывается кликой K.

Введём понятие минимального КПБ (по аналогии с минимальным КПР).

Определение 10. *Минимальным КПБ* (сокращенно МКПБ) графа G назовем КПБ графа G, содержащее наименьшее из всех возможных КПБ число клик.

Центральный результат, являющийся следствием из приведённых выше рассуждений и обуславливающий перспективность использования баз в задаче поиска МКПР, сформулирован в следующем утверждении.

Утверждение 4. Пусть P — МКПБ графа G для некоторого полного набора баз B. Тогда P является МКПР графа G.

Доказательство. Пусть P' — МКПР графа G и $|P'| = r$. Из утверждения 3 следует, что любое МКПР графа G является его КПБ. Предположим, существует P'' — МКПБ графа G, такое, что $|P''| < r$. Тогда P'' является КПР графа G и $|P''| < |P'|$, что противоречит тому, что P' является минимальным КПР. Значит, P' является МКПБ графа G. Следовательно, $|P'| = |P|$. Значит, P является МКПР графа G. ■

Таким образом, можно строить покрытие только тех рёбер, которые являются порождающими для баз, входящих в полный набор, не заботясь об остальных рёбрах.

Далее опишем простейший метод построения некоторого полного набора баз в графе G. Изначально считая множество B пустым, для каждого ребра $e \in E_G$ последовательно проведем следующие действия:

1. Найдём множество доминирующих вершин окружения ребра e, обозначим его как b_e.
2. Добавим элемент b_e в множество B.
3. Рёбра, входящие в состав b_e, исключим из дальнейшего рассмотрения.

Когда закончатся рассматриваемые рёбра, будем считать множество B сформированным.

С учётом трудоёмкости $O(n^2)$ нахождения базы ребра и оценки сверху n^2 на количество рёбер, трудоёмкость предложенного алгоритма составляет $O(n^4)$.

Утверждение 5. Множество B, полученное по вышеописанному алгоритму, является полным набором баз графа G.

Доказательство. Так как любой доминирующей вершиной невыколотого окружения ребра e можно расширить любой полный подграф, содержащий ребро e, множество $H(e)_D$ является базой ребра e. Так как ребро исключается из рассмотрения только в том случае, если оно покрывается базой, порождённой другим ребром, то множество B покрывает все рёбра. Таким образом, B является полным набором баз графа G. ■

Подавляющее большинство существующих алгоритмов построения МКПР строит покрытия последовательно, добавляя в уже построенное покрытие новые клики. Утверждение 5 позволяет останавливать задачу поиска МКПР в случае, когда в уже найденный набор клик входят все порождающие рёбра некоторого полного набора баз. При таком подходе возникают следующие аспекты.

1) К трудоёмкости поиска МКПР добавляется трудоёмкость построения полного набора баз. Выше предложен алгоритм такого построения с трудоёмкостью $O(n^4)$. Однако полный набор из малого числа баз снижает количество рёбер, которые необходимо покрыть в процессе построения МКПР. Соответственно перед поиском МКПР необходимо найти полный набор из как можно меньшего числа баз; можно рассмотреть другие, более трудоёмкие алгоритмы построения полного набора баз как можно меньшей мощности. Вместе с тем неясно, какой выигрыш при построении МКПР даст выделение порождающих рёбер, которые необходимо покрыть при построении, поэтому такое увеличение трудоёмкости может оказаться нецелесообразным.

2) Используемый алгоритм (эвристический или точный) поиска МКПР можно модернизировать следующим образом: если на каком-то шаге возникает возможность выбрать произвольным образом следующее ребро для добавления к уже покрытым рёбрам, в первую очередь необходимо выбирать порождающие рёбра баз.

Итак, основным достоинством предложенного метода является тот факт, что поиск полного набора баз добавляет к трудоёмкости алгоритма построения МКПР небольшую величину (ограниченную полиномом от числа вершин графа), в то время как при его применении трудоёмкость построения МКПР может существенно уменьшиться за счёт того, что на некотором шаге алгоритма будут покрыты все порождающие рёбра баз (но ещё не все рёбра графа).

2. Редукция графа в задаче построения МКПР: сжатие графа

2.1. Основные понятия

Пусть дано покрытие графа кликами $K_1, \ldots, K_s$. Для удобства примем обозначение $I = \{1, \ldots, s\}$. Далее буквой U будем обозначать некоторое подмножество I: $U \subseteq I$.

Определение 11. В принятых выше обозначениях для графа $G = (V, E)$ *классом, порождённым множеством* U, назовём множество

$$M_U = \bigcap_{i \in U} V_{K_i} \setminus \bigcup_{j \in I \setminus U} V_{K_j}.$$

В дальнейшем будем рассматривать в качестве U непустые множества, поскольку в случае $U = \varnothing$ множество M_U также пусто. На рис. 1 изображён граф G и все клас-

сы M_U для него. В графе G пять непустых классов M_U, три из которых содержат по две вершины, а два — по одной.

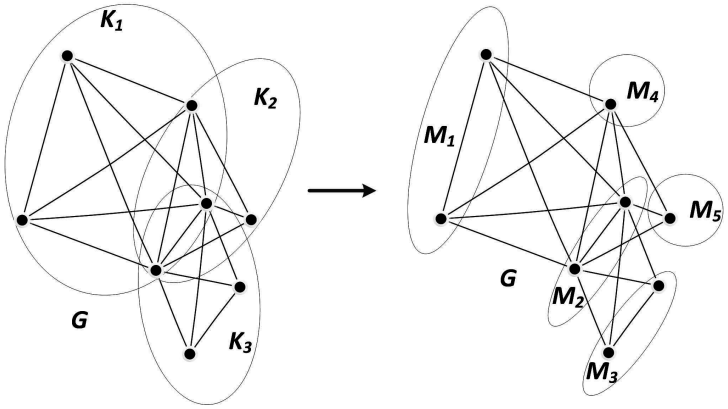

Рис. 1. Покрытие графа G (слева); множества $M_1 = M_{\{1\}}$, $M_2 = M_{\{1,3\}}$, $M_3 = M_{\{3\}}$, $M_4 = M_{\{1,2\}}$, $M_5 = M_{\{2\}}$ (справа)

Отметим очевидный факт: все вершины, принадлежащие одному классу M_U, образуют полный подграф. Это следует из того, что любые две вершины, принадлежащие одному классу, принадлежат одной клике с номером $i \in U$ и, следовательно, соединены ребром в графе G.

Утверждение 6. Две вершины $v, w \in V$ входят в множество M_U для некоторого U тогда и только тогда, когда их невыколотые окружения совпадают.

Доказательство. Необходимость. Пусть вершины v и w находятся в одном множестве M_U. Отметим, что вершины v и w соединены ребром, так как M_U — полный подграф. Пусть существует вершина $u \in V$, соединённая ребром с вершиной v. Значит, вершины u и v входят в общую клику, следовательно, $u \in V_{K_p}$ для некоторого $p \in U$. Так как в силу $p \in U$ и $w \in M_U$ клика K_p содержит и вершину w, то $(w, u) \in E$. Аналогично из смежности произвольной вершины с вершиной w следует её смежность с вершиной v. Таким образом, невыколотые окружения вершин v и w совпадают.

Достаточность. Пусть вершины v и w соединены ребром и $H(v) = H(w)$. Тогда любая клика либо содержит обе эти вершины, либо не содержит ни одной из них, ведь если K — полный подграф, $v \in V_K$ и $w \notin V_K$, то K не является кликой, так как существует полный подграф $K' = K \cup \{w\}$, содержащий K. Это значит, что для любого $p \in I$ выполнено: из $v \in V_{K_p}$ следует $w \in V_{K_p}$ и из $w \in V_{K_p}$ следует $v \in V_{K_p}$, то есть $v \in M_U \iff w \in M_U$. В качестве U возьмём множество номеров тех клик, в которые входят обе вершины v и w. Тогда $v, w \in M_U$. ■

Следствие 1. Отношение принадлежности двух вершин одному классу M_U является отношением эквивалентности.

Таким образом, покрытие графа кликами задает группировку вершин в классы с одинаковыми невыколотыми окружениями. В дальнейшем будем называть классы вершин M_U *классами эквивалентности*. Введём следующее обозначение: для вершины v под U_v будем понимать такое подмножество I, для которого $v \in M_{U_v}$.

Определение 12. *Множеством классов вершин* графа G для некоторого покрытия назовём множество $\mathcal{M} = \{M \subset V_G : M \neq \varnothing \ \& \ \exists U \subseteq I (M = M_U)\}$.

Так как эквивалентные между собой вершины, формирующие единый класс, определяются исходя из совпадающих невыколотых окружений, не зависящих от выбора КПР, то верно

Утверждение 7. Для любых покрытий графа кликами множества классов вершин совпадают.

Так как любая вершина извне конкретного класса либо входит в невыколотое окружение каждой вершины класса, либо не смежна ни с одной из них, для классов эквивалентности вершин также выполнено следующее

Утверждение 8. Для любых двух классов эквивалентности M_1 и M_2 либо все вершины класса M_1 смежны со всеми вершинами класса M_2, либо ни одна из вершин класса M_1 не смежна ни с одной вершиной класса M_2.

Определение 13. Назовём два класса вершин M_1 и M_2 *соседними*, если все вершины класса M_1 смежны со всеми вершинами класса M_2.

В случае, когда требуется указать, о каком графе идет речь, будем использовать следующие обозначения. Под множеством U^G для некоторого покрытия $\{K_1, \ldots, K_s\}$ графа G будем понимать подмножество множества $I = \{1, \ldots, s\}$. Класс эквивалентности в графе G, порождённый множеством U^G, обозначается M_{U^G}.

Для обозначения класса эквивалентности вершин, в который входит вершина u, будем иногда прибегать к такой упрощённой записи: M_u (или M_u^G, если необходимо указать, что рассматривается класс эквивалентности вершин графа G).

2.2. Основные результаты

Пусть дан простой граф $G = (V, E)$ и некоторое его покрытие $\{K_1, \ldots, K_s\}$. Рассмотрим взвешенный граф $T(G) = (V_{T(G)}, E_{T(G)})$, полученный следующим образом: каждому непустому классу эквивалентности M графа G соответствует одна вершина $t(M)$ графа $T(G)$, вес которой равен $|M|$, и других вершин в графе $T(G)$ нет. Две вершины $t(M_1)$ и $t(M_2)$ графа $T(G)$ соединены ребром в графе $T(G)$ в том и только в том случае, если классы M_1 и M_2 соседние. Класс всевозможных графов, каждый из которых может быть получен описанным выше образом из какого-то графа, назовём классом $\mathcal{G}$.

На рис. 2 изображён граф G и соответствующий ему граф $T(G)$.

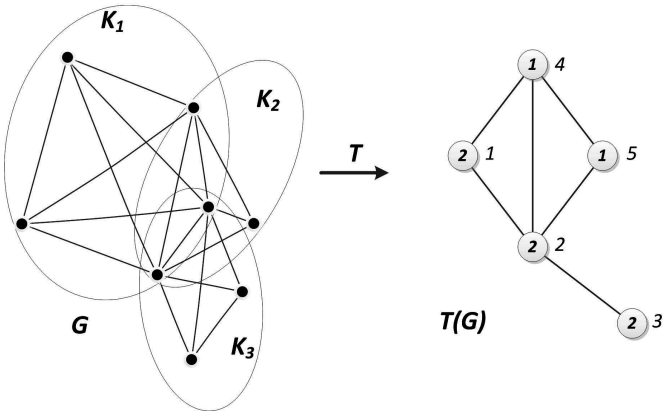

Рис. 2. Покрытие графа G (слева); взвешенный граф $T(G)$ (справа)

Исследуем некоторые свойства, которыми обладает класс $\mathcal{G}$.

Легко видеть, что по данному графу $T(G)$ можно с точностью до изоморфизма восстановить граф G следующим способом:

1. Каждой вершине v графа $T(G)$ поставить в соответствие полный граф K_v размера, равного весу этой вершины.
2. Для каждой пары смежных вершин v, w графа $T(G)$ все вершины графа K_v попарно соединить со всеми вершинами графа K_w.

Согласно построению, для каждого полного подграфа графа G получается полный подграф графа $T(G)$, и наоборот. При этом двум разным полным подграфам графа G соответствуют два разных полных подграфа графа $T(G)$, и также наоборот. Это следует из того, что вершины каждого класса эквивалентности графа G либо все входят в некоторый полный подграф, либо все в него не входят.

Отметим, что в графе $T(G)$ может присутствовать изолированная вершина, даже если в графе G изолированных вершин нет. Это возможно в том и только в том случае, если в графе G есть отдельная компонента связности, являющаяся кликой. Случай наличия изолированной вершины не приводит ни к каким сложностям, поскольку любая такая вершина образует клику, которая входит в любое покрытие графа.

Лемма 1. Если в графе G вершины a и b принадлежат разным классам M_a и M_b, то в графе $T = T(G)$ вершины $t(M_a)$ и $t(M_b)$ принадлежат разным классам $M^T_{t(M_a)}$ и $M^T_{t(M_b)}$.

Доказательство. Из того, что вершины a и b принадлежат разным классам M_a и M_b, в силу утверждения 6 и определения невыколотого окружения вершины следует, что существует вершина $u \in V$, такая, что $(u, a) \in E$ и $(u, b) \notin E$.

С л у ч а й 1. Пусть вершины a и b не соединены ребром в графе G. Тогда $t(M_a)$ и $t(M_b)$ не соединены ребром в графе T, так как вершины класса M_a не смежны с вершинами класса M_b. Значит, невыколотые окружения вершин $t(M_a)$ и $t(M_b)$ не совпадают и эти вершины принадлежат разным классам эквивалентности $M^T_{t(M_a)}$ и $M^T_{t(M_b)}$.

С л у ч а й 2. Пусть вершины a и b соединены ребром в графе G. Тогда, согласно утверждению 8, все вершины класса M_a смежны с вершинами класса M_b. Значит, вершины $t(M_a)$ и $t(M_b)$ соединены ребром в графе T. По аналогичным соображениям, вершины $t(M_u)$ и $t(M_a)$ соединены ребром в графе T. По соображениям, аналогичным приведённым в п. 1, вершины $t(M_u)$ и $t(M_b)$ не соединены ребром в графе T. Из этого следует, что $M^T_{t(M_a)}$ и $M^T_{t(M_b)}$ различны. ■

Теорема 1 (критерий принадлежности графа классу $\mathcal{G}$). Граф Γ принадлежит классу $\mathcal{G}$ тогда и только тогда, когда мощность любого класса M^Γ_u не превосходит 1.

Доказательство. Необходимость. Предположим, что в графе $\Gamma \in \mathcal{G}$ мощность некоторого класса M^Γ_u больше 1. Рассмотрим граф $G = T^{-1}(\Gamma)$ (как показано ранее, граф G однозначно с точностью до изоморфизма восстанавливается из графа $T(G)$). Из сделанного предположения следует, что в графе $\Gamma = T(G)$ найдутся две различные вершины α и β, принадлежащие одному классу $M^\Gamma_\alpha = M^\Gamma_\beta$. Пусть вершины $a, b \in E_G$ таковы, что $t(M_a) = \alpha$, $t(M_b) = \beta$. Согласно лемме 1, если бы вершины a и b принадлежали разным классам эквивалентности в графе G, то и вершины α и β также принадлежали бы разным классам эквивалентности в графе Γ. Следовательно, вершины a и b принадлежат одному классу $M_a = M_b$. Значит, по построению графа $T(G) = \Gamma$ вершины $\alpha = t(M_a)$ и $\beta = t(M_b)$ совпадают. Приходим к противоречию с тем, что вершины α и β различны. Значит, мощность любого класса M^Γ не превосходит 1.

Достаточность. Пусть в графе $\Gamma \in \mathcal{G}$ мощность каждого класса эквивалентности M^Γ_u равна 1 и в графе Γ содержится ровно s вершин. Построим граф $G = T^{-1}(\Gamma)$ по

тому алгоритму, который предложен в начале п. 2.2. Получим s полных подграфов, порожденных множествами вершин $M_1, \ldots, M_s$, где каждое множество M_i соответствует вершине $v_i \in G_\Gamma$. При этом вершины из разных множеств M_i и M_j смежны в том и только в том случае, если смежны вершины v_i и v_j в графе Γ, т. е. если $u \notin M_i$, то либо все вершины множества M_i смежны с u, либо все они не смежны с u. Таким образом, для двух вершин $v, w \in M_i$ выполнено равенство $H(v) = H(w)$.

Рассмотрим теперь две вершины v, w из разных классов: $v \in M_i$, $w \in M_j$, $i \neq j$. Предположим, что $H(v) = H(w)$. Все вершины M_i и M_j смежны со всеми вершинами одних и тех же множеств $M_{k_1}, \ldots, M_{k_m}$, среди которые есть и M_i, и M_j. Это значит, что в графе Γ соответствующие данным множествам вершины соединены рёбрами с одними и теми же вершинами и между собой, то есть их невыколотые окружения совпадают. Согласно утверждению 6, они принадлежат одному классу эквивалентности $M^\Gamma_{u_i}$, что противоречит условию. Значит, $H(v) \neq H(w)$.

Итак, $H(v) = H(w)$ тогда и только тогда, когда v и w принадлежат одному множеству M_i. Согласно утверждению 6, множества M_i есть классы эквивалентности вершин. По построению графа G граф $T(G)$ есть Γ, откуда получаем, что $\Gamma \in \mathcal{G}$. ■

Таким образом, в любом графе $\Gamma \in \mathcal{G}$ каждая вершина составляет отдельный класс эквивалентности и все такие графы входят в $\mathcal{G}$.

Следствие 2. Пусть граф Γ входит в $\mathcal{G}$. Тогда граф $T(\Gamma)$ совпадает с графом Γ.

Доказательство. В графе Γ каждая вершина α составляет отдельный класс M_α, сумма весов вершин в котором равна весу α. Две вершины α и β графа Γ, соединённые ребром, составляют два соседних класса вершин. Таким образом, при построении графа $T(\Gamma)$ каждая вершина этого графа переводится в отдельную вершину графа Γ, имеющую такой же вес, а каждое ребро — в ребро между соответствующими вершинами графа Γ. ■

Из доказанного выше, в частности, следует, что множество $\mathcal{G}$ является замкнутым относительно операции построения графа на классах вершин. Далее граф, входящий в множество $\mathcal{G}$, будем называть сжатым графом. Докажем теорему о связи между изоморфизмом двух графов и изоморфизмом соответствующих им сжатых графов.

Теорема 2. Графы G_1 и G_2 изоморфны в том и только в том случае, если взвешенные графы $T_1 = T(G_1)$ и $T_2 = T(G_2)$ изоморфны и веса их соответствующих вершин совпадают.

Доказательство. Обозначим вершины $t(M^{G_1}_a)$ и $t(M^{G_1}_b)$ как α и β соответственно, будем считать, что $M_a \neq M_b$.

Необходимость. Пусть графы G_1 и G_2 изоморфны, а $f : V_{G_1} \to V_{G_2}$ — соответствующая функция изоморфизма. Это значит, что функция f является биекцией и $(u, v) \in E_{G_1}$ тогда и только тогда, когда $(f(u), f(v)) \in E_{G_2}$. Рассмотрим вершину a графа G_1 и вершину $f(a)$ графа G_2. Пусть $H(a) = \{a, h_1, h_2, \ldots, h_k\}$ — невыколотое окружение вершины a. В силу того, что функция f смежные вершины переводит в смежные, а несмежные — в несмежные, $H(f(a)) = \{b, f(h_1), f(h_2), \ldots, f(h_k)\}$. Поэтому для двух вершин a и b графа G_1 выполнено $H(a) = H(b)$ тогда и только тогда, когда для вершин $f(a)$ и $f(b)$ графа G_2 выполнено $H(f(a)) = H(f(b))$.

Обозначим множество вершин графа G_2, в которые переводится множество M_a функцией f, как $F(M_a)$. Из того, что равенство $H(a) = H(b)$ равносильно $H(f(a)) = H(f(b))$, и из утверждения 6 следует, что множества $F(M_a)$ и $F(M_b)$ являются классами эквивалентности вершин. Поскольку $f(a) \in F(M_a)$ и $f(b) \in F(M_b)$, выполнены

следующие равенства: $F(M_a) = M_{f(a)}$, $F(M_b) = M_{f(b)}$. Значит, каждый класс эквивалентности в графе G_1 функцией f поэлементно переводится в класс эквивалентности в графе G_2.

Определим функцию $g : V_{T_1} \to V_{T_2}$ следующим образом: $g(\alpha) = t(M^{G_2}_{f(a)})$. Так как f — функция изоморфизма графов G_1 и G_2, $|M^{G_1}_a| = |M^{G_2}_{f(a)}|$. Следовательно, вес каждой вершины α в графе T_1 совпадает с весом вершины $g(\alpha)$ в графе T_2.

Таким образом, функция g переводит каждую вершину графа T_1 в вершину графа T_2, при этом все вершины графа T_2 покрыты, так как в T_2 каждая вершина соответствует классу эквивалентности графа G_2, порождённому функцией f из некоторого класса эквивалентности в графе G_1.

Осталось показать, что вершины α и β соединены ребром в графе T_1 в том и только в том случае, если вершины $g(\alpha)$ и $g(\beta)$ соединены ребром в графе T_2.

С л у ч а й 1. Пусть вершины α и β соединены ребром в графе T_1. Это значит, что соответствующие классы вершин M_a и M_b являются соседними в графе G_1. Так как функция f смежные вершины переводит в смежные, вершины $f(a) \in M_{f(a)}$ и $f(b) \in M_{f(b)}$ соединены ребром в графе G_2 и, следовательно, все вершины класса $M_{f(a)}$ соединены со всеми вершинами класса $M_{f(b)}$, т. е. эти классы соседние. Значит, $g(\alpha) = t(M^{G_2}_{f(a)})$ и $g(\beta) = t(M^{G_2}_{f(b)})$ соединены ребром в графе T_2.

С л у ч а й 2. Пусть вершины α и β не соединены ребром в графе T_1, т. е. классы вершин M_a и M_b не являются соседними в графе G_1. Так как функция f несмежные вершины переводит в несмежные, вершины $f(a) \in M_{f(a)}$ и $f(b) \in M_{f(b)}$ не соединены ребром в графе G_2 и, следовательно, ни одна вершина класса $M_{f(a)}$ не соединена ни с одной из вершин класса $M_{f(b)}$, т. е. эти классы не являются соседними. Значит, $g(\alpha) = t(M^{G_2}_{f(a)})$ и $g(\beta) = t(M^{G_2}_{f(b)})$ не соединены ребром в графе T_2.

Таким образом, g — функция изоморфизма для графов T_1 и T_2.

Достаточность. Пусть графы T_1 и T_2 изоморфны, $g : V_{T_1} \to V_{T_2}$ — функция изоморфизма и веса соответствующих вершин совпадают.

Рассмотрим класс эквивалентности $t^{-1}(g(\alpha))$ графа G_2. В графе T_2 ему соответствует вершина $g(\alpha)$ с весом, равным мощности рассматриваемого класса. Поскольку вес вершины $g(\alpha)$ равен весу вершины α, мощность класса $t^{-1}(g(\alpha))$ равна мощности класса $M^{G_1}_a$.

Определим функцию $f : V_{G_1} \to V_{G_2}$ следующим образом. Каждой вершине a графа G_1 ставится в соответствие одна вершина $f(a)$ графа G_2, принадлежащая классу эквивалентности $t^{-1}(g(\alpha))$. При этом если две вершины a и b графа G_1 принадлежат одному классу эквивалентности $M_a = M_b$, то $f(a) \neq f(b)$. Поставленное требование выполнимо в силу того, что классы эквивалентности M_a и $t^{-1}(g(\alpha))$ равномощны. Других условий на функцию не налагается. Покажем, что f является функцией изоморфизма.

Во-первых, данная функция каждой вершине графа G_1 ставит в соответствие одну вершину графа G_2. Во-вторых, для каждой вершины графа G_2 найдётся её вершина-прообраз в графе G_1, так как функция f каждому элементу из множества M_a ставит в соответствие уникальный элемент из $t^{-1}(g(\alpha))$ и такими классами покрываются все вершины графа.

