МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПРИКЛАДНАЯ ДИСКРЕТНАЯ МАТЕМАТИКА

Научный журнал

2013 №4(22)

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

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

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 11.12.2013.
Формат $60 \times 84\frac{1}{8}$. Усл. п. л. 12,33. Уч.-изд. л. 13,83. Тираж 300 экз.

Издательство ТГУ. 634029, Томск, ул. Никитина, 4

СОДЕРЖАНИЕ

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ПРИКЛАДНОЙ ДИСКРЕТНОЙ МАТЕМАТИКИ

МАТЕМАТИЧЕСКИЕ ОСНОВЫ КОМПЬЮТЕРНОЙ БЕЗОПАСНОСТИ

ПРИКЛАДНАЯ ТЕОРИЯ ГРАФОВ

ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ В ДИСКРЕТНОЙ МАТЕМАТИКЕ

ДИСКРЕТНЫЕ МОДЕЛИ РЕАЛЬНЫХ ПРОЦЕССОВ

CONTENTS

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ПРИКЛАДНОЙ ДИСКРЕТНОЙ МАТЕМАТИКИ

DOI 10.17223/20710410/22/1

УДК 519.7

ПОДСТАНОВКИ, ИНДУЦИРОВАННЫЕ РАЗРЯДНО-ИНЪЕКТИВНЫМИ ПРЕОБРАЗОВАНИЯМИ МОДУЛЯ НАД КОЛЬЦОМ ГАЛУА

А. В. Аборнев

ООО «Центр сертификационных исследований», г. Москва, Россия

E-mail: abconf.c@gmail.com

Для произвольного кольца Галуа $R = \mathrm{GR}(q^2, p^2)$, $q = p^r$, построен большой класс $m \times 2m$-матриц над R, называемых разрядно-инъективными (РИ-матрицами), которым соответствует нелинейная подстановка π на модуле R^m. В качестве криптографической характеристики таких подстановок изучаются свойства множества $\Sigma\pi$, где Σ — регулярное представление группы $(R^m, +)$ в симметрической группе $S(R^m)$. В случаях $R = \mathrm{GR}(q^2, p^2)$, $p > 2$, $m = 1$ и $R = \mathrm{GR}(q^2, 4)$, $m > 1$ описаны классы разрядно-инъективных матриц с минимально возможным показателем 2-транзитивности множества подстановок $\Sigma\pi$ равным 4. В случае $R = \mathbb{Z}_{p^2}$, $m = 1$ показано также, что группа, порождённая множеством $\Sigma\pi$, содержит знакопеременную группу подстановок.

Ключевые слова: *разрядно-инъективная матрица, РИ-матрица, подстановка, кольцо Галуа.*

Введение

Пусть $R = \mathrm{GR}(q^2, p^2)$ — кольцо Галуа с полем вычетов $\overline{R} = R/pR = \mathrm{GF}(q)$, $q = p^r$. В частности, при $r = 1$ имеем $R = \mathbb{Z}_{p^2}$. Подмножество $P = \Gamma(R) = \{a \in R : a^q = a\}$ называют *p-адическим координатным множеством* или *координатным множеством Тейхмюллера* кольца R. Будем называть его также *разрядным* множеством.

Каждый элемент $a \in R$ однозначно представляется в виде

$$a = a_0 + pa_1, \quad a_i \in P, \quad i \in \{0, 1\},$$

называемом *p-адическим разложением элемента a*. Отображения

$$\gamma_i \colon R \to P, \quad \gamma_i(a) = a_i, \quad i \in \{0, 1\},$$

будем называть *разрядными функциями в разрядном множестве* P, а элементы $a_i = \gamma_i(a)$ — *p-адическими разрядами элемента a*. Алгебра $(P, \oplus, \cdot)$ с операцией сложения $a \oplus b = \gamma_0(a + b)$, $a, b \in P$, является полем.

Понятия p-адического разложения и значения функции γ_i естественным образом (поэлементно) распространяются на матрицы $A \in R_{m \times m}$, при этом используется обозначение $A_i = \gamma_i(A)$. Так же естественным образом операции $\oplus$ и $\cdot$ распространяются на матрицы и векторы над полем P, при этом операция умножения матриц обозначается через $\odot$.

Назовём матрицу W размеров $m \times n$ над кольцом Галуа R *разрядно-инъективной* (*РИ-матрицей*), если любая ненулевая строка $\mathbf{a} \in R^m$ однозначно восстанавливается по строке $\gamma_1(\mathbf{a}W)$.

Каждой РИ-матрице соответствует эффективно реализуемая нелинейная подстановка на модуле ${}_R R^m$, определяемая равенством

$$\pi(\mathbf{x}) = \gamma_1(\mathbf{x}W)Z,$$

где $Z = (E \mid pE)^T$ — $2m \times m$-матрица над R.

В работе построен большой класс РИ-матриц вида

$$K = U(E \mid E + 2G), \tag{1}$$

где $U, G \in R_{m,m}$ — обратимые матрицы. Соответствующую матрице K подстановку на модуле R^m обозначим через π.

В качестве криптографического приложения изучается блочный шифр, реализующий подстановки из множества $(\Sigma\pi)^k$, где Σ — регулярная подгруппа группы $S(R^m)$. Изучаются следующие характеристики, определяющие криптографические свойства данного шифра:

1) показатель 2-транзитивности $d_2(\Sigma\pi)$ — минимальное k, для которого множество $(\Sigma\pi)^k$ 2-транзитивно;
2) группа, порождаемая множеством подстановок $\Sigma\pi$.

В случаях $R = \mathrm{GR}(q^2, p^2)$, $p > 2$, $m = 1$ и $R = \mathrm{GR}(q^2, 4)$, $m > 1$ описаны классы РИ-матриц с оптимальным значением показателя 2-транзитивности соответствующего множества подстановок $\Sigma\pi$, то есть минимального $k \in \mathbb{N}$, для которого множество $(\Sigma\pi)^k$ 2-транзитивно. Для кольца $R = \mathbb{Z}_{p^2}$ и циклической группы Σ показано, что группа $\langle\Sigma\pi\rangle$, порождённая множеством $\Sigma\pi$, содержит знакопеременную группу подстановок.

1. Построение разрядно-инъективных матриц

Из определения РИ-матрицы $K_{m\times n}$ следует, что её размеры должны удовлетворять условию $n \geqslant 2m$. Профессором А. А. Нечаевым предложен следующий способ построения большого класса РИ-матриц размера $m \times 2m$.

Теорема 1. Если $U_{m\times m}$ — обратимая матрица над кольцом R, $G_{m\times m}$ — обратимая матрица над полем P, то матрица $W_{m\times 2m}$ над кольцом R вида

$$W = U(E \mid E + pG) = (U \mid V) \tag{2}$$

является разрядно-инъективной.

Доказательство. Покажем, что для любой ненулевой строки $\mathbf{x} \in R^m$ по строке $\mathbf{y}$, равной

$$\mathbf{y} = \gamma_1(\mathbf{x}W), \tag{3}$$

можно однозначно восстановить строку $\mathbf{x}$.

Приведём алгоритм восстановления. Подставляя (2) в (3), получаем

$$\mathbf{y} = (\gamma_1(\mathbf{x}U) \mid \gamma_1(\mathbf{x}V)) = (\gamma_1(\mathbf{x}_0U_0) \oplus \mathbf{x}_1 \odot U_0 \oplus \mathbf{x}_0 \odot U_1 \mid \gamma_1(\mathbf{x}_0V_0) \oplus \mathbf{x}_1 \odot V_0 \oplus \mathbf{x}_0 \odot V_1). \tag{4}$$

Так как ввиду (2) справедливы равенства $V_0 = U_0$, $V_1 = U_1 \oplus (U_0 \odot G)$, то из (4) имеем

$$\gamma_1(\mathbf{x}V) \ominus \gamma_1(\mathbf{x}U) = \mathbf{x}_0 \odot U_0 \odot G.$$

Таким образом по строке $\mathbf{y}$ однозначно восстанавливается строка $\mathbf{x}_0$:

$$\mathbf{x}_0 = (\gamma_1(\mathbf{x}V) \ominus \gamma_1(\mathbf{x}U)) \odot G^{-1} \odot U_0^{-1}.$$

После этого по строке $\mathbf{y}$ и известным $U, \mathbf{x}_0$ однозначно восстанавливается из (4) строка $\mathbf{x}_1$: $\mathbf{x}_1 = (\gamma_1(\mathbf{x}U) \ominus \gamma_1(\mathbf{x}_0U_0) \ominus \mathbf{x}_0U_1) \odot U_0^{-1}$. ■

Расширить класс разрядно-инъективных матриц позволяет

Утверждение 1. Пусть $W_{m\times n}$ — разрядно-инъективная матрица над кольцом R, $D = \mathrm{diag}(c_1, \ldots, c_n)$ — диагональная обратимая матрица над полем P, Π — подстановочная матрица размеров $n \times n$. Тогда матрицы K_1 и K_2, заданные равенствами

$$K_1 = WD, \quad K_2 = W\Pi,$$

являются разрядно-инъективными.

Доказательство. Пусть $\mathbf{x} = (x_1, \ldots, x_m)$ — произвольная ненулевая строка над кольцом R. Для доказательства утверждения достаточно доказать равенства

$$\gamma_1(\mathbf{x}D) = \gamma_1(\mathbf{x}) \odot D; \tag{5}$$
$$\gamma_1(\mathbf{x}\Pi) = \gamma_1(\mathbf{x}) \odot \Pi. \tag{6}$$

Рассмотрим i-й элемент h_i строки $\gamma_1(\mathbf{x}D)$: $h_i = \gamma_1(x_ic_i)$. Воспользовавшись свойствами функции γ_1, получаем

$$h_i = \gamma_1(x_i)\gamma_0(c_i) = \gamma_1(x_i)c_i = (\gamma_1(\mathbf{x}) \odot D)_i.$$

Равенство (5) доказано. Равенство (6) очевидно. ■

Следствие 1. Матрица $K_{m\times 2m} = WD\Pi$ над кольцом R (W, D, Π определены выше) является разрядно-инъективной.

2. Алгоритм блочного шифрования

Приведём описание криптографического примитива, построенного на основе разрядно-инъективной матрицы:

1) $U_{m\times m}$ — обратимая матрица над кольцом R;
2) $G_{m\times m}$ — обратимая матрица над полем P;
3) $K = U(E \mid E + pG)$ — матрица размера $m \times 2m$ над кольцом R;
4) $Z = (E \mid pE)^T$, где E — единичная матрица размера $m \times m$[1];
5) $\mathbf{c}_0, \mathbf{c}_1, ..., \mathbf{c}_{n-1} \in R^m$ — набор раундовых ключей;
6) $\mathbf{x} = \mathbf{x}_0$ — открытый текст;
7) раундовое преобразование определим равенством $\pi(\mathbf{x}) = \gamma_1(\mathbf{x}K)Z$. Тогда преобразование текста на i-м шаге, $i \in \{0, \ldots, n-1\}$, задается формулой

$$\mathbf{x}^{(i+1)} = \gamma_1((\mathbf{x}^{(i)} + \mathbf{c}_i)K)Z = \pi(\mathbf{x}^{(i)} + \mathbf{c}_i). \tag{7}$$

В качестве криптографического примитива рассматривается преобразование открытого текста $\mathbf{x}$ путём n-кратного применения преобразований (7).

[1]Заметим, что умножение произвольного вектора $\mathbf{x} = (\mathbf{x}' \mid \mathbf{x}'') \in P^{2m}$ на матрицу Z даёт вектор $\mathbf{x}' + p\mathbf{x}'' \in R^m$.

3. Криптографические свойства примитива

Пусть $\Sigma = (\Omega, \cdot)$ — регулярное представление группы $(R^m, +)$, $|\Omega| = N$ и подстановка $\pi : R^m \to R^m$ задана равенством

$$\pi(\mathbf{x}) = \gamma_1(\mathbf{x}K)Z. \tag{8}$$

Далее считается, что раундовые ключи $\mathbf{c}_0, \ldots, \mathbf{c}_{n-1}$ выбираются независимо, случайно и равновероятно. Тогда с помощью описанного выше криптографического примитива нетрудно реализовать подстановки из множества $(\Sigma\pi)^k$. Свойства данного множества, согласно [1], определяют криптографические характеристики изучаемого примитива.

Показателем 2-транзитивности $d_2(\Sigma\pi)$ называют минимальную степень k, в которой множество $(\Sigma\pi)^k$ является 2-транзитивным. Интерес представляют подстановки с минимальным значением данного параметра. Известно [1, 2], что для произвольной подстановки $g \in S(\Omega)$ верна оценка $d_2(\Sigma g) \geqslant 3$. Заметим, что подстановочная матрица $\mathcal{P}_2(\pi)$ не влияет на показатель 2-транзитивности и скорость сходимости последовательности степеней матриц переходных вероятностей биграмм к предельной.

Согласно [1], достаточно рассмотреть матрицу Q_π переходов ненулевых разностей биграмм вида

$$(Q_\pi)_{N-1\times N-1} = \frac{1}{N}(\nu_{\mathbf{ab}})_{\mathbf{a},\mathbf{b}\in R^m\setminus\mathbf{0}},$$

где $\nu_{\mathbf{ab}} = |\{\mathbf{x} \in R^m : \pi(\mathbf{x}+\mathbf{a}) - \pi(\mathbf{x}) = \mathbf{b}\}|$. Заметим, что для нахождения показателя 2-транзитивности достаточно описать степени матрицы Q_π, поскольку $d_2(\Sigma\pi) = k$ тогда и только тогда, когда $Q_\pi^{k-1} > 0$.

Для оценки элементов $\nu_{\mathbf{ab}}$ матрицы Q_π используем

Утверждение 2. Элемент $\nu_{\mathbf{ab}}$ для матрицы Q_π равен числу решений системы нелинейных уравнений

$$\begin{cases} \mathbf{b}_0 = \gamma_1(\mathbf{a}U) \oplus \gamma_1(\mathbf{x}_0 + \gamma_0(\mathbf{a}U)), \\ \mathbf{b}_1 = \gamma_0(\mathbf{a}U) \odot G \oplus \mathbf{b}_0 \ominus \gamma_1(\mathbf{x}_1 + \mathbf{b}_0) \end{cases}$$

относительно $\mathbf{x}_0, \mathbf{x}_1 \in P^m$.

Доказательство. Для произвольной матрицы K из (1), по определению, элемент $\nu_{\mathbf{a},\mathbf{b}}$ есть число решений уравнения

$$\mathbf{b} = \pi(\mathbf{y}+\mathbf{a}) - \pi(\mathbf{y}) = \gamma_1(\mathbf{y}K + \mathbf{a}K)Z - \gamma_1(\mathbf{y}K)Z \tag{9}$$

относительно $\mathbf{y} \in R^m$.

Пусть $\mathbf{y}K = (\mathbf{y}' \mid \mathbf{y}'')$, $\mathbf{a}K = (\mathbf{a}' \mid \mathbf{a}'')$. Пользуясь определением матрицы Z, из (9) для нулевого и первого разрядов $\mathbf{b}_0, \mathbf{b}_1$ строки $\mathbf{b}$ получаем равенства

$$\begin{aligned} \mathbf{b}_0 &= \gamma_0(\gamma_1(\mathbf{y}' + \mathbf{a}' \mid \mathbf{y}'' + \mathbf{a}'')Z - \gamma_1(\mathbf{y}' \mid \mathbf{y}'')Z) = \\ &= \gamma_0((\gamma_1(\mathbf{y}' + \mathbf{a}') + p\gamma_1(\mathbf{y}'' + \mathbf{a}'')) - (\gamma_1(\mathbf{y}') + p\gamma_1(\mathbf{y}'')) = \\ &= \gamma_1(\mathbf{y}' + \mathbf{a}') \ominus \gamma_1(\mathbf{y}') = \gamma_1(\mathbf{y}'_0 + \mathbf{a}'_0) \oplus \mathbf{y}'_1 \oplus \mathbf{a}'_1 \ominus \mathbf{y}'_1; \\ \mathbf{b}_1 &= \gamma_1(\gamma_1(\mathbf{y}' + \mathbf{a}') + p\gamma_1(\mathbf{y}'' + \mathbf{a}'') - \gamma_1(\mathbf{y}') - p\gamma_1(\mathbf{y}'')) = \\ &= \gamma_0(\mathbf{y}''_1 \oplus \mathbf{a}''_1 \oplus \gamma_1(\mathbf{y}''_0 + \mathbf{a}''_0) \ominus \mathbf{y}''_1) \oplus \gamma_1((\mathbf{y}'_1 \oplus \mathbf{b}_0) - \mathbf{y}'_1). \end{aligned}$$

Окончательно получаем

$$\begin{aligned} \mathbf{b}_0 &= \gamma_1(\mathbf{y}'_0 + \mathbf{a}'_0) \oplus \mathbf{a}'_1, \\ \mathbf{b}_1 &= \mathbf{a}''_1 \oplus \gamma_1(\mathbf{y}''_0 + \mathbf{a}''_0) \oplus \gamma_1((\mathbf{y}'_1 \oplus \mathbf{b}_0) - \mathbf{y}'_1). \end{aligned}$$

Заметим, что $\mathbf{y}'_0 = \mathbf{y}''_0$ и $\mathbf{a}'_0 = \mathbf{a}''_0$, откуда

$$\begin{aligned} \mathbf{b}_0 &= \mathbf{a}'_1 \oplus \gamma_1(\mathbf{y}'_0 + \mathbf{a}'_0), \\ \mathbf{b}_1 &= \mathbf{a}'_1 \oplus \mathbf{a}''_1 \oplus \mathbf{b}_0 \oplus \gamma_1((\mathbf{y}'_1 \oplus \mathbf{b}_0) - \mathbf{y}'_1). \end{aligned}$$

Преобразуем второе равенство. Пусть $a, b \in \Gamma(R)$, тогда значение $\gamma_1((a \oplus b) - a)$ можно получить, используя равенство $a + b = (a \oplus b) + p\gamma_1(a + b)$. Имеем $a \oplus b - a = b - p\gamma_1(a+b)$, откуда $\gamma_1((a \oplus b) - a) = \gamma_1(b) \ominus \gamma_1(a+b)$. Заметим также, что $\mathbf{a}'_1 \oplus \mathbf{a}''_1 = \mathbf{a}'_0 G$. Следовательно,

$$\begin{aligned} \mathbf{b}_0 &= \mathbf{a}'_1 \oplus \gamma_1(\mathbf{y}'_0 + \mathbf{a}'_0), \\ \mathbf{b}_1 &= \mathbf{a}'_0 G \oplus \mathbf{b}_0 \ominus \gamma_1(\mathbf{y}'_1 + \mathbf{b}_0). \end{aligned}$$

Подставим значение $\mathbf{a}' = \mathbf{a}U$ и сделаем замену $\mathbf{x} = \mathbf{y}' = \mathbf{y}U$. Получим искомую систему уравнений. ■

Приведём оценку снизу показателя 2-транзитивности $d_2(\Sigma\pi)$ для класса подстановок π вида (8).

Теорема 2. Пусть π — подстановка на R^m, определённая равенством (8). Тогда $d_2(\Sigma\pi) \geqslant 4$.