Наконец, покажем, что вершины a и b графа G_1 смежны в том и только в том случае, если смежны вершины $f(a)$ и $f(b)$ графа G_2.

С л у ч а й 1. Пусть вершины a и b графа G_1 соединены ребром. В этом случае возможны два варианта. Первый вариант: $M_a = M_b$, в этом случае $t(M_a)$ и $t(M_b)$ — одна вершина в графе T_1, которая функцией g переводится в вершину $g(t(M_a))$ графа T_2, вес

которой совпадает с весом вершины $t(M_a)$ и с мощностью класса эквивалентности M_a. Согласно определению функции f, вершины a и b переводятся в различные вершины $f(a)$ и $f(b)$, принадлежащие одному классу эквивалентности. Значит, они соединены ребром. Второй вариант: $M_a \neq M_b$, вершины $t(M_a)$ и $t(M_b)$ различны и соединены ребром в графе T_1. В этом случае вершины $g(t(M_a))$ и $g(t(M_b))$ различны и соединены ребром в графе T_2; значит, классы эквивалентности $t^{-1}(g(\alpha))$ и $t^{-1}(g(\beta))$ — соседние в графе G_2. Отсюда следует, что все элементы $t^{-1}(g(\alpha))$ соединены со всеми элементами $t^{-1}(g(\beta))$, т. е. вершина $f(a)$ соединена с $f(b)$.

С л у ч а й 2. Пусть вершины a и b графа G_1 не соединены ребром. Тогда множества M_a и M_b не являются соседними, и вершины $t(M_a)$ и $t(M_b)$ не соединены ребром в графе T_1, а значит, и вершины $g(t(M_a))$ и $g(t(M_b))$ не соединены ребром в графе T_2 в силу того, что g – функция изоморфизма. Согласно описанию функции f, отсюда следует, что $f(a)$ и $f(b)$ не соединены ребром в графе G_2.

Таким образом, f — функция изоморфизма для графов G_1 и G_2. ■

Доказанная теорема говорит о том, что при изучении свойств кликовых покрытий рёбер можно ограничиться рассмотрением графов, входящих в класс $\mathcal{G}$.

3. Графы с единственным тупиковым КПР

Графы, имеющие единственное тупиковое покрытие, интересны с той точки зрения, что для них процедура поиска минимального покрытия очень проста: необходимо всего лишь породить какое-нибудь тупиковое покрытие, оно и будет единственным и минимальным. Далее приведён алгоритм порождения всех графов, обладающих единственным тупиковым покрытием.

Теорема 3. Клика K присутствует в любом покрытии графа G тогда и только тогда, когда она является зафиксированной.

Доказательство. Пусть клика K присутствует в любом покрытии графа. Рассмотрим некоторое покрытие P графа G. Предположим, что клика K не является зафиксированной, т. е. все рёбра клики K не являются собственными. Но тогда можно в P заменить K на множество клик, каждая из которых, не совпадая с K, покрывает своё ребро клики K. Получили противоречие — построенное покрытие не содержит K.

В обратную сторону утверждение очевидно. ■

Следствие 3. Граф $G = (V, E)$ обладает единственным тупиковым КПР тогда и только тогда, когда все его рёбра покрываются зафиксированными кликами.

Полученный критерий может быть сформулирован в несколько иной форме.

Теорема 4. Покрытие $\{K_1, \ldots, K_s\}$ графа G является единственным тупиковым покрытием этого графа тогда и только тогда, когда для любой клики K_i найдутся такие вершины a и b графа G, что $U_a \cap U_b = \{i\}$.

Доказательство. Согласно следствию из теоремы 3, единственность покрытия графа равносильна тому, что покрытие $\{K_1, \ldots, K_s\}$ содержит только зафиксированные клики. Рассмотрим клику K_i, $i = 1, \ldots, s$. В ней существуют две вершины a и b, не принадлежащие одновременно ни одной клике, кроме K_i, что равносильно условию $U_a \cap U_b = \{i\}$. ■

Полученное утверждение поможет построить алгоритм порождения всех графов, имеющих единственное покрытие кликами.

Утверждение 9. Граф G имеет единственное тупиковое покрытие тогда и только тогда, когда граф $T(G)$ имеет единственное тупиковое покрытие.

Доказательство. Необходимость. Пусть граф G обладает единственным тупиковым покрытием $\{K_1^{(G)}, \ldots, K_s^{(G)}\}$. Очевидно, вершины, содержащиеся в одном классе эквивалентности графа G, целиком принадлежат любой клике, которая содержит хотя бы одну из этих вершин.

Рассмотрим множество $\{K_1^{(T(G))}, \ldots, K_s^{(T(G))}\}$ полных подграфов графа $T(G)$, где каждый полный подграф $K_i^{(T(G))}$ получается из клики $K_i^{(G)}$ заменой в ней каждого набора эквивалентных вершин одной вершиной, соответствующей в графе $T(G)$ данному классу эквивалентности.

Каждая клика $K_i^{(G)}$ из покрытия графа G является зафиксированной, т. е. содержит собственное ребро. Каждый полный подграф $K_i^{(T(G))}$ является кликой, так как если бы была возможность достраивания его до клики, то клику $K_i^{(G)}$ также можно было бы достроить до большей клики, что противоречит определению клики. Для рассматриваемой клики $K_i^{(G)}$ возможны два варианта.

1. Каждое собственное ребро клики $K_i^{(G)}$ соединяет вершины из одного класса эквивалентности. В этом случае данная клика является отдельной компонентой связности графа G, потому что иначе существует другой класс эквивалентности $M_U^G \subseteq V_{K_i^{(G)}}$, такой, что $i \in U$ и $\{i\} \neq U$, и в этом случае $U \cap \{i\} = \{i\}$, то есть любое ребро (w, u), где $w \in U$, $u \in \{i\}$, является собственным, что противоречит рассматриваемому условию. Значит, $K_i^{(G)}$ — отдельная компонента связности. В этом случае клика $K_i^{(T(G))}$ — изолированная вершина в графе $T(G)$, таким образом, она входит в любое покрытие.

2. В клике $K_i^{(G)}$ существует собственное ребро (u, v), соединяющее вершины из разных классов эквивалентности, т. е. $M_u^{(G)} \neq M_v^{(G)}$. Рассмотрим в графе $T(G)$ ребро $(t(M_u), t(M_v))$. Оно является собственным для клики $K_i^{(T(G))}$, так как если существует другая клика, содержащая данное ребро, то соответствующее множество вершин графа G, очевидно, является кликой, отличной от $K_i^{(G)}$, и содержит ребро (u, v). Таким образом, клика $K_i^{(T(G))}$ входит в любое покрытие графа $T(G)$.

Наконец, осталось заметить, что любое ребро графа $T(G)$ покрыто одной из клик набора $\{K_1^{(T(G))}, \ldots, K_s^{(T(G))}\}$ по построению. Значит, данный набор является тупиковым покрытием графа $T(G)$, а из сказанного ранее следует, что данное тупиковое покрытие единственно.

Достаточность. Пусть граф $T(G)$ обладает единственным тупиковым покрытием. Это значит, что все клики данного покрытия являются зафиксированными (либо содержат единственную изолированную вершину).

Пусть $\{K_1^{(T(G))}, \ldots, K_s^{(T(G))}\}$ — единственное покрытие графа $T(G)$. Рассмотрим множество клик $\{K_1^{(G)}, \ldots, K_s^{(G)}\}$, где каждую клику $K_i^{(G)}$ получим из клики $K_i^{(T(G))}$, заменив в ней каждую вершину на все вершины из соответствующего класса эквивалентности графа G. Как нетрудно убедиться, полученные полные подграфы будут кликами.

Очевидно, данное множество покрывает все рёбра графа G, так как любое ребро либо принадлежит классу эквивалентности, либо соединяет между собой вершины из соседних классов. Все рёбра первого типа покрыты благодаря тому, что все вершины одного класса эквивалентности принадлежат одной клике. Все рёбра второго типа покрыты по построению графа $T^{-1}(G)$.

Рассмотрим клику $K_i^{(T(G))}$. Если она состоит из одной вершины, то клика $K_i^{(G)}$ является отдельной компонентой связности графа G и поэтому входит в любое его покрытие. Если она состоит из более чем одной вершины, то она содержит собствен-

ное ребро $(t(M_u), t(M_v))$ (при этом классы $M_u^{(G)}$ и $M_v^{(G)}$ не пересекаются). Рассмотрим вершины $u \in M_u^{(G)}$ и $v \in M_v^{(G)}$ графа G. Они смежны, так как принадлежат соседним классам. Ребро (u, v) является собственным ребром клики $K_i^{(G)}$, так как если бы существовала другая клика $\overline{K_i^{(G)}} \neq K_i^{(G)}$, содержащая это ребро, то существовала бы соответствующая клика $\overline{K_i^{(T(G))}} \neq K_i^{(T(G))}$ графа $T(G)$, содержащая ребро $(t(M_u), t(M_v))$.

Таким образом, любая клика данного тупикового покрытия входит в любое покрытие графа G и, следовательно, данное покрытие единственно. ■

Породить все графы на n вершинах, имеющие единственное тупиковое КПР, можно с помощью алгоритма, существенно более простого, чем полный перебор графов порядка n. Далее приводим описание такого алгоритма.

Алгоритм состоит из трёх стадий, применяемых для всех возможных количеств клик в покрытии.

На первой стадии для фиксированного числа s клик в покрытии работает рекурсивная процедура $P(s, i)$. На i-м уровне рекурсии в процедуре перебираются функции $f_i : I \setminus \{i\} \to \{0, 1, 2\}$, для каждой из которых вызывается копия процедуры $P(s, i+1)$. Глубина рекурсии равна s, вектора значений всех функций f_i хранятся в глобальной памяти и доступны с любого уровня рекурсии. Когда сформирован набор векторов функций f_i для всех $i = 1, \ldots, s$, он сохраняется в памяти для дальнейшего использования.

Алгоритм 1. Процедура $P(s, i)$

1: Выбрать функцию f_i с нулевым вектором значений на области её определения: $(0, \ldots, 0)$.
2: **Пока** вектор значений функции f_i не равен $(2, \ldots, 2)$, выполнять:
3: **Если** $i < s$, **то**
4: вызвать процедуру $P(s, i + 1)$,
5: **иначе**
6: сохранить текущий набор векторов функций f_i для $i = 1, \ldots, s$ в памяти.
7: Изменить функцию f_i, в качестве вектора значений выбрав следующий в лексикографическом порядке вектор.
8: **Выход** из процедуры.

На второй стадии из каждого набора функций, порождённого в процедуре $P(s, 1)$, строится некоторый взвешенный граф, вес каждой вершины в котором равен 1. При этом рассматриваются множества вершин вида L_V, каждое из которых определяется множеством $V \subseteq I$. Каждое такое множество изначально пустое, однако в процессе работы алгоритма в некоторые из них добавляются вершины.

На третьей стадии строятся все возможные варианты весовой функции данного графа, дающие в сумме значений n, и по каждому из них строится граф, обладающий единственным тупиковым КПР.

Вопрос о корректности алгоритма 2 возникает в связи с построением графа $T^{-1}(\Gamma)$. Для графа Γ множества L_V являются классами эквивалентности вершин и каждый из них содержит ровно одну вершину. Значит, согласно теореме 1, взвешенный граф Γ (независимо от весовой функции) принадлежит классу $\mathcal{G}$. Поэтому граф $T^{-1}(\Gamma)$ может быть построен.

Утверждение 10. Алгоритм 2 порождает с точностью до изоморфизма все графы на n вершинах, обладающие единственным тупиковым КПР, и только их.

Алгоритм 2. Построение графов, обладающих единственным тупиковым КПР

1: **Для всех** s от 1 до n^2
2: вызвать процедуру $P(s, 1)$;
3: каждый набор векторов значений функций преобразовать во взвешенный граф следующим образом:
4: **Для всех** функций f_i
5: выбрать из множеств вида L_V два множества L_{V_1} и L_{V_2}, для которых $V_1 = \{k \in I : f_i(k) = 1\}$ и $V_2 = \{k \in I : f_i(k) = 2\}$;
6: если множество L_{V_1} пустое, то добавить к нему вершину;
7: если множество L_{V_2} пустое, то добавить к нему вершину;
8: если общее число добавленных вершин больше чем n, то данный набор векторов не порождает граф — перейти к п. 3 и рассмотреть следующий набор.
9: Построить взвешенный граф Γ следующим образом:
каждому непустому множеству L_V поставить в соответствие вершину v;
две вершины v_1 и v_2 соединить ребром в том и только в том случае, если соответствующие множества V_1 и V_2 пересекаются;
каждой вершине полученного графа поставить в соответствие вес 1.
10: Преобразовать полученный взвешенный граф размера k в множество графов на $n \geqslant k$ вершинах следующим образом:
11: **Для всех** размещений из $n - k$ по k
в соответствии с размещением добавить к весу каждой из k вершин графа Γ соответствующее число;
построить граф $G = T^{-1}(\Gamma)$ и выдать пользователю.
12: **Выход** из алгоритма.

Доказательство. Во-первых, покажем, что любой граф, порожденный алгоритмом 2, обладает единственным покрытием. В графе Γ для каждой i-й клики непустыми оказываются некоторые множества $L_{V_1^{(i)}}$ и $L_{V_2^{(i)}}$ (по построению графа являющиеся его классами эквивалентности), такие, что $V_1^{(i)} \cap V_2^{(i)} = \{i\}$. Согласно теореме 4, из этого следует, что граф Γ обладает единственным покрытием. При увеличении веса некоторых вершин графа Γ множество его покрытий остаётся неизменным. Из утверждения 9 следует, что граф G также обладает единственным покрытием.

Во-вторых, покажем, что любой граф, обладающий единственным покрытием, порождается алгоритмом 2 с точностью до изоморфизма. Пусть граф G обладает единственным покрытием. Тогда, согласно утверждению 9, граф $\Gamma = T(G)$ также обладает единственным покрытием. Так как множество покрытий не зависит от весовой функции, граф Γ, у которого каждой вершине приписан вес 1, обладает единственным покрытием. Согласно теореме 4, из этого следует, что для любой клики K_i, входящей в покрытие, найдутся две вершины u и v, соединённые ребром, для которых $U_u \cap U_v = \{i\}$. Таким образом, множество $I \setminus \{i\}$ разбивается на три подмножества: $U_u \setminus \{i\}$, $U_v \setminus \{i\}$ и $I \setminus (\{i\} \cup U_u \cup U_v)$. Любое такое деление однозначно задается функцией $f_i : I \setminus \{i\} \to \{0, 1, 2\}$. Поскольку в процессе работы алгоритма перебираются все такие функции (и все возможные количества клик в покрытии), можно сделать вывод, что рассматриваются все возможные графы $\Gamma \in \mathcal{G}$. При переходе от наборов векторов к графу Γ отбрасываются только те наборы, которые не порождают графа на n

вершинах. Значит, любой граф, обладающий единственным КПР, строится данным алгоритмом. ■

ЛИТЕРАТУРА

1. *Kou L. T., Stockmeyer L. J., and Wong C. K.* Cliques with regard to keyword conflicts and intersection graphs // Comm. ACM. 1978. V. 21. No. 2. P. 135–139.
2. *Orlin J.* Contentment in graph theory: covering graphs with cliques // Indagat. Math. 1977. No. 39. P. 406–424.
3. *Gramm J., Guo J., Huffner F., and Niedermeier R.* Data reduction, exact, and heuristic algorithms for clique cover // Proc. 8th Workshop on Algorithm Engineering and Experiments, Miami, Fl. January 21, 2006. P. 86–94.
4. *Cavers M. S.* Clique partitions and coverings of graphs. Masters thesis, University of Waterloo, 2005. `http://www.math.uwaterloo.ca/co/graduate-students/files/mmath/Mike-Cavers.pdf`
5. *Харари Ф.* Теория графов М.: Мир, 1973. 301 с.

2013 Прикладная теория графов №1(19)

DOI 10.17223/20710410/19/7

УДК 519.178

АЛГОРИТМ ПОИСКА КРАТЧАЙШИХ ПУТЕЙ ДЛЯ РАЗРЕЖЕННЫХ ГРАФОВ БОЛЬШОЙ РАЗМЕРНОСТИ

А. Р. Ураков, Т. В. Тимеряев

Уфимский государственный авиационный технический университет, г. Уфа, Россия

E-mail: urakov@ufanet.ru, timeryaev@yandex.ru

Рассматривается задача поиска кратчайших путей между всеми вершинами взвешенного ненаправленного разреженного графа. Предлагается алгоритм «разборки и сборки графа», использующий решение задачи на графе малой размерности для получения решения для исходного графа. Приводится сравнение предлагаемого алгоритма с одним из быстрых классических алгоритмов на данных из открытого доступа.

Ключевые слова: *задача о кратчайших путях, APSP, разборка графа, сборка графа, сжатие графа, разреженные графы.*

Введение

Задача поиска кратчайших путей между всеми парами вершин графа (APSP — All-Pairs Shortest Path problem) является одной их классических в теории графов. Такой её статус обусловлен тем, что при решении оптимизационных задач на графах из многих областей (задача коммивояжёра, транспортная задача и многие другие) необходимо знать расстояния между всеми вершинами графа. Более того, сама задача представляет большой исследовательский интерес, так как её решение позволяет ответить, например, на такие вопросы:

— Как попасть из любого пункта A в любой пункт $Б$ с наименьшими затратами времени/денежных средств?
— Каков ближайший/самый удалённый пункт для A и каково расстояние до него?

Новый приток интереса к задаче произошёл с появлением создаваемых полуавтоматически графов высокой подробности, описывающих структуры реального мира. Подобные графы имеют размерность 10^6 и более, а их подробность со временем будет только увеличиваться. Поэтому актуальным становится вопрос об ускорении поиска кратчайших путей в графах, обладающих большой размерностью.

Для решения задачи APSP существует множество алгоритмов, но нет способа, получающего решения одинаково быстро для любого вида входов. В связи с этим алгоритмы решения APSP принято делить по типу графов, на которых задача решается. Известны алгоритмы, в которых входами являются направленные графы [1], полные графы [2], взвешенные графы [3], невзвешенные графы [4], разреженные графы [5].

В данной работе рассматривается задача APSP для взвешенных ненаправленных разреженных графов большой размерности с неотрицательными весами ребер. Предлагается алгоритм, сводящий решение задачи к замене исходного графа графом меньшей размерности и решению задачи на нём. Приводится сравнение предлагаемого алгоритма с быстрым классическим алгоритмом на графах дорожных сетей.

1. Постановка задачи и определения

1.1. Термины и определения

Рассматривается задача поиска кратчайших путей на связном ненаправленном разреженном взвешенном графе $G = (V, E, w)$ с неотрицательными весами рёбер, где $V = \{v_1, v_2, \ldots, v_n\}$ — множество вершин графа; $E = \{e_1, e_2, \ldots, e_m\} \subseteq V \times V$ — множество рёбер графа; $w : E \to [0, \infty)$ — весовая функция на рёбрах. Ребро между вершинами v_i и v_j будем обозначать $e(v_i, v_j)$. Будем рассматривать простые графы — без петель и кратных рёбер.

Граф называется *взвешенным* (относительно рёбер), если каждому его ребру поставлено в соответствие число — вес ребра. Вес ребра между вершинами v_i и v_j обозначается $w(i, j)$, для вершин v_i и v_j, не связанных ребром, полагается $w(i, j) = \infty$. *Степень* $d(v_i)$ *вершины* v_i — количество рёбер графа, инцидентных v_i. Граф *разреженный*, если справедливо $m \ll n^2$.

Путь в графе — чередующаяся последовательность вершин и рёбер v_0, e_1, v_1, …, v_{k-1}, e_k, v_k, каждое ребро которой инцидентно двум вершинам — непосредственно предшествующей ему и непосредственно следующей за ним. *Длина пути* — сумма весов входящих в него рёбер. *Кратчайший путь* $p_{ij}^s = p^s(v_i, v_j)$ между вершинами v_i и v_j — путь между v_i и v_j минимальной длины. *Расстояние* $m(i, j)$ между v_i и v_j — длина кратчайшего пути между v_i и v_j ($m(i, i) = 0$, $i = 1, \ldots, n$). *Матрица расстояний* графа — матрица $M_{n\times n}$ с элементами $m_{ij} = m(i, j)$. Граф *связный*, если между любыми двумя его вершинами существует путь, то есть $m_{ij} < \infty$ для всех i, j.

Между двумя вершинами графа может существовать несколько кратчайших путей. Этот аспект в данной работе не является существенным, поэтому при всех упоминаниях кратчайшего пути будем подразумевать любой кратчайший путь. *Матрица предшествования* — матрица $P_{n\times n}$, элемент p_{ij} которой представляет собой метку вершины, которая предшествует вершине v_j в кратчайшем пути от v_i до v_j. То есть элементы матрицы P определяются по формуле

$$p_{ij} = \begin{cases} v_k, & \exists v_k (p_{ij}^s = \ldots v_k, e(v_k, v_j), v_j), \\ \infty & \text{иначе.} \end{cases}$$

С использованием матрицы P кратчайший путь p_{ij}^s для вершин $v_i \neq v_j$ в связном графе может быть определён по рекурсивной формуле

$$p_{ij}^s = \begin{cases} p^s(v_i, p_{ij}), e(p_{ij}, v_j), v_j, & p_{ij} \neq v_i, \\ v_i, e(v_i, v_j), v_j, & p_{ij} = v_i. \end{cases}$$

Обозначим через S и будем называть *сжимающей граф* $G_0 = (V_0, E_0, w_0)$ последовательность графов $S = G_1, G_2, \ldots, G_r$, где $G_p = (V_p, E_p, w_p)$; $V_p = \left\{v_1^p, v_2^p, \ldots, v_{n(p)}^p\right\}$; $E_p = \left\{e_1^p, e_2^p, \ldots, e_{m(p)}^p\right\}$. Каждый следующий граф G_{p+1} последовательности получается из предыдущего G_p удалением k вершин и инцидентных им рёбер, добавлением новых рёбер и пересчётом весов рёбер между вершинами, смежными с удалёнными. Для таких графов справедливо $|V_p| > |V_{p+1}|$, $p = 0, \ldots, r-1$. Будем обозначать v_i^{p+1} вершину графа G_{p+1}, соответствующую вершине v_i^p графа G_p, и $e^{p+1}\left(v_j^{p+1}, v_k^{p+1}\right)$ — ребро графа G_{p+1}, соответствующее ребру $e^p\left(v_j^p, v_k^p\right)$ графа G_p. Граф, полученный удалением вершин $v_1^p, v_2^p, \ldots, v_k^p$ и инцидентных им рёбер из графа G_p, будем обозначать $G_{p+1} = R_p(v_1^p, v_2^p, \ldots, v_k^p)$. Для такого графа справедливо: $w_{p+1}(i, j) = w_p(i, j)$ для

всех i, j, таких, что $v_i^{p+1}, v_j^{p+1} \in V_{p+1}$. Расстояние между вершинами v_i^p и v_j^p графа G_p будем обозначать $m^p(i,j)$, а его матрицу расстояний — $M_p = \left(m_{ij}^p\right)$. Наконец, обозначим A_i^p множество всех смежных с v_i^p вершин в графе G_p.

1.2. Постановка задачи

Задача. Дан связный ненаправленный разреженный простой взвешенный граф $G = (V, E, w)$ с неотрицательной вещественной весовой функцией на ребрах $w : E \to [0, \infty)$. Необходимо для всех пар вершин графа найти кратчайший путь между вершинами в паре, т. е. найти матрицу расстояний M и матрицу предшествования P.

2. Алгоритм поиска решения

2.1. Общая идея

Принцип предлагаемого алгоритма состоит в сведении задачи на исходном графе большой размерности к задаче на графе малой размерности. Алгоритм можно разделить на три этапа:

1) сжатие — замена исходного графа графом меньшей размерности;
2) микрорешение — решение задачи о кратчайших путях на графе меньшей размерности с использованием существующих методов;
3) восстановление — перенос решения с графа меньшей размерности на исходный граф; пересчёт кратчайших расстояний для части вершин исходного графа.

Такой подход требует выполнения следующих условий: а) достоверности — сжатие должно сохранять информацию о кратчайших путях исходного графа; б) скорости — выполнение всех этапов в сумме должно требовать меньше времени, чем решение задачи известными способами на исходном графе.

Алгоритм, использующий похожие идеи, опубликован, например, в [6].

Для разреженных графов предлагается алгоритм «разборки и сборки графа». На этапе разборки из графа последовательно удаляются вершины, далее производится поиск решения на малом графе, после чего исходный граф собирается с получением искомых кратчайших расстояний. Отличие предлагаемого алгоритма от уже известных алгоритмов со сжатием или подменой рёбер состоит в том, что он достаточно прост, выполняет расчёт кратчайших путей сразу между всеми вершинами, при этом гарантированно находит точное решение (не является эвристическим). В частности, в [6] решается задача поиска кратчайшего пути от источника до пункта назначения (point-to-point), одной из основных целей которой является уменьшение времени запроса, тогда как время решения задачи APSP для графов большой размерности таким алгоритмом может быть весьма велико.

2.2. Разборка

Этап разборки графа является многоступенчатым и состоит в последовательном приближении исходного графа $G_0 = (V_0, E_0, w_0)$ графами сжимающей G_0 последовательности $S = \{G_1, G_2, \ldots, G_r\}$. Будем рассматривать частный случай, когда каждый следующий граф G_{p+1} последовательности S получается из предыдущего графа G_p удалением одной вершины.

Пусть удаляемая вершина v_i^p графа G_p имеет степень k, т. е. с ней смежны вершины $v_{i_z}^p$, $z = 1, \ldots, k$. Если какой-то из кратчайших путей графа проходит через вершину v_i^p (не считая путей от и до самой v_i^p), то этот путь обязательно содержит подпуть вида $v_{i_j}^p, e^p(v_{i_j}^p, v_i^p), v_i^p, e^p(v_i^p, v_{i_l}^p), v_{i_l}^p$, $j, l \in \{1, 2, \ldots, k\}$. Таким образом, при уда-

лении вершины v_i^p необходимо гарантировать сохранение кратчайших путей только между вершинами, смежными с v_i^p.

Обозначим $w_p^{\mathrm{mv}(1,2,\ldots,h)}(i_j, i_l) = \min\limits_{g=1,2,\ldots,h}(w_p(i_j, g) + w_p(g, i_l))$ минимальную сумму весов двух рёбер, соединяющих вершины $v_{i_j}^p$ и $v_{i_l}^p$ и проходящих через одну из вершин $v_1^p, v_2^p, \ldots, v_h^p$ графа G_p. Для сохранения расстояний достаточно для всех пар вершин $\left(v_{i_j}^p, v_{i_l}^p\right)$, смежных с v_i^p в следующем графе G_{p+1} последовательности, положить

$$w_{p+1}(i_j, i_l) = \begin{cases} \min\left(w_p^{\mathrm{mv}(i)}(i_j, i_l), w_p(i_j, i_l)\right), & \text{если } w_p^{\mathrm{mv}(i)}(i_j, i_l) < w_p^{\mathrm{mv}(h\neq i)}(i_j, i_l), \\ w_p(i_j, i_l) & \text{иначе.} \end{cases} \quad (1)$$

В начале алгоритма вводится матрица $P'_{n\times n}$ с элементами $p'_{ij} = \infty$. Для сохранения информации о путях элементы матрицы P', для соответствующих номеров которых справедливо $w_p^{\mathrm{mv}(i)}(i_j, i_l) < \min\left(w_p(i_j, i_l), w_p^{\mathrm{mv}(h\neq i)}(i_j, i_l)\right)$, изменяются по формуле

$$p'_{i_j i_l} = \begin{cases} v_i, & \text{если } p'_{ii_l} = \infty, \\ p'_{ii_l}, & p'_{ii_l} \neq \infty. \end{cases} \quad (2)$$

В частности, если удаляемая вершина v_i^p смежна только с одной вершиной графа G_p, то сохранять кратчайшие пути надобности нет (так как через v_i^p они не проходят), и в этом случае вершина v_i^p и инцидентное ей ребро просто удаляются из графа.

В качестве параметров на этапе разборки выступают $d_{\max}$ — максимальная степень удаляемых вершин, $n_{\min}$ — число вершин в самом малом графе G_r и $I_{\max}$ — ограничение на рост числа рёбер при удалении вершины. Вопрос совместного определения $d_{\max}$, $n_{\min}$ и $I_{\max}$ является весьма объёмным, поэтому его рассмотрение остаётся вне рамок данной работы.

Пусть на предмет удаления рассматривается вершина v_i^p с k инцидентными ребрами. Обозначим через $I(v_i^p)$ величину изменения количества ребер при удалении вершины v_i^p. Само удаление вершины v_i^p уменьшит число ребер в графе на k, поэтому полагаем $I(v_i^p) = -k$. Пересчёт весов ребер по формуле (1) между каждой парой $\left(v_{i_j}^p, v_{i_l}^p\right)$ смежных с v_i^p вершин изменит величину $I(v_i^p)$ согласно формуле

$$I(v_i^p) = \begin{cases} I(v_i^p) + 1, & \text{если } w_p(i_j, i_l) = \infty \ \&\ w_p^{\mathrm{mv}(i)}(i_j, i_l) < w_p^{\mathrm{mv}(h\neq i)}(i_j, i_l), \\ I(v_i^p) & \text{иначе.} \end{cases} \quad (3)$$

Таким образом получим число, на которое изменится количество рёбер в графе G_{p+1} по сравнению с G_p после удаления вершины v_i^p. Если $I(v_i^p) > 0$, число рёбер увеличится, иначе — уменьшится или останется неизменным. Использование формулы (3) предполагает применение ограничения $I_{\max}$ на рост числа рёбер при удалении вершины: удаляются лишь те вершины v_i^p, для которых выполняется неравенство $I(v_i^p) \leqslant I_{\max}$.

В процессе разборки просмотр вершин — претендентов на удаление — происходит следующим образом. Так как при $d_{\max} \geqslant 2$ вершины степени меньше 3 удаляются в любом случае, то рассмотрение вершин происходит в порядке роста их степеней от 1 до $d_{\max}$. Это позволяет уменьшить степени оставшихся вершин и ускорить работу алгоритма за счёт меньшего числа просмотров вершин со степенями, близкими к $d_{\max}$. После удаления вершины v_i^p степени смежных с ней вершин могут измениться. Поэтому для того чтобы обеспечить удаление всех вершин со степенью $d\left(v_j^p\right) \leqslant d_{\max}$, после

удаления вершины v_i^p рекурсивно рассматриваются бывшие смежные с ней вершины, степень которых уменьшилась в результате удаления. После завершения рекурсивного вызова просмотр вершин продолжается в обычном порядке. Алгоритм 1 разборки графа использует вспомогательный алгоритм 2 проверки и удаления вершины.

Алгоритм 1. Разборка графа

Вход: граф $G_0 = (V_0, E_0, w_0)$, $|V_0| = n$; $d_{\max}$, $n_{\min}$, $I_{\max}$; $P'_{n\times n} = \left(p'_{ij}\right)$, $p'_{ij} = \infty$ для всех $i, j = 1, \ldots, n$.

Шаг 0. Подготовка

$d_c = 1$, $i = 0$, $n_c = n$, $p = 0$.

Шаг 1. Выбор вершины

Если $n_c = n_{\min}$, **то**
  конец алгоритма,
иначе
  Если $\exists v_j^p \in V_p$ $(j > i$ & $d\left(v_j^p\right) = d_c)$, **то**
    $i = j$, переход к шагу 2,
  иначе
    Если $d_c < d_{\max}$, **то**
      $d_c = d_c + 1$, $i = 0$ и переход к шагу 1,
    иначе
      конец алгоритма.

Шаг 2. Проверка и удаление вершины

Проверка и удаление вершины $(v_i^p, n_c, I_{\max}, d_{\max}, n_{\min}, p, P')$, переход к шагу 1.

Выход: граф $G_r = G_p$.

2.3. Микрорешение

На этом этапе решается задача APSP для графа G_r последовательности S, полученного на этапе разборки. Результатом является матрица расстояний M_r графа G_r. Вводится матрица $M'_r = M_r$ и производится изменение матрицы P' по формулам

$$p'_{ij} = \begin{cases} p^r_{ij}, & p'_{ij} = \infty \ \& \ p'_{p^r_{ij}j} = \infty, \\ p'_{p^r_{ij}j}, & p'_{ij} = \infty \ \& \ p'_{p^r_{ij}j} \neq \infty; \end{cases} \tag{4}$$

$$p'_{ij} = \begin{cases} p^r_{ij}, & w_r(i,j) > m^r_{ij} \ \& \ p'_{p^r_{ij}j} = \infty, \\ p'_{p^r_{ij}j}, & w_r(i,j) > m^r_{ij} \ \& \ p'_{p^r_{ij}j} \neq \infty, \end{cases} \tag{5}$$

где p^r_{ij} — элементы матрицы предшествования P_r графа G_r. Найденные на этом этапе пути между вершинами графа G_r будут гарантированно кратчайшими, так как удаление вершин, произведённое на этапе разборки, сохраняет расстояния. То есть справедливо $m'^r_{ij} = m^r_{ij} = m^0_{ij}$ для всех i, j, таких, что $v^r_i, v^r_j \in V_r$.

Если G_r содержит всего одну вершину $|V_r| = 1$, то данный этап алгоритма пропускается и сразу выполняется сборка графа.

2.4. Сборка

До начала работы данного этапа (алгоритма 3) определена последовательность графов $S = \{G_0, G_1, \ldots, G_r\}$. Здесь G_0 — исходный граф, G_r — наименьший граф последовательности, у которого на этапе микрорешения найдены кратчайшие пути между

Алгоритм 2. Вспомогательный алгоритм проверки и удаления вершины

Вход: вершина v_i^p, текущее количество вершин n_c, $I_{\max}$, $d_{\max}$, $n_{\min}$, p, $P^{'}$.

Шаг 1. Проверка вершины

Если $d\left(v_i^p\right) < 3$, **то**
  переход к шагу 2,
иначе
  $I\left(v_i^p\right) = -d\left(v_i^p\right)$. Рассматриваются все пары вершин множества A_i^p и изменяется значение $I\left(v_i^p\right)$ по формуле (3).
  Если $I\left(v_i^p\right) \leqslant I_{\max}$, **то**
    переход к шагу 2,
  иначе
    конец алгоритма.

Шаг 2. Удаление вершины

Формируется новый граф $G_{p+1} = R_p\left(v_i^p\right)$.
Пересчитываются веса рёбер между вершинами графа G_{p+1}, соответствующими вершинам множества A_i^p, по формуле (1).
Элементы матрицы $P^{'}$ изменяются по формуле (2), $n_c = n_c - 1$, $t = p$, $p = p + 1$.
Если $n_c = n_{min}$, **то**
  конец алгоритма,
иначе
  Пока $n_c > n_{\min}$
    для вершин $v_{i_l}^p$, таких, что $d\left(v_{i_l}^p\right) < d\left(v_{i_l}^t\right)$, выполняем
    Проверка и удаление вершины $(v_{i_l}^p, n_c, I_{\max}, d_{\min}, n_{\min}, p, P^{'})$.

всеми вершинами. На этапе сборки производится обратный ход от G_r к G_0 через графы $G_{r-1}, G_{r-2}, \ldots, G_1$ с расчётом кратчайших путей на каждом шаге p для вершин v_i^{r-p}, таких, что $v_i^{r-p+1} \notin V_{r-p+1}$ и $v_i^{r-p} \in V_{r-p}$.

Вершина v_i^{r-1}, «добавляемая» на первом шаге к графу G_r при переходе к G_{r-1}, соединена ребрами с вершинами $v_{i_z}^{r-1}$, $z = 1, \ldots, k$, графа G_{r-1}. У G_r вычислены кратчайшие пути, т. е. известна матрица $M_r^{'} = M_r$. Поэтому чтобы найти матрицу расстояний $M_{r-1}^{'}$ графа G_{r-1}, надо вычислить кратчайшие пути от вершины v_i^{r-1} до всех вершин графа G_{r-1}, положив остальные элементы матрицы $M_{r-1}^{'}$ равными соответствующим элементам $M_r^{'}$: $m_{jl}^{'r-1} = m_{jl}^{'r}$ для всех $v_j^r, v_l^r \in V_r$.

Так как кратчайший путь от любой вершины графа G_{r-1} до v_i^{r-1} проходит через одну из вершин $v_{i_z}^{r-1}$, $z = 1, \ldots, k$, кратчайшие расстояния от v_i^{r-1} до любой вершины v_l^{r-1} графа G_{r-1} можно рассчитать по формуле

$$m^{'r-1}(i,l) = \min_{z \in \{1,\ldots,k\}} \left(w_{r-1}(i, i_z) + m^{'r}(i_z, l) \right).$$

Формулы расчёта матрицы расстояний $M_{r-p}^{'}$ при переходе от графа G_{r-p+1} к графу G_{r-p} с добавлением вершины v_i^{r-p} выглядят следующим образом:

$$m_{jl}^{'r-p} = m_{jl}^{'r-p+1}, \quad v_j^{r-p+1}, v_l^{r-p+1} \in V_{r-p+1}; \tag{6}$$

$$m_{il}^{'r-p} = m_{li}^{'r-p} = \min_{z \in \{1,\ldots,k\}} \left(w_{r-p}(i, i_z) + m^{'r-p+1}(i_z, l) \right), \quad v_l^{r-p+1} \in V_{r-p+1}. \tag{7}$$

Обозначим $x(l)$ номер i_z для l, на котором достигается минимум формулы (7). Элементы матрицы $P^{'}$, для вершин которых либо $w_{r-p}(i,l) > m_{il}^{'r-p} \vee w_{r-p}(l,i) > m_{li}^{'r-p}$, либо $p_{il}^{'} = \infty \ \vee \ p_{li}^{'} = \infty$, изменяются по формулам

$$p_{il}^{'} = \begin{cases} v_i, & p_{x(l)l}^{'} = \infty, \\ p_{x(l)l}^{'}, & p_{x(l)l}^{'} \neq \infty; \end{cases} \tag{8}$$

$$p_{li}^{'} = \begin{cases} v_l, & p_{x(l)i}^{'} = \infty, \\ p_{x(l)i}^{'}, & p_{x(l)i}^{'} \neq \infty. \end{cases} \tag{9}$$

Алгоритм 3. Сборка графа

Вход: $S = \{G_0, G_1, \ldots, G_r\}$, $M_r^{'}$, $P^{'}$

1: $p = 1$.
2: **Пока** $p \leqslant r$
3: расчёт матрицы $M_{r-p}^{'}$ по формулам (6), (7) и матрицы $P^{'}$ по формулам (8) и (9); $p = p + 1$.
4: Выход: матрица $M_0^{'}$, матрица $P^{'}$.

2.5. Корректность алгоритма

Утверждение 1. Предложенные алгоритмы «сборки и разборки графа» являются корректным, т. е. найденные в результате сборки матрицы $M_0^{'}$ и $P^{'}$ являются соответственно матрицами расстояний и предшествования исходного графа G_0.

Доказательство. Сначала докажем корректность относительно матрицы $M_0^{'}$. Производимое на этапе разборки графа удаление вершин сохраняет расстояния между оставшимися вершинами, это следует из формулы (1). На втором этапе алгоритма находятся кратчайшие (относительно графа G_0) расстояния: $m_{jl}^{'r} = m_{jl}^0$ для всех $v_j^r, v_l^r \in V_r$. На этапе сборки вычисляются кратчайшие расстояния между всеми вершинами графа; для доказательства этого достаточно показать, что при переходе от графа G_r к G_{r-1} находятся кратчайшие расстояния в графе G_{r-1} (в силу формулы (6) достаточно доказать это для расстояний от «добавляемой» вершины v_i^{r-1} до всех остальных вершин графа G_{r-1}). Это справедливо в силу сохранения расстояний для графа G_{r-1} и того, что кратчайшие пути от любой вершины проходят через смежные с ней вершины, т. е. в силу формулы (7).

Теперь докажем корректность относительно матрицы $P^{'}$. Покажем, что на этапе микрорешения для всех вершин графа G_r находятся корректные элементы матрицы $P^{'}$. В силу сохранения расстояний в графе G_r кратчайший путь $p^s(v_j, v_l)$ от v_j до v_l содержит подпуть $p_{ij}^r, \ldots, v_l$. Согласно формуле (2), если $p_{p_{ij}^r j}^{'} = \infty$, то $p^s\left(p_{ij}^r, v_j\right)$ не содержит вершин, кроме p_{ij}^r и v_j, т. е. $p_{ij} = p_{ij}^r$. Если же $p_{p_{ij}^r j}^{'} \neq \infty$, то, согласно (2), $p^s\left(p_{ij}^r, v_j\right) = p_{ij}^r, \ldots, p_{p_{ij}^r j}^{'}, e(p_{p_{ij}^r j}^{'}, v_j), v_j$, т. е. $p_{ij} = p_{p_{ij}^r j}^{'}$. Расчёт по формулам (4), (5) производится для тех $p_{jl}^{'}$, $v_j^r, v_l^r \in V_r$, которые либо ещё не известны, либо определены неверно. Для $p_{jl}^{'} \neq \infty$ и $w_r(j,l) = m_{jl}^r$ в силу (2) значения уже определены верно на этапе разборки и не требуют пересчёта, т. е. $p_{jl}^{'} = p_{jl}$ для всех $v_j^r, v_l^r \in V_r$.

По индукции, для доказательства корректности матрицы $P^{'}$ достаточно показать, что при переходе от графа G_r к G_{r-1} корректно определяются элементы матрицы $P^{'}$ для «добавляемой» вершины v_i^{r-1}. Это следует из сохранения расстояний для G_{r-1},

корректности P' для $v_j^r, v_l^r \in V_r$ и того, что кратчайшие пути от любой вершины проходят через смежные с ней вершины, т. е. в силу формул (8), (9). ■

3. Результаты эксперимента

Все тесты проводились на компьютере с процессором Intel Core 2 Duo E8400 с частотой 3 ГГц и объемом памяти 2 Гбайт в 32-разрядной версии Windows XP. Программный код написан на языке C++ с использование среды разработки Borland C++ Builder 6. В качестве тестовых данных использовались взвешенные графы дорожных сетей США из открытого доступа ($G1$–$G10$) [7]. Из графов $G1$–$G10$ были получены связные подграфы размерностью от 10^3 до 10^4 в количестве 10 шт. для каждой размерности. Еще одним набором тестовых данных были графы дорожных сетей городов России (GR, подробные характеристики см. в [8]). Характеристики тестовых графов представлены в табл. 1.

Таблица 1

Характеристики тестовых графов

Группа графов	Среднее кол-во вершин	Среднее кол-во рёбер	Средняя степень вершин	Макс. степень вершин	Графов
$G1$	10^3	$2{,}5 \cdot 10^3$	2,48	6	10
$G2$	$2 \cdot 10^3$	$5{,}21 \cdot 10^3$	2,6	5	
$G3$	$3 \cdot 10^3$	$7{,}88 \cdot 10^3$	2,62	6	
$G4$	$4 \cdot 10^3$	$1{,}07 \cdot 10^4$	2,68	6	
$G5$	$5 \cdot 10^3$	$1{,}33 \cdot 10^4$	2,66	6	
$G6$	$6 \cdot 10^3$	$1{,}58 \cdot 10^4$	2,63	7	
$G7$	$7 \cdot 10^3$	$1{,}85 \cdot 10^4$	2,64	6	
$G8$	$8 \cdot 10^3$	$2{,}08 \cdot 10^4$	2,6	6	
$G9$	$9 \cdot 10^3$	$2{,}36 \cdot 10^4$	2,62	7	
$G10$	10^4	$2{,}72 \cdot 10^4$	2,72	7	
GR	$2{,}1 \cdot 10^3$	$6 \cdot 10^3$	2,86	14	20

Тестирование проводилось при $d_{\max} = I_{\max} = \infty$, $n_{\min} = 1$. То есть при разборке удаляются вершины с любой степенью до тех пор, пока в наименьшем графе G_r не останется лишь одна вершина. Соответственно второй этап алгоритма — микрорешение — не выполняется. Разработанный алгоритм (обозначение БП) сравнивался с одним из быстрейших алгоритмов для разреженных графов — алгоритмом Дейкстры в реализации с двоичной кучей (обозначение ДД, [4]), выполняемым для всех вершин тестовых графов. Результаты тестов приведены в табл. 2.

Использование предлагаемого алгоритма позволяет ускорить вычисление кратчайших путей графа в среднем в 47 раз в сравнении с алгоритмом Дейкстры. На всех тестовых графах алгоритм БП оказался быстрее алгоритма Дейстры, наибольшее по всем группам графов время счёта БП оказалось в 34 раза меньше аналогичного времени ДД. Рост максимальной степени вершины в графах последовательности $S = (G_1, G_2, \ldots, G_r)$ во время работы алгоритма явился допустимым и не превысил 17, что говорит о невысоком росте сложности процедуры удаления вершины на этапе разборки для исследованных графов.

Таблица 2

Результаты тестов

Группа графов	БП, среднее время, с	ДД, среднее время, с	БП, макс. время, с	ДД, макс. время, с	Среднее ускор. БП/ДД, раз	БП, макс. степень удал. верш.
$G1$	0,03	1,5	0,05	1,7	50	11
$G2$	0,13	6,7	0,14	7,1	52	12
$G3$	0,31	16	0,33	17	52	12
$G4$	0,67	30	0,88	32	45	22
$G5$	1,1	48	1,2	52	44	16
$G6$	1,5	72	1,8	82	48	20
$G7$	2,1	97	2,2	104	46	17
$G8$	2,6	131	2,8	145	50	21
$G9$	3,7	177	4,7	189	48	24
$G10$	5,4	218	6,4	230	40	23
GR	0,17	7,5	0,4	18,8	45	17

Заключение

Представленный алгоритм позволяет находить кратчайшие пути между всеми вершинами взвешенных ненаправленных графов. Эффективность алгоритма достигается за счёт разборки и сборки исходного графа с поиском промежуточного решения на графе малой размерности. Результаты тестов, проведённых на разреженных графах, позволяют говорить о значительном (в среднем в 47 раз) ускорении счёта в сравнении с одним из быстрых классических алгоритмов, решающих данную задачу.

Объектами дальнейших исследований являются подбор параметров алгоритма на основе быстрого анализа свойств рассматриваемого графа, модификация порядка разборки и сборки графа и вопрос масштабируемости алгоритма с увеличением размерности исследуемых графов. Представляет интерес возможность адаптации использованного в алгоритме подхода для решения задач поиска кратчайших путей в направленных графах и поиска кратчайших путей с заданной погрешностью.

ЛИТЕРАТУРА

1. *Bellman R.* On a routing problem // Quarter. Appl. Math. 1958. No. 16. P. 87–90.
2. *Floyd R. W.* Algorithm 97: Shortest Path // Comm. ACM. 1962. No 5(6). P. 345.
3. *Dijkstra E. W.* A note on two problems in connexion with graphs // Numer. Math. 1959. No. 1. P. 269–271.
4. *Кормен Т. Х. и др.* Алгоритмы: построение и анализ. М.: Вильямс, 2006. 1296 с.
5. *Johnson D. B.* Efficient algorithms for shortest paths in sparse graph // J. ACM. 1977. No. 24. P. 1–13.
6. *Geisberger R., Sanders P., et al.* Contraction hierarchies: faster and simpler hierarchical routing in road networks // International Workshop on Experimental Algorithms (WEA 2008). Provincetown: Springer, 2008. P. 319–333.
7. `http://www.dis.uniroma1.it/challenge9/download.shtml` — 9th DIMACS Implementation Challenge — Shortest Paths (дата обращения: ноябрь 2012).
8. *Ураков А. Р., Тимеряев Т. В.* Использование особенностей взвешенных графов для более быстрого определения их характеристик // Прикладная дискретная математика. 2012. № 2. С. 95–99.