Доказательство. Достаточно показать, что матрица Q_π^2 содержит нулевые элементы. Элементы $\nu_{\mathbf{ac}}^{(2)}$ матрицы Q_π^2 опишем, используя утверждение 2. Заметим, что

$$\nu_{\mathbf{ac}}^{(2)} = \frac{1}{N^2} \sum_{\mathbf{b} \in R^m} \nu_{\mathbf{ab}} \nu_{\mathbf{bc}}$$

и, согласно утверждению 2, произведение $\nu_{\mathbf{ab}}\nu_{\mathbf{bc}}$ равно числу решений $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{y}_0, \mathbf{y}_1) \in (P^m)^4$ системы

$$\begin{cases} \mathbf{b}_0 = \gamma_1(\mathbf{a}U) \oplus \gamma_1(\mathbf{x}_0 + \gamma_0(\mathbf{a}U)), \\ \mathbf{b}_1 = \gamma_0(\mathbf{a}U) \odot G \oplus \mathbf{b}_0 \ominus \gamma_1(\mathbf{x}_1 + \mathbf{b}_0), \\ \mathbf{c}_0 = \gamma_1(\mathbf{b}U) \oplus \gamma_1(\mathbf{y}_0 + \gamma_0(\mathbf{b}U)), \\ \mathbf{c}_1 = \gamma_0(\mathbf{b}U) \odot G \oplus \mathbf{c}_0 \ominus \gamma_1(\mathbf{y}_1 + \mathbf{c}_0), \end{cases} \tag{10}$$

а $\nu_{\mathbf{ac}}^{(2)}$ равно числу решений системы (10) относительно системы независимых переменных $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{y}_0, \mathbf{y}_1, \mathbf{b}_0, \mathbf{b}_1) \in (P^m)^6$. При этом условие $\nu_{\mathbf{ac}}^{(2)} > 0$ равносильно совместности системы (10) относительно последнего набора неизвестных.

Покажем, что в матрице Q_π^2 всегда есть нулевые элементы. Пусть $\mathbf{a}_0 = \mathbf{c}_0 = \mathbf{0}$. Заметим, что $\gamma_0(\mathbf{b}U) = \mathbf{b}_0 \odot U_0$. Первое и последнее уравнения системы (10) примут вид

$$\begin{cases} \mathbf{b}_0 = \gamma_1(\mathbf{a}U), \\ \mathbf{c}_1 = \mathbf{b}_0 \odot U_0 \odot G. \end{cases}$$

Следствием данной системы является уравнение $\mathbf{c}_1 = \mathbf{a}_1 \odot U_0^2 \odot G$. Пусть $\mathbf{a}_1, \mathbf{c}_1 \in P^m$ выбраны так, что последнее равенство не выполняется. Тогда для $\mathbf{a} = p\mathbf{a}_1, \mathbf{c} = p\mathbf{c}_1$ элемент $\nu_{\mathbf{ac}}^{(2)}$ равен нулю. ■

Далее показано, что в ряде случаев указанная оценка достижима.

4. Примитив над кольцом $R = \mathrm{GR}(q^2, p^2)$ на основе разрядно-инъективной матрицы K размера 1×2

Пусть $R = \mathrm{GR}(q^2, p^2)$, $P = \Gamma(R)$, $q = p^r$, $m = 1$, $p > 2$. В этом случае преобразование одного раунда шифрования будем записывать в виде

$$\pi(x) = \gamma_1(xu(e \mid e + pg))Z = \gamma_1(xu) + p(\gamma_1(xu) \oplus x_0 u_0 g_0), \tag{11}$$

где $u \in R^*$; $g \in \Gamma(R)$.

Заметим, что всегда справедливо включение $\{\gamma_1(a + e) \ominus \gamma_1(b + e) : a, b \in P\} \subset P$. Следующая теорема описывает кольца Галуа при $p > 2$, для которых любая нетривиальная РИ-матрица индуцирует 2-транзитивное множество подстановок с минимально возможным показателем 2-транзитивности.

Теорема 3. Пусть для кольца Галуа $R = \mathrm{GR}(q^2, p^2)$, $p > 2$, выполнено условие

$$\{\gamma_1(a + e) \ominus \gamma_1(b + e) : a, b \in P\} = P. \tag{12}$$

Тогда для любой подстановки π вида (11) выполнено равенство $d_2(\Sigma\pi) = 4$.

Доказательство. Как было замечено в п. 3, достаточно доказать, что $Q_\pi^3 > 0$. Опишем элементы $\nu_{ad}^{(3)}$ матрицы Q_π^3, используя утверждение 2. Заметим, что

$$\nu_{ad}^{(3)} = \frac{1}{N^3} \sum_{b,c \in R} \nu_{ab}\nu_{bc}\nu_{cd}$$

и, согласно утверждению 2, произведение $\nu_{ab}\nu_{bc}\nu_{cd}$ равно числу решений $(x_0, x_1, y_0, y_1, z_0, z_1) \in P^6$ системы

$$\begin{cases} b_0 = \gamma_1(au) \oplus \gamma_1(x_0 + \gamma_0(au)), \\ b_1 = \gamma_0(au)g \oplus b_0 \ominus \gamma_1(x_1 + b_0), \\ c_0 = \gamma_1(bu) \oplus \gamma_1(y_0 + \gamma_0(bu)), \\ c_1 = \gamma_0(bu)g \oplus c_0 \ominus \gamma_1(y_1 + c_0), \\ d_0 = \gamma_1(cu) \oplus \gamma_1(z_0 + \gamma_0(cu)), \\ d_1 = \gamma_0(cu)g \oplus d_0 \ominus \gamma_1(z_1 + d_0), \end{cases} \tag{13}$$

а $\nu_{ad}^{(3)}$ равно числу решений системы (13) относительно системы независимых переменных $(x_0, x_1, y_0, y_1, z_0, z_1, b_0, b_1, c_0, c_1) \in P^{10}$. При этом условие $\nu_{ad}^{(3)} > 0$ равносильно совместности системы (13) относительно последнего набора неизвестных. Покажем, что при условии (12) неравенство $\nu_{ad}^{(3)} > 0$ выполняется при всех $a, d \in R \setminus \{0\}$.

Покажем, что из пятого и шестого уравнений системы (13) можно выразить переменные c_1, c_0 соответственно через остальные переменные.

Поскольку $m = 1$, справедливо равенство $\gamma_1(cu) = c_0u_1 \oplus c_1u_0$, и пятое уравнение системы (13) равносильно уравнению

$$c_1 = (d_0 \ominus \gamma_1(z_0 + c_0u_0)) \ominus c_0u_1)u_0^{-1}. \tag{14}$$

Заметим, что последнее уравнение системы (13) имеет вид

$$d_1 = c_0u_0g \oplus d_0 \ominus \gamma_1(z_1 + d_0),$$

откуда $c_0 = (d_1 \ominus d_0 \oplus \gamma_1(z_1 + d_0))g^{-1}u_0^{-1}$.

Преобразуем (13) следующим образом. Заметим, что третье уравнение системы имеет вид

$$c_0 = b_0u_1 \oplus b_1u_0 \oplus \gamma_1(y_0 + \gamma_0(bu)).$$

Подставив в него выражение для b_1 из второго уравнения системы (13), получим

$$c_0 \ominus b_0(u_0 \oplus u_1) \ominus \gamma_0(augu) = \gamma_1(y_0 + b_0u_0) \ominus \gamma_1(x_1 + b_0)u_0.$$

Подставив выражение для c_1 из (14) в четвёртое уравнение системы (13), получим

$$d_0 \ominus c_0(u_0 \oplus u_1) \ominus \gamma_0(bugu) = \gamma_1(z_0 + c_0u_0) \ominus \gamma_1(y_1 + c_0)u_0.$$

Из приведённых рассуждений следует, что система (13) равносильна системе уравнений

$$\begin{cases} \gamma_1(y_0 + b_0u_0) \ominus \gamma_1(x_1 + b_0)u_0 = c_0 \ominus b_0(u_0 \oplus u_1) \ominus \gamma_0(augu), \\ \gamma_1(z_0 + c_0u_0) \ominus \gamma_1(y_1 + c_0)u_0 = d_0 \ominus c_0(u_0 \oplus u_1) \ominus \gamma_0(bugu), \\ b_0 = \gamma_1(au) \oplus \gamma_1(x_0 + \gamma_0(au)), \\ c_0 = (d_1 \ominus d_0 \oplus \gamma_1(z_1 + d_0))g^{-1}u_0^{-1}, \\ b_1 = \gamma_0(au)g \oplus b_0 \ominus \gamma_1(x_1 + b_0), \\ c_1 = (d_0 \ominus \gamma_1(z_0 + c_0u_0)) \ominus c_0u_1)u_0^{-1} \end{cases} \tag{15}$$

и совместна тогда и только тогда, когда совместна система из первых четырёх уравнений системы (15).

Лемма 1. Пусть $a \in R \setminus \{0\}$ — произвольный элемент. Тогда в уравнении

$$b_0 = \gamma_1(au) \oplus \gamma_1(x_0 + \gamma_0(au))$$

найдётся $x_0 \in P$, при котором $b_0 \neq \mathbf{0}$.

Доказательство. Для доказательства рассмотрим два случая. Пусть $a_0 = \mathbf{0}$, тогда $b_0 = \gamma_1(au) \neq \mathbf{0}$, так как $a \neq \mathbf{0}$.

Если же $a_0 \neq \mathbf{0}$, то существование искомого x_0 следует из того, что функция γ_1 не является константой. ■

Лемма 2. Пусть $d \in R \setminus \{0\}$ — произвольный элемент. Тогда в уравнении

$$c_0 = (d_1 \ominus d_0 \oplus \gamma_1(z_1 + d_0))g^{-1}u_0^{-1}$$

найдётся $z_1 \in P$, при котором $c_0 \neq \mathbf{0}$.

Доказательство. Лемма 2 доказывается аналогично лемме 1. ■

Зафиксируем произвольные $a, d \in R \setminus \{0\}$. По леммам 1 и 2 элементы $x_0, z_1 \in P$ выберем так, чтобы выполнялось условие $b_0 \neq 0$, $c_0 \neq 0$.

Нам понадобятся свойства функции переноса γ_1. Известно [3], что для $x, y \in \Gamma(R)$ она имеет вид

$$\gamma_1(x + y) \equiv \sum_{j=1}^{p-1} \frac{1}{j!(p-j)!} x^{rj} y^{r(p-j)} \pmod{p}.$$

Данный многочлен является однородным, поэтому при $y \neq 0$ верно сравнение $\gamma_1(x + y) \equiv y\gamma_1(x/y + e) \pmod{p}$.

В системе (15) сделаем замену переменных $y_0' = \dfrac{y_0}{b_0u_0}, x_1' = \dfrac{x_1}{b_0}, z_0' = \dfrac{z_0}{c_0u_0}, y_1' = \dfrac{y_1}{c_0}$.

Тогда условие $\nu_{ad}^{(3)} > 0$ выполняется в том и только в том случае, когда для данных a, d, b_0, c_0, x_0, z_1 существует решение $(x_1', y_0', y_1', z_0') \in P^4$ системы уравнений

$$\begin{cases} \gamma_1(y_0' + e) \ominus \gamma_1(x_1' + e) = (c_0 \ominus b_0(u_0 \oplus u_1) \ominus \gamma_0(augu))u_0^{-1}b_0^{-1}, \\ \gamma_1(z_0' + e) \ominus \gamma_1(y_1' + e) = (d_0 \ominus c_0(u_0 \oplus u_1) \ominus \gamma_0(bugu))u_0^{-1}c_0^{-1}. \end{cases} \tag{16}$$

Если выполнено условие (12), то очевидно, что система (16) совместна. Таким образом, доказано, что все элементы матрицы Q_π^3 положительны. ■

Замечание. Экспериментально подтверждено, что равенство (12) справедливо для $\mathbb{Z}_{p^2}$, где $p \leqslant 283$, а также для колец Галуа $\mathrm{GR}(9^2, 3^2)$, $\mathrm{GR}(27^2, 3^2)$. Описание колец Галуа, для которых выполнено условие (12), остаётся открытой задачей. Примеров, когда оно не выполняется, не найдено.

Утверждение 3. Пусть $p = 3$, $R = \mathrm{GR}(q^2, p^2)$. Тогда выполнено (12).

Доказательство. Функция переноса в первый разряд имеет вид

$$\gamma_1(x+e) = \sum_{j=1}^{p-1} \frac{(-1)^j}{j} x^j.$$

Отсюда $\gamma_1(a+e) \ominus \gamma_1(b+e) = \left(\dfrac{a^2}{2} \ominus a\right) \ominus \left(\dfrac{b^2}{2} \ominus b\right) = 2^{-1}((a \ominus e)^2 \ominus (b \ominus e)^2)$. Достаточно показать, что при любом $c \in P$ уравнение

$$x^2 \ominus y^2 = c \tag{17}$$

имеет решение в P. Пусть $y = x \ominus e$. Тогда $x^2 \ominus (x \ominus e)^2 = 2x \ominus e$, и решением уравнения (17) будет пара $\left(\dfrac{c \oplus e}{2}, \dfrac{c \ominus e}{2}\right)$. ■

В предположении, что верна S-гипотеза (все конечные простые группы известны [4]), удаётся описать свойства группы $\langle \Sigma\pi \rangle$.

Утверждение 4. Пусть верна S-гипотеза, $R = \mathbb{Z}_{p^2}$. Если множество Σ содержит полный цикл $t(x) \equiv x + 1 \pmod{p^2}$, то $\langle \Sigma\pi \rangle \supseteq A_{p^2}$.

Доказательство. Из теоремы 3 следует 2-транзитивность группы $\langle \Sigma\pi \rangle$. Из справедливости S-гипотезы следует, что все 2-транзитивные группы известны, и поэтому список таких групп в [4, 5] является исчерпывающим. Так как группа $\langle \Sigma\pi \rangle$ примарна, то либо она подгруппа аффинной группы, либо содержит знакопеременную.

Однако известно, что $t \notin AGL(2, p)$. Тогда, согласно списку 2-транзитивных групп, $\langle \Sigma\pi \rangle \supseteq A_{p^2}$. ■

5. Примитив над кольцом $R = \mathrm{GR}(q^2, 4)$ на основе разрядно-инъективной $m \times 2m$-матрицы

Пусть далее $R = \mathrm{GR}(q^2, 4)$, $P = \Gamma(R)$.

Теорема 4. Пусть $R = \mathrm{GR}(q^2, 4)$, $P = \Gamma(R)$, $m \in \mathbb{N}$, подстановка π вида (8) выбрана так, что все миноры матрицы U_0 отличны от нуля. Тогда $d_2(\Sigma\pi) = 4$.

Доказательство. Достаточно доказать, что $Q_\pi^3 > 0$. Опишем элементы $\nu_{\mathbf{ad}}^{(3)}, \mathbf{a}, \mathbf{d} \in R^m$ матрицы Q_π^3, используя утверждение 2. Заметим, что

$$\nu_{\mathbf{ad}}^{(3)} = \frac{1}{N^3} \sum_{\mathbf{b},\mathbf{c} \in R^m} \nu_{\mathbf{ab}} \nu_{\mathbf{bc}} \nu_{\mathbf{cd}}$$

и, согласно утверждению 2, произведение $\nu_{\mathbf{ab}}\nu_{\mathbf{bc}}\nu_{\mathbf{cd}}$ равно числу решений $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{y}_0, \mathbf{y}_1, \mathbf{z}_0, \mathbf{z}_1) \in (P^m)^6$ системы

$$\begin{array}{l} \mathbf{b}_0 = \gamma_1(\mathbf{a}U) \oplus \gamma_1(\mathbf{x}_0 + \gamma_0(\mathbf{a}U)), \\ \mathbf{b}_1 = \gamma_0(\mathbf{a}U) \odot G \oplus \mathbf{b}_0 \ominus \gamma_1(\mathbf{x}_1 + \mathbf{b}_0), \\ \mathbf{c}_0 = \gamma_1(\mathbf{b}U) \oplus \gamma_1(\mathbf{y}_0 + \gamma_0(\mathbf{b}U)), \\ \mathbf{c}_1 = \gamma_0(\mathbf{b}U) \odot G \oplus \mathbf{c}_0 \ominus \gamma_1(\mathbf{y}_1 + \mathbf{c}_0), \\ \mathbf{d}_0 = \gamma_1(\mathbf{c}U) \oplus \gamma_1(\mathbf{z}_0 + \gamma_0(\mathbf{c}U)), \\ \mathbf{d}_1 = \gamma_0(\mathbf{c}U) \odot G \oplus \mathbf{d}_0 \ominus \gamma_1(\mathbf{z}_1 + \mathbf{d}_0), \end{array} \tag{18}$$

а $\nu^{(3)}_{\mathbf{ad}}$ равно числу решений системы (18) относительно системы независимых переменных $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{y}_0, \mathbf{y}_1, \mathbf{z}_0, \mathbf{z}_1, \mathbf{b}_0, \mathbf{b}_1, \mathbf{c}_0, \mathbf{c}_1) \in (P^m)^{10}$. При этом условие $\nu^{(3)}_{\mathbf{ad}} > 0$ равносильно совместности системы (18) относительно последнего набора неизвестных. Покажем, что при условии теоремы неравенство $\nu^{(3)}_{\mathbf{ad}} > 0$ выполняется при всех $\mathbf{a}, \mathbf{d} \in R^m \setminus \{\mathbf{0}\}$.

Покажем, что из пятого и шестого уравнений системы (18) можно выразить переменные $\mathbf{c}_1, \mathbf{c}_0$ соответственно через остальные переменные.