2013 Прикладная теория графов №1(19)

DOI 10.17223/20710410/19/8

УДК 519.248:62-192+519.176

АСИМПТОТИКА ВЕРОЯТНОСТИ СВЯЗНОСТИ ГРАФА С НИЗКОНАДЁЖНЫМИ РЁБРАМИ[1]

Г. Ш. Цициашвили, М. А. Осипова, А. С. Лосев

Институт прикладной математики ДВО РАН, г. Владивосток, Россия

E-mail: guram@iam.dvo.ru, mao1975@list.ru, alexax@bk.ru

Для графов с низконадёжными ребрами построены асимптотики вероятностей связности всего графа и любой пары его вершин. Параметрами полученных соотношений являются характеристики остовных деревьев графа и кратчайших путей. Для вычисления характеристик остовных деревьев получены формулы с помощью теорем Кирхгофа — Трента, а для вычисления характеристик кратчайших путей разработаны модификации классических алгоритмов.

Ключевые слова: *остовное дерево, матрица Кирхгофа, кратчайший путь, вероятность связности, вычислительная сложность.*

Введение

В работе [1] для планарных графов с высоконадёжными рёбрами построен алгоритм вычисления вероятности несвязности на основе асимптотической формулы Буртина — Питтеля [2], параметрами которой являются минимальное число рёбер в разрезах графа и число таких разрезов. Асимптотические константы вычислены с помощью теоремы Уитни [3] о соответствии разрезов планарного графа циклам в двойственном графе и формул Харари [4], определяющих число простых циклов. Предложенный алгоритм имеет сложность не более кубической по числу граней двойственного графа, что значительно проще известных методов перечисления всех разрезов минимального объёма, имеющих геометрическую сложность.

В настоящей работе для случайных графов с низконадёжными рёбрами построены удобные в реализации алгоритмы вычисления вероятности связности в основном кубической сложности. При различных условиях, накладываемых на вероятности работоспособности рёбер, доказаны асимптотические соотношения для вероятности связности всего графа и любой пары его вершин. Параметрами полученных соотношений являются характеристики остовных деревьев графа и кратчайших путей. Для вычисления характеристик остовных деревьев получены формулы с помощью теорем Кирхгофа — Трента, а для вычисления характеристик кратчайших путей разработаны модификации классических алгоритмов. Особенностью предлагаемых алгоритмов является тот факт, что в них требуется не перечислять экстремальные подграфы (остовные деревья, кратчайшие пути между узлами), а лишь определять их количество. Ещё одним существенным фактором упрощения вычислений является рассмотрение графов с ограниченным диаметром, которые в последние годы вызывают большой теоретический и практический интерес [5 – 7].

1. Вероятность связности всего графа

Рассмотрим неориентированный связный простой (без петель и кратных рёбер) граф G с множеством узлов $U = \{1, \ldots, n\}$ и множеством рёбер V. Определим матрицу

[1]Работа поддержана грантом РФФИ № 12-01-00114-а.

Кирхгофа $K = ||k(i,j)||_{i,j=1}^{n}$:

$$k(i,j) = \begin{cases} \text{степень узла } i,\ i = j, \\ -1,\ (i,j) \in V, \\ 0 \text{ иначе.} \end{cases}$$

Под степенью узла понимается число инцидентных этому узлу рёбер. Обозначим m число остовных деревьев в графе G и предположим, что каждое ребро $v = (i,j)$, $i \neq j$, $1 \leqslant i,j \leqslant n$, графа G с вероятностью $p(v)$ работоспособно, причём все рёбра функционируют независимо. Если $p(v) = h$, $v \in V$, то для вероятности связности $P(G)$ графа G справедливо соотношение [8, формула (5)] (далее запись $f(x) \sim g(x)$, $x \to x_0$ означает, что существует и равен единице предел $\lim\limits_{x \to x_0} f(x)/g(x)$)

$$P(G) \sim mh^{n-1},\ h \to 0. \tag{1}$$

В силу теоремы Кирхгофа — Трента (см., например, [9]) алгебраические дополнения всех элементов матрицы K равны между собой и совпадают с m. Известно (см., например, [10]), что вычисление определителя порядка $n-1$ и, значит, коэффициента m методом Гаусса требует $\mathrm{O}(n^3)$ арифметических операций.

Замечание 1. Если $p(v) = h^{l(v)}$, $v \in V$, при некоторых натуральных $l(v)$, то, заменяя каждое ребро графа на $l(v)$ последовательно соединённых рёбер, можно к построенному таким образом графу применить формулу (1). Одним из приложений этого результата является исследование распределения случайного времени потери связности сети при $p(v) = P(\tau(v) > T)$, $T \to \infty$, где $\tau(v)$ — случайное время до отказа ребра v.

Теорема 1. Если при некоторых $s(v) > 0$, $v \in V$, выполняется $p(v) \sim s(v)h$, $h \to 0$, то

$$P(G) \sim m_1 h^{n-1},\ h \to 0. \tag{2}$$

Здесь m_1 — алгебраическое дополнение любого элемента (они все совпадают) матрицы $K_1 = ||k_1(i,j)||_{i,j=1}^{n}$ вида

$$k_1(i,j) = \begin{cases} \sum\limits_{t \in U,\ (i,t) \in V} s((i,t)),\ i = j, \\ -s((i,j)),\ (i,j) \in V, \\ 0 \text{ иначе,} \end{cases}$$

причём вычислительная сложность определения коэффициента m_1 равна $\mathrm{O}(n^3)$.

Доказательство. Обозначим $G_1, \ldots, G_m$ остовные деревья графа G, каждое из которых имеет $n-1$ ребро. Положим A_k событие, состоящее в работоспособности всех рёбер дерева G_k, $1 \leqslant k \leqslant m$. Тогда справедлива формула $P(G) = P\left(\bigcup\limits_{k=1}^{m} A_k\right)$ и вытекающие из неё неравенства

$$\sum_{k=1}^{m} P(A_k) - \sum_{1 \leqslant k_1 < k_2 \leqslant m} P(A_{k_1} A_{k_2}) \leqslant P(G) \leqslant \sum_{k=1}^{m} P(A_k).$$

Из условий теоремы на $p(v)$ следует, что

$$\sum_{k=1}^{m} P(A_k) \sim \sum_{k=1}^{m} \prod_{v \in G_k} hs(v) = h^{n-1} m_1,\ h \to 0,\ m_1 = \sum_{k=1}^{m} \prod_{v \in G_k} s(v),$$

$$\sum_{1\leqslant k_1<k_2\leqslant m} P(A_{k_1}A_{k_2}) \sim \sum_{1\leqslant k_1<k_2\leqslant m} \prod_{v\in G_{k_1}\cup G_{k_2}} hs(v) \sim h^n \sum_{1\leqslant k_1<k_2\leqslant m} \prod_{v\in G_{k_1}\cup G_{k_2}} s(v) = \mathrm{O}(h^n).$$

Из полученных соотношений приходим к формуле (2). В силу обобщения теоремы Кирхгофа — Трента (см., например, [9, теорема 1]) выражение m_1 совпадает с алгебраическим дополнением любого элемента матрицы K_1, вычислительная сложность определения которого $\mathrm{O}(n^3)$. ■

Замечание 2. Утверждение теоремы 1 автоматически переносится на мультиграф, полученный из графа G заменой каждого ребра $v \in V$ на $s(v)$ параллельно соединённых и независимо функционирующих рёбер с вероятностью работоспособности h. Такое параллельное соединение имеет вероятность работоспособности $\sim s(v)h,\ h \to 0$.

2. Вероятность связности пар вершин графа

Обозначим $D(i,j)$ минимальное число рёбер в путях, соединяющих узлы i,j графа G, $C(i,j)$ — число путей с $D(i,j)$ рёбрами, $\Gamma_1(i,j),\ldots,\Gamma_{C(i,j)}(i,j)$ — пути с $D(i,j)$ рёбрами. Для вероятности связности $P_{ij}(G)$ узлов i,j графа G доказаны следующие утверждения.

Теорема 2.

1. Если $p(v) = h,\ v \in V$, то

$$P_{ij}(G) \sim C(i,j)h^{D(i,j)},\ h \to 0. \tag{3}$$

2. Если при некоторых $s(v) > 0,\ v \in V$, выполняется $p(v) \sim s(v)h,\ h \to 0$, то

$$P_{ij}(G) \sim m(i,j)h^{D(i,j)},\ h \to 0,\ m(i,j) = \sum_{t=1}^{C(i,j)} \prod_{v\in\Gamma_t(i,j)} s(v). \tag{4}$$

Доказательство данной теоремы аналогично доказательству теоремы 1, с тем лишь отличием, что в нем остовные деревья заменяются на кратчайшие пути между узлами i,j.

Следствие 1. Если $p(v) = h,\ v \in V$, то

$$\min_{1\leqslant i,j\leqslant n} P_{ij}(G) \sim Ch^D,\ h \to 0,$$

$$D = \max_{1\leqslant i,j\leqslant n} D(i,j),\ C = \min_{(i,j):\ D(i,j)=D} C(i,j).$$

Остановимся на вычислении коэффициентов $D(i,j),\ C(i,j)$ асимптотической формулы (3). Найдем все элементы матриц $||D(i,j)||_{i,j=1}^n,\ ||C(i,j)||_{i,j=1}^n$, так как это более экономичная процедура, чем последовательное определение элементов этой матрицы.

Для вычисления матрицы $||D(i,j)||_{i,j=1}^n$ воспользуемся алгоритмом Флойда — Стейнберга. Следуя [11], введём матрицу $R = ||r(i,j)||_{i,j=1}^n$ равенствами

$$r(i,j) = \begin{cases} 1,\ (i,j) \in V, \\ 0,\ i = j, \\ +\infty \text{ иначе.} \end{cases}$$

На множестве квадратных матриц размера $n \times n$ введём операцию произведения $\otimes$ следующего вида: $T \otimes Q = ||(t \otimes q)(i,j)||_{i,j=1}^n$, где

$$(t \otimes q)(i,j) = \min_{1\leqslant p\leqslant n} (t(i,p) + q(p,j)),\ 1 \leqslant i,j \leqslant n.$$

Обозначим $t = \min(f : 2^f \geqslant n)$, тогда матрицу $||D(i,j)||_{i,j=1}^n$ можно вычислить с помощью рекуррентной процедуры

$$R^2 = R \otimes R,\ R^{2^{f+1}} = R^{2^f} \otimes R^{2^f},\ 1 \leqslant f < t,\ ||D(i,j)||_{i,j=1}^n = R^n = R^{2^t}. \tag{5}$$

Для вычисления матрицы $||D(i,j)||_{i,j=1}^n$ в соответствии с алгоритмом Флойда — Стейнберга требуется $2n^3 \log_2 n$ арифметических операций. Зная матрицу $||D(i,j)||_{i,j=1}^n$, можно вычислить диаметр D графа G.

Перейдём теперь к вычислению матрицы $||C(i,j)||_{i,j=1}^n$. Обозначим $||C_1(i,j)||_{i,j=1}^n$ матрицу смежности графа G :

$$C_1(i,j) = \begin{cases} 1,\ (i,j) \in V, \\ 0\ \text{ иначе.} \end{cases}$$

Известно [12], что матрицы $||C_k(i,j)||_{i,j=1}^n$, элементами которых являются количества путей длины k между узлами i, j в графе G, удовлетворяют равенствам

$$||C_{k+1}(i,j)||_{i,j=1}^n = ||C_k(i,j)||_{i,j=1}^n \cdot ||C_1(i,j)||_{i,j=1}^n,\ 1 \leqslant k < n. \tag{6}$$

В свою очередь, справедливы очевидные соотношения

$$C(i,j) = C_{D(i,j)}(i,j),\ 1 \leqslant i,j \leqslant n. \tag{7}$$

Таким образом, для вычислении матрицы $||C(i,j)||_{i,j=1}^n$ с помощью формул (6), (7) требуется определить матрицы $||C_1(i,j)||_{i,j=1}^n, \ldots, ||C_{n-1}(i,j)||_{i,j=1}^n$. Эта процедура имеет вычислительную сложность $\mathrm{O}(n^4.)$

Замечание 3. Для сетей с ограниченным диаметром D в формуле (5) можно заменить величину t на $\min\{f : 2^f \geqslant D\}$, при этом сложность вычисления матрицы $||D(i,j)||_{i,j=1}^n$ составляет $2n^3 \log_2 D$ арифметических операций. В свою очередь, для вычисления матрицы $||C(i,j)||_{i,j=1}^n$ с помощью формул (6), (7) понадобится вычислить матрицы $||C_l(i,j)||_{i,j=1}^n$, $l = 1, \ldots, D$, что потребует $\mathrm{O}(Dn^3)$ арифметических операций.

Замечание 4. Для вычисления матрицы $||m(i,j)||_{i,j=1}^n$ в соотношении (4) надо ввести матрицу $||C_1(i,j)||_{i,j=1}^n$ следующим образом:

$$C_1(i,j) = \begin{cases} s((i,j)),\ (i,j) \in V, \\ 0\ \text{ иначе} \end{cases}$$

и, воспользовавшись равенством (6) и $m(i,j) = C_{D(i,j)}(i,j),\ 1 \leqslant i,j \leqslant n$, найти матрицы $||C_l(i,j)||_{i,j=1}^n,\ l = 1, \ldots, n-1$.

3. Вычислительный эксперимент

Зададим граф G графически (рис. 1).

Составим для графа G матрицу Кирхгофа:

$$K = \begin{pmatrix} 3 & -1 & 0 & -1 & 0 & -1 \\ -1 & 3 & -1 & 0 & 0 & -1 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ -1 & 0 & -1 & 3 & -1 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 \\ -1 & -1 & 0 & 0 & -1 & 3 \end{pmatrix}.$$

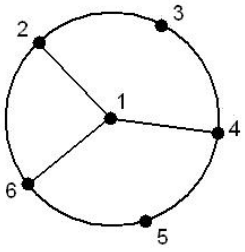

Рис. 1. Граф G

Число остовных деревьев графа m совпадает с алгебраическим дополнением любого элемента матрицы K. В нашем случае $m = 35$. Полагая, что вероятность работоспособности рёбер равна $p(v) = h = 0{,}1$, вычислим вероятность связности графа по формуле (1) и методом Монте-Карло при 10^7 итераций, обозначив её $P^*(G)$:

$$P(G) \approx 0{,}00035,\ P^*(G) \approx 0{,}000283.$$

Время счета по формуле (1) составило 2 с, а методом Монте-Карло — 12 ч.

Для заданного графа вычислим вероятности связности между всеми парами вершин, полагая $p(v) = h = 0{,}01$. С использованием рекуррентных процедур (5) и (6) вычислены матрицы $||D(i,j)||_{i,j=1}^n$, $||C(i,j)||_{i,j=1}^n$, характеризующие минимальное число рёбер в путях и количество путей с минимальным числом рёбер:

$$||D(i,j)||_{i,j=1}^n = \begin{pmatrix} 0 & 1 & 2 & 1 & 2 & 1 \\ 1 & 0 & 1 & 2 & 2 & 1 \\ 2 & 1 & 0 & 1 & 2 & 2 \\ 1 & 2 & 1 & 0 & 1 & 2 \\ 2 & 2 & 2 & 1 & 0 & 1 \\ 1 & 1 & 2 & 2 & 1 & 0 \end{pmatrix}, \quad ||C(i,j)||_{i,j=1}^n = \begin{pmatrix} 0 & 1 & 2 & 1 & 2 & 1 \\ 1 & 0 & 1 & 2 & 1 & 1 \\ 2 & 1 & 0 & 1 & 1 & 1 \\ 1 & 2 & 1 & 0 & 1 & 2 \\ 2 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 2 & 1 & 0 \end{pmatrix}.$$

Результаты вычислений вероятностей связности пар вершин $P_{ij}(G)$, $1 \leqslant i, j \leqslant n$, по формуле (3) и методом Монте-Карло ($P_{ij}^*(G)$) при 10^6 итераций следующие:

$$||P_{ij}(G)||_{i,j=1}^n = \begin{pmatrix} 1 & 0{,}01 & 0{,}0002 & 0{,}01 & 0{,}0002 & 0{,}01 \\ 0{,}01 & 1 & 0{,}01 & 0{,}0002 & 0{,}0001 & 0{,}01 \\ 0{,}0002 & 0{,}01 & 1 & 0{,}01 & 0{,}0001 & 0{,}0001 \\ 0{,}01 & 0{,}0002 & 0{,}01 & 1 & 0{,}01 & 0{,}0002 \\ 0{,}0002 & 0{,}0001 & 0{,}0001 & 0{,}01 & 1 & 0{,}01 \\ 0{,}01 & 0{,}01 & 0{,}0001 & 0{,}0002 & 0{,}01 & 1 \end{pmatrix},$$

$$||P_{ij}^*(G)||_{i,j=1}^n = \begin{pmatrix} 1 & 0{,}01035 & 0{,}000203 & 0{,}010027 & 0{,}000192 & 0{,}010205 \\ 0{,}01035 & 1 & 0{,}010001 & 0{,}000198 & 0{,}000095 & 0{,}009764 \\ 0{,}000203 & 0{,}010001 & 1 & 0{,}010083 & 0{,}000094 & 0{,}000103 \\ 0{,}010027 & 0{,}000198 & 0{,}010083 & 1 & 0{,}010051 & 0{,}000208 \\ 0{,}000192 & 0{,}000095 & 0{,}000094 & 0{,}010051 & 1 & 0{,}009973 \\ 0{,}010205 & 0{,}009764 & 0{,}000103 & 0{,}000208 & 0{,}009973 & 1 \end{pmatrix}.$$

Время счета по формуле (3) составило 10 с, методом Монте-Карло — 6 ч.

ЛИТЕРАТУРА

1. *Tsitsiashvili G. Sh.*Complete calculation of disconnection probability in planar graphs // Reliability: Theory and Applications. 2012. V. 1. No. 1. P. 154–159.

2. *Буртин Ю., Питтель Б.* Асимптотические оценки надёжности сложных систем // Техническая кибернетика. 1972. Т. 10. № 3. С. 90–96.
3. *Whithney H.* Nonseparable and planar graphs // Transact. Amer. Math. Soc. 1932. V. 34. P. 339–369.
4. *Харари Ф.* Теория графов. М.: Мир, 1973. 314 с.
5. *Мигов Д. А.* Расчет надежности сети с ограничением на диаметр с применением точек сочленения // Автоматика и телемеханика. 2011. № 7. С. 69–74.
6. *Мигов Д. А.* Расчет надежности сети с ограничением на диаметр с использованием сочленений // Проблемы информатики. 2011. № 3. С. 4–9.
7. *Райгородский А. М.* Модели случайных графов и их применение // Труды МФТИ. 2010. Т. 2. № 4. С. 130-140.
8. *Ломоносов М. В., Полесский В. П.* Нижняя оценка надежности сетей // Проблемы передачи информации. 1972. Т. 8. № 2. С. 47–53.
9. *Чеботарев П. Ю., Шамис Е. В.* Матричная теорема о лесах и измерение связей в малых социальных группах // Автоматика и телемеханика. 1997. Т. 9. С. 125–137.
10. *Ильин В. А., Позняк Э. Г.* Линейная алгебра. М.: Физматлит, 2004. 280 с.
11. *Floid R. W. and Steinberg L.* An adaptive algorithm for spatial grayscale // SID 75 Digest. New York, N.Y.: Lewis Winner, 1975. P. 36–37.
12. *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы: построение и анализ. М.: МЦНМО, 2000. 893 с.

ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ В ДИСКРЕТНОЙ МАТЕМАТИКЕ

DOI 10.17223/20710410/19/9

УДК 519.8

ИССЛЕДОВАНИЕ ЗАДАЧ ДИСКРЕТНОЙ ОПТИМИЗАЦИИ С ЛОГИЧЕСКИМИ ОГРАНИЧЕНИЯМИ НА ОСНОВЕ МЕТОДА РЕГУЛЯРНЫХ РАЗБИЕНИЙ[1]

А. А. Колоколов*, А. В. Адельшин*, Д. И. Ягофарова**

*Омский филиал Института математики им. С. Л. Соболева СО РАН, г. Омск, Россия

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

E-mail: kolo@ofim.oscsbras.ru, adelshin@ofim.oscsbras.ru, ydarya@mail.ru

Рассматриваются различные постановки задач дискретной оптимизации с логическими ограничениями и возможности их применения для решения некоторых задач проектирования сложных изделий. Представлен обзор результатов исследования указанных задач на основе моделей целочисленного программирования и метода регулярных разбиений. С использованием данного подхода проведен анализ структуры и сложности задач, предложены алгоритмы решения задач выполнимости и максимальной выполнимости.

Ключевые слова: *задача выполнимости, логические ограничения, целочисленное программирование, перебор L-классов.*

Введение

Значительное число исследований в области дискретной оптимизации посвящено задаче выполнимости логической формулы и её обобщениям. Широкое практическое применение этих задач в экономике, управлении, проектировании и других областях является стимулом для разработки алгоритмов их решения. Алгоритмы решения задач, связанных с выполнимостью логической формулы, представлены, например, в [1 – 4]. Одним из подходов к анализу и решению задач выполнимости и максимальной выполнимости является использование аппарата целочисленного линейного программирования (ЦЛП).

Для анализа задач целочисленного программирования (ЦП), разработки и исследования алгоритмов, основанных на релаксации условия целочисленности, А. А. Колоколовым предложен метод регулярных разбиений [5]. Ранее с помощью этого подхода была исследована сложность решения ряда задач, изучена структура релаксационных множеств, введены новые классы отсечений, построены оценки числа итераций для некоторых известных алгоритмов, проведён анализ устойчивости задач и алгоритмов, разработаны новые алгоритмы.

В работах [6, 7] получен ряд теоретических результатов для задач с логическими условиями. Исследована L-структура многогранников задач выполнимости и максимальной выполнимости. Выделены семейства логических формул, для которых мощности L-накрытий указанных задач растут экспоненциально с увеличением числа пе-

[1]Работа поддержана грантом РФФИ № 10-01-00598.

ременных. Получены оценки числа итераций некоторых алгоритмов целочисленного линейного программирования при решении задач из построенных семейств.

В данной работе рассматриваются некоторые из перечисленных результатов при исследовании задач выполнимости и максимальной выполнимости. Кроме того, строятся модели ЦЛП для задач проектирования на основе задач дискретной оптимизации с логическими ограничениями и приводятся алгоритмы решения задач выполнимости и максимальной выполнимости с использованием метода регулярных разбиений.

1. Задачи с логическими ограничениями

Рассмотрим постановку задачи выполнимости (SAT). Пусть $x_1, \ldots, x_n$ — переменные, принимающие значение `истина` или `ложь`. Под литералом понимается либо переменная x_j, либо ее отрицание. Пусть логическая формула F представляет собой конъюнкцию формул (скобок) $C_i, i = 1, \ldots, m$, каждая из которых является дизъюнкцией литералов. Требуется определить, выполнима ли формула F, т. е. существует ли такой набор значений переменных, при котором F принимает значение `истина`. Известно, что задача SAT является NP-полной, а в случае, когда каждая скобка содержит не более двух литералов, задача полиномиально разрешима [8]. Многие известные задачи теории графов, построения расписаний, криптографии могут быть сформулированы в виде задачи выполнимости некоторой логической формулы в конъюнктивной нормальной форме.

Важным обобщением задачи SAT является задача максимальной выполнимости (MAX SAT). Предположим, что каждой скобке C_i соответствует неотрицательный вес c_i, тогда задача MAX SAT состоит в отыскании набора значений логических переменных, при котором суммарный вес выполненных скобок будет наибольшим. Актуальными также являются постановки так называемых смешанных задач [4, 9], в которых присутствуют два вида логических ограничений — «жёсткие» и «мягкие», причем веса приписаны только «мягким». При этом требуется найти набор значений логических переменных, при котором выполнены все «жёсткие» ограничения, а суммарный вес выполненных «мягких» является наибольшим. Указанные задачи находят применение при решении многих практических задач, в частности в проектировании изделий сложной структуры [9 – 11].

Перейдём к построению моделей ЦЛП рассматриваемых задач. Введём булевы переменные $y_1, \ldots, y_n$, такие, что y_j соответствует переменной x_j, а $(1 - y_j)$ — её отрицанию, т. е. $y_j = 1$ тогда и только тогда, когда переменная x_j принимает значение `истина`. Нетрудно показать, что условие выполнимости логической формулы F эквивалентно существованию решения системы

$$\sum_{j \in C_i^+} y_j - \sum_{j \in C_i^-} y_j \geqslant 1 - |C_i^-|, \ i = 1, \ldots, m; \tag{1}$$

$$0 \leqslant y_j \leqslant 1, \ j = 1, \ldots, n; \tag{2}$$

$$y_j \in \mathbb{Z}, \ j = 1, \ldots, n, \tag{3}$$

где C_i^- и C_i^+ — множества индексов переменных, входящих в скобку C_i с отрицанием и без него соответственно; $|C_i^-|$ — мощность множества C_i^-.

Модель ЦЛП для задачи выполнимости будет получена, если к последней системе добавить целевую функцию. При этом может быть выбрана, например, функция $f(y) = y_1 \to \max$ или $f(y) = \sum\limits_{j=1}^{n} y_j \to \max$, где $y = (y_1, \ldots, y_n)$. Здесь рассматри-

вается лексикографическая постановка задачи ЦЛП, т. е. ищется лексикографически максимальный вектор y, удовлетворяющий системе ограничений (1) — (3).

Модель ЦЛП для задачи максимальной выполнимости выглядит следующим образом:

$$y_0 = \sum_{i=1}^{m} c_i z_i \to \max; \tag{4}$$

$$\sum_{j \in C_i^-} y_j - \sum_{j \in C_i^+} y_j + z_i \leqslant |C_i^-|, \ i = 1, \ldots, m; \tag{5}$$

$$0 \leqslant y_j \leqslant 1, \ y_j \in \mathbb{Z}, \ j = 1, \ldots, n; \tag{6}$$

$$0 \leqslant z_i \leqslant 1, \ z_i \in \mathbb{Z}, \ i = 1, \ldots, m. \tag{7}$$

Здесь каждой скобке C_i поставлена в соответствие переменная z_i, причем в любом допустимом целочисленном решении $z_i = 1$ только в том случае, когда C_i выполнена. Таким образом, оптимальное значение целевой функции равно наибольшему суммарному весу выполненных скобок. Отметим, что если к (4) — (7) добавить ограничения (1), то получим модель ЦЛП для смешанной задачи с «жёсткими» ограничениями (1) и «мягкими» ограничениями (5).

2. Прикладные постановки

Постановки задач с логическими ограничениями могут использоваться при анализе и решении многих прикладных задач. Отметим некоторые из них. В [9, 10] исследуются и решаются задачи проектирования сложных изделий из набора составляющих, работа [12] посвящена задаче формирования производственных групп с учетом межличностных отношений, в [13, 14] решаются задачи логического криптоанализа. Кроме того, большое число приложений задачи выполнимости приведено в обзоре [2].

Остановимся подробнее на задачах проектирования. На основе моделей задач дискретной оптимизации с логическими условиями в последнее время успешно развивается подход к автоматизации проектирования сложных изделий. Отметим, что вопросам автоматизации проектирования уделяется большое внимание, что обусловлено актуальностью этого направления в различных отраслях промышленности. К примеру, в [9, 11] отмечается применимость рассматриваемых задач для проектирования одежды, в [10] решаются задачи проектирования химического состава резин специального назначения.

Рассмотрим процесс проектирования сложных изделий, которые формируются из множества составляющих. В случае проектирования одежды к ним относятся различные покрои рукавов, виды воротников, карманов, застёжек и т. д. Естественно, что не все составляющие входят в изделие одновременно, поскольку помимо функциональной нецелесообразности определённых сочетаний некоторые из них приводят к визуальной дисгармонии. Такие условия могут отражаться с помощью логических ограничений и приводить к задачам выполнимости и максимальной выполнимости логической формулы. При проектировании химического состава резин возникают ограничения на включение различных химических элементов в состав смесей, которые также могут описываться с помощью логических условий. Кроме того, наряду с логическими ограничениями учитываются конструкторские, технологические, экономические и др.

Приведём математическую модель для задачи проектирования сложных изделий. Введём необходимые обозначения:

J — множество номеров составляющих изделия, $J = \{1, \ldots, n\}$;

v_j — составляющая изделия с номером $j \in J$;

x_j — логическая переменная, принимающая значение `истина`, если v_j входит в состав изделия, и значение `ложь` в противном случае, $j \in J$;

s_j — вес составляющей v_j, характеризующий степень целесообразности её включения в изделие;

p — нижняя граница для суммарного веса составляющих v_j, включённых в изделие;

I — множество номеров логических формул, используемых в задаче, $I = \{1, \ldots, m\}$;

I_1 — множество номеров логических формул, которые должны быть обязательно выполнены, $I_1 \subseteq I$;

C_i — логическая формула, соответствующая i-му логическому ограничению, которая представляет собой дизъюнкцию переменных x_j или их отрицаний, $i \in I$;

c_i — вес формулы C_i, характеризующий степень необходимости её выполнения, $i \in I \backslash I_1$;

K — множество номеров используемых ресурсов, $K = \{1, \ldots, q\}$;

a_{kj} — объём k-го ресурса, требуемый для изготовления j-й составляющей изделия, $k \in K$;

b_k — имеющийся объём k-го ресурса, $k \in K$.

Задача состоит в отыскании значений логических переменных, при которых выполняются формулы C_i с номерами $i \in I_1$, ограничения по ресурсам и по суммарному весу включённых в изделие составляющих v_j, а вес выполненных формул C_i для $i \in I \backslash I_1$ максимален. Если поставленная задача разрешима, то в результате может быть получен один или несколько наборов составляющих, из которых формируются проектируемые изделия.

Модель ЦЛП рассматриваемой задачи имеет вид

$$f(z) = \sum_{i \in I \backslash I_1} c_i z_i \to \max; \tag{8}$$

$$\sum_{j \in C_i^-} y_j - \sum_{j \in C_i^+} y_j + z_i \leqslant |C_i^-|, \ i \in I \backslash I_1; \tag{9}$$

$$\sum_{j \in C_i^-} y_j - \sum_{j \in C_i^+} y_j \leqslant |C_i^-| - 1, \ i \in I_1; \tag{10}$$

$$\sum_{j \in J} a_{kj} y_j \leqslant b_k, \ k \in K; \tag{11}$$

$$\sum_{j \in J} s_j y_j \geqslant p; \tag{12}$$

$$0 \leqslant y_j \leqslant 1, \ y_j \in \mathbb{Z}, \ j \in J; \tag{13}$$

$$0 \leqslant z_i \leqslant 1, \ z_i \in \mathbb{Z}, \ i \in I. \tag{14}$$

Здесь система (9) представляет собой набор «мягких» ограничений, а система (10) — «жёстких». Условия (11) описывают ресурсные ограничения, (12) — возможные требования проектировщика. В [9, 11] приводятся также более сложные модели, в которых учитываются группы составляющих и другие ограничения.

Рассмотрим задачу проектирования, состоящую в разработке рецептуры огнестойкой резины на основе смеси каучуков [10]. В дополнение к введённым обозначениям, через J_1 обозначим множество номеров антипиренов (ингредиентов, обладающих свойством огнестойкости), $J_1 \subset J$, а через r_j — вес, соответствующий ингредиенту v_j, который характеризует его огнестойкость, $j \in J_1$. Двухкритериальная модель ЦЛП этой

задачи может быть записана в виде системы (9), (10), (13), (14) с критериями

$$f(z) = \sum_{i \in I \setminus I_1} c_i z_i \to \max; \quad g(y) = \sum_{j \in J_1} r_j y_j \to \max,$$

в которых максимизируются общий вес «мягких» ограничений и суммарная огнестойкость включённых в состав ингредиентов.

В работе [10] представлены результаты решения указанной задачи. Все ингредиенты были разбиты на несколько групп в зависимости от их свойств. Разработаны логические ограничения, которые отражают недопустимость или желательность определённых комбинаций составляющих. Проведён вычислительный эксперимент с использованием алгоритма перебора L-классов и показана перспективность данного подхода.

3. О методе регулярных разбиений

Рассмотрим кратко подход, идея которого заключается в выделении семейства специальных разбиений релаксационного множества задачи. Особенности данного подхода подробно описаны в [5]. Приведём необходимые для дальнейшего изложения определения и обозначения.

Введём понятие лексикографического порядка. Для этого рассмотрим функцию

$$\eta(x, y) = \min\{i : x_i \neq y_i,\ i = 1, \ldots, n\},\ x, y \in \mathbb{R}^n,\ x \neq y.$$

Вектор x *лексикографически больше* (*меньше*) вектора y, т. е. $x \succ y$ ($x \prec y$), если $x \neq y$ и $x_w > y_w$ ($x_w < y_w$) для $w = \eta(x, y)$. Отношение $\succ$ представляет собой отношение строгого линейного порядка. Запись $x \succeq y$ означает, что либо $x \succ y$, либо $x = y$. Аналогично понимается $x \preceq y$.

Для множеств $X, Y \subset \mathbb{R}^n$ полагаем $X \succ Y$ ($X \prec Y$), если $x \succ y$ ($x \prec y$) при любых $x \in X$ и $y \in Y$.

Пусть $x, y \in \mathbb{R}^n$ и $x \succ y$. Будем говорить, что точки x и y *отделимы*, если найдётся точка $z \in \mathbb{Z}^n$, для которой выполняется $x \succeq z \succeq y$. Точку z назовем *отделяющей*.

Дадим определение L-разбиения. Точки $x, y \in \mathbb{R}^n$ ($x \succ y$) называются *L-эквивалентными*, если не существует отделяющей их точки $z \in \mathbb{Z}^n$. Это отношение эквивалентности порождает разбиение любого множества $X \subset \mathbb{R}^n$ на непересекающиеся *L-классы*. Фактор-множество X/L называется *L-разбиением* множества X. Указанное разбиение обладает рядом полезных свойств, применяемых при разработке и исследовании алгоритмов целочисленного программирования, в частности алгоритмов перебора L-классов. Отметим некоторые из них:

1) каждая точка $z \in \mathbb{Z}^n$ образует отдельный L-класс, остальные классы состоят из нецелочисленных точек и называются *дробными*;
2) если X — ограниченное множество, то фактор-множество X/L конечно;
3) любой дробный L-класс $V \in X/L$ можно представить в виде

$$V = X \cap \{x :\ x_1 = a_1, \ldots, x_{r-1} = a_{r-1}, a_r < x_r < a_r + 1\},$$

где $a_i \in \mathbb{Z}$ $(i = 1, \ldots, r)$, $1 \leqslant r \leqslant n$.

Рассмотрим лексикографическую задачу ЦЛП следующего вида: найти лексикографически максимальную точку y^* множества $(M \cap \mathbb{Z}^n)$, т. е.

$$\text{найти } y^* = \operatorname{lexmax}(M \cap \mathbb{Z}^n), \tag{15}$$

где M — некоторый выпуклый многогранник. Важную роль в исследовании задачи (15) и методов её решения играет множество

$$M_* = \{y \in M : \ \forall z \in M \cap \mathbb{Z}^n \ (y \succ z)\},$$

которое называется *дробным накрытием* задачи (15). Фактор-множество M_*/L называется L-*накрытием* задачи.

Выделение этой части релаксационного множества задачи связано с тем, что в некоторых методах ЦП (методах отсечения, перебора L-классов) происходит последовательное исключение точек из M_*, т. е. эти методы можно рассматривать как определённые способы «снятия» дробных накрытий, поэтому мощность L-накрытия является параметром сложности задачи для указанного класса алгоритмов.

Ранее с использованием данного подхода получены оценки числа итераций для некоторых известных алгоритмов ЦП. В частности, для задачи ЦЛП (15) установлено $I \leqslant |M_*/L|$, где I — количество L-регулярных отсечений, т. е. таких отсечений, которые сохраняют все допустимые целочисленные решения и отсекают текущее лексикографическое решение задачи линейного программирования вместе со всем L-классом, содержащим его [5]. К примеру, некоторые отсечения первого алгоритма Гомори являются L-регулярными.

При анализе структуры многогранников интересным представляется построение семейств задач ЦЛП, для которых мощности L-накрытий достаточно велики. Такие задачи оказываются труднорешаемыми для определённых классов алгоритмов, основанных на аппарате непрерывной оптимизации. В работах [6, 7] описываются различные семейства задач выполнимости и максимальной выполнимости, для которых мощности L-накрытий растут экспоненциально с увеличением числа переменных в формуле. Приведём одно из таких семейств для невыполнимых формул.

Пусть $C_i = (x_{2i-1} \vee x_{2i})$, $i = 1, \ldots, n/2-1$, $C'_j = (\bar{x}_{2j-1} \vee x_{n-1} \vee x_n)$, $j = 1, \ldots, n/2-1$, где n — чётное число, большее 2, и $C' = (x_{n-1} \vee \bar{x}_n)(\bar{x}_{n-1} \vee x_n)(\bar{x}_{n-1} \vee \bar{x}_n)$. Рассмотрим формулу $F = C_1 \wedge C_2 \wedge \ldots \wedge C_r \wedge C'_1 \wedge C'_2 \wedge \ldots \wedge C'_r \wedge C'$, где $r = n/2-1$. Структура матрицы A системы ограничений (1), соответствующей формуле F, выглядит следующим образом:

$$A = \begin{pmatrix}
1 & 1 & & & & & & & & \\
 & & 1 & 1 & & & & & & \\
 & & & & \cdots & & & & & \\
 & & & & & & 1 & 1 & & \\
-1 & & & & & & & & 1 & 1 \\
 & & -1 & & & & & & 1 & 1 \\
 & & & & \cdots & & & & & \\
 & & & & & -1 & & & 1 & 1 \\
 & & & & & & & & 1 & -1 \\
 & & & & & & & & -1 & 1 \\
 & & & & & & & & -1 & -1
\end{pmatrix}$$

В матрице A указаны только ненулевые элементы. Можно показать, что для многогранника задачи выполнимости с формулой F выполняется $|M_*/L| = 2 \cdot 3^r - 1$.

4. Разработка алгоритмов

Приведём алгоритмы для задач выполнимости и максимальной выполнимости, основанные на методе регулярных разбиений. Начальный вариант алгоритма перебора

L-классов для решения задачи выполнимости предложен в работе [15]. Подробное его изложение и некоторые модификации, а также экспериментальные исследования содержатся в [1]. Первое описание алгоритма точного решения для невзвешенного варианта задачи максимальной выполнимости представлено в [16], во взвешенном случае — в [17].

4.1. Алгоритм перебора L-классов для задачи выполнимости

Рассмотрим лексикографическую постановку задачи ЦЛП (15), где M — многогранник, определяемый системой ограничений (1), (2).

Опишем идею метода перебора L-классов для задачи (15). Пусть L-накрытие задачи состоит из классов $V_1, V_2 \ldots, V_p$, причем $V_1 \succ V_2 \succ \ldots \succ V_p$.

Основной шаг метода заключается в переходе от одного класса из M_*/L к другому в порядке лексикографического убывания, начиная с лексикографически максимального L-класса. Алгоритм порождает последовательность точек $y^{(t)} \in M_*$, обладающую следующими свойствами:

— $y^{(t)} \succ y^{(t+1)},\ t = 1, 2, \ldots, s-1;\ s \leqslant p$;
— все точки принадлежат различным L-классам.

В общем случае для задачи ЦЛП переход от одного L-класса к другому выполняется путём решения некоторого числа задач линейного программирования. Для решения задачи выполнимости поиск представителей $y^{(t)}$ указанных классов можно осуществлять, не решая задач линейного программирования, что заметно повышает эффективность алгоритма. Это основано на следующем свойстве: в каждом L-классе многогранника M содержится *полуцелочисленная точка*, т. е. точка со значениями координат из множества $\{0, 1/2, 1\}$. Поиск представителей осуществляется с помощью специальной комбинаторной *процедуры* W, в процессе выполнения которой строится лексикографически убывающая последовательность недопустимых целочисленных точек с координатами, равными 0, 1 или 2. Отметим, что значение 2 используется для обеспечения лексикографической монотонности указанной последовательности. С этой целью можно также выбрать любое число, большее 1.

Приведём алгоритм перебора L-классов для задачи выполнимости (алгоритм LCE), который за конечное число шагов либо находит лексикографически максимальное решение в случае выполнимости формулы, либо указывает на её невыполнимость.

Предполагается, что перед началом работы алгоритма формула не содержит единичных скобок, т. е. скобок вида x_j или $\bar{x}_j$. В противном случае число переменных и скобок можно сократить путём присвоения таким переменным соответствующих значений. Если в ходе работы алгоритма в формуле появляется единичная скобка x_j, то будем считать, что переменной y_j значение 1 присвоено «*автоматически*» (аналогично для скобки $\bar{x}_j$ — значение 0).

Алгоритм завершает работу в двух случаях: если получено оптимальное решение задачи (15) или если не удаётся найти очередной L-класс. Показано, что трудоёмкость алгоритма LCE составляет $O(n^3 m \cdot |M_*/L|)$.

Алгоритм LCE

Шаг 0 (*поиск представителя первого L-класса*). Выполняем *процедуру* W (она описана ниже), начиная с недопустимой точки $w^{(0)1} = (2, \ldots, 2)$. Если находим полуцелочисленную точку $y^{(1)} \in V_1$, то переходим к шагу 1 первой итерации.

Итерация t ($t \geqslant 1$).

Шаг 1. Начинаем с точки $y^{(t)} = (a_1^{(t)}, \ldots, a_{s-1}^{(t)}, 1/2, y_{s+1}^{(t)}, \ldots, y_n^{(t)})$, где $a_i^{(t)} \in \{0, 1\}$, $i = 1, \ldots, s-1$; $y_j^{(t)} \in \{0, 1/2, 1\}$, $j = s+1, \ldots, n$. Первую дробную координату $y_s^{(t)}$ округляем до 0. Выполняем *процедуру упрощения формулы*. В ходе этого преобразования некоторые координаты $y_j^{(t)}$ $(j = s+1, \ldots, n)$ «автоматически» примут значения 0 или 1. Обозначим полученную точку $\tilde{y}^{(t)}$. После упрощения формулы возникает один из следующих случаев.

— *Все скобки исключены.* Рассматривая точку $\tilde{y}^{(t)}$, переходим на шаг 4.
— *Скобки остались, противоречие не обнаружено.* Из вектора $\tilde{y}^{(t)}$ строим недопустимую точку $w^{(t)1} = (a_1^{(t)1}, \ldots, a_{s-1}^{(t)1}, 0, w_{s+1}^{(t)1}, \ldots, w_n^{(t)1})$, где $w_j^{(t)1} \in \{0, 1, 2\}$, причём если $\tilde{y}_j^{(t)} = 1/2$, то $w_j^{(t)1} = 2$, $j = s+1, \ldots, n$. Переходим на шаг 2.
— *Обнаружено противоречие.* Возвращаемся к точке $y^{(t)}$, переходим на шаг 3.

Шаг 2. Выполняем *процедуру* W, начиная с недопустимой точки $w^{(t)1}$. Если находим полуцелочисленную точку $y^{(t+1)} \in V_{t+1}$, то переходим к шагу 1 итерации $(t+1)$.

Шаг 3. Среди координат вектора $y^{(t)}$ ищем максимальную по номеру координату $y_p^{(t)} = 1$, которой это значение не было присвоено «автоматически» $(p < s)$. Возможны следующие случаи.

— Такая координата *существует*, тогда продолжаем просматривать вектор $y^{(t)}$. Находим все переменные $y_j^{(t)}$ $(p < j \leqslant n)$, которым значение 0 или 1 присвоено после назначения $a_p^{(t)} = 1$. Изменяем их значения на $1/2$. Значение координаты с номером p также уменьшаем до $1/2$. В результате получаем точку $y^{(t+1)} \in V_{t+1}$ и переходим на шаг 1 итерации $(t+1)$.
— Такой координаты *не существует*. В этом случае алгоритм завершает работу: *формула невыполнима*.

Шаг 4. Алгоритм завершает работу: *формула выполнима*. Оптимальное решение y^* задачи (15) строим из текущего вектора $\tilde{y}^{(t)}$ следующим образом:

$$y_j^* = \begin{cases} \tilde{y}_j^{(t)}, & \text{если } \tilde{y}_j^{(t)} \in \{0, 1\}, \\ 1, & \text{если } \tilde{y}_j^{(t)} = 1/2, \end{cases}$$

для $j = 1, \ldots, n$.

Отметим, что на шаге 4 координатам со значением $1/2$ можно произвольным образом присвоить булевы значения. Это дает множество различных допустимых решений задачи (15).

Приведём описание процедуры W, используемой в алгоритме LCE.

Процедура W

Полагаем $\gamma = 1$.

Итерация γ.

Начинаем с вектора $w^{(t)\gamma} = (a_1^{(t)\gamma}, \ldots, a_{s-1}^{(t)\gamma}, 0, w_{s+1}^{(t)\gamma}, \ldots, w_n^{(t)\gamma})$, где $a_i^{(t)\gamma} \in \{0, 1\}$, $i = 1, \ldots, s-1$; $w_j^{(t)\gamma} \in \{0, 1, 2\}$, $j = s+1, \ldots, n$. Для получения представителя первого L-класса начинаем с вектора $w^{(0)1} = (2, \ldots, 2)$. Среди координат этого вектора ищем первую по номеру координату $w_q^{(t)\gamma}$, значение которой равно 2, и изменяем его на 1. Выполняем *процедуру упрощения формулы*. В результате некоторые компоненты вектора «автоматически» примут значения 0 или 1. Обозначим полученный вектор $\tilde{w}$. После упрощения формулы возникает один из следующих случаев:

— *Все скобки исключены.* В векторе $\tilde{w}$ заменяем все координаты, равные 2, на 1/2. Полученную точку обозначаем $\tilde{y}^{(t)}$ и переходим на шаг 4 алгоритма LCE.
— *Скобки остались, противоречия не обнаружено.* Вектор $\tilde{w}$ обозначаем $w^{(t)\gamma+1}$ и переходим на $(\gamma+1)$-ю итерацию.
— *Обнаружено противоречие.* Из вектора $w^{(t)\gamma}$ строим точку $y^{(t+1)}$: если $w_j^{(t)\gamma} = 2$, то полагаем $y_j^{(t+1)} = 1/2$, $j = s+1, \ldots, n$. Таким образом, значение q-й координаты меняется на 1/2. Возвращаемся на шаг 2 алгоритма LCE.

4.2. Алгоритм для задачи максимальной выполнимости

Приведём описание точного алгоритма решения задачи максимальной выполнимости, который за конечное число итераций находит оптимальное решение задачи. Алгоритм основан на лексикографическом переборе целочисленных векторов и сведении к последовательности задач выполнимости, для решения которых используется алгоритм LCE. По сравнению с алгоритмом из работы [16] для невзвешенной задачи, в данном алгоритме содержатся дополнительные процедуры, оптимизирующие перебор указанных векторов с целью сокращения числа решаемых подзадач выполнимости.