Из равенства $\gamma_1(\mathbf{c}U) = \mathbf{c}_0 \odot U_1 \oplus \mathbf{c}_1 \odot U_0 \oplus \gamma_1(\mathbf{c}_0U_0)$ следует, что пятое уравнение системы (18) равносильно уравнению

$$\mathbf{c}_1 = (\mathbf{d}_0 \ominus \gamma_1(\mathbf{z}_0 + \mathbf{c}_0 \odot U_0)) \ominus \mathbf{c}_0 \odot U_1 \ominus \gamma_1(\mathbf{c}_0U_0)) \odot U_0^{-1}. \tag{19}$$

Заметим, что последнее уравнение системы (18) имеет вид

$$\mathbf{d}_1 = \mathbf{c}_0 \odot U_0 \odot G \oplus \mathbf{d}_0 \ominus \gamma_1(\mathbf{z}_1 + \mathbf{d}_0),$$

откуда

$$\mathbf{c}_0 = (\mathbf{d}_1 \ominus \mathbf{d}_0 \oplus \gamma_1(\mathbf{z}_1 + \mathbf{d}_0)) \odot G^{-1} \odot U_0^{-1}.$$

Преобразуем систему (18) следующим образом. Заметим, что третье уравнение системы имеет вид

$$\mathbf{c}_0 = \mathbf{b}_0 \odot U_1 \oplus \mathbf{b}_1 \odot U_0 \oplus \gamma_1(\mathbf{b}_0U_0) \oplus \gamma_1(\mathbf{y}_0 + \gamma_0(\mathbf{b}U)).$$

Подставив в него выражение для $\mathbf{b}_1$ из второго уравнения, получим

$$\mathbf{c}_0 \ominus \mathbf{b}_0 \odot (U_0 \oplus U_1) \ominus \gamma_0(\mathbf{a}UGU) \ominus \gamma_1(\mathbf{b}_0U_0) = \gamma_1(\mathbf{y}_0 + \mathbf{b}_0 \odot U_0) \ominus \gamma_1(\mathbf{x}_1 + \mathbf{b}_0) \odot U_0.$$

Подставив выражение для $\mathbf{c}_1$ из (19) в четвёртое уравнение системы, получим

$$\mathbf{d}_0 \ominus \mathbf{c}_0 \odot (U_0 \oplus U_1) \ominus \gamma_0(\mathbf{b}UGU) \ominus \gamma_1(\mathbf{c}_0U_0) = \gamma_1(\mathbf{z}_0 + \mathbf{c}_0 \odot U_0) \ominus \gamma_1(\mathbf{y}_1 + \mathbf{c}_0) \odot U_0.$$

Из приведённых рассуждений следует, что система (18) равносильна системе уравнений

$$\begin{cases} \gamma_1(\mathbf{y}_0 + \mathbf{b}_0 \odot U_0) \ominus \gamma_1(\mathbf{x}_1 + \mathbf{b}_0) \odot U_0 = \mathbf{c}_0 \ominus \mathbf{b}_0 \odot (U_0 \oplus U_1) \ominus \gamma_0(\mathbf{a}UGU) \ominus \gamma_1(\mathbf{b}_0U_0), \\ \gamma_1(\mathbf{z}_0 + \mathbf{c}_0 \odot U_0) \ominus \gamma_1(\mathbf{y}_1 + \mathbf{c}_0) \odot U_0 = \mathbf{d}_0 \ominus \mathbf{c}_0 \odot (U_0 \oplus U_1) \ominus \gamma_0(\mathbf{b}UGU) \ominus \gamma_1(\mathbf{c}_0U_0), \\ \mathbf{b}_0 = \gamma_1(\mathbf{a}U) \oplus \gamma_1(\mathbf{x}_0 + \gamma_0(\mathbf{a}U)), \\ \mathbf{c}_0 = (\mathbf{d}_1 \ominus \mathbf{d}_0 \oplus \gamma_1(\mathbf{z}_1 + \mathbf{d}_0)) \odot G^{-1} \odot U_0^{-1}, \\ \mathbf{b}_1 = \gamma_0(\mathbf{a}U) \odot G \oplus \mathbf{b}_0 \ominus \gamma_1(\mathbf{x}_1 + \mathbf{b}_0), \\ \mathbf{c}_1 = (\mathbf{d}_0 \ominus \gamma_1(\mathbf{z}_0 + \mathbf{c}_0 \odot U_0)) \ominus \mathbf{c}_0 \odot U_1 \ominus \gamma_1(\mathbf{c}_0U_0)) \odot U_0^{-1} \end{cases} \tag{20}$$

и совместна тогда и только тогда, когда совместна система из первых четырёх уравнений системы (20).

Лемма 3. Пусть $\mathbf{a}\in R^m\setminus\{\mathbf{0}\}$ — произвольный вектор. Тогда в уравнении

$$\mathbf{b}_0=\gamma_1(\mathbf{a}U)\oplus\gamma_1(\mathbf{x}_0+\gamma_0(\mathbf{a}U))$$

найдётся $\mathbf{x}_0\in P^m$, при котором $\mathbf{b}_0\neq\mathbf{0}$.

Доказательство. Для доказательства рассмотрим два случая. Пусть $\mathbf{a}_0=\mathbf{0}$, тогда $\mathbf{b}_0=\gamma_1(\mathbf{a}U)\neq\mathbf{0}$, так как $\mathbf{a}\neq\mathbf{0}$.

Если же $\mathbf{a}_0\neq\mathbf{0}$, то существование искомого $\mathbf{x}_0$ следует из того, что функция γ_1 не является константой. ■

Лемма 4. Пусть $\mathbf{d}\in R^m\setminus\{\mathbf{0}\}$ — произвольный вектор. Тогда в уравнении

$$\mathbf{c}_0=(\mathbf{d}_1\ominus\mathbf{d}_0\oplus\gamma_1(\mathbf{z}_1+\mathbf{d}_0))\odot G^{-1}\odot U_0^{-1}$$

найдётся $\mathbf{z}_1\in P^m$, при котором $\mathbf{c}_0\neq\mathbf{0}$.

Доказательство. Лемма 4 доказывается аналогично лемме 3. ■

Зафиксируем произвольные $\mathbf{a},\mathbf{d}\in R^m\setminus\{\mathbf{0}\}$. По леммам 3 и 4 векторы $\mathbf{x}_0,\mathbf{z}_1\in P^m$ выберем так, чтобы выполнялось условие $\mathbf{b}_0\neq\mathbf{0},\mathbf{c}_0\neq\mathbf{0}$.

Следовательно, достаточно показать, что для произвольных $\mathbf{a},\mathbf{d}$ и произвольных ненулевых $\mathbf{b}_0,\mathbf{c}_0$ при условии теоремы система уравнений

$$\begin{cases}\gamma_1(\mathbf{y}_0+\mathbf{b}_0\odot U_0)\ominus\gamma_1(\mathbf{x}_1+\mathbf{b}_0)\odot U_0=\mathbf{c}_0\ominus\mathbf{b}_0\odot(U_0\oplus U_1)\ominus\gamma_0(\mathbf{a}UGU)\ominus\gamma_1(\mathbf{b}_0U_0),\\ \gamma_1(\mathbf{z}_0+\mathbf{c}_0\odot U_0)\ominus\gamma_1(\mathbf{y}_1+\mathbf{c}_0)\odot U_0=\mathbf{d}_0\ominus\mathbf{c}_0\odot(U_0\oplus U_1)\ominus\gamma_0(\mathbf{b}UGU)\ominus\gamma_1(\mathbf{c}_0U_0)\end{cases}\tag{21}$$

относительно неизвестных $(\mathbf{y}_0,\mathbf{x}_1,\mathbf{z}_0,\mathbf{y}_1)$ совместна.

Поскольку правые части уравнений системы (21) при фиксированных $\mathbf{a},\mathbf{d},\mathbf{b}_0,\mathbf{c}_0$ являются постоянными векторами, преобразуем отдельно левые части.

Пусть $U_0=(u_{ij})$, $\mathbf{b}_0=(b_1^{(0)},\ldots,b_m^{(0)})$ и $\mathbf{y}_0=(y_1^{(0)},\ldots,y_m^{(0)})$, $\mathbf{x}_1=(x_1^{(1)},\ldots,x_m^{(1)})$. Заметим, что первое уравнение системы (21) равносильно системе уравнений относительно независимых переменных $(y_1^{(0)},\ldots,y_m^{(0)},x_1^{(1)},\ldots,x_m^{(1)})\in P$ с матрицей

$$M=\begin{pmatrix}\sum\limits_{i=1}^m b_i^{(0)}s_{i1} & 0 & \cdots & 0 & 0 & b_1^{(0)}s_{11} & \ldots & b_m^{(0)}s_{m1}\\ & & \ddots & & & \vdots & & \vdots\\ 0 & 0 & \ldots & 0 & \sum\limits_{i=1}^m b_i^{(0)}s_{im} & b_1^{(0)}s_{1m} & \ldots & b_m^{(0)}s_{mm}\end{pmatrix}.$$

Покажем, что если все миноры матрицы U_0 ненулевые, то $\operatorname{rank}M=m$ при любом ненулевом векторе $\mathbf{b}_0$.

Из определения максимально-рассеивающей матрицы $U_0\in P_{m,m}$ следует, что если вектор $\mathbf{b}_0\in P^m$ имеет вес k, то $\|\mathbf{b}_0U_0\|\geqslant m-k+1$.

Пусть $\|\mathbf{b}_0\|=k$. Можно считать, что $\mathbf{b}_0=(b_1^{(0)},\ldots,b_k^{(0)},0,\ldots,0)$. Тогда последние $m-k$ столбцов матрицы M ненулевые и все миноры, соответствующие подматрицам, содержащимся в этих столбцах, также ненулевые. Следовательно, $\|\mathbf{b}_0U_0\|\geqslant m-k+1$. Это означает, что

$$\left\|\left(\sum_{i=1}^m b_i^{(0)}s_{i1},\ldots,\sum_{i=1}^m b_i^{(0)}s_{im}\right)\right\|\geqslant m-k+1.$$

Получаем, что все строки матрицы M линейно независимы, т. е. $\operatorname{rank}M=m$.

Аналогичные рассуждения справедливы и для второго уравнения системы (21). Поэтому система (21) совместна относительно неизвестных $(\mathbf{y}_0,\mathbf{x}_1,\mathbf{z}_0,\mathbf{y}_1)$, т. е. $\nu_{\mathbf{ad}}^{(3)}>0$ при всех $\mathbf{a},\mathbf{d}\in R^m\setminus\{\mathbf{0}\}$. ■

6. Реализация

Опишем представление элементов кольца Галуа $R = \mathrm{GR}(q^2, p^2)$, $q = p^r$, операции сложения и умножения, а также способ вычисления разрядной функции γ_1 в разрядном множестве $\Gamma(R)$.

Имеет место изоморфизм $R \cong \mathbb{Z}_{p^2}[x]/(F(x))$, где $F(x)$ — многочлен Галуа над $\mathbb{Z}_{p^2}$, в соответствии с которым элементу $a \in R$ сопоставлен многочлен $a(x) \in \mathbb{Z}_{p^2}[x]$ степени $r-1$.

Сумма $c = a + b$ и произведение $d = ab$ реализуются с помощью равенств

$$c(x) = a(x) + b(x), \quad d(x) = a(x)b(x) \pmod{F(x)}.$$

Из описания разрядных функций, приведённых в работе [3], следует, что $\gamma_0(a) = a^q$. Поэтому для нахождения первого разряда $\gamma_1(a)$ вычислим многочлен $t(x)$ из равенства $a(x) - a(x)^q = pt(x) \pmod{F(x)}$. Тогда $t(x)^q \pmod{F(x)}$ есть многочлен, соответствующий элементу $\gamma_1(a)$ кольца R.

Заметим, что возведение в степень $q = p^r$ реализуется с помощью не более чем $\log_2 q$ возведений многочлена в квадрат по модулю многочлена $F(x)$ степени r.

Заключение

Полученные в работе результаты показывают, что, используя только линейные преобразования модуля и разрядные функции кольца Галуа, можно строить эффективно реализуемые нелинейные подстановки большой степени, которые порождают множество с малым показателем 2-транзитивности. В дальнейшем представляет интерес изучение других криптографических свойств описанных примитивов и построение новых классов РИ-матриц.

Автор выражает искреннюю благодарность профессору А. А. Нечаеву за постановку задачи и внимание к проводимым исследованиям.

ЛИТЕРАТУРА

1. *Глухов М. М.* О 2-транзитивности произведения регулярных групп подстановок // Труды по дискретной математике. М.: Физматлит, 2000. С. 37–52.
2. *Глухов М. М., Зубов А. Ю.* О длинах симметрических и знакопеременных групп подстановок в различных системах образующих (обзор) // Математические вопросы кибернетики. 1999. Вып. 8. С. 2–32.
3. *Кузьмин А. С., Нечаев А. А.* Линейные рекуррентные последовательности над кольцами Галуа // Алгебра и логика. 1995. Т. 34. № 2. С. 169–189.
4. *Cameron P. J.* Finite permutation groups and finite simple groups // Bull. London Math. Soc. 1981. V. 13. No. 1. P. 1–22.
5. *Погорелов Б. А.* Основы теории групп подстановок. Ч. I. Общие вопросы. М., 1986. 316 с.

2013 Теоретические основы прикладной дискретной математики №4(22)

DOI 10.17223/20710410/22/2
УДК 512.714

ПЕРЕСТАНОВОЧНЫЕ МНОГОЧЛЕНЫ НАД ПРИМАРНЫМИ КОЛЬЦАМИ

А. В. Карпов

Национальный исследовательский Томский государственный университет, г. Томск, Россия

E-mail: karpov@isc.tsu.ru

Изучаются вопросы обращения перестановочных многочленов над примарным кольцом $\mathbb{Z}_{p^k}$, где p — простое и $k > 1$. Устанавливаются необходимые и достаточные условия того, что два перестановочных многочлена являются взаимно обратными по модулю p^k. Доказывается рекуррентная формула для нахождения обратного перестановочного многочлена по модулю p^k на основе известного обращения по модулю p^2. Предлагается метод построения пар взаимно обратных многочленов по модулю p^k на основе одной такой пары.

Ключевые слова: *перестановочные многочлены, примарные кольца, полиномиальные перестановки.*

Введение

Рассматриваются многочлены с целыми коэффициентами от одной переменной. Каждый многочлен $f(x)$ индуцирует соответствующую функцию $\tilde{f} : \mathbb{Z} \to \mathbb{Z}$, значения которой можно рассматривать по разным модулям n, т. е. $\tilde{f}/n : \mathbb{Z}_n \to \mathbb{Z}_n$. Многочлен $f(x)$ называется *перестановочным* по модулю n (над $\mathbb{Z}_n$), если соответствующая ему функция $\tilde{f}/n : \mathbb{Z}_n \to \mathbb{Z}_n$ является биективной. Многочлен $g(x)$ называется *обратным* к $f(x)$ по модулю n, если для любого целого c выполняется $g(f(c)) = c \pmod n$. При этом многочлены $f(x)$ и $g(x)$ называются *взаимно обратными* по модулю n (над $\mathbb{Z}_n$).

Изучение перестановочных многочленов как таковых было начато в 1863 г. [1]. История развития этой области до 1922 г. освещена в [2]. Наиболее полный обзор современных и не только результатов, связанных с перестановочными многочленами над конечными полями, приведён в [3]. Вопросы, связанные с представимостью функций k-значной логики полиномами, в наиболее общем виде обсуждаются в [4]. В настоящее время появляются работы, связанные с перестановочными многочленами специального вида и их обращением [5 – 9], а также с различными прикладными вопросами, в том числе с применением перестановочных многочленов в криптографии и теории кодирования [10, 11]. Изучается строение группы перестановочных многочленов над конечным целочисленным кольцом [12].

В [13] приведен критерий перестановочности над произвольным кольцом классов вычетов целых чисел, из которого видно, что ключевым является случай примарных колец $\mathbb{Z}_{p^k}$, о котором и пойдёт речь далее.

1. Построение взаимно обратных многочленов

Исследование перестановочных многочленов над примарными кольцами начато в [14] для случая $p = 2$. Изучение некоторых частных случаев можно найти в [15]. В общем случае перестановочные многочлены над кольцами $\mathbb{Z}_{p^k}$ рассматриваются в [13], откуда известен следующий

Критерий перестановочности над $\mathbb{Z}_{p^k}$. Целочисленный многочлен является перестановочным над кольцом $\mathbb{Z}_{p^k}$ тогда и только тогда, когда он является перестановочным над полем $\mathbb{Z}_p$ и его производная не обращается в ноль в $\mathbb{Z}_p$.

Из приведённого критерия видно, что перестановочный над $\mathbb{Z}_{p^2}$ многочлен является перестановочным и по всем примарным модулям p^k и что задача построения перестановочных над $\mathbb{Z}_{p^k}$ многочленов сводится к построению перестановочных над полем $\mathbb{Z}_p$ многочленов $f_p(x)$ и многочленов $f_0(x)$, индуцирующих нулевую по модулю p функцию, таких, что $f_p'(x) + f_0'(x)$ не имеет корней в $\mathbb{Z}_p$.

Далее предлагается метод построения пар взаимно обратных многочленов по модулю p^k на основе одной такой пары.

Пусть $f_p(x)$ и $g_p(x)$ — взаимно обратные многочлены по модулю p^k. Следующая теорема позволяет обратить многочлен $f(x) = f_p(x) + f_0(x)$, используя известное обращение $g_p(x)$, при дополнительных ограничениях на $f_0(x)$.

Теорема 1. Пусть многочлены $f_p(x)$ и $g_p(x)$ — взаимно обратные по модулю p^k, а многочлены $f_0(x)$ и $f_0'(x)$ индуцируют нулевые функции по модулям $p^{\lceil k/2 \rceil}$ и $p^{\lfloor k/2 \rfloor}$ соответственно. Тогда обратным к $f(x)$ по модулю p^k является многочлен

$$g(x) = g_p(x) - f_0(g_p(x))g_p'(x).$$

Доказательство. Выберем произвольный $x \in \mathbb{Z}_{p^k}$ и рассмотрим суперпозицию

$$g(f(x)) = g_p(f(x)) - \underbrace{f_0(g_p(f(x)))}_{(*)} g_p'(f(x)). \tag{1}$$

Так как $f(x) = f_p(x) + f_0(x)$ и $f_0(x) = 0 \pmod{p^{\lceil k/2 \rceil}}$, то $g_p(f(x)) = g_p(f_p(x) + f_0(x)) = g_p(f_p(x)) + f_0(x)g_p'(f_p(x)) \pmod{p^k}$, $g_p(x)$ — обратный к $f_p(x)$ над $\mathbb{Z}_{p^k}$, следовательно,

$$g_p(f(x)) = x + f_0(x)g_p'(f_p(x)) \pmod{p^k}. \tag{2}$$

Подставляя (2) в (*) выражения (1), получаем

$$f_0(g_p(f(x))) = f_0(x + f_0(x)g_p'(f_p(x))) = f_0(x) + f_0'(x)f_0(x)g_p'(f_p(x)) \pmod{p^k}.$$

Так как $f_0'(x) = 0 \pmod{p^{\lfloor k/2 \rfloor}}$, то второе слагаемое в правой части последнего равенства равно нулю по модулю p^k и

$$f_0(g_p(f(x))) = f_0(x) \pmod{p^k}. \tag{3}$$

Далее заметим, что $g_p'(f(x)) = g_p'(f_p(x)) + f_0(x)g_p''(f_p(x)) \pmod{p^k}$, а значит, ввиду $f_0^2(x) = 0 \pmod{p^k}$, с учётом (3) получим

$$f_0(g_p(f(x)))g_p'(f(x)) = f_0(x)g_p'(f_p(x)) \pmod{p^k}. \tag{4}$$

Подставляя (2) и (4) в (1), получаем

$$g(f(x)) = x + f_0(x)g_p'(f_p(x)) - f_0(x)g_p'(f_p(x)) = x \pmod{p^k}.$$

Так как x — произвольный, то $g(x)$ является обратным к $f(x)$ над $\mathbb{Z}_{p^k}$. ■

Пример 1. Построим на основе теоремы 1 обратный многочлен к многочлену $f(x) = 2x + 2x^2 - x^4 + x^6 + 3x^8 + 3x^{10} + x^{12}$ по модулю 27. Многочлен $f_p(x) = 2x$ линеен, и обратный к нему над $\mathbb{Z}_{27}$ равен $g_p(x) = 14x$. В этом случае

$$f_0(x) = 2x^2 - x^4 + x^6 + 3x^8 + 3x^{10} + x^{12}$$

и вместе со своей производной он индуцирует нулевые функции по модулям 9 и 3 соответственно. Многочлен $f(x)$ удовлетворяет условиям теоремы 1, и обратный к нему найдём как

$$g(x) = g_p(x) - f_0(g_p(x))g_p'(x) = 14x + 20x^2 + 11x^4 + 4x^6 + 3x^8 + 21x^{10} + 22x^{12}.$$

Многочлен $g(x)$ индуцирует перестановку

$$(1, 14, 7, 26, 13, 20, 19, 23, 25, 8, 3, 2, 10, 5, 16, 19, 22, 11)(3, 6, 21, 15, 12, 24)(9, 18),$$

которая, как видно, является обратной к

$$(1, 11, 22, 17, 16, 5, 10, 2, 4, 8, 25, 23, 19, 20, 13, 26, 7, 14)(3, 24, 12, 15, 21, 6)(9, 18),$$

индуцированной $f(x)$.