Пусть формуле $F = C_1 \wedge \ldots \wedge C_m$ соответствует вектор $Z = (Z_1, \ldots, Z_m)$, координаты которого принадлежат множеству $\{0, 1, 2\}$. Основные процедуры алгоритма заключаются в направленном переборе указанных векторов, каждому из которых соответствует следующая задача: найти такой набор значений переменных, что скобки C_p, для которых $Z_p = 1$, становятся истинными, а скобки C_q, где $Z_q = 0$, — ложными. Если задача выполнимости, соответствующая вектору Z, разрешима, то будем называть этот вектор выполнимым, иначе — невыполнимым. Отметим, что координаты $Z_i = 2$ используются для обеспечения монотонности лексикографического перебора векторов, с этой целью можно использовать любое значение, большее 1. Пусть все скобки, входящие в формулу, предварительно упорядочены по невозрастанию их весов. При описании алгоритма будем пользоваться следующими обозначениями:

$W_0(Z)$ — суммарный вес всех скобок C_i формулы F, для которых $Z_i = 0$;

$W_1(Z)$ — суммарный вес всех скобок C_i формулы F, для которых $Z_i = 1$;

$subSAT(F, Z)$ — функция для решения специальной задачи выполнимости с помощью алгоритма LCE. Её особенность заключается в следующем. Если находится такой выполняющий набор, при котором скобки C_p (для координат $Z_p = 1$) становятся истинными, а скобки C_q (для $Z_q = 0$) — ложными, то функция принимает значение 0. В противном случае функция принимает такое значение s, что если положить $Z_i = 2, i = s, \ldots, m$, то вектор Z останется невыполнимым.

Схема точного алгоритма для MAX SAT

Шаг 0. Получаем приближённое решение $\tilde{x}$ задачи каким-нибудь эвристическим алгоритмом. Пусть $\tilde{f} = f(\tilde{x})$ — суммарный вес выполненных скобок формулы F, соответствующий набору $\tilde{x}$. Полагаем $Z_i = 2,\ i = 1, \ldots, m$, и переходим на шаг 1.

Шаг 1. Если $\tilde{f} = \sum\limits_{i=1}^{m} w_i$, алгоритм завершает работу ($\tilde{x}$ — оптимальное решение задачи); в противном случае переходим на шаг 2.

Шаг 2. Если выполняется $W_1(Z) \leqslant \tilde{f}$, то переходим на шаг 3; в противном случае — на шаг 7.

Шаг 3. Если имеет место $\sum\limits_{i=1}^{m} w_i - W_0(Z) \leqslant \tilde{f}$, то переходим на шаг 4; в противном случае — на шаг 6.

Шаг 4. Пусть $\mu = \min\{i : Z_i = 0,\ i = 1, \ldots, m\}$. Если среди координат Z_i, $i = \mu, \ldots, m$, не более одной, равной 0, то алгоритм завершает работу ($\tilde{x}$ — оптимальное решение задачи); в противном случае — на шаг 5.

Шаг 5. Находим $\nu = \max\{i : Z_i = 0,\ i = 1, \ldots, m\}$, $k = \max\{i : Z_i = 0,\ i = 1, \ldots, \nu - 1\}$ и $l = \max\{i : Z_i = 1,\ i = 1, \ldots, k - 1\}$. Полагаем $Z_l = 0$, $Z_i = 2$, $i = l + 1, \ldots, m$, и переходим на шаг 2.

Шаг 6. Пусть $r = \max\{i : Z_i \neq 2,\ i = 1, \ldots, m\}$ при $Z_1 \neq 2$ и $r = 0$ в противном случае. Полагаем $Z_{r+1} = 1$ и переходим на шаг 2.

Шаг 7. Пусть $s = subSAT(F, Z)$. Если найден выполняющий набор x (в этом случае $s = 0$), то обновляем рекордное значение целевой функции и переходим на шаг 1; в противном случае переходим на шаг 8.

Шаг 8. Если $\{i : Z_i = 1,\ i = 1, \ldots, s\} = \varnothing$, алгоритм завершает работу. В противном случае находим номер $l = \max\{i : Z_i = 1,\ i = 1, \ldots, s\}$ и полагаем $Z_l = 0$, $Z_i = 2$, $i = l + 1, \ldots, m$. Переходим на шаг 2.

Предложенные алгоритмы реализованы и проведены вычислительные эксперименты как на случайно сгенерированных задачах, так и на семействах логических формул из электронных библиотек тестовых задач. В [1, 17] приведены результаты расчётов для различных серий задач и сравнение алгоритмов с другими известными алгоритмами для задач выполнимости и максимальной выполнимости. В частности, в работе [17] алгоритм для задачи максимальной выполнимости сравнивается с основанным на методе ветвей и границ алгоритмом WMaxSatz-2.5 из библиотеки MAX-SAT Evaluation 2009. Кроме того, в указанной работе описывается возможность использования предложенного алгоритма для приближённого решения задачи максимальной выполнимости. Результаты эксперимента показали эффективность используемого подхода на ряде серий задач и целесообразность его дальнейшего развития. В настоящее время ведётся работа по применению разработанных схем алгоритмов для эффективного решения смешанных задач с логическими ограничениями.

Заключение

Метод регулярных разбиений, предложенный для анализа и решения задач целочисленного программирования, получил значительное развитие при исследовании задач о покрытии и упаковке множества, размещения, задачи о рюкзаке и других. В данной работе описаны результаты последних лет по его использованию для исследования задач выполнимости и максимальной выполнимости, которые находят широкое практическое применение во многих областях. В дальнейшем представляется целесообразным продолжение исследований задач дискретной оптимизации с логическими ограничениями на основе указанного подхода.

ЛИТЕРАТУРА

1. *Колоколов А. А., Адельшин А. В., Ягофарова Д. И.* Решение задачи выполнимости с использованием метода перебора L-классов // Информационные технологии. 2009. № 2. С. 54–59.
2. *Gu J., Purdom P., Franco J., and Wah B.* Algorithms for the satisfiability (SAT) problem: a survey // DIMACS Series in Discrete Mathematics and Theoretical Computer Science. Princeton: AMS, 1996. P. 19–152.
3. *Hansen P. and Jaumard B.* Algorithms for the maximum satisfiability problem // Computing. 1990. No. 4. P. 279–303.

4. *Thornton J., Bain S., Sattar A., and Pham D. N.* A two level local search for MAX-SAT problems with hard and soft constraints // Proc. 15th Australian Joint Conference on Artificial Intelligence, AustAI-2002. Canberra, Australia, 2002. P. 603–614.
5. *Колоколов А. А.* Регулярные разбиения и отсечения в целочисленном программировании // Сиб. журнал исследования операций. 1994. № 2. С. 18–39.
6. *Адельшин А. В.* Исследование задач максимальной и минимальной выполнимости с использованием L-разбиения // Автоматика и телемеханика. 2004. № 1. С. 35–42.
7. *Колоколов А. А., Адельшин А. В., Чередова Ю. Н.* Применение L-разбиения к исследованию некоторых задач выполнимости // Труды XII Байкальской междунар. конф. «Методы оптимизации и их приложения». Иркутск, 2001. Т. 1. С. 166–172.
8. *Cook S. A.* The complexity of theorem-proving procedure // Proc. 3rd Annual ACM Symposium on the Theory of Computing. New York: ACM, 1971. P. 151–159.
9. *Колоколов А. А., Ярош А. В.* Автоматизация проектирования сложных изделий с использованием дискретной оптимизации и информационных технологий // Омский научный вестник. 2010. № 2(90). С. 234–238.
10. *Адельшин А. В., Жовнер Е. Н.* Применение задач выполнимости логической формулы для проектирования химического состава резин // Вестник Омского университета. 2011. № 2. С. 14–18.
11. *Гуселетова О. Н., Колоколов А. А.* Решение задач дискретной оптимизации с логическими ограничениями при проектировании сложных изделий // Автоматика и телемеханика. 2008. № 10. С. 176–182.
12. *Колоколов А. А., Серышева Ю. С., Шулепова Л. Д.* Решение задач формирования малых групп с учетом межличностных отношений // Труды XV Байкальской междунар. школы-семинара «Методы оптимизации и их приложения». Иркутск, 2011. Т. 5. С. 61–66.
13. *Посыпкин М. А., Заикин О. С., Беспалов Д. В., Семенов А. А.* Решение задач криптоанализа поточных шифров в распределенных вычислительных средах // Труды ИСА. 2009. Т. 46. С. 119–137.
14. *Massacci F. and Marraro L.* Logical cryptanalysis as a SAT problem // J. Automated Reasoning. 2000. No. 24. P. 165–203.
15. *Колоколов А. А., Чередова Ю. Н.* Исследование и решение задачи о выполнимости с использованием L-разбиения // Материалы междунар. конф. «Дискретный анализ и исследование операций». Новосибирск, 2000. С. 150.
16. *Колоколов А. А., Адельшин А. В., Ягофарова Д. И.* Алгоритмы лексикографического перебора для решения задачи выполнимости и некоторых её обобщений // Труды XIII Байкальской междунар. школы-семинара «Методы оптимизации и их приложения». Иркутск, 2005. Т. 1. С. 503–508.
17. *Адельшин А. В., Кучин А. К.* Алгоритмы точного и приближенного решения задачи максимальной выполнимости // Омский научный вестник. 2011. № 1(97). С. 5–9.

2013 Вычислительные методы в дискретной математике №1(19)
DOI 10.17223/20710410/19/10
УДК 512.54

БЫСТРОЕ УМНОЖЕНИЕ ЭЛЕМЕНТОВ В КОНЕЧНЫХ ДВУПОРОЖДЁННЫХ ГРУППАХ ПЕРИОДА ПЯТЬ[1]

А. А. Кузнецов, А. С. Кузнецова

Сибирский государственный аэрокосмический университет, г. Красноярск, Россия

E-mail: alex_kuznetsov80@mail.ru

Представлен алгоритм, основанный на полиномах Холла, для быстрого умножения элементов в конечных двупорождённых группах периода пять.

Ключевые слова: *периодическая группа, собирательный процесс, полиномы Холла.*

Введение

Пусть $B_0(2,5,k)$ — максимальная конечная двупорождённая бернсайдова группа периода 5 ступени нильпотентности k. В данном классе групп наибольшей является группа $B_0(2,5,12)$, порядок которой равен 5^{34} [1]. Для каждой из $B_0(2,5,k)$ известны PC-представления (Power Commutator presentation [2, 3]), которые несложно получить, используя систему компьютерной алгебры GAP. Для исследования различных свойств $B_0(2,5,k)$, например таких, как функция роста или диаметр Кэли группы, часто требуется вычислять произведения её элементов. Пусть $a_1^{x_1}\dots a_n^{x_n}$ и $a_1^{y_1}\dots a_n^{y_n}$ — два произвольных элемента в группе $B_0(2,5,k)$, записанных в коммутаторном виде. Тогда их произведение равно

$$a_1^{x_1}\dots a_n^{x_n}\cdot a_1^{y_1}\dots a_n^{y_n}=a_1^{z_1}\dots a_n^{z_n}.$$

Основой для нахождения коэффициентов z_i является собирательный процесс [2, 3], который реализован в системах компьютерной алгебры GAP и MAGMA. Однако это не единственный способ для вычисления произведений элементов группы. В работе Ф. Холла [4] показано, что z_i представляют собой полиномиальные функции (в нашем случае над полем $\mathbb{Z}_5$), зависящие от переменных $x_1,\dots,x_i,y_1,\dots,y_i$, которые называют сейчас полиномами Холла. Согласно [4],

$$z_i=x_i+y_i+p_i(x_1,\dots,x_{i-1},y_1,\dots,y_{i-1}).$$

Ч. Симс в [5] анонсировал, что вычислил указанные полиномиальные функции, однако они так и не были нигде опубликованы. Кроме того, авторам не удалось найти компьютерные программы, в которых реализован метод полиномов Холла для вычисления произведений элементов группы. Для восполнения указанного пробела в настоящей работе приведено подробное описание алгоритма для вычисления полиномов Холла в группах $B_0(2,5,k)$. Полиномы z_i найдены при помощи машинных вычислений в системе компьютерной алгебры MATLAB, после чего полученные формулы реализованы в виде отдельной программы на языке Си. Сравнение скорости вычисления около 10^4 произведений случайно выбранных элементов в каждой из групп $B_0(2,5,k)$

[1]Работа выполнена при финансовой поддержке гранта РФФИ (код проекта № 10-01-00509-а).

показало, что метод полиномов Холла имеет значительное преимущество перед собирательным процессом. В среднем использование полиномов позволяет получить произведение двух элементов группы на порядок (т. е. в 10 раз) быстрее собирательного процесса.

При описании алгоритма ограничимся случаями $k \leqslant 4$. Для $k > 4$ полиномы вычисляются аналогично, однако их вывод занимает значительно больше места, что делает невозможным проверить доказательство «вручную». Полиномы для наибольшего случая $k = 12$ содержат также полиномы для всех других k и занимают 42 страницы текста, вследствие чего в работе не приводятся. Их можно легко получить, связавшись с авторами по электронной почте.

Из описания алгоритма, приведённого далее, следует

Теорема 1. Пусть $a_1^{x_1} \dots a_n^{x_n}$ и $a_1^{y_1} \dots a_n^{y_n}$ — два произвольных элемента в группе $B_0(2, 5, k)$, записанных в коммутаторном виде, где $k \in \mathbb{N}$ и $k \leqslant 4$. Тогда их произведение равно $a_1^{x_1} \dots a_n^{x_n} \cdot a_1^{y_1} \dots a_n^{y_n} = a_1^{z_1} \dots a_n^{z_n}$, где $z_i \in \mathbb{Z}_5$ — полиномы Холла, задаваемые формулами (1), (2) при $k = 1$, (1)–(3) при $k = 2$, (1)–(5) при $k = 3$ и (1)–(8) при $k = 4$:

$$z_1 = x_1 + y_1; \tag{1}$$

$$z_2 = x_2 + y_2; \tag{2}$$

$$z_3 = x_3 + y_3 + x_2 y_1; \tag{3}$$

$$z_4 = x_4 + y_4 + x_2 \binom{y_1}{2} + x_3 y_1; \tag{4}$$

$$z_5 = x_5 + y_5 + x_2 y_1 y_2 + \binom{x_2}{2} y_1 + x_3 y_2; \tag{5}$$

$$z_6 = x_6 + y_6 + x_2 \binom{y_1}{3} + x_3 \binom{y_1}{2} + x_4 y_1; \tag{6}$$

$$z_7 = x_7 + y_7 + x_2 \binom{y_1}{2} y_2 + \binom{x_2}{2} \binom{y_1}{2} + x_3 y_1 y_2 + x_4 y_2 + x_5 y_1; \tag{7}$$

$$z_8 = x_8 + y_8 + x_2 y_1 \binom{y_2}{2} + \binom{x_2}{2} y_1 y_2 + \binom{x_2}{3} y_1 + x_3 \binom{y_2}{2} + x_5 y_2. \tag{8}$$

Описание алгоритма

Вычислим полиномы Холла для группы $B_0(2, 5, 4)$. Работая с данной группой, автоматически получим полиномы для групп $B_0(2, 5, 1)$, $B_0(2, 5, 2)$ и $B_0(2, 5, 3)$, поэтому нет необходимости отдельно рассматривать эти случаи. При $k < 4$ коммутаторы, вес которых больше k, не рассматриваются, поскольку они равны единице группы по определению.

Запишем PC-представление группы $B_0(2, 5, 4)$ согласно [1].

Коммутаторы веса 1:

a_1, a_2 — образующие группы.

Коммутаторы веса 2:

$$a_3 = [a_2, a_1].$$

Коммутаторы веса 3:

$$a_4 = [a_3, a_1] = [a_2, a_1, a_1],$$
$$a_5 = [a_3, a_2] = [a_2, a_1, a_2].$$

Коммутаторы веса 4:

$$a_6 = [a_4, a_1] = [a_2, a_1, a_1, a_1],$$

$$a_7 = [a_5, a_1] = [a_2, a_1, a_2, a_1],$$
$$a_8 = [a_5, a_2] = [a_2, a_1, a_2, a_2].$$

Список определяющих соотношений R для базисных коммутаторов:
$a_i^5 = 1$ $(1 \leqslant i \leqslant 8)$, $[a_2, a_1] = a_3$, $[a_3, a_1] = a_4$, $[a_3, a_2] = a_5$, $[a_4, a_1] = a_6$,
$[a_4, a_2] = a_7$, $[a_4, a_3] = 1$, $[a_5, a_1] = a_7$, $[a_5, a_2] = a_8$, $[a_5, a_3] = 1$, $[a_5, a_4] = 1$,
$[a_6, a_1] = 1$, $[a_6, a_2] = 1$, $[a_6, a_3] = 1$, $[a_6, a_4] = 1$, $[a_6, a_5] = 1$, $[a_7, a_1] = 1$,
$[a_7, a_2] = 1$, $[a_7, a_3] = 1$, $[a_7, a_4] = 1$, $[a_7, a_5] = 1$, $[a_7, a_6] = 1$, $[a_8, a_1] = 1$,
$[a_8, a_2] = 1$, $[a_8, a_3] = 1$, $[a_8, a_4] = 1$, $[a_8, a_5] = 1$, $[a_8, a_6] = 1$, $[a_8, a_7] = 1$.
Таким образом,

$$B_0(2, 5, 4) = \langle\, a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8 \mid R \,\rangle.$$

Каждый элемент группы выражается единственным образом в виде нормального коммутаторного слова:

$$\forall g \in B_0(2, 5, 4) \quad g = a_1^{x_1} a_2^{x_2} a_3^{x_3} a_4^{x_4} a_5^{x_5} a_6^{x_6} a_7^{x_7} a_8^{x_8}, \quad x_i \in \mathbb{Z}_5.$$

Иногда будем писать $g = (x_1, \ldots, x_8)$.

Для того чтобы определить функции z_i, сначала необходимо вычислить произведения вида $a_j^y\, a_i^x$ для всех $1 \leqslant i < j \leqslant 8$, $x, y = 1, 2, 3, 4$. Для пары (j, i) требуется по 16 значениям произведений (y, x) найти интерполяционный полином для каждого из 8 коммутаторов.

Начнем с первого случая $a_2^y\, a_1^x$:

$$a_2^1 a_1^1 = (1,1,1,0,0,0,0,0), \quad a_2^1 a_1^2 = (2,1,2,1,0,0,0,0),$$
$$a_2^1 a_1^3 = (3,1,3,3,0,1,0,0), \quad a_2^1 a_1^4 = (4,1,4,1,0,4,0,0),$$
$$a_2^2 a_1^1 = (1,2,2,0,1,0,0,0), \quad a_2^2 a_1^2 = (2,2,4,2,2,0,1,0),$$
$$a_2^2 a_1^3 = (3,2,1,1,3,2,3,0), \quad a_2^2 a_1^4 = (4,2,3,2,4,3,1,0),$$
$$a_2^3 a_1^1 = (1,3,3,0,3,0,0,1), \quad a_2^3 a_1^2 = (2,3,1,3,1,0,3,2),$$
$$a_2^3 a_1^3 = (3,3,4,4,4,3,4,3), \quad a_2^3 a_1^4 = (4,3,2,3,2,2,3,4),$$
$$a_2^4 a_1^1 = (1,4,4,0,1,0,0,4), \quad a_2^4 a_1^2 = (2,4,3,4,2,0,1,3),$$
$$a_2^4 a_1^3 = (3,4,2,2,3,4,3,2), \quad a_2^4 a_1^4 = (4,4,1,4,4,1,1,1).$$

Запишем

$$a_2^y\, a_1^x = a_1^x\, a_2^y\, a_3^{f_3^{(1,2)}(x,y)}\, a_4^{f_4^{(1,2)}(x,y)} \ldots\, a_8^{f_8^{(1,2)}(x,y)},$$

где $f_r^{(1,2)}(x, y) = \sum\limits_{p=1}^{4} \sum\limits_{q=1}^{4} \beta_{pq}^r x^p y^q$ — некоторые полиномы над полем $\mathbb{Z}_5$. Для их определения выполним интерполяцию для каждого коммутатора $r = 3, 4, \ldots, 8$.

Для нахождения $f_r^{(1,2)}(x, y)$ требуется решить систему линейных уравнений над указанным полем:

$$\sum_{p=1}^{4} \sum_{q=1}^{4} \beta_{pq}^r x^p y^q = z_r^{yx}, \quad x, y = 1, 2, 3, 4, \tag{9}$$

где z_r^{yx} — значение r-го коммутатора для пары (y, x). Данная система имеет 16 переменных и состоит из 16 уравнений.

Покажем на примере 8-го коммутатора нахождение $f_8^{(1,2)}(x,y)$. Для краткости вместо β_{pq}^8 будем писать β_{pq}. Подставив в систему (9) все значения z_8^{yx}, получим

$$\begin{bmatrix}
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
2&4&2&3&4&2&1&3&4&2&1&3&4&1&3&1\\
3&4&3&2&4&3&1&2&4&3&1&2&4&1&2&1\\
4&1&4&4&1&4&1&4&1&4&1&4&1&1&4&1\\
2&2&4&2&4&3&2&4&3&1&4&3&1&3&1&1\\
4&3&3&1&1&1&2&2&2&2&4&4&4&3&3&1\\
1&3&2&4&1&4&2&3&2&3&4&1&4&3&2&1\\
3&2&1&3&4&2&2&1&3&4&4&2&1&3&4&1\\
3&3&4&3&4&2&3&4&2&1&4&2&1&2&1&1\\
1&2&3&4&1&4&3&2&3&2&4&1&4&2&3&1\\
4&2&2&1&1&1&3&3&3&3&4&4&4&2&2&1\\
2&3&1&2&4&3&3&1&2&4&4&3&1&2&4&1\\
4&4&1&4&1&4&4&1&4&1&1&4&1&4&1&1\\
3&1&2&2&4&3&4&3&1&2&1&2&4&4&3&1\\
2&1&3&3&4&2&4&2&1&3&1&3&4&4&2&1\\
1&4&4&1&1&1&4&4&4&4&1&1&1&4&4&1
\end{bmatrix}\cdot\begin{pmatrix}\beta_{11}\\ \beta_{21}\\ \beta_{12}\\ \beta_{31}\\ \beta_{22}\\ \beta_{13}\\ \beta_{41}\\ \beta_{32}\\ \beta_{23}\\ \beta_{14}\\ \beta_{42}\\ \beta_{33}\\ \beta_{24}\\ \beta_{43}\\ \beta_{34}\\ \beta_{44}\end{pmatrix}=\begin{pmatrix}0\\0\\0\\0\\0\\0\\0\\0\\1\\2\\3\\4\\4\\3\\2\\1\end{pmatrix}.$$

Ранг матрицы системы равен 16, поэтому она имеет единственное решение $\beta_{11}=2$, $\beta_{12}=2$, $\beta_{13}=1$, все остальные коэффициенты равны нулю. Следовательно,

$$f_8^{(1,2)}(x,y)=2xy+2xy^2+xy^3.$$

Аналогичным образом вычисляются все полиномы $f_r^{(1,2)}(x,y)$. Перечислим их:
$f_1^{(1,2)}(x,y)=x$,
$f_2^{(1,2)}(x,y)=y$,
$f_3^{(1,2)}(x,y)=xy$,
$f_4^{(1,2)}(x,y)=2xy+3x^2y=\binom{x}{2}y$,
$f_5^{(1,2)}(x,y)=2xy+3xy^2=x\binom{y}{2}$,
$f_6^{(1,2)}(x,y)=2xy+2x^2y+x^3y=\binom{x}{3}y$,
$f_7^{(1,2)}(x,y)=4xy+xy^2+x^2y+4x^2y^2=\binom{x}{2}\binom{y}{2}$,
$f_8^{(1,2)}(x,y)=2xy+2xy^2+xy^3=x\binom{y}{3}$.

Таким образом,

$$a_2^y\,a_1^x=\left(x,y,xy,\binom{x}{2}y,x\binom{y}{2},\binom{x}{3}y,\binom{x}{2}\binom{y}{2},x\binom{y}{3}\right).$$

Вторая пара $a_3^y\, a_1^x$:

$$a_3^1a_1^1=(1,0,1,1,0,0,0,0),\quad a_3^1a_1^2=(2,0,1,2,0,1,0,0),$$
$$a_3^1a_1^3=(3,0,1,3,0,3,0,0),\quad a_3^1a_1^4=(4,0,1,4,0,1,0,0),$$
$$a_3^2a_1^1=(1,0,2,2,0,0,0,0),\quad a_3^2a_1^2=(2,0,2,4,0,2,0,0),$$
$$a_3^2a_1^3=(3,0,2,1,0,1,0,0),\quad a_3^2a_1^4=(4,0,2,3,0,2,0,0),$$
$$a_3^3a_1^1=(1,0,3,3,0,0,0,0),\quad a_3^3a_1^2=(2,0,3,1,0,3,0,0),$$
$$a_3^3a_1^3=(3,0,3,4,0,4,0,0),\quad a_3^3a_1^4=(4,0,3,2,0,3,0,0),$$
$$a_3^4a_1^1=(1,0,4,4,0,0,0,0),\quad a_3^4a_1^2=(2,0,4,3,0,4,0,0),$$
$$a_3^4a_1^3=(3,0,4,2,0,2,0,0),\quad a_3^4a_1^4=(4,0,4,1,0,4,0,0);$$
$$a_3^y\, a_1^x=(x,0,y,xy,0,2xy+3x^2y,0,0).$$

Третья пара $a_3^y\, a_2^x$:

$$a_3^1a_2^1=(0,1,1,0,1,0,0,0),\quad a_3^1a_2^2=(0,2,1,0,2,0,0,1),$$
$$a_3^1a_2^3=(0,3,1,0,3,0,0,3),\quad a_3^1a_2^4=(0,4,1,0,4,0,0,1),$$
$$a_3^2a_2^1=(0,1,2,0,2,0,0,0),\quad a_3^2a_2^2=(0,2,2,0,4,0,0,2),$$
$$a_3^2a_2^3=(0,3,2,0,1,0,0,1),\quad a_3^2a_2^4=(0,4,2,0,3,0,0,2),$$
$$a_3^3a_2^1=(0,1,3,0,3,0,0,0),\quad a_3^3a_2^2=(0,2,3,0,1,0,0,3),$$
$$a_3^3a_2^3=(0,3,3,0,4,0,0,4),\quad a_3^3a_2^4=(0,4,3,0,2,0,0,3),$$
$$a_3^4a_2^1=(0,1,4,0,4,0,0,0),\quad a_3^4a_2^2=(0,2,4,0,3,0,0,4),$$
$$a_3^4a_2^3=(0,3,4,0,2,0,0,2),\quad a_3^4a_2^4=(0,4,4,0,1,0,0,4);$$
$$a_3^y\, a_2^x=(0,x,y,0,xy,0,0,2xy+3x^2y).$$

Четвёртая пара $a_4^y\, a_1^x$:

$$a_4^1a_1^1=(1,0,0,1,0,1,0,0),\quad a_4^1a_1^2=(2,0,0,1,0,2,0,0),$$
$$a_4^1a_1^3=(3,0,0,1,0,3,0,0),\quad a_4^1a_1^4=(4,0,0,1,0,4,0,0),$$
$$a_4^2a_1^1=(1,0,0,2,0,2,0,0),\quad a_4^2a_1^2=(2,0,0,2,0,4,0,0),$$
$$a_4^2a_1^3=(3,0,0,2,0,1,0,0),\quad a_4^2a_1^4=(4,0,0,2,0,3,0,0),$$
$$a_4^3a_1^1=(1,0,0,3,0,3,0,0),\quad a_4^3a_1^2=(2,0,0,3,0,1,0,0),$$
$$a_4^3a_1^3=(3,0,0,3,0,4,0,0),\quad a_4^3a_1^4=(4,0,0,3,0,2,0,0),$$
$$a_4^4a_1^1=(1,0,0,4,0,4,0,0),\quad a_4^4a_1^2=(2,0,0,4,0,3,0,0),$$
$$a_4^4a_1^3=(3,0,0,4,0,2,0,0),\quad a_4^4a_1^4=(4,0,0,4,0,1,0,0);$$
$$a_4^y\, a_1^x=(x,0,0,y,0,xy,0,0).$$

Пятая пара $a_4^y\,a_2^x$:

$$\begin{gathered}
a_4^1a_2^1=(0,1,0,1,0,0,1,0),\quad a_4^1a_2^2=(0,2,0,1,0,0,2,0),\\
a_4^1a_2^3=(0,3,0,1,0,0,3,0),\quad a_4^1a_2^4=(0,4,0,1,0,0,4,0),\\
a_4^2a_2^1=(0,1,0,2,0,0,2,0),\quad a_4^2a_2^2=(0,2,0,2,0,0,4,0),\\
a_4^2a_2^3=(0,3,0,2,0,0,1,0),\quad a_4^2a_2^4=(0,4,0,2,0,0,3,0),\\
a_4^3a_2^1=(0,1,0,3,0,0,3,0),\quad a_4^3a_2^2=(0,2,0,3,0,0,1,0),\\
a_4^3a_2^3=(0,3,0,3,0,0,4,0),\quad a_4^3a_2^4=(0,4,0,3,0,0,2,0),\\
a_4^4a_2^1=(0,1,0,4,0,0,4,0),\quad a_4^4a_2^2=(0,2,0,4,0,0,3,0),\\
a_4^4a_2^3=(0,3,0,4,0,0,2,0),\quad a_4^4a_2^4=(0,4,0,4,0,0,1,0);\\
a_4^y\,a_2^x=(0,x,0,y,0,0,xy,0).
\end{gathered}$$

Шестая пара $a_4^y\,a_3^x$. Коммутаторы a_4 и a_3 перестановочны, поэтому $a_4^y\,a_3^x=a_3^x\,a_4^y$. Теперь рассмотрим $a_5^y\,a_1^x$:

$$\begin{gathered}
a_5^1a_1^1=(1,0,0,0,1,0,1,0),\quad a_5^1a_1^2=(2,0,0,0,1,0,2,0),\\
a_5^1a_1^3=(3,0,0,0,1,0,3,0),\quad a_5^1a_1^4=(4,0,0,0,1,0,4,0),\\
a_5^2a_1^1=(1,0,0,0,2,0,2,0),\quad a_5^2a_1^2=(2,0,0,0,2,0,4,0),\\
a_5^2a_1^3=(3,0,0,0,2,0,1,0),\quad a_5^2a_1^4=(4,0,0,0,2,0,3,0),\\
a_5^3a_1^1=(1,0,0,0,3,0,3,0),\quad a_5^3a_1^2=(2,0,0,0,3,0,1,0),\\
a_5^3a_1^3=(3,0,0,0,3,0,4,0),\quad a_5^3a_1^4=(4,0,0,0,3,0,2,0),\\
a_5^4a_1^1=(1,0,0,0,4,0,4,0),\quad a_5^4a_1^2=(2,0,0,0,4,0,3,0),\\
a_5^4a_1^3=(3,0,0,0,4,0,2,0),\quad a_5^4a_1^4=(4,0,0,0,4,0,1,0);\\
a_5^y\,a_1^x=(x,0,0,0,y,0,xy,0).
\end{gathered}$$

И наконец, последняя неперестановочная пара $a_5^y\,a_2^x$:

$$\begin{gathered}
a_5^1a_2^1=(0,1,0,0,1,0,0,1),\quad a_5^1a_2^2=(0,2,0,0,1,0,0,2),\\
a_5^1a_2^3=(0,3,0,0,1,0,0,3),\quad a_5^1a_2^4=(0,4,0,0,1,0,0,4),\\
a_5^2a_2^1=(0,1,0,0,2,0,0,2),\quad a_5^2a_2^2=(0,2,0,0,2,0,0,4),\\
a_5^2a_2^3=(0,3,0,0,2,0,0,1),\quad a_5^2a_2^4=(0,4,0,0,2,0,0,3),\\
a_5^3a_2^1=(0,1,0,0,3,0,0,3),\quad a_5^3a_2^2=(0,2,0,0,3,0,0,1),\\
a_5^3a_2^3=(0,3,0,0,3,0,0,4),\quad a_5^3a_2^4=(0,4,0,0,3,0,0,2),\\
a_5^4a_2^1=(0,1,0,0,4,0,0,4),\quad a_5^4a_2^2=(0,2,0,0,4,0,0,3),\\
a_5^4a_2^3=(0,3,0,0,4,0,0,2),\quad a_5^4a_2^4=(0,4,0,0,4,0,0,1);\\
a_5^y\,a_2^x=(0,x,0,0,y,0,0,xy).
\end{gathered}$$

Все оставшиеся пары $a_j^y\,a_i^x$ перестановочны, т. е. $a_j^y\,a_i^x=a_i^x\,a_j^y$.

Таким образом, имеем полный набор соотношений для осуществления собирательного процесса в аналитическом виде:

$$a_j^y\,a_i^x=a_i^x\,a_j^y\,a_{j+1}^{f_{j+1}^{(i,j)}(x,y)}\,a_{j+2}^{f_{j+2}^{(i,j)}(x,y)}\,\ldots\,a_8^{f_8^{(i,j)}(x,y)},\ 1\leqslant i<j\leqslant 8. \tag{10}$$

Пользуясь (10), можем вычислить произведение $a_1^{x_1}\ldots a_8^{x_8}\cdot a_1^{y_1}\ldots a_8^{y_8}=a_1^{z_1}\ldots a_8^{z_8}$. После выполнения данной процедуры найдём все z_i (1)–(8).

Теорема доказана.

ЛИТЕРАТУРА

1. *Havas G., Wall G., and Wamsley J.* The two generator restricted Burnside group of exponent five // Bull. Austral. Math. Soc. 1974. No. 10. P. 459–470.
2. *Sims C.* Computation with Finitely Presented Groups. Cambridge: Cambridge University Press, 1994. 628 p.
3. *Holt D., Eick B., and O'Brien E.* Handbook of computational group theory. Boca Raton: Chapman & Hall/CRC Press, 2005. 514 p.
4. *Hall P.* Nilpotent groups: Notes of lectures given at the Canadian Mathematical Congress summer seminar, University of Alberta, 12–30 August, 1957. London: Queen Mary College, 1969.
5. *Sims C.* Fast multiplication and growth in groups // Proc. Intern. Symp. Symbolic and Algebraic Computation. New York, NY, USA, 1998. P. 165–170.

2013 Вычислительные методы в дискретной математике №1(19)

DOI 10.17223/20710410/19/11

УДК 511

О ВЕРХНЕЙ ГРАНИЦЕ ПЛОТНОСТИ ИНЪЕКТИВНЫХ ВЕКТОРОВ

Д. М. Мурин

Ярославский государственный университет им. П. Г. Демидова, г. Ярославль, Россия

E-mail: nirum87@mail.ru

Рассматривается последовательность Штерна $b_1 = 1$, $b_2 = 1$, $b_3 = 2$, $b_4 = 3$, $b_5 = 6$, $b_6 = 11$, $b_7 = 20$, $b_8 = 40\ldots$ Устанавливаются верхние и нижние границы для значений элементов b_i последовательности Штерна. В предположении, что вектор $(a_1, \ldots, a_r)$, элементы которого строятся по правилу $a_1 = b_r$, $a_2 = b_r + b_{r-1}$, …, $a_r = \sum\limits_{i=1}^{r} b_i$, является инъективным вектором с наименьшим возможным среди инъективных векторов размера r максимальным элементом, устанавливается верхняя граница плотности инъективных векторов для $r \geqslant 4$.

Ключевые слова: *плотность инъективных векторов, последовательность Штерна.*

Введение

Напомним некоторые определения.

Определение 1. Вектор $A = (a_1, \ldots, a_r)$ называется *возрастающим*, если и только если условие $a_j > a_{j-1}$ выполняется для всех j, $2 \leqslant j \leqslant r$.

Определение 2. Вектор $A = (a_1, \ldots, a_r)$ называется *инъективным*, если для любых различных подмножеств $A^*, A^{**} \subseteq \{a_1, \ldots, a_r\}$ суммы их элементов различны.

Определение 3. *Плотностью* вектора $A = (a_1, \ldots, a_r)$ называется число

$$d = \frac{r}{\log_2 \max\limits_{1 \leqslant i \leqslant r} a_i}.$$

В работе [1] рассмотрен вопрос о порядке роста числа инъективных векторов с ростом максимального элемента вектора. Среди всех инъективных векторов заданного размера r можно выделить векторы, обладающие наименьшим максимальным элементом, то есть векторы, обладающие таким максимальным элементом, что инъективных векторов с меньшим максимальным элементом не существует. В силу определения 3 эти векторы обладают наибольшей плотностью среди всех инъективных векторов размера r. В данной работе устанавливается верхняя граница для плотности инъективных векторов в предположении, что один из инъективных векторов $(a_1, \ldots, a_r)$, обладающий наименьшим максимальным элементом среди всех таких векторов, может быть построен по правилу $a_1 = b_r$, $a_2 = b_r + b_{r-1}$, …, $a_r = \sum\limits_{i=1}^{r} b_i$, где $b_1, \ldots, b_r$ — суть первые r элементов последовательности Штерна.

1. О последовательности Штерна и результатах экспериментов

В ходе вычислительных экспериментов по подсчёту числа возрастающих инъективных и сверхрастущих векторов получена информация о векторах, обладающих наименьшим среди всех векторов фиксированного размера r максимальным элементом. Во втором столбце таблицы приведены все возрастающие инъективные векторы,

обладающие наименьшим среди всех таких векторов максимальным элементом, для $1 \leqslant r \leqslant 9$.

Рассмотрим для каждого возрастающего инъективного вектора $(a_1, a_2, \ldots, a_r)$, обладающего наименьшим среди всех векторов размера r максимальным элементом, разностный вектор

$$(a_2 - a_1, \ldots, a_{r-1} - a_{r-2}, a_r - a_{r-1})$$

размера $r-1$ (третий столбец таблицы). Для каждого $2 \leqslant r \leqslant 9$ среди разностных векторов встречается вектор, образованный первыми $r-1$ элементами последовательности $b_1 = 1$, $b_2 = 1$, $b_3 = 2$, $b_4 = 3$, $b_5 = 6$, $b_6 = 11$, $b_7 = 20$, $b_8 = 40, \ldots$, кроме того, элементы вектора $(a_1, a_2, \ldots, a_r)$, соответствующего этому разностному вектору, построены по правилу $a_1 = b_r$, $a_2 = b_r + b_{r-1}$, …, $a_r = \sum\limits_{i=1}^{r} b_i$.

Инъективные векторы с наименьшим максимальным элементом и их разностные векторы

Размер вектора r	Инъективные векторы с наименьшим максимальным элементом	Разностный вектор
1	(1)	
2	(1, 2)	(1)
3	(1, 2, 4) **(2, 3, 4)**	(1, 2) **(1, 1)**
4	(3, 5, 6, 7)	(2, 1, 1)
5	(3, 6, 11, 12, 13) **(6, 9, 11, 12, 13)**	(3, 5, 1, 1) **(3, 2, 1, 1)**
6	(11, 17, 20, 22, 23, 24)	(6, 3, 2, 1, 1)
7	(20, 31, 37, 40, 42, 43, 44)	(11, 6, 3, 2, 1, 1)
8	(20, 40, 71, 77, 80, 82, 83, 84) (39, 59, 70, 77, 78, 79, 81, 84) **(40, 60, 71, 77, 80, 82, 83, 84)**	(20, 31, 6, 3, 2, 1, 1) (20, 11, 7, 1, 1, 2, 3) **(20, 11, 6, 3, 2, 1, 1)**
9	(77, 117, 137, 148, 154, 157, 159, 160, 161)	(40, 20, 11, 6, 3, 2, 1, 1)

Последовательность чисел $b_1 = 1$, $b_2 = 1$, $b_3 = 2$, $b_4 = 3$, $b_5 = 6$, $b_6 = 11$, $b_7 = 20$, $b_8 = 40, \ldots$ образована по следующему правилу: первый элемент последовательности равен 1, второй — предыдущему элементу, следующие два — сумме двух предыдущих элементов, следующие три — сумме трех предыдущих элементов и так далее:

$$b_1 = 1; \qquad b_2 = b_1;$$
$$b_3 = b_2 + b_1; \qquad b_4 = b_3 + b_2;$$
$$b_5 = b_4 + b_3 + b_2; \qquad b_6 = b_5 + b_4 + b_3; \qquad b_7 = b_6 + b_5 + b_4;$$
$$b_8 = b_7 + b_6 + b_5 + b_4;\ b_9 = b_8 + b_7 + b_6 + b_5;\ b_{10} = b_9 + b_8 + b_7 + b_6;\ b_{11} = b_{10} + b_9 + b_8 + b_7;$$
$$\ldots$$

Известно [2, с. 73 и 535], что в рекуррентном виде i-й член последовательности Штерна $b_1, b_2, \ldots$ при $i \geqslant 2$ может быть записан следующим образом:

$$b_i = \sum_{j=1}^{\left[\sqrt{2(i-2)+1/4}+1/2\right]} b_{i-j},$$

где через $[x]$ обозначена целая часть числа $x \in \mathbb{R}$.

Последовательности такого типа впервые в 1838 г. рассмотрел М. А. Штерн в работе [3]; последовательность $b_1, b_2, \ldots$ также носит его имя [4], хотя в рассмотренной Штерном последовательности сумма каждого числа слагаемых берётся только 2 раза.

Последовательность Штерна широко применяется при изучении процедур взвешенного голосования [5], а также в штрафной логике [6]. Более того, в работе [6] есть указание на то, что в 1983 г. А. Родригез [7] доказал, что не существует «свободных от коллизий» (что в нашей терминологии можно понимать как инъективных) векторов размера r, максимальный элемент которых меньше $\sum\limits_{i=1}^{r} b_i$, однако в работе [5], полностью основанной на работе [7], таких указаний нет. Тем не менее в обеих работах [5, 6] говорится о том, что А. Родригез [7] показал, что вектор $(a_1, a_2, \ldots, a_r)$, элементы которого построены по правилу $a_1 = b_r$, $a_2 = b_r + b_{r-1}$, …, $a_r = \sum\limits_{i=1}^{r} b_i$, является инъективным (и, более того, «сохраняющим большинство», то есть при $k < h$ любая сумма k его элементов меньше, чем любая сумма h его элементов, что важно для взвешенного голосования, но, вообще говоря, является дополнительным к инъективности условием).

Основываясь на результатах экспериментов, полагаем, что является справедливой следующая гипотеза, необходимая для обоснования основного результата.

Гипотеза 1. Не существует инъективного вектора размера r, максимальный элемент которого строго меньше величины $\sum\limits_{i=1}^{r} b_i$, где $b_1, \ldots, b_r$ — суть первые r элементов последовательности Штерна.

2. О плотности инъективных векторов

Прежде чем перейти к доказательству основного результата, получим верхние и нижние оценки для значений элементов последовательности Штерна. Теорема 1 показывает, что каждый следующий элемент последовательности Штерна не превосходит удвоенного предыдущего элемента.

Теорема 1. Пусть $b_1, b_2, \ldots$ — последовательность Штерна, тогда $2b_i \geqslant b_{i+1}$ для всех $i \in \mathbb{N}$.

Доказательство. Утверждение теоремы сводится к неравенству

$$\sum_{j=1}^{\left[\sqrt{2(i-2)+1/4}+1/2\right]} b_{i-j} \geqslant \sum_{j=1}^{\left[\sqrt{2(i-1)+1/4}+1/2\right]-1} b_{i-j},$$

поскольку

$$b_{i+1} = b_i + \sum_{j=2}^{\left[\sqrt{2(i-1)+1/4}+1/2\right]} b_{i+1-j} = b_i + \sum_{j=1}^{\left[\sqrt{2(i-1)+1/4}+1/2\right]-1} b_{i-j}.$$

Покажем, что $\left[\sqrt{2(x-1)+1/4}+1/2\right] - \left[\sqrt{2x+1/4}+1/2\right] + 1 \geqslant 0$ для всех $x \in \mathbb{N}$.

Сначала заметим, что для всех $x \in \mathbb{N}$

$$\sqrt{2x+1/4} > \sqrt{2(x-1)+1/4} \geqslant \sqrt{2x+1/4} - 1.$$

Поэтому $0 > \sqrt{2(x-1)+1/4}+1/2 - \sqrt{2x+1/4} - 1/2 \geqslant -1$ и

$$0 > \left[\sqrt{2(x-1)+1/4}+1/2\right] + \left\{\sqrt{2(x-1)+1/4}+1/2\right\} - \\ - \left[\sqrt{2x+1/4}+1/2\right] - \left\{\sqrt{2x+1/4}+1/2\right\} \geqslant -1,$$

где через $\{x\}$ обозначена дробная часть числа $x \in \mathbb{R}$.

Но $1 > \left\{\sqrt{2x+1/4}+1/2\right\} - \left\{\sqrt{2(x-1)+1/4}+1/2\right\} > -1$, следовательно,

$$1 > \left[\sqrt{2(x-1)+1/4}+1/2\right] - \left[\sqrt{2x+1/4}+1/2\right] > -2,$$

и так как $\left[\sqrt{2(x-1)+1/4}+1/2\right] - \left[\sqrt{2x+1/4}+1/2\right] + 1$ — целое число, то

$$1 \geqslant \left[\sqrt{2(x-1)+1/4}+1/2\right] - \left[\sqrt{2x+1/4}+1/2\right] + 1 \geqslant 0,$$

что завершает доказательство теоремы. ■

Первое следствие теоремы 1 определяет случаи, в которых следующий элемент последовательности Штерна в точности равен удвоенному предыдущему элементу.

Следствие 1. Пусть $b_1, b_2, \ldots$ — последовательность Штерна, тогда $2b_i = b_{i+1}$ при $i = n(n-1)/2 + 1$ и $n > 1$.

Доказательство. Достаточно показать, что при $i = n(n-1)/2 + 1$ и $n > 1$

$$\left[\sqrt{2(i-2)+1/4}+1/2\right] = \left[\sqrt{2(i-1)+1/4}+1/2\right] - 1.$$

С одной стороны, если $i = n(n-1)/2 + 1$, то

$$\left[\sqrt{2(i-1)+1/4}+1/2\right] - 1 = \left[\sqrt{(n-1/2)^2}+1/2\right] - 1 = n - 1.$$

Так как $(1-A)^2 = 1 - 2A + A^2 > 1 - 2A$ для $A \in \mathbb{R}$, то

$$1 - \frac{2}{(n-1/2)^2} < \left(1 - \frac{1}{(n-1/2)^2}\right)^2,$$

и при $n > 1$ выполнены неравенства

$$n - 1 \leqslant \sqrt{(n-1/2)^2 - 2} + 1/2 < n - \frac{1}{n-1/2} < n.$$

Следовательно, с другой стороны, если $i = n(n-1)/2 + 1$ и $n > 1$, то

$$\left[\sqrt{2(i-2)+1/4}+1/2\right] = \left[\sqrt{(n-1/2)^2 - 2}+1/2\right] = n - 1.$$

Следствие доказано. ■

Второе следствие из теоремы 1 дает верхнюю оценку величины i-го элемента последовательности Штерна.

Следствие 2. Пусть $b_1, b_2, \ldots$ — последовательность Штерна, тогда $2^{i-2} \geqslant b_i$ при $i \geqslant 2$.

Доказательство. Поскольку $2b_i \geqslant b_{i+1}$ для всех $i \in \mathbb{N}$, то для всех $i \geqslant 2$ выполняется $2^{i-2} = 2^{i-2}b_2 \geqslant 2^{i-3}b_3 \geqslant \ldots \geqslant b_i$. ■

Следующие леммы необходимы для получения нижней оценки величины i-го элемента последовательности Штерна. Первые две из них связаны с нижними оценками биномиальных коэффициентов вида C_{2n}^n. Третья напоминает о некоторых особенностях средних величин.

Лемма 1. Для всех $n \in \mathbb{N}$ имеет место $C_{2n}^{n} \geqslant 2^{2n-1-2^{-1}\log_2 n}$.