Помимо порождения целого класса взаимно обратных по модулю p^k многочленов из одной пары взаимно обратных, теорема 1 позволяет свести задачу обращения $f(x)$ над $\mathbb{Z}_{p^k}$ к задаче обращения $f_p(x) = f(x) - f_0(x)$ над $\mathbb{Z}_{p^k}$, которая может оказаться проще. Так, в примере 1 многочлен $f_2(x)$ имеет достаточно сложную форму, но в силу того, что $f_p(x)$ линеен, обратить его оказывается несложно.

2. Условия на взаимно обратные многочлены, формула подъёма

Следующая теорема дает необходимые и достаточные условия того, что два многочлена являются взаимно обратными.

Теорема 2. Пусть многочлен $f(x)$ — перестановочный по модулю p^k. Тогда многочлен $g(x)$ является обратным к $f(x)$ по модулю p^k, если и только если выполняются следующие два условия:

1) для любого целого x выполняется

$$f'(x)g'(f(x)) = 1 \pmod{p^{\lfloor k/2 \rfloor}}; \tag{5}$$

2) $g(x)$ обращает $f(x)$ в $p^{\lceil k/2 \rceil}$ различных по модулю $p^{\lceil k/2 \rceil}$ точках.

Доказательство. Необходимость. Второе условие выполняется очевидным образом. Покажем, что выполняется условие 1. Для этого зафиксируем произвольный $x \in \mathbb{Z}_{p^k}$, придадим ему приращение $h = p^{\lceil k/2 \rceil}$. Тогда $h^2 = 0 \pmod{p^k}$ и разложение $g(f(x + h))$ по степеням h принимает вид

$$g(f(x + h)) = g(f(x)) + hf'(x)g'(f(x)) \pmod{p^k}. \tag{6}$$

Так как $g(x)$ является обратным к $f(x)$ над $\mathbb{Z}_{p^k}$, то (6) можно переписать в виде

$$h = hf'(x)g'(f(x)) \pmod{p^k},$$

откуда делением обеих частей на h получаем (5), и условие 1 выполняется в силу произвольности выбранного x.

Достаточность. Очевидно, что из $p^{\lceil k/2\rceil}$ различных по модулю $p^{\lceil k/2\rceil}$ точек путём прибавления к ним слагаемых $p^{\lceil k/2\rceil}l$, где $l \in \{0, 1, \ldots, p^{\lfloor k/2\rfloor} - 1\}$, можно получить все элементы $\mathbb{Z}_{p^k}$. Следовательно, так как выполняется условие 2, произвольный элемент кольца $\mathbb{Z}_{p^k}$ можно представить в виде $x + p^{\lceil k/2\rceil}l$, где x такой, что $g(f(x)) = x \pmod{p^k}$, а $l \in \{0, 1, \ldots, p^{\lfloor k/2\rfloor} - 1\}$. Тогда $g(f(x + p^{\lceil k/2\rceil}l)) = x + p^{\lceil k/2\rceil}lf'(x)g'(f(x)) \pmod{p^k}$, что, в силу условия 1, эквивалентно $g(f(x + p^{\lceil k/2\rceil}l)) = x + p^{\lceil k/2\rceil}l \pmod{p^k}$. Так как $x + p^{\lceil k/2\rceil}l$ — произвольный элемент кольца $\mathbb{Z}_{p^k}$, то $g(x)$ — обратный к $f(x)$ многочлен над $\mathbb{Z}_{p^k}$. ■

В ходе исследования было замечено, что многочлен $g(x)$, удовлетворяющий условию 1 теоремы 2, скорее всего, оказывается перестановочным. Можно также отметить, что, судя по примерам, перестановки, порождаемые взаимно обратными по модулю p^k многочленами, коммутируют по модулю p^{k+1}.

Далее излагается метод обращения целочисленного многочлена, перестановочного по всем примарным модулям p^n. Обращение осуществляется рекурсивно, начиная с известного обращения по модулю p^k при произвольном $k > 1$.

Понадобится

Следствие 1. Пусть многочлены $f(x)$ и $g(x)$ — взаимно обратные по модулю p^k. Тогда имеет место равенство

$$g(f(x)) = x + h_0(x), \tag{7}$$

где многочлены $h_0(x)$ и $h'_0(x)$ индуцируют нулевые функции по модулям p^k и $p^{\lfloor k/2\rfloor}$ соответственно.

Доказательство. Так как $g(x)$ является обратным к $f(x)$ над $\mathbb{Z}_{p^k}$, то их суперпозиция — тождественная функция, а значит, $g(f(x)) = x + h_0(x)$ и $h_0(x) = 0 \pmod{p^k}$ для всех $x \in \mathbb{Z}_{p^k}$. Из этого же следует, что выполняется условие 1 теоремы 2, а значит, $[g(f(x)]' = g'(f(x))f'(x) = 1 = 1 + h'_0(x) \pmod{p^{\lfloor k/2\rfloor}}$ для произвольного $x \in \mathbb{Z}_{p^k}$. ■

Следующая теорема позволяет «поднимать» обратный по модулю p^k многочлен до обратного по модулю $p^{k+\lfloor k/2\rfloor}$.

Теорема 3. Пусть многочлен $f(x)$ — перестановочный над $\mathbb{Z}_{p^k}$, $g_k(x)$ — обратный к $f(x)$ над $\mathbb{Z}_{p^k}$, $k > 1$. Тогда обратным к $f(x)$ по модулю $p^{k+\lfloor k/2\rfloor}$ является многочлен

$$g_{k+\lfloor k/2\rfloor}(x) = 2g_k(x) - g_k(f(g_k(x))). \tag{8}$$

Доказательство. Используя (7), равенство (8) можно переписать в виде $g_{k+\lfloor k/2\rfloor}(x) = g_k(x) - h_0(g_k(x))$. Выберем произвольный $x \in \mathbb{Z}_{p^k}$ и рассмотрим суперпозицию $g_{k+\lfloor k/2\rfloor}(f(x)) = g_k(f(x)) - h_0(g_k(f(x)))$. Так как $h_0(x) = g_k(f(x)) - x$, получаем $g_{k+\lfloor k/2\rfloor}(f(x)) = x + h_0(x) - h_0(x + h_0(x))$. Далее $h_0(x + h_0(x)) = h_0(x) + h_0(x)h'_0(x)$ и, в силу следствия 1, слагаемое $h_0(x)h'_0(x)$ обращается в ноль по модулю $p^{k+\lfloor k/2\rfloor}$ при любом $x \in \mathbb{Z}_{p^k}$. Окончательно получаем $g_{k+\lfloor k/2\rfloor}(f(x)) = x + h_0(x) - h_0(x) = x \pmod{p^{k+\lfloor k/2\rfloor}}$, что, в силу произвольности выбранного x, и доказывает теорему. ■

Пример 2. Рассмотрим многочлен $f(x) = x^3 + x$. Над $\mathbb{Z}_9$ он индуцирует перестановку $(1, 2)(4, 5)(7, 8)$ и, как видно, является обратным к самому себе. Над $\mathbb{Z}_{27}$ $f(x)$ индуцирует следующую перестановку:

$$(1, 2, 10, 11, 19, 20)(4, 14)(5, 22)(7, 26, 25, 17, 16, 8)(13, 23),$$

и над $\mathbb{Z}_{27}$ себя уже не обращает. По теореме 3 найдём обратный к $f(x)$ над $\mathbb{Z}_{27}$ многочлен. Используем (8), где $g_2(x) = f(x)$. Найдём $g_3(x) = 2f(x) - f(f(f(x)))$:

$$g_3(x) = x - x^3 + 18x^5 + 3x^7 + 6x^{11} + 6x^{13} + 24x^{15} + 15x^{19} + 21x^{21} + 18x^{23} + 18x^{25} - x^{27}.$$

Перестановка, индуцируемая полученным $g_3(x)$:

$$(1, 20, 19, 11, 10, 2)(4, 14)(5, 22)(7, 8, 16, 17, 25, 26)(13, 23),$$

обратна к перестановке, индуцированной $f(x)$.

Теорема 3 позволяет свести задачу обращения заданного перестановочного многочлена $f(x)$ по модулю p^k к задаче обращения $f(x)$ по модулю p^2. Таким образом, имея известное обращение по модулю p^2, по формуле (8) можно обратить заданный многочлен по модулю p^k для произвольного k.

Заключение

Рассмотрение вопросов перестановочности над конечным целочисленным кольцом сводится к случаю колец классов вычетов целых чисел по примарным модулям p^k.

Теорема 1 даёт метод построения пар взаимно обратных по модулю p^k перестановочных многочленов на основе одной такой пары, а также позволяет в некоторых случаях существенно упростить обращение. Формула (8) сводит задачу обращения по модулю p^k к обращению по модулю p^2 с последующим «подъёмом» решения, осуществляемым с модуля p^k на модуль $p^{k+\lfloor k/2 \rfloor}$. Получены необходимые и достаточные условия того, что два перестановочных многочлена являются взаимно обратными по модулю p^k (теорема 2).

Краткое изложение представленных результатов можно найти в [16].

ЛИТЕРАТУРА

1. *Hermite C.* Sur les fonctions de sept lettres // C. R. Acad. Sci. Paris. 1863. V. 57. P. 750–757.
2. *Dickson L. E.* History of the Theory of Numbers. Carnegie Institute, Washington, D. C., 1923. V. 3.
3. *Лидл Р., Нидеррайтер Г.* Конечные поля. Т. 1, 2. М.: Мир, 1988.
4. *Мещанинов Д. Г.* Метод построения полиномов для функций k-значной логики // Дискретная математика. 1995. Т. 7. № 3. С. 48–60.
5. *Caceres A. and Colon-Reyes O.* Some criteria for permutation binomials. Preprint. University of Puerto Rico at Humacao, 1997.
6. *Akbary A. and Wang Q.* On some permutation polynomials over finite fields // Intern. J. Math. Math. Sci. 2005. V. 2005. Iss. 16. P. 2631–2640.
7. *Diaz-Vargas J., Rubio-Barrios C. J., Sozaya-Chan J. A., and Tapia-Recillas H.* Self-invertible permutation polynomials over $\mathbb{Z}_m$ // Intern. J. Algebra. 2011. V. 5. No. 23. P. 1135–1153.
8. *Ryu J. and Takeshita O. Y.* On quadratic inverses for quadratic permutation polynomials over integer rings // IEEE Trans. Inform. Theory. 2006. V. 52. Iss. 3. P. 1254–1260.
9. *Wu B. and Liu Z.* The compositional inverse of a class of bilinear permutation polynomials over finite fields of characteristic 2 // `arxiv.org/abs/1301.0070`. January 2013.
10. *Varadharajan V.* Cryptosystems based on permutation polynomials // Intern. J. Computer Math. 1988. V. 23. Iss. 3–4. P. 237–250.
11. *Sun J. and Takeshita O. Y.* Interleavers for turbo codes using permutation polynomials over integer rings //IEEE Trans. Inform. Theory. 2005. V. 51. Iss. 1. P. 101–119.
12. *Frisch S. and Krenn D.* Sylow p-groups of polynomial permutations on the integers mod p^n // `arxiv.org/abs/1112.1228`. December 2011.

13. *Li S.* Permutation polynomials modulo m // `arxiv.org/pdf/math/0509523.pdf`. February 2008.
14. *Rivest R. L.* Permutation polynomials modulo 2^w // Finite Fields and Their Applications. 2001. No. 7. P. 287–292.
15. *Singh R. P. and Maity S.* Permutation polynomials modulo p^n // `eprint.iacr.org/2009/393.pdf`. 2009.
16. *Карпов А. В.* Обращение перестановочного многочлена над примарным кольцом // Междунар. конф. «Алгебра и логика: теория и приложения». Тез. докл. Красноярск: СФУ, 2013. С. 60–61.

МАТЕМАТИЧЕСКИЕ ОСНОВЫ КОМПЬЮТЕРНОЙ БЕЗОПАСНОСТИ

DOI 10.17223/20710410/22/3

УДК 004.94

АДМИНИСТРИРОВАНИЕ СИСТЕМЫ В РАМКАХ МАНДАТНОЙ СУЩНОСТНО-РОЛЕВОЙ ДП-МОДЕЛИ УПРАВЛЕНИЯ ДОСТУПОМ И ИНФОРМАЦИОННЫМИ ПОТОКАМИ В ОС СЕМЕЙСТВА LINUX

П. Н. Девянин

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

E-mail: peter_devyanin@hotmail.com

Рассматриваются де-юре правила администрирования параметров механизма управления доступом системы, заданной в рамках мандатной сущностно-ролевой ДП-модели и ориентированной на реализацию в отечественной защищённой операционной системе *Astra Linux Special Edition*. Данные правила позволяют администрировать учётные записи пользователей, права доступа, множество ролей, иерархию ролей и административных ролей, уровни конфиденциальности и целостности сущностей, субъект-сессий, ролей и административных ролей.

Ключевые слова: *мандатная сущностно-ролевая ДП-модель, операционная система Linux.*

Введение

При разработке формальных моделей безопасности управления доступом и информационными потоками часто правилам преобразования состояний системы, задающим порядок её администрирования, уделяется второстепенное внимание. Это объясняется ориентированностью большинства таких моделей только на теоретический анализ условий безопасности системы, при осуществлении которого, как правило, предполагается, что все действия по администрированию системы выполнены её доверенными субъект-сессиями (субъектами) и функционирование системы осуществляется только в результате реализации правил, инициированных недоверенными субъект-сессиями. Однако в разрабатываемой в настоящее время автором мандатной сущностно-ролевой ДП-модели операционных систем (ОС) семейства Linux (МРОСЛ ДП-модели) [1 – 3] значительное внимание уделяется её адаптации к условиям функционирования механизма управления доступом отечественной защищённой операционной системы *Astra Linux Special Edition* [4]. Для этого в МРОСЛ ДП-модели детально задаётся порядок реализации её элементов непосредственно в коде программного обеспечения защищённой ОС. Следовательно, целесообразно, кроме типовых правил управления доступом (получения доступа к сущностям, создания, удаления, переименования сущностей или «жёстких» ссылок на них, создания или удаления субъект-сессий), чётко определить правила, позволяющие администрировать учётные записи пользователей, административные права доступа, множества ролей, иерархию ролей и административных ролей, уровни конфиденциальности и целостности сущностей, субъект-сессий, ролей и административных ролей.

1. Основные элементы МРОСЛ ДП-модели

Так как формирование МРОСЛ ДП-модели ещё не окончено и её элементы, определения, предположения корректируются с учётом опыта их практической реализации в защищённой ОС, приведём актуальное описание модели в объёме, достаточном для рассмотрения правил администрирования системы.

Предположение 1. В рамках МРОСЛ ДП-модели пользователям соответствуют их учётные записи. Каждая учётная запись пользователя является учётной записью либо доверенного, либо недоверенного пользователя. Каждая субъект-сессия является либо доверенной, либо недоверенной и функционирует от имени учётной записи доверенного или недоверенного пользователя соответственно.

Используем следующие обозначения:

— $E = O \cup C$ — множество сущностей, где O — множество объектов; C — множество контейнеров; $O \cap C = \varnothing$;
— S — множество субъект-сессий учётных записей пользователей, где $S \cap E = \varnothing$;
— $entity\_name : C \times E \to NAMES$ — функция имён сущностей в составе сущностей-контейнеров, где $NAMES$ — некоторое множество допустимых имён сущностей, включающее элемент «» — пустая строка. При этом, по определению, если $e \notin H_E(c)$, то $entity\_name(c, e) =$ «», иначе $entity\_name(c, e) \neq$ «».

Определение 1. Иерархией сущностей называется заданное на множестве сущностей E отношение частичного порядка $\leqslant$, удовлетворяющее условию: если для сущности-контейнера $e \in C$ существуют сущности $e_1, e_2 \in C$, такие, что $e \leqslant e_2$, $e \leqslant e_1$, то $e_1 \leqslant e_2$ или $e_2 \leqslant e_1$. В случае, когда для двух сущностей $e_1, e_2 \in E$ выполняются условия $e_1 \leqslant e_2$ и $e_1 \neq e_2$, будем говорить, что сущность e_1 содержится в сущности-контейнере e_2, и будем использовать обозначение $e_1 < e_2$. Определим $H_E : E \to 2^E$ — функцию иерархии сущностей (сопоставляющую каждой сущности $e \in E$ множество сущностей $H_E(e) \subset E$, непосредственно в ней содержащихся), удовлетворяющую следующим условиям:

— если $e \in H_E(c)$, то $e < c$, при этом если $e \in C$, то не существует сущности-контейнера $d \in C$, такой, что $e < d < c$;
— для любых сущностей $e_1, e_2 \in E$, $e_1 \neq e_2$, выполняется равенство $H_E(e_1) \cap H_E(e_2) \cap \cap C = \varnothing$;
— если $o \in O$, то справедливо равенство $H_E(o) = \varnothing$.

Определение 2. Иерархией субъект-сессий называется заданное на множестве сущностей S отношение частичного порядка $\leqslant$, удовлетворяющее условию: если для субъект-сессии $s \in S$ существуют субъект-сессии $s_1, s_2 \in S$, такие, что $s \leqslant s_2$, $s \leqslant s_1$, то $s_1 \leqslant s_2$ или $s_2 \leqslant s_1$. В случае, когда для двух субъект-сессий $s_1, s_2 \in S$ выполняются условия $s_1 \leqslant s_2$ и $s_1 \neq s_2$, будем говорить, что субъект-сессия s_1 является потомком s_2, и будем использовать обозначение $s_1 < s_2$. Определим $H_S : S \to 2^S$ — функцию иерархии субъект-сессий (сопоставляющую каждой субъект-сессии $s \in S$ множество субъект-сессий $H_S(s) \subset S$, непосредственно в ней содержащихся), удовлетворяющую условиям:

— если $s_1 \in H_S(s_2)$, то $s_1 < s_2$ и не существует субъект-сессии $s \in S$, такой, что $s_1 < s < s_2$;
— для любых субъект-сессий $s_1, s_2 \in S$, $s_1 \neq s_2$, выполняется равенство $H_S(s_1) \cap \cap H_S(s_2) = \varnothing$.