Доказательство. Для $n = 1$ выполняется равенство $C_2^1 = 2^{2-1-2^{-1}\log_2 1} = 2$. Рассмотрим случай $n \geqslant 2$. По формуле Стирлинга [8]

$$n! = (2\pi n)^{1/2} n^n e^{-n} e^{\theta_n/(12n)}$$

для некоторого $0 < \theta_n < 1$, поэтому

$$C_{2n}^{n} = \frac{(2n)!}{n!n!} = \frac{((2\pi(2n))^{1/2}(2n)^{2n}e^{-(2n)}e^{\theta_{1n}/(24n)}}{(2\pi n)n^{2n}e^{-2n}e^{\theta_{2n}/(6n)}} = \frac{2^{2n-C_n/n}}{(\pi n)^{1/2}} = 2^{2n-2^{-1}\log_2(\pi n)-C_n/n},$$

где $0 < \theta_{1n} < 1$; $0 < \theta_{2n} < 1$; $C_n = \log_2 e\,\dfrac{4\theta_{2n} - \theta_{1n}}{24}$.

Теперь, поскольку $\dfrac{|C_n|}{n} < \dfrac{1}{6}$ для всех $n \geqslant 2$ и $\dfrac{\log_2 \pi}{2} = 0{,}82574...$,

$$C_{2n}^{n} > 2^{2n-1-2^{-1}\log_2 n}$$

для всех $n \geqslant 2$. ■

Лемма 2. Для всех $n \in \mathbb{N}$

$$\prod_{j=1}^{n}(2 - j^{-1}) \geqslant 2^{n-1-2^{-1}\log_2 n}.$$

Доказательство.
$\prod\limits_{j=1}^{n}(2 - j^{-1}) = \frac{(2n-1)!!}{n!} = \dfrac{(2n)!}{2^n(n!)^2} = \dfrac{C_{2n}^{n}}{2^n} \geqslant \dfrac{2^{2n-1-2^{-1}\log_2 n}}{2^n} = 2^{n-1-2^{-1}\log_2 n}$. ■

Лемма 3. Пусть $A = (a_1, a_2, \ldots, a_r)$ — возрастающий вектор, тогда

$$s\frac{\sum\limits_{i=1}^{r} a_i}{r} \geqslant \sum_{i=1}^{s} a_i \text{ для всех } 1 \leqslant s \leqslant r.$$

Доказательство. Для $s = r$, очевидно, выполняется равенство.
Для $1 \leqslant s \leqslant r - 1$

$$s\frac{\sum\limits_{i=s+1}^{r} a_i}{r-s} \geqslant sa_{s+1} \geqslant \sum_{i=1}^{s} a_i,$$

поэтому $s\sum\limits_{i=s+1}^{r} a_i \geqslant r\sum\limits_{i=1}^{s} a_i - s\sum\limits_{i=1}^{s} a_i$, откуда $\sum\limits_{i=1}^{r} a_i \geqslant r\sum\limits_{i=1}^{s} a_i$. ■

Теорема 2 даёт нижнюю оценку i-го элемента последовательности Штерна.

Теорема 2. Пусть $b_1, b_2, \ldots$ — последовательность Штерна, тогда для любого натурального $l \geqslant 3$

$$b_l \geqslant 2^{l-4-2^{-1}\log_2([\sqrt{2(l-2)+1/4}-1/2])}.$$

Доказательство. Пусть $t = n(n-1)/2 + 1$ и $k = n - 1$, тогда, согласно следствию 1 из теоремы 1, $b_{t+1} = 2b_t$. Рассмотрим возрастающий вектор

$$\left(b_{t-(n-1)}, \ldots, b_{t-1}\right) = \left(b_{\frac{(n-1)(n-2)}{2}+1+1}, \ldots, b_{\frac{(n-1)(n-2)}{2}+1+n-1}\right)$$

размера $r = n - 1$. По лемме 3 для всех $1 \leqslant s \leqslant n-1$ получим

$$\sum_{j=1}^{s} b_{\frac{(n-1)(n-2)}{2}+1+j} \leqslant \frac{s}{n-1} \sum_{i=1}^{n-1} b_{t-i}$$

и, так как $\sum\limits_{i=1}^{n-1} b_{t-i} = b_t$, то для всех $1 \leqslant s \leqslant n-1$

$$b_t - \sum_{j=1}^{s} b_{\frac{(n-1)(n-2)}{2}+1+j} \geqslant \frac{n-1-s}{n-1} b_t,$$

поэтому

$$b_{t+2} = b_{t+1} + b_t + \ldots + b_{\frac{(n-1)(n-2)}{2}+3} = 3b_t + \left(b_t - b_{\frac{(n-1)(n-2)}{2}+2}\right) \geqslant 3b_t + \frac{k-1}{k} b_t,$$

$$b_{t+3} \geqslant b_{t+2} + b_{t+1} + b_t + \frac{k-2}{k} b_t \geqslant 6b_t + \frac{k-1}{k} b_t + \frac{k-2}{k} b_t,$$

$$b_{t+4} \geqslant 12b_t + 2\frac{k-1}{k} b_t + \frac{k-2}{k} b_t + \frac{k-3}{k} b_t,$$

$$\ldots$$

В случае $2 \leqslant i \leqslant n$ получаем

$$b_{t+i} \geqslant \left(3 \cdot 2^{i-2} + 2^{i-3}\frac{k-1}{k} + 2^{i-4}\frac{k-2}{k} + \ldots + 2\frac{k-(i-3)}{k} + \frac{k-(i-2)}{k} + \frac{k-(i-1)}{k}\right) b_t =$$

$$= \left(2^i + 2^{i-3}\frac{-1}{k} + 2^{i-4}\frac{-2}{k} + \ldots + 2\frac{-(i-3)}{k} + \frac{-(i-2)}{k} + \frac{-(i-1)}{k}\right) b_t =$$

$$= \left(2^i - 2^{i-3}\frac{1}{k}\left(\sum_{j=1}^{i-2}\frac{j}{2^{j-1}} + \frac{(i-1)}{2^{i-3}}\right)\right) b_t \geqslant \left(2^i - 2^{i-3}\frac{1}{k}\left(\sum_{j=1}^{i-1}\frac{j}{2^{j-1}} + \frac{(i-1)}{2^{i-2}}\right)\right) b_t \geqslant$$

$$\geqslant \left(2^i - 2^{i-3}\frac{1}{k}\sum_{j=1}^{\infty}\frac{j}{2^{j-1}}\right) b_t = \left(2^i - 2^{i-1}k^{-1}\right) b_t = 2^{i-1}\left(2 - k^{-1}\right) b_t,$$

так как

$$S = \sum_{j=1}^{\infty}\frac{j}{2^{j-1}} = \sum_{j=1}^{\infty}\frac{1}{2^{j-1}} + \sum_{l=1}^{\infty}\frac{l-1}{2^{l-1}} = 2 + 2^{-1}\sum_{j=1}^{\infty}\frac{j}{2^{j-1}} = 2 + 2^{-1}S.$$

Таким образом, $b_{t+1} = 2b_t$ для $t = n(n-1)/2 + 1$ и $b_{t+i} \geqslant 2^{i-1}(2 - (n-1)^{-1})b_t$ для $2 \leqslant i \leqslant n$. Из этого следует, что

$$b_{t+1} = 2b_t \geqslant 2 \cdot 2^{n-2}\left(2 - (n-2)^{-1}\right) b_{t-(n-1)} \geqslant 2\prod_{j=1}^{n-2}\left(2^j\left(2 - j^{-1}\right)\right) b_2,$$

и по лемме 2

$$2\prod_{j=1}^{n-2}\left(2^j\left(2 - j^{-1}\right)\right) b_2 = 2^{(n-2)(n-1)/2+1} \cdot 2^{n-3-2^{-1}\log_2(n-2)} \geqslant 2^{t+1-4-2^{-1}\log_2(n-2)}.$$

При $2 \leqslant i \leqslant n$

$$b_{t+i} \geqslant 2^{i-1}\left(2 - (n-1)^{-1}\right) b_t \geqslant 2^{i-1}\left(2 - (n-1)^{-1}\right) 2^{n-2}\left(2 - (n-2)^{-1}\right) b_{t-(n-1)} \geqslant$$

$$\geqslant 2^{i-1}\prod_{j=1}^{n-1}\left(2^j\left(2 - j^{-1}\right)\right) b_2 = 2^{i-1}2^{(n-2)(n-1)/2}\prod_{j=1}^{n-1}\left(2 - j^{-1}\right),$$

и по лемме 2

$$2^{i-1}2^{(n-2)(n-1)/2}\prod_{j=1}^{n-1}\left(2-j^{-1}\right)=2^{t+i-4-2^{-1}\log_2(n-1)}.$$

Пусть $l=t+i$, где $1\leqslant i\leqslant n$, тогда при $l\geqslant 2$

$$n=\max\left\{j\in\mathbb{N}:\frac{j(j-1)}{2}+2\leqslant l\right\}=\max\left\{j\in\mathbb{N}:(j-1/2)^2\leqslant 2(l-2)+1/4\right\}=$$
$$=\max\left\{j\in\mathbb{N}:j\leqslant\sqrt{2(l-2)+1/4}+1/2\right\}=\left[\sqrt{2(l-2)+1/4}+1/2\right].$$

Следовательно, при $l\geqslant 3$

$$b_l=b_{t+i}\geqslant 2^{t+i-4-2^{-1}\log_2(n-1)}=2^{l-4-2^{-1}\log_2([\sqrt{2(l-2)+1/4}+1/2]-1)}=$$
$$=2^{l-4-2^{-1}\log_2([\sqrt{2(l-2)+1/4}-1/2])}.$$

Теорема доказана. ■

Наконец, теорема 3 устанавливает верхнюю границу для плотности инъективных векторов.

Теорема 3. При условии, что гипотеза 1 является верной, плотность инъективных векторов размера $r\geqslant 4$ удовлетворяет неравенству

$$d_{in}\leqslant\frac{r}{r-3-2^{-1}\log_2([\sqrt{2(r-2)+1/4}-1/2])}.$$

Доказательство. Для возрастающих инъективных векторов размера r при условии, что гипотеза 1 верна, выполнено

$$a_r=\sum_{i=1}^{r}b_i\geqslant 2+\sum_{i=3}^{r}2^{i-4-2^{-1}\log_2([\sqrt{2(i-2)+1/4}-1/2])}\geqslant 2+2^{-4-2^{-1}\log_2([\sqrt{2(r-2)+1/4}-1/2])}\sum_{i=3}^{r}2^i=$$
$$=2+2^{-1-2^{-1}\log_2([\sqrt{2(r-2)+1/4}-1/2])}(2^{r-2}-1)\geqslant 2^{r-3-2^{-1}\log_2([\sqrt{2(r-2)+1/4}-1/2])}.$$

Обратим внимание на то, что в данном случае

$$\log_2 a_r\geqslant r-3-2^{-1}\log_2([\sqrt{2(r-2)+1/4}-1/2]),$$

но для вычисления верхней границы плотности необходимо потребовать, чтобы выполнялось неравенство

$$r-3-2^{-1}\log_2([\sqrt{2(r-2)+1/4}-1/2])>0,$$

поскольку $d_{in}>0$. Это условие выполнено при $r\geqslant 4$.

Из этого следует, что при $r\geqslant 4$

$$d_{in}=\frac{r}{\max\limits_{1\leqslant i\leqslant r}\log_2 a_i}=\frac{r}{\log_2 a_r}\leqslant\frac{r}{r-3-2^{-1}\log_2([\sqrt{2(r-2)+1/4}-1/2])}.$$

Теорема доказана. ■

Заключение

Возможно, кому-то покажется более интересной следующая граница, легко получаемая из теоремы 3.

Следствие 3. При условии, что гипотеза 1 является верной, плотность инъективных векторов размера $r \geqslant 4$ удовлетворяет неравенству

$$d_{in} \leqslant \frac{r}{r - 3{,}25 - 2^{-2}\log_2(r-2)}.$$

Заметим, что границы, приведенные в теореме 2 и следствии 3, на наш взгляд, улучшают встречающуюся, например в [9], границу $r/(r - \log_2 r)$, поскольку при $r \geqslant 20$ справедливо неравенство

$$\frac{r}{r - 3{,}25 - 2^{-2}\log_2(r-2)} < \frac{r}{r - \log_2 r}.$$

ЛИТЕРАТУРА

1. *Мурин Д. М.* О порядке роста числа инъективных и сверхрастущих рюкзачных векторов // Моделирование и анализ информационных систем. 2012. Т. 19. № 3. С. 103–115.
2. *Кнут Д.* Искусство программирования. Т. 1. М.: Издательский дом «Вильямс», 2008.
3. *Stern M. A.* Aufgaben // J. Reine Angew. Math. 1838. No. 18. P. 100.
4. *Sloane N. J. A. and Plouffe S.* The Encyclopedia of Integer Sequences. San Diego: Academic Press, 1995.
5. *Kreweras G.* Sur quelques problemes relatifs au vote pondere [Some problems of weighted voting] // Math. Sci. Humaines. 1983. No. 84. P. 45–63.
6. *Chetcuti-Sperandio N. and Lagrue S.* How to choose weightings to avoid collisions in a restricted penalty logic // Principles of Knowledge Representation and Reasoning: Proc. 11-th International Conf. Sydney: AAAI Press, 2008. P. 340–347.
7. *Rodriguez A. É*tude des propriétés d'une suite numérique liée à un probléme de vote pondéré // Thèse de docteur-ingénieur, Université Pierre et Marie Curie. 1983.
8. *Фихтенгольц Г. М.* Курс дифференциального и интегрального исчисления. Т. 2. М.: Физматлит, 2001.
9. *Николенко С.* Криптография и решетки. `http://logic.pdmi.ras.ru/~sergey/teaching/cscryp09/05-lattices.pdf`. 2009.

СВЕДЕНИЯ ОБ АВТОРАХ

АДЕЛЬШИН Александр Владимирович — доцент, кандидат физико-математических наук, старший научный сотрудник Омского филиала Института математики им. С. Л. Соболева СО РАН, г. Омск. E-mail: **adelshin@ofim.oscsbras.ru**

БАДЕХА Иван Александрович — сотрудник Российского государственного социального университета, г. Москва. E-mail: **ivanbadekha@gmail.com**

БИЛЯК Иван Богданович — г. Москва. E-mail: **bil-ib@mail.ru**

КОЛОКОЛОВ Александр Александрович — профессор, доктор физико-математических наук, заведующий лабораторией Омского филиала Института математики им. С. Л. Соболева СО РАН, г. Омск. E-mail: **kolo@ofim.oscsbras.ru**

КОЛОМЕЕЦ Николай Александрович — аспирант Института математики СО РАН, г. Новосибирск. E-mail: **nkolomeec@gmail.com**

КУЗНЕЦОВ Александр Алексеевич — доцент, доктор физико-математических наук, заведующий кафедрой высшей математики Сибирского государственного аэрокосмического университета им. акад. М. Ф. Решетнева, г. Красноярск.
E-mail: **alex_kuznetsov80@mail.ru**

КУЗНЕЦОВА Александра Сергеевна — аспирантка кафедры высшей математики Сибирского государственного аэрокосмического университета им. акад. М. Ф. Решетнева, г. Красноярск. E-mail: **alexakulch@rambler.ru**

ЛОСЕВ Александр Сергеевич — кандидат физико-математических наук, младший научный сотрудник Института прикладной математики ДВО РАН, г. Владивосток. E-mail: **alexax@bk.ru**

МУРИН Дмитрий Михайлович — аспирант Ярославского государственного университета им. П. Г. Демидова, г. Ярославль. E-mail: **nirum87@mail.ru**

ОСИПОВА Марина Анатольевна — доцент, кандидат физико-математических наук, научный сотрудник Института прикладной математики ДВО РАН, г. Владивосток.
E-mail: **mao1975@list.ru**

СЕМЕНОВА Наталья Андреевна — аспирантка Московского государственного института электроники и математики (ТУ), г. Москва.
E-mail: **natasha_sem@inbox.ru**

СМОЛЬЯНИНОВ Владимир Юрьевич — научный сотрудник Учебно-методического объединения по информационной безопасности, г. Москва.
E-mail: **VladimirSmall@gmail.com**

ТИМЕРЯЕВ Тимофей Валерьевич — аспирант Уфимского государственного авиационного технического университета, г. Уфа. E-mail: **timeryaev@yandex.ru**

УРАКОВ Айрат Ренатович — кандидат физико-математических наук, доцент Уфимского государственного авиационного технического университета, г. Уфа.
E-mail: **urakov@ufanet.ru**

ЦИЦИАШВИЛИ Гурами Шалвович — профессор, доктор физико-математических наук, заместитель директора Института прикладной математики ДВО РАН, г. Владивосток. E-mail: **guram@iam.dvo.ru**

ШОЛОМОВ Лев Абрамович — профессор, доктор физико-математических наук, главный научный сотрудник Института системного анализа РАН, г. Москва.
E-mail: **sholomov@isa.ru**

ЯГОФАРОВА Дарья Ивановна — доцент, кандидат физико-математических наук, доцент Омского государственного университета им. Ф. М. Достоевского, г. Омск.
E-mail: **ydarya@mail.ru**

АННОТАЦИИ СТАТЕЙ НА АНГЛИЙСКОМ ЯЗЫКЕ

Bilyak I. B. **ESTIMATES FOR THE NUMBER OF ELEMENT APPEARANCES IN SEGMENTS OF LINEAR RECURRENT SEQUENCES.** Trigonometric sums of linear recurrent sequences are considered. They are studied by using the Sidelnikov's method. Based on these results, estimates for the number of element appearance are obtained. In some occurrences, these estimates specify earlier known results.
Keywords: *exponential sums, linear recurrences, the number of element appearences.*

Kolomeec N. A. **AN UPPER BOUND FOR THE NONLINEARITY OF SOME BOOLEAN FUNCTIONS WITH MAXIMAL POSSIBLE ALGEBRAIC IMMUNITY.** An estimation for the nonlinearity of Dalai's Boolean functions with the maximal algebraic immunity in even number of variables is given. It is proved that the estimation is achieved.
Keywords: *Boolean functions, nonlinearity, algebraic immunity.*

Sholomov L. A. **BINARY REPRESENTATIONS OF UNDERDETERMINED DATA AND SUPERIMPOSED CODES.** For underdetermined data, binary representations are defined, making it possible to reconstruct the initial data entirely (not only their specifications) and being fairly compact. To solve the problem of their design, some special matrices called selective ones are introduced and studied. They generalize the disjunct (cover-free) matrices widely applied in Computer Science. Some characteristics of selective matrices and estimates of data representation length via some data parameters are investigated. Problems related to the complexity of representations design are considered too.
Keywords: *underdetermined data, compression, binary representation, sets system basis, representation length, disjunct matrix, superimposed code, cover-free family, polynomial algorithm.*

Semenova N. A. **REPRESENTATION OF SEMANTIC ROLE BASED ACCESS CONTROL SYSTEM BY A COLORED PETRI NET.** This paper describes the process of updating Semantic RBAC system by several administrators concurrently. Processes of system transition between states are represented by a Petri net. Secure transitions are defined, and security conditions are proved for each transition type.
Keywords: *RBAC, automated role management.*

Smolyaninov V. Y. **TRANSFORMATION RULES FOR STATES IN THE DBMS DP-MODEL.** In this article, the DBMS DP-model of access control is introduced for relational database management systems (abbreviated DBMS). For taking into account an access control features in a databases, the model includes procedures, triggers, access rights specific for databases, access rights inheritance and context dynamic change when activating procedures and triggers. The claim about granting access rights to the procedures and containers is proved.
Keywords: *computer security, DBMS DP-model, database management system.*

Badekha I. A. **EDGE CLIQUE COVERINGS OF GRAPHS.** In the article, the following problems connected with the edge clique coverings of graphs are investigated: the

structure of graphs whose edges can be covered by cliques with only one way; the possibility of graph reducing with the order decrease not changing the edge clique covering structure; algorithms for construction of minimal edge clique coverings; the interrelation between isomorphisms of graphs and corresponding reduced graphs.
Keywords: *graphs, edge clique covering, graph isomorphism, NP-complete problems.*

Urakov A. R., Timeryaev T. V. **ALL-PAIRS SHORTEST PATHS ALGORITHM FOR HIGH-DIMENSIONAL SPARSE GRAPHS.** All-pairs shortest path problem on weighted undirected sparse graphs is considered. "Disassembly and assembly of graph" algorithm is proposed to obtain the solution of the problem for the given graph using the solution of the problem for a small-dimensional graph. The algorithm is compared with one of the fastest classic algorithms on public source data.
Keywords: *APSP, graph disassembly, graph assembly, sparse graphs.*

Tsitsiashvili G. Sh., Osipova M. A., Losev A. C. **ASYMPTOTICS FOR CONNECTIVITY PROBABILITY OF GRAPH WITH LOW RELIABLE ARCS.** Asymptotics of connectivity probabilities for complete graphs with the low reliable arcs and for all pairs of nodes in them are constructed. Parameters of these asymptotics are characteristics of spanning trees and shortest paths. The calculation of the spanning trees characteristics is based on the Kirchhoff — Trent theorem. Modifications of classical algorithms are applied to calculate the characteristics of shortest paths.
Keywords: *spanning tree, Kirchhoff's matrix, shortest path, connectivity probability, calculation complexity.*

Kolokolov A. A., Adelshin A. V., Yagofarova D. I. **STUDY OF DISCRETE OPTIMIZATION PROBLEMS WITH LOGICAL CONSTRAINTS BASED ON REGULAR PARTITIONS.** Some discrete optimization problems with logical constraints are considered, and the possibility of applying these problems in complex products design is investigated. The results of the studying these problems with the use of the integer programming models and regular partitions approach are reviewed. The structure and the complexity of the problems are analysed, and the algorithms for SAT and MAX-SAT problems based on this approach are proposed.
Keywords: *satisfiability problem, logical constraints, integer programming, L-class enumeration.*

Kuznetsov A. A., Kuznetsova A. S. **FAST MULTIPLICATION IN FINITE TWO-GENERATED GROUPS OF EXPONENT FIVE.** An algorithm for fast multiplication of elements in finite two-generated groups of exponent five based on Hall's polynomials is described.
Keywords: *periodical groups, gathering process, Hall's polynomials.*

Murin D. M. **ON THE UPPER BOUND FOR THE DENSITY OF ANY INJECTIVE VECTOR.** In this work, the Stern's sequence $b_1 = 1$, $b_2 = 1$, $b_3 = 2$, $b_4 = 3$, $b_5 = 6$, $b_6 = 11$, $b_7 = 20$, $b_8 = 40, \ldots$ is considered, and the upper and lower bounds for b_i are determined. Supposing that the vector $(a_1, \ldots, a_r)$, where $r \geqslant 4$, $a_1 = b_r$, $a_2 = b_r + b_{r-1}$, $\ldots$, $a_r = \sum\limits_{i=1}^{r} b_i$, is the injective one having the least maximum element among all other injective vectors of length r, the upper bound for density of any injective vector is stated.
Keywords: *density of injective vector, Stern's sequence.*

Журнал «Прикладная дискретная математика» включен в перечень ВАК рецензируемых российских журналов, в которых должны быть опубликованы основные результаты диссертаций, представляемых на соискание учёной степени кандидата и доктора наук, а также в перечень журналов, рекомендованных УМО в области информационной безопасности РФ в качестве учебной литературы по специальности «Компьютерная безопасность».

Журнал «Прикладная дискретная математика» распространяется по подписке; его подписной индекс 38696 в объединённом каталоге «Пресса России». Полнотекстовые электронные версии вышедших номеров журнала доступны на его сайте vestnik.tsu.ru/pdm и на Общероссийском математическом портале www.mathnet.ru. На сайте журнала можно найти также и правила подготовки рукописей статей в журнал.

Тематика публикаций журнала:

- *Теоретические основы прикладной дискретной математики*
- *Математические методы криптографии*
- *Математические методы стеганографии*
- *Математические основы компьютерной безопасности*
- *Математические основы надёжности вычислительных и управляющих систем*
- *Прикладная теория кодирования*
- *Прикладная теория автоматов*
- *Прикладная теория графов*
- *Логическое проектирование дискретных автоматов*
- *Математические основы информатики и программирования*
- *Вычислительные методы в дискретной математике*
- *Дискретные модели реальных процессов*
- *Математические основы интеллектуальных систем*
- *Исторические очерки по дискретной математике и её приложениям*