В рамках МРОСЛ ДП-модели учтено наличие механизма создания «жёстких» ссылок (*hard link*) в файловой системе ОС семейства Linux, обеспечивающего возможность

размещения сущностей-объектов одновременно в нескольких сущностях-контейнерах. Для всех сущностей заданы их имена в составе сущностей-контейнеров, что позволяет описать правила предоставления содержимого контейнеров (например, списков файлов и каталогов, выдаваемых в реальной защищённой ОС по команде `ls`) с учётом параметров мандатного управления доступом. Впервые, по сравнению с предыдущими ДП-моделями, определено, что множество субъект-сессий не входит в множество сущностей. Это связано с тем, что в реальной защищённой ОС функции управления доступом к субъект-сессиям (процессам) и сущностям (файлам, каталогам) реализуются раздельно. Таким образом, иерархия на множестве субъект-сессий определяется независимо от иерархии сущностей и при реализации в ОС может быть задана двумя способами. Первый способ, когда существует явная связь между родительскими субъект-сессиями и их потомками (например, когда при завершении работы родительской субъект-сессии завершают работу её субъект-сессии — потомки, подчинённые родительской). Второй способ, когда порождённая субъект-сессией другая субъект-сессия функционирует независимо, в этом случае подчинение её в иерархии родительской субъект-сессии нецелесообразно.

Также определим:

— U — множество учётных записей пользователей;
— L_U — множество учётных записей доверенных пользователей;
— N_U — множество учётных записей недоверенных пользователей, при этом по определению справедливы равенства $L_U \cup N_U = U$, $L_U \cap N_U = \varnothing$;
— L_S — множество доверенных субъект-сессий;
— N_S — множество недоверенных субъект-сессий, при этом по определению справедливы равенства $L_S \cup N_S = S$, $L_S \cap N_S = \varnothing$;
— $user : S \to U$ — функция принадлежности субъект-сессии учётной записи пользователя, задающая для каждой субъект-сессии учётную запись пользователя, от имени которой она активизирована;
— R — множество ролей;
— AR — множество административных ролей, при этом по определению $AR \cap R = \varnothing$ (административные роли — особый вид ролей, предназначенный для изменения множеств прав доступа ролей, авторизации на роли, а также выполнения функций по администрированию системы, например, управления мандатными уровнями конфиденциальности сущностей и субъект-сессий);
— $R_r = \{read_r, write_r, execute_r, own_r\}$ — множество видов прав доступа;
— $R_a = \{read_a, write_a\}$ — множество видов доступа;
— $R_f = \{write_m, write_t\}$ — множество видов информационных потоков (по памяти и по времени).

Поскольку традиционно в ОС семейства Linux рассматриваются три вида прав доступа к сущностям: на чтение, на запись и на выполнение (или на использование контейнера-каталога), а также при управлении доступом к сущностям и субъект-сессиям учитывается наличие у каждой из них уникального владельца, имеющего право передавать права доступа к ним другим учётным записям пользователей, то в рамках МРОСЛ ДП-модели используются виды прав доступа $read_r$, $write_r$, $execute_r$, own_r и виды доступов $read_a$ и $write_a$. В отличие от предыдущих ДП-моделей, доступ владения own_a исключён из рассмотрения, как не имеющий явной реализации в реальных ОС.

Далее введём следующие обозначения:

- $P \subset (E \times R_r) \cup (S \times \{own_r\})$ — множество прав доступа к сущностям и субъект-сессиям;
- $AP \subset (R \cup AR) \times R_r$ — множество административных прав доступа к ролям или административным ролям;
- $A \subset S \times E \times R_a$ — множество доступов субъект-сессий к сущностям;
- $AA \subset S \times (R \cup AR) \times R_a$ — множество доступов субъект-сессий к ролям или административным ролям;
- $F \subset (E \cup S \cup R \cup AR) \times (E \cup S \cup R \cup AR) \times R_f$ — множество информационных потоков;
- $PA : R \cup AR \to 2^P$ — функция прав доступа к сущностям ролей и административных ролей, при этом для каждого права доступа $p \in P$ существует роль $r \in R \cup AR$, такая, что выполняется условие $p \in PA(r)$;
- $APA : AR \to 2^{AP}$ — функция административных прав доступа к ролям и административным ролям административных ролей, при этом для каждого административного права доступа $ap \in AP$ существует административная роль $ar \in AR$, такая, что выполняется условие $ap \in APA(ar)$;
- $shared\_container : C \cup R \cup AR \to \{\texttt{true}, \texttt{false}\}$ — функция разделяемых контейнеров, такая, что сущность-контейнер, роль или административная роль $c \in C \cup R \cup AR$ является разделяемой, когда $shared\_container(c) = \texttt{true}$, в противном случае $shared\_container(c) = \texttt{false}$;
- $V : E \cup R \cup AR \to \{(a_1, \ldots, a_n) : a_i \in \{0, 1\}, 1 \leqslant i \leqslant n, n \geqslant 1\} \cup \{\varnothing\}$ — функция значений сущностей, ролей и административных ролей (как аналогов сущностей-контейнеров), задающая возможность для каждой из них принимать значение любой (в том числе пустой) конечной последовательности бит;
- $(LC, \leqslant)$ — решётка многоуровневой безопасности уровней конфиденциальности (декартово произведение линейной шкалы уровней конфиденциальности данных и множества всех подмножеств конечного множества неиерархических категорий данных);
- $(f_u, f_e, f_r, f_s) \in FC$ — четвёрка функций уровней конфиденциальности, при этом:
  - $f_u : U \to LC$ — функция, задающая для каждой учётной записи пользователя её уровень доступа — максимальный разрешённый уровень доступа субъект-сессий, функционирующих от её имени;
  - $f_e : E \to LC$ — функция, задающая уровень конфиденциальности для каждой сущности;
  - $f_r : R \cup AR \to LC$ — функция, задающая для каждой роли или административной роли её уровень конфиденциальности;
  - $f_s : S \to LC$ — функция, задающая для каждой субъект-сессии её текущий уровень доступа;
- FC — множество всех четвёрок функций заданного вида;
- $CCR : E \cup R \cup AR \to \{\texttt{true}, \texttt{false}\}$ — функция, задающая способ доступа к сущностям внутри контейнеров или ролям в иерархии ролей (с учётом их мандатных уровней конфиденциальности). Если сущность $e \in C$ является контейнером и доступ к сущностям, содержащимся внутри контейнера e, разрешён без учёта уровня конфиденциальности контейнера e, то по определению выполняется равенство $CCR(e) = \texttt{false}$, в противном случае $CCR(e) = \texttt{true}$. При этом по определению для каждой сущности-объекта $o \in O$ выполняется равенство $CCR(o) = \texttt{true}$, для ролей или административных ролей $r \in R \cup AR$ всегда выполняется равенство $CCR(r) = \texttt{false}$;

— $(LI, \leqslant)$ — линейная шкала двух уровней целостности данных, где $LI = \{i\_low, i\_high\}$, $i\_low < i\_high$;
— $(i_u, i_e, i_r, i_s) \in I$ — четвёрка функций уровней целостности, при этом:
  — $i_u : U \to LI$ — функция, задающая для каждой учётной записи пользователя её уровень целостности — максимальный разрешённый уровень целостности субъект-сессий, функционирующих от её имени;
  — $i_e : E \to LI$ — функция, задающая уровень целостности для каждой сущности;
  — $i_r : R \cup AR \to LI$ — функция, задающая для каждой роли или административной роли её уровень целостности;
  — $i_s : S \to LI$ — функция, задающая для каждой субъект-сессии её текущий уровень целостности;
— I — множество всех четвёрок функций заданного вида;
— $CCRI : E \cup R \cup AR \to \{\texttt{true}, \texttt{false}\}$ — функция, задающая способ доступа к сущностям внутри контейнеров, ролям или административным ролям в иерархии ролей (с учётом их мандатных уровней целостности). Если сущность $e \in C$ является контейнером и доступ к сущностям, содержащимся внутри контейнера e, разрешён без учёта уровня целостности контейнера e, то по определению выполняется равенство $CCRI(e) = \texttt{false}$, в противном случае $CCRI(e) = \texttt{true}$. При этом по определению для каждой сущности-объекта $o \in O$ выполняется равенство $CCRI(o) = \texttt{true}$, для ролей или административных ролей $r \in R \cup AR$ всегда выполняется равенство $CCRI(r) = \texttt{false}$.

Определение 3. Иерархией ролей называется заданное на множестве ролей R отношение частичного порядка $\leqslant$. При этом по определению справедливо: для административной роли $ar \in AR$, если роли $r, r' \in R$ таковы, что $(r, read_r) \in APA(ar)$ и $r' \leqslant r$, то выполняется условие $(r', read_r) \in APA(ar)$. Иерархией административных ролей называется заданное на множестве ролей AR отношение частичного порядка $\leqslant$. При этом по определению справедливо: для административной роли $ar \in AR$, если административные роли $r, r' \in AR$ таковы, что $(r, read_r) \in APA(ar)$ и $r' \leqslant r$, то выполняется условие $(r', read_r) \in APA(ar)$. По определению роли и административные роли несравнимы друг с другом, т. е. их иерархии всегда независимы. В случае, когда для двух ролей или административных ролей $r, r' \in R \cup AR$ выполняются условия $r \leqslant r'$ и $r \neq r'$, будем использовать обозначение $r < r'$. Определим $H_R : R \cup AR \to 2^R \cup 2^{AR}$ — функцию иерархии ролей и административных ролей, сопоставляющую каждой роли $r \in R \cup AR$ множество ролей $H_R(r) \subset R \cup AR$ и удовлетворяющую условию: если $r' \in H_R(r)$, то $r' < r$ и не существует роли $r'' \in R \cup AR$, такой, что $r' < r''$, $r'' < r$.

Используем следующее обозначение:

— $role\_name : (R \cup AR) \times (R \cup AR) \to NAMES$ — функция имён ролей и административных ролей в составе роли или административной роли (как контейнера). При этом по определению если $r \in H_R(r')$, то $role_name(r', r) = $ «», иначе $role\_name(r', r) \neq$ «».

Предположение 2. Для каждой учётной записи пользователя $u \in U$ в зависимости от её уровня целостности задаются следующие индивидуальные административные роли, на которые авторизована только эта учётная запись:

— $u\_admin\_i\_low \in AR$, $i_r(u\_admin\_i\_low) = i\_low$, $f_r(u\_admin\_i\_low) = \otimes LC$, когда $i_u(u) = i\_low$ ($\otimes LC$ — минимальный элемент решётки уровней конфиденциальности LC);

— $u\_admin\_i\_low$, $u\_admin\_i\_high \in AR$, $i_r(u\_admin\_i\_low) = i\_low$, $i_r(u\_admin\_i\_high) = i\_high$, $f_r(u\_admin\_i\_low) = f_r(u\_admin\_i\_high) = \otimes LC$, $u\_admin\_i\_low < u\_admin\_i\_high$, когда $i_u(u) = i\_high$.

Множество других ролей учётной записи пользователя u задается с использованием административных прав доступа соответственно административной роли $u\_admin\_i\_low$ или административных ролей $u\_admin\_i\_low$, $u\_admin\_i\_high$, определяемых функцией APA. Иерархия индивидуальных административных ролей изменяется только при создании, удалении и изменении уровней доступа или целостности соответствующих учётных записей пользователей. На траекториях функционирования системы у этих административных ролей не изменяются имена, уровни конфиденциальности или целостности.

В рамках МРОСЛ ДП-модели, в отличие от моделей семейства $RBAC$ и других ролевых ДП-моделей, впервые вместо функций UA и AUA для задания авторизованных ролей и административных ролей учётных записей пользователей используются права доступа их индивидуальных административных ролей, задаваемых функцией APA. Такой подход, с одной стороны, позволяет реализовать ролевое управление доступом — назначение прав доступа учётным записям пользователей только через роли, с другой стороны, он направлен на обеспечение приоритетности и «монолитности» мандатного управления доступом как при управлении доступом к сущностям, так и при использовании или администрировании ролей. Достигается также большая совместимость со штатными механизмами управления доступом реальной защищённой ОС. Минимальный уровень конфиденциальности административных ролей обеспечивает возможность получения их в качестве текущих (получения доступа $read_a$) субъект-сессией с любым текущим уровнем доступа.

В моделях семейства $RBAC$ и других ролевых ДП-моделях предполагалось, что функция авторизованных ролей учётных записей пользователей UA обладает свойством «наследования» подчинённых ролей, т. е. выполняется следующее условие: для учётной записи пользователя $u \in U$, если роли $r, r' \in R$ таковы, что $r \in UA(u)$ и $r' \leqslant r$, то выполняется условие $r' \in UA(u)$. В рамках МРОСЛ ДП-модели с учётом предположения 2 это обеспечивается «наследованием» административной ролью права доступа $read_r$ от данной роли ко всем подчинённым ей ролям в иерархии. При этом права доступа $write_r$ и own_r таким свойством не обладают, что может позволить гибко задавать «диапазоны» администрируемых ролей по аналогии с моделью ролевого администрирования $ARBAC$ [1]. При реализации иерархий ролей в реальной защищённой ОС по аналогии с файловой системой можно использовать виртуальную структуру сущностей-ролей, отличающуюся от структур для файлов наличием «жёстких» ссылок не только на роли-«объекты» (роли, которым в иерархии не подчинена ни одна другая роль), но и на роли-«контейнеры» (роли, которым в иерархии подчинена хотя бы одна роль).

Используем следующее обозначение:

— $execute\_container : S \times E \to \{\texttt{true}, \texttt{false}\}$ — функция доступа субъект-сессии к сущностям в контейнерах, такая, что по определению для субъект-сессии $s \in S$ и сущности $e \in E$ справедливо равенство $execute\_container(s, e) = \texttt{true}$ тогда и только тогда, когда существует последовательность сущностей $e_1, \ldots, e_n \in E$, где $n \geqslant 1$, $e = e_n$, удовлетворяющих следующим условиям:
  — не существует сущности-контейнера $e_0 \in E$, такой, что $e_1 \in H_E(e_0)$;
  — $e_i \in H_E(e_{i-1})$, где $1 < i \leqslant n$;

— существует $r_i \in R \cup AR$, такая, что $(s, r_i, read_a) \in AA$ и $(e_i, execute_r) \in PA(r_i)$, $[f_e(e_i) \leqslant f_s(s)$ или $CCR(e_i) = \texttt{false}]$ и $[i_e(e_i) \leqslant i_s(s)$ или $CCRI(e_i) = \texttt{false}]$, где $1 \leqslant i < n$.

Данная функция используется для краткости и удобства описания элементов модели с учётом мандатного управления доступом и мандатного контроля целостности и не требует непосредственной реализации в реальной защищённой ОС (она «реализуется» путём последовательной проверки прав доступа к сущностям-контейнерам, содержащим данную сущность, на которую указывает её полное имя, начиная с корневой сущности-контейнера).

Зададим в рамках МРОСЛ ДП-модели механизм ограничений ($Constraints$), при этом используем следующие обозначения:

— APA^* — множество всех функций административных прав доступа административных ролей;
— PA^* — множество всех функций прав доступа ролей и административных ролей;
— AA^* — множество множеств доступов субъект-сессий к ролям или административным ролям;
— $C^{APA} : APA^* \to \{\texttt{true}, \texttt{false}\}$ — функция, задающая ограничение на значения множеств административных прав доступа административных ролей, т. е. по определению множества административных прав доступа административных ролей, заданные функцией $APA \in APA^*$, удовлетворяют ограничению C^{APA}, если выполняется равенство $C^{APA}(APA) = \texttt{true}$;
— $C^{PA} : PA^* \to \{\texttt{true}, \texttt{false}\}$ — функция, задающая ограничение на значения множеств прав доступа ролей и административных ролей, т. е. по определению множества прав доступа ролей и административных ролей, заданные функцией $PA \in PA^*$, удовлетворяют ограничению C^{PA}, если выполняется равенство $C^{PA}(PA) = \texttt{true}$;
— $C^{AA} : AA^* \to \{\texttt{true}, \texttt{false}\}$ — функция, задающая ограничение на значение множества административных доступов субъект-сессий к ролям или административным ролям, т. е. по определению множества административных доступов субъект-сессий к ролям или административным ролям, заданные множеством $AA \in AA^*$, удовлетворяют ограничению C^{AA}, если выполняется равенство $C^{AA}(AA) = \texttt{true}$;
— $Constraint_{APA} = \{C_1^{APA}, \ldots, C_{n_{APA}}^{APA}\}$ — множество функций, задающих некоторый набор ограничений на значения множеств административных прав доступа административных ролей, где $n_{APA} \geqslant 0$. При этом будем писать $Constraint_{APA}(APA) = \texttt{true}$ тогда и только тогда, когда либо $n_{APA} = 0$, либо $n_{APA} > 0$ и $C_i^{APA}(APA) = \texttt{true}$ для $1 \leqslant i \leqslant n_{APA}$;
— $Constraint_{PA} = \{C_1^{PA}, \ldots, C_{n_{PA}}^{PA}\}$ — множество функций, задающих некоторый набор ограничений на значения множеств прав доступа ролей, где $n_{PA} \geqslant 0$. При этом будем писать $Constraint_{PA}(PA) = \texttt{true}$ тогда и только тогда, когда либо $n_{PA} = 0$, либо $n_{PA} > 0$ и $C_i^{PA}(PA) = \texttt{true}$ для $1 \leqslant i \leqslant n_{PA}$;
— $Constraint_{AA} = \{C_1^{AA}, \ldots, C_{n_{AA}}^{AA}\}$ — множество функций, задающих некоторый набор ограничений на значение множества административных доступов субъект-сессий к ролям или административным ролям, где $n_{AA} \geqslant 0$. При этом будем писать $Constraint_{AA}(AA) = \texttt{true}$ тогда и только тогда, когда либо $n_{AA} = 0$, либо $n_{AA} > 0$ и $C_i^{AA}(AA) = \texttt{true}$ для $1 \leqslant i \leqslant n_{AA}$.

Механизм ограничений как мощное средство ролевого управления доступом целесообразно реализовать, как минимум, на перспективу. Ограничения, например, вза-

имного исключения ролей, количественные на обладание ролью, могут стать удобным средством администрирования защищённой ОС. Ограничениям на функции UA и $roles$, заданным в моделях семейства $RBAC$ и других ролевых ДП-моделях, в МРОСЛ ДП-модели соответствуют ограничения на функцию APA и множество AA.

Определение 4. Пусть определены:

— множества учётных записей пользователей U, сущностей E, субъект-сессий S, прав доступа к сущностям P, учётных записей доверенных пользователей L_U, доверенных субъект-сессий L_S, доступов субъект-сессий к сущностям A, доступов субъект-сессий к ролям и административным ролям AA, информационных потоков F, ограничений на значения множеств административных прав доступа административных ролей $Constraint_{APA}$, множеств прав доступа ролей $Constraint_{PA}$, множеств доступов субъект-сессий к ролям или административным ролям $Constraint_{AA}$;

— функции административных прав доступа к ролям административных ролей APA, прав доступа ролей и административных ролей PA, принадлежности субъект-сессий учётным записям пользователей $user$, уровней конфиденциальности (f_u, f_e, f_r, f_s), мандатных атрибутов конфиденциальности CCR, уровней целостности (i_u, i_e, i_r, i_s), мандатных атрибутов целостности $CCRI$, иерархии ролей H_R, иерархии сущностей H_E, иерархии субъект-сессий H_S.

Определим $G = (APA, PA, user, (f_u, f_e, f_r, f_s), CCR, (i_u, i_e, i_r, i_s), CCRI, A, AA, F, H_R, H_E, H_S, Constraint_{APA}, Constraint_{PA}, Constraint_{AA}, L_U, L_S)$ — состояние системы.

Используем обозначение:

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

Используемые в МРОСЛ ДП-модели правила преобразования состояний из множества OP классифицированы на де-юре правила, реализуемые в реальной защищённой ОС, т. е. приводящие к «реальным» изменениям её параметров (изменению множеств прав доступа ролей, получению доступов субъект-сессий к сущностям или ролям и т. д.), и де-факто правила, не требующие реализации в ОС, так как используются в модели для отражения факта получения субъект-сессией де-факто владения субъект-сессиями или факта реализации информационного потока по памяти или по времени.

2. Параметрически ассоциированные сущности. Специальные административные роли

Сформулируем предположения, позволяющие учесть наличие в реальных защищённых ОС сущностей, параметрически ассоциированных с учётными записями пользователей или ролями.

Предположение 3. Для каждой учётной записи пользователя задаётся множество сущностей, параметрически с ней ассоциированных, получение доступов на чтение или на запись к которым субъект-сессией позволяет ей создать субъект-сессию от имени данной учётной записи пользователя. Множество сущностей, параметрически ассоциированных с учётной записью пользователя, не изменяется в процессе функционирования системы.

Предположение 4. Для каждой роли или административной роли задаётся (возможно, пустое) множество сущностей, параметрически с ней ассоциированных. При этом для получения или удаления субъект-сессией доступа к роли, кроме наличия к данной роли соответствующего права доступа, этой субъект-сессии необходимо

получить доступ на чтение или на запись ко всем сущностям, параметрически ассоциированным с данной ролью. Множество сущностей, параметрически ассоциированных с ролью или административной ролью, не изменяется в процессе функционирования системы. Множество сущностей, параметрически ассоциированных с каждой индивидуальной административной ролью учётной записи пользователя, совпадает с множеством сущностей, параметрически ассоциированных с данной учётной записью пользователя.

Примерами параметрически ассоциированных с учётными записями пользователей или ролями сущностей являются файлы паролей, `cookies` или ключей и т. д., т. е. то, «знание» (получение доступа на чтение) или «подмена» (получение доступа на запись) чего позволяет недоверенной субъект-сессии создать в системе субъект-сессию от имени данной учётной записи пользователя. Лишь у небольшого числа ролей могут быть такие сущности, их наличие позволяет обеспечить дополнительную защиту при получении субъект-сессией доступа к роли, например, потребовав дополнительно повторно ввести пароль.

С учётом сделанных предположений используем следующие обозначения:

— $]u[\in E$ — множество сущностей, параметрически ассоциированных с учётной записью пользователя $u \in U$;
— $UE = \{e \in]u[\colon u \in U\}$ — множество сущностей, каждая из которых параметрически ассоциирована хотя бы с одной учётной записью пользователя;
— $]r[\in E$ — множество сущностей, параметрически ассоциированных с ролью или административной ролью $r \in R \cup AR$, при этом для каждой учётной записи пользователя $u \in U$ в соответствии с предположением 4 выполняется условие $]u[=]u\_admin\_i\_low[$, и если $i_u(u) = i\_high$, то $]u[=]u\_admin\_i\_high[$;
— $RE = \{e \in]r[\colon r \in R \cup AR\}$ — множество сущностей, параметрически ассоциированных хотя бы с одной ролью или административной ролью.

Предположение 5. В рамках МРОСЛ ДП-модели на траекториях функционирования системы не изменяются функции $Constraint_{APA}$, $Constraint_{PA}$ и $Constraint_{AA}$. Новые значения для функций f_s и i_s задаются только при создании соответствующих новых субъект-сессий. В дополнение к предположению 1, недоверенная субъект-сессия всегда может создать недоверенную субъект-сессию с низким уровнем доступа. В начальном состоянии G_0 любой системы $\Sigma(G^*, OP)$ выполняются следующие условия:

— функции APA_0, PA_0 и множество AA_0 удовлетворяют соответствующим ограничениям $Constraint_{APA}$, $Constraint_{PA}$ и $Constraint_{AA}$;
— для любых субъект-сессии $s \in S_0$ и сущности $e \in E_0$, если $(s, e, \alpha_a) \in A_0$, где $\alpha_a \in R_a$, то справедливо равенство $execute\_container_0(s, e) = \texttt{true}$.

В предположении 5 требуется, чтобы в начальном состоянии выполнялись ограничения и доступы субъект-сессий к сущностям соответствовали мандатному управлению доступом и мандатному контролю целостности. Кроме того, для строгого задания правил преобразования состояний системы, ориентированных на реализацию в реальной защищённой ОС, впервые по сравнению с предшествующими ДП-моделями в рамках МРОСЛ ДП-модели допускается администрирование существенных параметров системы, влияющих на её безопасность (множеств учётных записей пользователей, ролей или административных ролей, значений уровней доступа, конфиденциальности или целостности учётных записей пользователей, ролей, административных ролей, сущностей). Для этого в модель включены специальные административные роли.

Используем следующие обозначения:

— $ADMIN\_ROLES \subset AR$ — множество специальных административных ролей, которые не могут создаваться, удаляться, переименовываться, менять свои параметры в процессе функционирования системы. Все административные роли из множества $ADMIN\_ROLES$ по определению обладают высоким уровнем целостности: для административной роли $ar \in ADMIN\_ROLES$ справедливо равенство $i_r(ar) = i\_high$;

— $users\_admin\_role \in ADMIN\_ROLES$ — специальная административная роль, доступ субъект-сессии на чтение к которой требуется для изменения уровня доступа или целостности, удаления учётной записи пользователя;

— $entities\_admin\_role \in ADMIN\_ROLES$ — специальная административная роль, доступ субъект-сессии на чтение к которой требуется для изменения ею единственной роли-владельца сущности (роли, обладающей правом доступа владения own_r к сущности), получения данных о ролях или административных ролях, обладающих правами доступа к сущности, получения числа «жёстких» ссылок к сущности, изменения уровня конфиденциальности или целостности сущности, изменения мандатного атрибута конфиденциальности CCR, мандатного атрибута целостности $CCRI$ или метки разделяемости сущности-контейнера, переименования, удаления сущности-контейнера или «жёсткой» ссылки на него с мандатным атрибутом целостности $CCRI$ равным `false`;

— $subjects\_admin\_role \in ADMIN\_ROLES$ — специальная административная роль, доступ субъект-сессии на чтение к которой требуется для изменения единственной роли-владельца другой субъект-сессии, получения данные о ролях или административных ролях, обладающих правами доступа владения к другой субъект-сессии;

— $roles\_admin\_role, admin\_roles\_admin\_role \in ADMIN\_ROLES$ — специальные административные роли, являющиеся ролями-владельцами ролей или административных ролей соответственно, доступ субъект-сессии на чтение к каждой из которых требуется для управления доступом к роли или административной роли, изменения её уровней конфиденциальности или целостности, создания, переименования, удаления, получения параметров роли или административной роли, создания, изменения уровня доступа или целостности учётной записи пользователя;

— $downgrade\_admin\_role \in ADMIN\_ROLES$ — специальная административная роль, доступ субъект-сессии на чтение к которой требуется для нарушения ею правил мандатного управления доступом.

По определению справедливы равенства

$$f_r(users\_admin\_role) = f_r(entities\_admin\_role) = f_r(subjects\_admin\_role) = \\ = f_r(roles\_admin\_role) = f_r(admin\_roles\_admin\_role) = \otimes LC, \\ f_r(downgrade\_admin\_role) = \oplus LC.$$

3. Де-юре правила администрирования системы

Всего в рамках МРОСЛ ДП-модели заданы 34 де-юре правила преобразования состояний, условия и результаты применения которых соответствуют предположениям модели. Правила предназначены для формального описания (спецификации) следующих основных функций механизма управления доступом защищённой ОС:

— создания, удаления, переименования, получения или изменения параметров учётных записей пользователей, ролей, административных ролей, сущностей или «жёстких» ссылок на них, субъект-сессий;

— получения доступов субъект-сессий к сущностям, ролям или административным ролям;
— изменения прав доступа ролей или административных ролей к сущностям, субъект-сессиям, ролям или административным ролям;
— изменения иерархии, уровней конфиденциальности или целостности сущностей, ролей или административных ролей.

В таблице приведено детальное описание де-юре правил администрирования системы, при этом в ней не указываются элементы последующего состояния G', которые не изменяются в результате применения соответствующего правила к исходному состоянию G.

Де-юре правила администрирования системы в рамках МРОСЛ ДП-модели

Правило: *create_user*(x, x', u, uc, ui, ue)
Условия применения: $x, x' \in S$, $u \notin U$, $(x, users\_admin\_role, read_a) \in AA$, $(x, roles\_admin\_role, \alpha_a) \in AA$, $(x, admin\_roles\_admin\_role, \alpha_a) \in AA$, где $\alpha_a \in \{read_a, write_a\}$; $ue \subset UE$; [$uc = f_s(x)$ или ($uc \leqslant f_s(x)$ и $(x, downgrade\_admin\_role, read_a) \in AA$)]; $ui \leqslant i_s(x)$; [для $e \in ue$ выполняется $f_e(e) = uc$, $i_e(e) = ui$]; $Constraint_{APA}(APA') = \texttt{true}$; $Constraint_{PA}(PA') = \texttt{true}$; [если $ui = i\_high$, то $(x', f_s(x)\_i\_entity, write_a) \in A$]
Результаты применения: $U' = U \cup \{u\}$, $]u[= ue$, $f'_u(u) = uc$, $i'_u(u) = ui$, если $ui = i\_high$, то $L'_U = L_U \cup \{u\}$, иначе $N'_U = N_U \cup \{u\}$, для $u' \in U$ выполняется $f'_u(u') = f_u(u')$, $i'_u(u') = i_u(u')$, $AR' = AR \cup \{u\_admin\_li : li \leqslant ui\}$, $f'_r(u\_admin\_li) = \otimes LC$, $i'_r(u\_admin\_li) = li$, $shared\_container'(u\_admin\_li) = \texttt{true}$, $CCR'(u\_admin\_li) = CCRI'(u\_admin\_li) = \texttt{false}$, $role\_name'(u\_admin\_li) =$“u_admin_li”, где $li \leqslant ui$, $H'_R(u\_admin\_i\_low) = \varnothing$, если $ui = i\_high$, то $H'_R(u\_admin\_i\_high) = \{u\_admin\_i\_low\}$, $R' = R \cup \{u\_c\_l\_li : l \leqslant uc, li \leqslant ui\}$, $f'_r(u\_c\_l\_li) = l$, $i'_r(u\_c\_l\_li) = li$, $shared\_container'(u\_c\_l\_li) = \texttt{true}$, $CCR'(u\_c\_l\_li) = CCRI'(u\_c\_l\_li) = \texttt{false}$, $role\_name'(u\_c\_l\_li) =$ = “$u\_c\_l\_li$”, где $l \leqslant uc$, $li \leqslant ui$, $APA'(admin\_roles\_admin\_role) = APA(admin\_roles\_admin\_role) \cup \{(u\_admin\_li, own_r) : li \leqslant ui\}$, $APA'(roles\_admin\_role) = APA(roles\_admin\_role) \cup \{(u\_c\_l\_li, own_r) : l \leqslant uc, li \leqslant ui\}$, для $ar \in AR$ выполняется $APA'(ar) = APA(ar) \cup \{(u\_admin\_li, execute_r) : li \leqslant ui\} \cup \{(u\_c\_l\_li, execute_r) : l \leqslant uc, li \leqslant ui\}$, $APA'(u\_admin\_i\_low) = \{(u\_admin\_i\_low, \alpha_r) : \alpha_r \in \{read_r, write_r, execute_r\}\} \cup$ $\cup \{(u\_c\_l\_(i\_low), \alpha_r) : l \leqslant uc, \alpha_r \{read_r, write_r, execute_r\}\}$, если $ui = i\_high$, то $APA'(u\_admin\_i\_high) = \{(u\_admin\_li, \alpha_r) : li \leqslant ui, \alpha_r \in \{read_r, write_r, execute_r\}\} \cup \{(u\_c\_l\_(i\_high), \alpha_r) : l \leqslant uc, \alpha_r \in \{read_r, write_r, execute_r\}\}$, $H'_R(u\_c\_l\_li) = \{u\_c\_l'\_li' : l' < l, li' < li$ и не существуют $l'' \in LC (l' < l'' < l)$ или $li'' \in LI$ $(li' < li'' < li)\}$, $PA'(u\_c\_l\_li) = \varnothing$, где $l \leqslant uc$, $li \leqslant ui$
Правило: *set_user_labels*(x, x', u, uc, ui, ue)
Условия применения: $x, x' \in S$, $u \in U$, $user^{-1}(u) = \varnothing$, ($x$, $users\_admin\_role$, $read_a$) $\in AA$, (x, $roles\_admin\_role$, α_a) $\in AA$, (x, $admin\_roles\_admin\_role$, α_a) $\in AA$, где $\alpha_a \in \{read_a, write_a\}$, [если $f_u(u) \neq uc$, то $\max(f_u(u), uc) \leqslant f_s(x)$ и (x, $downgrade\_admin\_role$, $read_a$) $\in AA$], $\max(i_u(u), ui) \leqslant i_s(x)$, [для $e \in ue$ верно $f_e(e) = uc$, $i_e(e) = ui$], $Constraint_{APA}(APA') = \texttt{true}$, $Constraint_{PA}(PA') = \texttt{true}$, [если $ui = i\_high$, то (x', $f_s(x)\_i\_entity$, $write_a$) $\in A$]

Продолжение таблицы

Результаты применения: $]u[= ue$, $f'_u(u) = uc$, $i'_u(u) = ui$, если $ui = i\_high$, то $L'_U = L_U \cup \{u\}$, $N'_U = N_U \setminus \{u\}$, иначе $N'_U = N_U \cup \{u\}$, $L'_U = L_U \setminus \{u\}$, для $u' \in U \setminus \{u\}$ верно $f'_u(u') = f_u(u')$, $i'_u(u') = i_u(u')$, $AR' = (A \cup \{u\_admin\_li{:}\ i_u(u) < li \leqslant ui\}) \setminus \{u\_admin\_li{:}\ ui < li \leqslant i_u(u)\}$, $f'_r(u\_admin\_li) = \otimes LC$, $i'_r(u\_admin\_li) = li$, $shared\_container'(u\_admin\_li) =$ `true`, $CCR'(u\_admin\_li) = CCRI'(u\_admin\_li) =$ `false`, $role\_name'(u\_admin\_li) =$ "u_admin_li", где $i_u(u) < li \leqslant ui$, если $ui = i\_high$, то $H'_R(u\_admin\_i\_high) = \{u\_admin\_i\_low\}$, $R' = (R \cup \{u\_c\_l\_li : f_u(u) < l \leqslant uc, i_u(u) < li \leqslant ui\}) \setminus \{u\_c\_l\_li : uc < l \leqslant f_u(u), ui < li \leqslant i_u(u)\}$, $f'_r(u\_c\_l\_li) = l$, $i'_r(u\_c\_l\_li) = li$, $shared\_container'(u\_c\_l\_li) =$ `true`, $CCR'(u\_c\_l\_li) = CCRI'(u\_c\_l\_li) =$ `false`, $role\_name'(u\_c\_l\_li) =$ "$u\_c\_l\_li$", где $f_u(u) < l \leqslant uc$, $i_u(u) < li \leqslant ui$, $APA'(admin\_roles\_admin\_role) = (APA(admin\_roles\_admin\_role) \cup \{(u\_admin\_li, own_r) : i_u(u) < li \leqslant ui\}) \setminus \{(u\_admin\_li, own_r) : ui < li \leqslant i_u(u)\}$, $APA'(roles\_admin\_role) = (APA(roles\_admin\_role) \cup \{(u\_c\_l\_li, own_r) : f_u(u) < l \leqslant uc, i_u(u) < li \leqslant ui\}) \setminus$ $\setminus \{(u\_c\_l\_li, own_r) : uc < l \leqslant f_u(u), ui < li \leqslant i_u(u)\}$, для $ar \in AR$ выполняется $APA'(ar) = (APA(ar) \cup \{(u\_admin\_li, execute_r) : i_u(u) < li \leqslant ui\} \cup$ $\cup \{(u\_c\_l\_li, execute_r) : f_u(u) < l \leqslant uc, i_u(u) < li \leqslant ui\}) \setminus (\{(u\_admin\_li, execute_r) : ui < li \leqslant$ $\leqslant i_u(u)\} \cup \{(u\_c\_l\_li, execute_r) : uc < l \leqslant f_u(u), ui < li \leqslant i_u(u)\})$, $APA'(u\_admin\_i\_low) =$ $= (APA'(u\_admin\_i\_low) \cup \{(u\_c\_l\_(i\_low), \alpha_r) : f_u(u) < l \leqslant uc, \alpha_r \in \{read_r, write_r, execute_r\}\}) \setminus \{(u\_c\_l\_(i\_low), \alpha_r) : uc < l \leqslant f_u(u), \alpha_r \in \{read_r, write_r, execute_r\}\}$, если $ui = i\_high$, то $APA'(u\_admin\_i\_high) = (APA(u\_admin\_i\_high) \cup \{(u\_admin\_li, \alpha_r) : li \leqslant ui, \alpha_r \in \{read_r, write_r, execute_r\}\} \cup \{(u\_c\_l\_(i\_high), \alpha_r) : f_u(u) < l \leqslant uc, \alpha_r \in \{read_r, write_r, execute_r\}\}) \setminus \{(u\_c\_l\_(i\_high), \alpha_r) : uc < l \leqslant f_u(u), \alpha_r \in \{read_r, write_r, execute_r\}\}$, $H'_R(u\_c\_l\_li) = \{u\_c\_l'\_li' : l' < l$, $li' < li$ и не существуют $l'' \in LC(l' < l'' < l)$ или $li'' \in LI$ $(li' < li'' < li)\}$, $PA'(u\_c\_l\_li) = \varnothing$, где $f_u(u) < l \leqslant uc$, $i_u(u) < li \leqslant ui$
Правило: $delete\_user(x, x', u)$
Условия применения: $x, x' \in S$, $u \in U$, $user^{-1}(u) = \varnothing$, $(x, users\_admin\_role, read_a) \in AA$, $(x, users\_admin\_role, read_a) \in AA$, $(x, roles\_admin\_role, read_a) \in AA$, $[f_u(u) = f_s(x)$ или $(f_u(u) \leqslant f_s(x)$ и $(x, downgrade\_admin\_role, read_a) \in AA)]$, $i_u(u) \leqslant i_s(x)$, $Constraint_{APA}(APA') =$ `true`, $Constraint_{PA}(PA') =$ `true`, [если $i_u(u) = i\_high$, то $(x', f_s(x)\_i\_entity, write_a) \in A$]
Результаты применения: $U' = U \setminus \{u\}$, для $u' \in U'$ выполняется $f'_u(u') = f_u(u')$, $i'_u(u') = i_u(u')$, $AR' = AR \setminus \{u\_admin\_li : li \leqslant ui\}$, $R' = R \setminus \{u\_c\_l\_li : l \leqslant uc, li \leqslant ui\}$, для $ar \in AR'$ выполняются равенства $APA'(ar) = APA(ar) \setminus (\{(u\_admin\_li, \alpha_r) : li \leqslant ui$, $\alpha_r \in R_r\} \cup \{(u\_c\_l\_li, \alpha_r) : l \leqslant uc, li \leqslant ui, \alpha_r \in R_r\})$
Правило: $get\_user\_attr(x, u, z)$
Условия применения: $x \in S$, $u \in U$, $z \in O$, $[f_u(u) \leqslant f_s(x)$ или $(x, downgrade\_admin\_role, read_a) \in AA]$, $(x, z, write_a) \in A$
Результаты применения: $V'(z) = (f_u(u), i_u(u))$, (если $user(x) = u$ или $(x, users\_admin\_role, read_a) \in AA$, то $\{(r', \alpha_r) : (r', \alpha_r) \in APA(u\_admin\_li)$, $li \leqslant i_u(u)$ и $(f_r(r') \leqslant f_s(x)$ или $(x, downgrade\_admin\_role, read_a) \in AA)\}$, иначе "$\varnothing$"), (если $user(x) = u$ или $(x, users\_admin\_role, read_a) \in AA$, то $\{s \in S{:}\ user(s) = u$ и либо $f_s(s) \leqslant f_s(x)$ или $(x, downgrade\_admin\_role, read_a) \in AA)\}$, иначе "$\varnothing$"))
Правило: $grant\_rights(x, x', r, \{(y, \alpha_{r_j}){:}\ 1 \leqslant j \leqslant k\})$
Условия применения: $x, x' \in S$, $y \in E$, $r \in R \cup AR$, $\alpha_{r_j} \in \{read_r, write_r, execute_r\}$, $(x, r, write_a) \in AA$, $i_e(y) \leqslant i_s(x)$, [существует $r' \in R \cup AR$ $((x, r', read_a) \in AA$, $(y, own_r) \in PA(r'))]$, [либо $(execute\_container(x, y) =$ `true` и $f_e(y) = f_s(x))$, либо $(x, downgrade\_admin\_role, read_a) \in AA]$, [если $\alpha_{r_j} = write_r$, то $i_e(y) \leqslant i_r(r)]$, $Constraint_{PA}(PA') =$ `true`, [если $i_e(y) = i\_high$, то $(x', f_s(x)\_i\_entity, write_a) \in A]$, где $1 \leqslant j \leqslant k$
Результаты применения: $PA'(r) = PA(r) \cup \{(y, \alpha_{r_j}) : 1 \leqslant j \leqslant k\}$ и для $r' \in (R \cup AR) \setminus \{r\}$ выполняется равенство $PA'(r') = PA(r')$

Продолжение таблицы

Правило: *remove_rights*(x, x', r, $\{(y, \alpha_{r_j})$: $1 \leqslant j \leqslant k\}$)
Условия применения: $x, x' \in S$, $y \in E$, $r \in R \cup AR$, $\alpha_{r_j} \in \{read_r, write_r, execute_r\}$, $\{(y, \alpha_{r_j}) : 1 \leqslant j \leqslant k\} \subset PA(r)$, $(x, r, write_a) \in AA$, $i_e(y) \leqslant i_s(x)$, [существует $r' \in R \cup AR$ ($(x, r', read_a) \in AA$, $(y, own_r) \in PA(r')$)], [либо ($execute\_container(x, y) =$ `true` и $f_e(y) = f_s(x)$), либо $(x, downgrade\_admin\_role, read_a) \in AA$], $Constraint_{PA}(PA') =$ `true`, [если $i_e(y) = i\_high$, то $(x', f_s(x)\_i\_entity, write_a) \in A$], где $1 \leqslant j \leqslant k$
Результаты применения: $PA'(r) = PA(r) \setminus \{(y, \alpha_{r_j}) : 1 \leqslant j \leqslant k\}$, и для $r' \in (R \cup AR) \setminus \{r\}$ выполняется равенство $PA'(r') = PA(r')$
Правило: *set_entity_owner*(x, x', r, r', y)
Условия применения: $x, x' \in S$, $y \in E$, $r, r' \in R \cup AR$, $\{(x, r, read_a)$, $(x, r, write_a)$, $(x, r', write_a)$, $(x, entities\_admin\_role, read_a)\} \subset AA$, $(y, own_r) \in PA(r)$, $Constraint_{PA}(PA') =$ `true`, $i_e(y) \leqslant \min(i_r(r'), i_s(x))$, [либо ($execute\_container(x, y) =$ `true` и $f_e(y) = f_s(x)$), либо $((x, downgrade\_admin\_role, read_a) \in AA)$], [если $i_e(y) = i\_high$, то $(x', f_s(x)\_i\_entity, write_a) \in A$]
Результаты применения: $PA'(r) = PA(r) \setminus \{(y, own_r)\}$, $PA'(r') = PA(r') \cup \{(y, own_r)\}$, для $r'' \in (R \cup AR) \setminus \{r, r'\}$ выполняется равенство $PA'(r'') = PA(r'')$
Правило: *set_subject_owner*(x, x', r, r', y)
Условия применения: $x, x' \in S$, $y \in S$, $r, r' \in R \cup AR$, $\{(x, r, read_a)$, $(x, r, write_a)$, $(x, r', write_a)$, $(x, subjects\_admin\_role, read_a)\} \subset AA$, $(y, own_r) \in PA(r)$, $Constraint_{PA}(PA') =$ `true`, $i_s(y) \leqslant$ $\leqslant \min(i_r(r'), i_s(x))$, [либо $f_s(y) = f_s(x)$, либо $(x, downgrade\_admin\_role, read_a) \in AA$], [если $i_s(y) = i\_high$, то $(x', f_s(x)\_i\_entity, write_a) \in A$]
Результаты применения: $PA'(r) = PA(r) \setminus \{(y, own_r)\}$, $PA'(r') = PA(r') \cup \{(y, own_r)\}$, для $r'' \in (R \cup AR) \setminus \{r, r'\}$ выполняется равенство $PA'(r'') = PA(r'')$
Правило: *grant_admin_rights*(x, x', ar, $\{(r, \alpha_{r_j})$: $1 \leqslant j \leqslant k\}$)
Условия применения: $x, x' \in S$, $r \in R \cup AR$, $ar \in AR$, $\alpha_{r_j} \in \{write_r, read_r\}$, $(x, ar, write_a) \in AA$, $i_r(r) \leqslant i_r(ar)$, $i_r(r) \leqslant i_s(x)$, [если $r \in R$, то $(x, roles\_admin\_role, read_a) \in AA$, если $r \in AR$, то $(x, admin\_roles\_admin\_role, read_a) \in AA$], [либо $f_r(r) = f_s(x)$, либо $(x, downgrade\_admin\_role, read_a) \in AA$], $Constraint_{APA}(APA') =$ `true`, [если $i_r(r) = i\_high$, то $(x', f_s(x)\_i\_entity, write_a) \in A$], где $1 \leqslant j \leqslant k$
Результаты применения: $APA'(ar) = APA(ar) \cup \{(r, \alpha_{r_j}) : \alpha_{r_j} = write_r, 1 \leqslant j \leqslant k\} \cup \{(r', \alpha_{r_j})$: $\alpha_{r_j} = read_r$, $r' \leqslant r$, $1 \leqslant j \leqslant k\}$, для $ar' \in AR \setminus \{ar\}$ выполняется равенство $APA'(ar') = APA(ar')$
Правило: *remove_admin_rights*(x, x', ar, $\{(r, \alpha_{r_j})$: $1 \leqslant j \leqslant k\}$)
Условия применения: $x, x' \in S$, $r \in R \cup AR$, $ar \in AR$, $\alpha_{r_j} \in \{write_r, read_r\}$, $\{(r, \alpha_{r_j}) : \alpha_{r_j} = write_r, 1 \leqslant j \leqslant k\} \subset APA(ar)$, $(x, ar, write_a) \in AA$, $i_r(r) \leqslant i_s(x)$, [если $r \in R$, то $(x, roles\_admin\_role, read_a) \in AA$, если $r \in AR$, то $(x, admin\_roles\_admin\_role, read_a) \in AA$], [либо $f_r(r) = f_s(x)$, либо $(x, downgrade\_admin\_role, read_a) \in AA$], $Constraint_{APA}(APA') =$ `true`, [если $i_r(r) = i\_high$, то $(x', f_s(x)\_i\_entity, write_a) \in A$], где $1 \leqslant j \leqslant k$
Результаты применения: $APA'(ar) = APA(ar) \setminus (\{(r, \alpha_{r_j}) : \alpha_{r_j} = write_r, 1 \leqslant j \leqslant k\} \cup \{(r', \alpha_{r_j})$: $\alpha_{r_j} = read_r$, $r \leqslant r'$, $1 \leqslant j \leqslant k\})$, для $ar' \in AR \setminus \{ar\}$ верно равенство $APA'(ar') = APA(ar')$
Правило: *create_role*(x, x', r, rc, ri, re, $name$, rz)
Условия применения: $x, x' \in S$, $r \in R \cup AR$, [если $rz \in R$, то $(x, roles\_admin\_role, \alpha_a) \in AA$, если $rz \in AR$, то $(x, admin\_roles\_admin\_role, \alpha_a) \in AA$, где $\alpha_a \in \{read_a, write_a\}$], $(x, rz, write_a) \in AA$, $re \subset RE$, $name \in NAME \setminus \{«»\}$, [либо $r = f_r(rz) = f_s(x)$, либо $r \leqslant f_r(rz)$ и $(x, downgrade\_admin\_role, read_a) \in AA$], $ri \leqslant \min(i_r(rz), i_s(x))$, [для $e \in re$ выполняется $f_e(e) = rc$, $i_e(e) = ri$], $Constraint_{APA}(APA') =$ `true`, $Constraint_{PA}(PA') =$ `true`, [если $i_r(rz) = i\_high$, то $(x', f_s(x)\_i\_entity, write_a) \in A$]
Результаты применения: если $rz \in R$, то $R' = R \cup \{r\}$, $APA'(roles\_admin\_role) = APA(roles\_admin\_role) \cup \{(r, own_r), (r, execute_r)\} \cup \{(r, read_r) : (rz, read_r) \in APA(roles\_admin\_role)\}$, если $rz \in AR$, то $AR' = AR \cup \{r\}$, $APA'(admin\_roles\_admin\_role) = APA(admin\_roles\_admin\_role) \cup \{(r, own_r), (r, execute_r)\} \cup \{(r, read_r) : (rz, read_r) \in APA(admin\_roles\_admin\_role)\}$, для $ar \in AR \setminus \{admin\_roles\_admin\_role, roles\_admin\_role\}$ выполняется $APA'(ar) = APA(ar) \cup \{(r, execute_r)\} \cup \{(r, read_r) : (rz, read_r) \in APA(ar)\}$, $f'_r(r) = rc$, $i'_r(r) = ri$, $CCR'(r) =$ `false`, $CCRI'(r) =$ `false`, $role\_name'(rz, r) = name$, $]r[= re$, $shared\_container'(r) =$ `true`, $PA'(r) = \varnothing$, $H'_R(rz) = H_E(z) \cup \{r\}$, $H'_R(r) = \varnothing$, для $r \in (R \cup AR) \setminus \{r\}$ выполняется равенство $H'_R(r) = H_R(r)$

Окончание таблицы

Правило: *delete_role*(x, x', r, rz)
Условия применения: $x, x' \in S$, $r \in (R \cup AR) \setminus (\{u\_admin\_li : u \in U, li \leqslant i_u(u)\} \cup \{u\_c\_l\_li : u \in U,\ l \leqslant f_u(u),\ li \leqslant i_u(u)\} \cup ADMIN\_ROLES)$, [если $r \in R$, то $rz \in R$, $(x, roles\_admin\_role, \alpha_a) \in AA$, если $r \in AR$, то $rz \in AR$, $(x, admin\_roles\_admin\_role, \alpha_a) \in AA$, где $\alpha_a \in \{read_a, write_a\}$], $r \in H_R(rz)$, $HR(r) = \varnothing$, [не существует $rz' \in R \cup AR$, такой, что $rz' \neq rz$ и $r \in H_R(rz')$], [$(x, rz, write_a) \in AA$, $Constraint_{AA}(AA') =$ `true`, $Constraint_{APA}(APA') =$ `true`, $Constraint_{PA}(PA') =$ `true`], [либо $f_r(r) = f_s(x)$, либо $f_r(r) \leqslant f_s(x)$ и $(x, downgrade\_admin\_role, read_a) \in AA$], $i_r(r) \leqslant i_s(x)$, [если $i_e(z) = i\_high$, то $(x', f_s(x)\_i\_entity, write_a) \in A$]
Результаты применения: $R' = R \setminus \{r\}$, $AR' = AR \setminus \{r\}$, $H'_R(rz) = H_R(rz) \setminus \{r\}$, для всех $r' \in (R' \cup AR') \setminus \{rz\}$ выполняется равенство $H'_R(r') = H_R(r')$, для $ar \in AR'$ выполняются равенства $APA'(ar) = APA(ar) \setminus \{(r, \alpha_r) : \alpha_r \in R_r\}$, $AA' = AA \setminus \{(s, r, \alpha_a) : s \in S, \alpha_a \in R_a\}$
Правило: *create_hard_link_role*$(x, x', r, name, rz)$
Условия применения: $x, x' \in S$, $r \in (R \cup AR) \setminus (\{u\_admin\_li : u \in U, li \leqslant i_u(u)\} \cup \{u\_c\_l\_li : u \in U, l \leqslant f_u(u), li \leqslant i_u(u)\} \cup ADMIN\_ROLES)$, не выполняется условие $rz \leqslant r$, $Constraint_{APA}(APA') =$ `true`, [если $rz \in R$, то $(x, roles\_admin\_role, \alpha_a) \in AA$, если $rz \in AR$, то $(x, admin\_roles\_admin\_role, \alpha_a) \in AA$, где $\alpha_a \in \{read_a, write_a\}$], $(x, rz, write_a) \in AA$, $name \in NAME \setminus \{«»\}$, [либо $f_r(r) = f_r(rz) = f_s(x)$, либо $f_r(r) \leqslant f_r(rz)$ и $(x, downgrade\_admin\_role, read_a) \in AA$], $i_r(r) \leqslant \min(i_r(rz), i_s(x))$, [если $i_r(rz) = i\_high$, то $(x', f_s(x)\_i\_entity, write_a) \in A$]
Результаты применения: $role\_name'(rz, r) = name$, $H'_R(rz) = H_R(rz) \cup \{r\}$, $H'_R(r) = \varnothing$, для $r' \in (R \cup AR) \setminus \{r\}$ выполняется равенство $H'_R(r') = H_R(r')$, для $ar \in AR$, таких, что $(rz, read_r) \in APA(ar)$, выполняется $APA'(ar) = APA(ar) \cup \{(r', read_r) : r' \leqslant r\}$
Правило: *delete_hard_link_role*(x, x', r, rz)
Условия применения: $x, x' \in S$, $r \in (R \cup AR) \setminus (\{u\_admin\_li : u \in U, li \leqslant i_u(u)\} \cup \{u\_c\_l\_li : u \in U, l \leqslant f_u(u), li \leqslant i_u(u)\} \cup ADMIN\_ROLES)$, [если $rz \in R$, то $(x, roles\_admin\_role, \alpha_a) \in AA$, если $rz \in AR$, то $(x, admin\_roles\_admin\_role, \alpha_a) \in AA$, где $\alpha_a \in \{read_a, write_a\}$], $r \in H_R(rz)$, [существует $rz' \in R \cup AR$, такая, что $rz' \neq rz$ и $r \in H_R(rz')$], $(x, rz, write_a) \in AA$, [либо $f_r(r) = f_s(x)$, либо $f_r(r) \leqslant f_s(x)$ и $(x, downgrade\_admin\_role, read_a) \in AA$], $i_r(r) \leqslant i_s(x)$, [если $i_e(z) = i\_high$, то $(x', f_s(x)\_i\_entity, write_a) \in A$]
Результаты применения: $H'_R(rz) = H_R(rz) \setminus \{y\}$, для $r' \in (R \cup AR) \setminus \{r\}$ верно $H'_R(r') = H_R(r')$
Правило: *set_container_attr*$(x, x', y, ccr, ccri, t)$
Условия применения: $x, x' \in S$, $y \in C$, $ccr, ccri, t \in \{$`true`, `false`$\}$, $i_e(y) \leqslant i_s(x)$, [либо существует $r \in R \cup AR$ ($(x, r, read_a) \in AA$, $(y, own_r) \in PA(r)$), либо $(x, entities\_admin\_role, read_a) \in AA$], [либо ($execute\_container(x, y) =$ `true` и $f_e(y) = f_s(x)$), либо $(x, downgrade\_admin\_role, read_a) \in AA$], [$(x, entities\_admin\_role, read_a) \in AA$ или $(x, downgrade\_admin\_role, read_a) \in AA$], [если $i_e(y) = i\_high$, то $(x', f_s(x)\_i\_entity, write_a) \in A$]
Результаты применения: $CCR'(y) = ccr$, $CCRI'(y) = ccri$, $shared\_container'(y) = t$
Правило: *set_entity_labels*(x, x', y, yc, yi)
Условия применения: $x, x' \in S$, $y \in E$, $y \notin UE \cup RE$, $(x, entities\_admin\_role, read_a)$, $(x, downgrade\_admin\_role, read_a) \in AA$, $\{(s, y, \alpha_a) : s \in S, \alpha_a \in R_a\} \cap A = \varnothing$, $\max(f_e(y), yc) \leqslant f_s(x)$, $\max(i_e(y), yi) \leqslant i_s(x)$, [$yc \leqslant f_e(z)$, $yi \leqslant i_e(z)$ для всех $z \in E$, таких, что $y \in H_E(z)$], [$f_e(z) \leqslant yc$, $i_e(z) \leqslant yi$ для всех $z \in E$, таких, что $z \in H_E(y)$], [если $i_e(y) = i\_high$ или $yi = i\_high$, то $(x', f_s(x)\_i\_entity, write_a) \in A$]
Результаты применения: $f'_e(y) = yc$, $i'_e(y) = yi$
Правило: *set_role_labels*(x, x', r, rc, re)
Условия применения: $x, x' \in S$, $r \in (R \cup AR) \setminus (\{u\_admin\_li : u \in U, li \leqslant i_u(u)\} \cup \{u\_c\_l\_li : u \in U, l \leqslant f_u(u), li \leqslant i_u(u)\} \cup ADMIN\_ROLES)$, $(x, downgrade\_admin\_role, read_a) \in AA$, [если $r \in R$, то $(x, roles\_admin\_role, read_a) \in AA$, если $r \in AR$, то $(x, admin\_roles\_admin\_role, read_a) \in AA$], $\max(f_r(r), rc) \in f_s(x)$, $\{(s, r, \alpha_a) : s \in S, \alpha_a \in R_a\} \cap AA = \varnothing$, [$rc \leqslant f_r(rz)$ для всех $rz \in R \cup AR$, таких, что $r \in H_R(rz)$], [$f_r(rz) \leqslant rc$ для всех $rz \in R \cup AR$, таких, что $rz \in H_R(r)$], $re \subset RE$, [для $e \in re$ выполняется $f_e(e) = rc$, $i_e(e) = i_r(r)$], [если $i_r(r) = i\_high$, то $(x', f_s(x)\_i\_entity, write_a) \in A$]
Результаты применения: $f'_r(r) = rc$, $]r[= re$

Рассмотрим условия и результаты применения де-юре правил преобразования состояний.

Де-юре правила $create\_user(x, x', u, uc, ui, ue)$, $set\_user\_labels(x, x', u, uc, ui, ue)$, $delete\_user(x, x', u)$ и $get\_user\_attr(x, u, z)$ позволяют субъект-сессии x создать, изменить уровни конфиденциальности и целостности, удалить или получить параметры учётной записи пользователя u с уровнем доступа, равным текущему уровню доступа субъект-сессии x (за исключением случая, когда субъект-сессия x обладает административным доступом на чтение к специальной административной роли $downgrade\_admin\_role$), и уровнем целостности, не превосходящим текущего уровня целостности субъект-сессии x (кроме случая получения параметров). Во всех случаях, кроме получения параметров, требуется наличие у субъект-сессии x доступа на чтение к специальной административной роли $users\_admin\_role$, а также в последующем состоянии системы — выполнение ограничений на значения множеств административных прав доступа административных ролей $Constraint_{APA}$ (с учётом создания или удаления индивидуальных административных ролей и индивидуальных ролей учётной записи пользователя u), а если осуществляются действия над учётной записью пользователя u с высоким уровнем целостности, то необходимо обладание некоторой кооперирующей с x (либо самой x) субъект-сессией x' доступом на запись к сущности $f_s(x)\_i\_entity$, что моделирует ситуацию подачи сигнала системе (нажатия соответствующей кнопки), подтверждающего переход на высокий уровень целостности. При этом в последующем состоянии иерархия ролей модифицируется (создаются или удаляются индивидуальные роли или индивидуальные административные роли, назначаются или удаляются административные права доступа к ним).

Следует заметить, что эти действия осуществляются без явного получения субъект-сессией x административного доступа на запись ко всем административным ролям, административные права доступа которых модифицируются, что сделано, во-первых, для удобства реализации правил в реальной защищённой ОС, во-вторых, для использования правил требуются специальные административные роли $roles\_admin\_role$ и $admin\_roles\_admin\_role$, доступные только доверенным субъект-сессиям, не участвующим в реализации запрещённых информационных потоков по времени. Для создания, удаления и изменения уровней конфиденциальности и целостности учётной записи пользователя требуется наличие у субъект-сессии x доступов на чтение и запись к специальным административным ролям $roles\_admin\_role$ и $admin\_roles\_admin\_role$, так как именно эти роли получают права доступа владения к создаваемым при применении правил индивидуальным административным ролям и индивидуальным ролям учётной записи пользователя u, при этом (в том числе при изменении параметров) задаются множества сущностей, параметрически ассоциированных с учётной записью пользователя u.

В случае изменения параметров учётной записи пользователя или её удаления требуется, чтобы в этот момент времени от её имени в системе не функционировала ни одна субъект-сессия. При получении параметров в сущность z, к которой субъект-сессия x должна иметь доступ на запись, всегда записываются уровни доступа и целостности учётной записи пользователя u (т. е. если в системе не предъявляются дополнительные требования к анонимности, то самые общие данные о зарегистрированных учётных записях пользователей должны предоставляться всем субъект-сессиям с соответствующим уровнем доступа), а также, когда x функционирует от имени u или обладает текущим административным доступом на чтение к роли $users\_admin\_role$, записываются роли (с учётом их уровней конфиденциальности и уровня доступа x) и права доступа к ним, которыми обладают индивидуальные административные роли u, и записываются субъект-сессии (в реальной защищённой ОС — идентификаторы

субъект-сессий с учётом их уровней доступа и уровня доступа x), функционирующие от имени u (в этом случае полные данные об учётной записи пользователя предоставляются либо субъект-сессии, функционирующей от её имени, либо субъект-сессии, которая может администрировать эту учётную запись).

Де-юре правила $grant\_rights(x, x', r, \{(y, \alpha_{r_j}) : 1 \leqslant j \leqslant k\})$ и $remove\_rights(x, x', r, \{(y, \alpha_{r_j}) : 1 \leqslant j \leqslant k\})$ позволяют субъект-сессии x добавить или удалить соответственно права доступа к сущности y, имеющей уровень целостности не выше, чем у x, из множества прав доступа (за исключением права доступа владения) роли или административной роли r. Возможность с использованием правил одновременного изменения нескольких прав доступа к одной сущности соответствует возможностям типовых для реальных защищённых ОС функций администрирования прав доступа (например, функции *chmod*). Для применения правил необходимо наличие у субъект-сессии x доступа на запись к роли r и наличие текущей роли, обладающей правом доступа владения к сущности y. Во всех случаях (за исключением наличия у субъект-сессии x текущей административной роли $downgrade\_admin\_role$) требуется, чтобы x могла получить доступ к сущности y с учётом прав доступа к сущностям-контейнерам, содержащим y, и чтобы уровень конфиденциальности y равнялся уровню доступа x. Кроме того, если осуществляется добавление права доступа на запись, то требуется, чтобы уровень целостности сущности y не превосходил уровня целостности роли r. Также необходимо выполнение ограничений на значения множеств прав доступа ролей или административных ролей $Constraint_{PA}$ в последующем состоянии системы (с учётом изменения у роли r прав доступа к y). При этом в случае, когда уровень целостности сущности y равняется i_high, требуется наличие у кооперирующей субъект-сессии x' доступа на запись к сущности $f_s(x)\_i\_entity$.

Де-юре правила $set\_entity\_owner(x, x', r, r', y)$ и $set\_subject\_owner(x, x', r, r', y)$ позволяют субъект-сессии x изменить единственную роль-«владельца» (имеющую право доступа владения) к сущности или субъект-сессии y соответственно с роли или административной роли r на роль или административную роль r'. Для этого субъект-сессии x необходимо иметь административные доступы на чтение и запись к r и на запись к r' (чтобы иметь возможность менять права доступа данных ролей), а также иметь административный доступ на чтение соответственно либо к административной роли $entities\_admin\_role$, либо к $subjects\_admin\_role$. При этом уровень целостности y не должен превосходить уровней целостности r' и x и в последующем состоянии системы должны выполняться ограничения на значения множеств прав доступа ролей или административных ролей $Constraint_{PA}$ (с учётом изменения прав доступа ролей r и r'). За исключением случая наличия у субъект-сессии x текущей административной роли $downgrade\_admin\_role$, когда изменяется роль-«владелец» сущности y, требуется, чтобы x могла получить доступ к сущности y с учётом прав доступа к сущностям-контейнерам, содержащим y, и чтобы уровень конфиденциальности y равнялся уровню доступа x; когда изменяется роль-«владелец» субъект-сессии y, требуется, чтобы x и y имели равные уровни доступа. При этом в случае, когда уровень целостности y равняется i_high, требуется наличие у кооперирующей субъект-сессии x' доступа на запись к сущности $f_s(x)\_i\_entity$.

Де-юре правила $grant\_admin\_rights(x, x', ar, \{(r, \alpha_{r_j}) : 1 \leqslant j \leqslant k\})$ и $remove\_admin\_rights(x, x', ar, \{(r, \alpha_{r_j}) : 1 \leqslant j \leqslant k\})$ позволяют субъект-сессии x добавить или удалить соответственно права доступа на чтение или запись к роли или административной роли r, имеющей уровень целостности не выше, чем у x, из множества прав доступа административной роли ar, к которой x должна иметь административный до-

ступ на запись. Для изменения прав доступа к роли r требуется наличие у x текущей административной роли $roles\_admin\_role$, а если r является административной ролью, то к роли $admin\_roles\_admin\_role$. Во всех случаях (за исключением наличия у субъект-сессии x текущей административной роли $downgrade\_admin\_role$) требуется, чтобы уровень конфиденциальности r равнялся уровню доступа x. Также необходимо выполнение ограничений на значения множеств административных прав доступа административных ролей $Constraint_{APA}$ в последующем состоянии системы (с учётом изменения у административной роли ar прав доступа к r). При добавлении прав доступа к роли r её уровень целостности должен быть не выше уровня целостности административной роли ar. В случае, когда уровень целостности роли или административной роли r равняется i_high, требуется наличие у кооперирующей субъект-сессии x' доступа на запись к сущности $f_s(x)\_i\_entity$.

Де-юре правила $create\_role(x, x', r, rc, ri, re, name, rz)$, $create\_hard\_link\_role(x, x', r, name, rz)$, $delete\_role(x, x', r, rz)$ и $delete\_hard\_link\_role(x, x', r, rz)$ позволяют субъект-сессии x создать или удалить роль или административную роль r или «жёсткую» ссылку на неё (изменить иерархию ролей), входящую в состав роли или административной роли rz, к которой субъект-сессия x должна иметь доступ на запись. При этом нельзя удалять или создавать роли или административные роли, «жёсткие» ссылки на них, являющиеся индивидуальными ролями и индивидуальными административными ролями учётных записей пользователей, а также специальными административными ролями из множества $ADMIN\_ROLES$. Для применения правил необходимо наличие у субъект-сессии x административных доступов на чтение и запись к административным ролям $roles\_admin\_role$ или $admin\_roles\_admin\_role$ для действий ролями или административными ролями соответственно. При этом доступ на запись к этим административным ролям следует требовать, так как создание, удаление ролей или «жёстких» ссылок на них являются существенными преобразованиями параметров безопасности системы. При реализации правил в последующем состоянии иерархия ролей модифицируется (назначаются или удаляются административные права доступа), при этом должны выполняться ограничения на значения множеств административных прав доступа административных ролей $Constraint_{APA}$, а при удалении ролей или административных ролей — на значение множества административных доступов $Constraint_{AA}$.

Аналогично правилам администрирования учётных записей пользователей, при реализации правил создания или удаления ролей права доступа к ним могут изменяться у административных ролей без явного получения к ним субъект-сессией x административного доступа на запись. Так же, как при удалении сущностей или «жёстких» ссылок на них, при удалении роли или административной роли проверяется, что ей в иерархии не подчинены другие роли, а при удалении «жёсткой» ссылки на роль — что эта ссылка не последняя. Однако при создании «жёсткой» ссылки на роль (так как, в отличие от сущностей, могут создаваться «жёсткие» ссылки на роли-контейнеры, содержащие подчинённые роли) проверяется, что это не приведёт к возникновению в иерархии циклов, т. е. нарушению отношения частичного порядка. При применении правил уровень конфиденциальности роли или административной роли r должен равняться уровню доступа субъект-сессии x (за исключением случая, когда x имеет административный доступ на чтение к административной роли $downgrade\_admin\_role$), уровень целостности r должен быть не выше уровня целостности x, а при создании роли или «жёсткой» ссылки на неё уровень конфиденциальности роли r должен равняться уровню конфиденциальности роли-контейнера rz (за исключением случая, когда x имеет ад-

министративный доступ на чтение к административной роли $downgrade\_admin\_role$), в котором она создаётся, и уровень целостности r не должен превосходить уровня целостности rz. При создании роли или административной роли уровни конфиденциальности и целостности сущностей, параметрически ассоциированных с нею, устанавливаются равными её уровням конфиденциальности и целостности соответственно. Если осуществляются действия над ролями или административными ролями с высоким уровнем целостности i_high, то требуется наличие у кооперирующей субъект-сессии x' доступа на запись к сущности $f_s(x)\_i\_entity$.

Де-юре правило $set\_container\_attr(x, x', y, ccr, ccri, t)$ позволяет субъект-сессии x задать сущности-контейнеру y, обладающей уровнем целостности не выше, чем у x, либо её мандатный атрибут конфиденциальности CCR, либо уровень целостности $CCRI$, либо признак — является она разделяемой или нет. Для этого требуется наличие у субъект-сессии административного доступа на чтение к роли $entities\_admin\_role$ или $downgrade\_admin\_role$. При этом субъект-сессия x должна иметь либо административный доступ на чтение к роли-«владельцу» контейнера y, либо к административной роли $entities\_admin\_role$, а также требуется (за исключением случая, когда x имеет административный доступ на чтение к административной роли $downgrade\_admin\_role$), чтобы доступ к y мог быть предоставлен x с учётом её прав доступа к сущностям-контейнерам, содержащим y, и уровень доступа x равнялся уровню конфиденциальности y. Если сущность-контейнер обладает высоким уровнем целостности i_high, то требуется наличие у кооперирующей субъект-сессии x' доступа на запись к сущности $f_s(x)\_i\_entity$.

Де-юре правило $set\_entity\_labels(x, x', y, yc, yi)$ позволяет субъект-сессии x задать сущности y (к которой на момент применения правила ни одна субъект-сессия системы не имеет доступов) её новые уровни конфиденциальности и целостности, для чего требуется наличие у x административного доступа на чтение к административным ролям $entities\_admin\_role$ и $downgrade\_admin\_role$. При этом текущие и устанавливаемые уровни конфиденциальности и целостности сущности y не должны превосходить соответственно текущих уровней доступа и целостности субъект-сессии x. Кроме того, устанавливаемые уровни конфиденциальности и целостности сущности y не должны превосходить соответственно уровней конфиденциальности и целостности всех сущностей-контейнеров, в которых непосредственно находится y, и наоборот, должны быть не ниже уровней конфиденциальности и целостности соответственно всех сущностей, непосредственно входящих в y. Если либо текущий, либо устанавливаемый уровень целостности сущности y является высоким уровнем целостности i_high, то требуется наличие у кооперирующей субъект-сессии x' доступа на запись к сущности $f_s(x)\_i\_entity$.

Де-юре правило $set\_role\_labels(x, x', r, rc, re)$ позволяет субъект-сессии x задать роли или административной роли y (не являющейся индивидуальной ролью или индивидуальной административной ролью учётных записей пользователей, а также специальной административной ролью из множества $ADMIN\_ROLES$, и к которой на момент применения правила ни одна субъект-сессия системы не имеет доступов) её новый уровень конфиденциальности, для чего требуется наличие у x административного доступа на чтение к административным ролям $roles\_admin\_role$, $admin\_roles\_admin\_role$ соответственно и к административной роли $downgrade\_admin\_role$. При этом текущие и устанавливаемые уровни конфиденциальности роли r не должны превосходить соответственно текущего уровня доступа субъект-сессии x. Кроме того, устанавливаемый уровень конфиденциальности

роли r не должен превосходить уровня конфиденциальности всех ролей или административных ролей, в которых непосредственно находится r, и наоборот, должен быть не ниже уровней конфиденциальности всех ролей или административных ролей, непосредственно входящих в r. При применении правила уровни конфиденциальности и целостности сущностей, параметрически ассоциированных с ролью или административной ролью r, должны быть соответственно равными уровням конфиденциальности и целостности r. Если уровень целостности роли r является высоким уровнем целостности i_high, то требуется наличие у кооперирующей субъект-сессии x' доступа на запись к сущности $f_s(x)\_i\_entity$.

Заключение

Разработанные в рамках МРОСЛ ДП-модели де-юре правила администрирования системы позволяют задать детальные спецификации функций механизма управления доступом защищённой ОС *Astra Linux Special Edition*, что, в свою очередь, создаёт предпосылки для осуществления как минимум полуформальной верификации реализации модели непосредственно в программном коде ОС. Кроме того, описание рассмотренных правил позволило выявить ряд неточностей при определении элементов модели, что в целом повысило её качество.

ЛИТЕРАТУРА

1. *Девянин П. Н.* Модели безопасности компьютерных систем. Управление доступом и информационными потоками: учебн. пособие для вузов. 2-е изд., испр. и доп. М.: Горячая линия-Телеком, 2013. 338 с.
2. *Девянин П. Н.* Корректность правил преобразования состояний системы в рамках мандатной сущностно-ролевой ДП-модели ОС семейства Linux // Прикладная дискретная математика. Приложение. 2013. № 6. С. 58–59.
3. *Девянин П. Н.* Об опыте внедрения мандатной сущностно-ролевой ДП-модели управления доступом и информационными потоками в защищенную ОС Astra Linux Special Edition // Методы и технические средства обеспечения безопасности информации. Материалы 22-й науч.-технич. конф. 08–11 июля 2013 г. СПб.: Изд-во Политехн. ун-та, 2013. С. 78–80.
4. Операционные системы Astra Linux. `http://www.astra-linux.ru/`.

ПРИКЛАДНАЯ ТЕОРИЯ ГРАФОВ

DOI 10.17223/20710410/22/4

УДК 519.171.1+514.17

V-ГРАФЫ И ИХ СВЯЗЬ С ЗАДАЧАМИ РАЗМЕЩЕНИЯ ФИГУР НА ПЛОСКОСТИ

И. Г. Величко*, А. И. Зинченко**

*Запорожский национальный технический университет, г. Запорожье, Украина

**Запорожский национальный университет, г. Запорожье, Украина

E-mail: wig64@mail.ru, andriver@znu.edu.ua

Изучаются такие расположения двух конгруэнтных фигур на плоскости, при которых они не имеют общих внутренних точек. Прямая, параллельная вектору сдвига, пересекает эти фигуры по двум одинаковым системам интервалов, смещённым на вектор сдвига. Строится ориентированный V_n-граф, вершины которого соответствуют топологически различным вариантам взаимного расположения двух систем из n интервалов, а рёбра — допустимым переходам между вершинами. Вводится понятие W_n-графа как минимального транзитивного графа, содержащего V_n-граф, пополненный истоком. Исследованы свойства V_n- и W_n-графов.

Ключевые слова: *размещение фигур на плоскости, ориентированный V_n-граф, W_n-граф, числа Каталана, пути Дика, системы интервалов, конгруэнтные фигуры.*

Введение

В задачах, связанных с размещениями фигур на плоскости, необходимо уметь определять такие взаимные положения двух фигур из заданного множества, при которых они не имеют внутренних общих точек. Для фигур сложной геометрии наиболее популярными являются методы Ф-функций [1] и No-Fit-полигонов [2]. Для выпуклых многоугольников эффективным является метод опорных прямых [3]. Для однотипных невыпуклых фигур авторами предложен точный метод решения таких задач для многоугольников, не являющихся выпуклыми [4]. В данной работе для исследования вариантов взаимного расположения систем интервалов, полученных в результате сечения двух одинаковых фигур прямой, параллельной вектору трансляции, переводящему одну фигуру в другую, строится так называемый V_n-граф. Его исследование позволяет уточнить трудоёмкость предложенного в [4] алгоритма. Другой аспект связи теории графов и задач взаимного расположения однотипных фигур в полосе освещен в [5].

1. V_n-графы

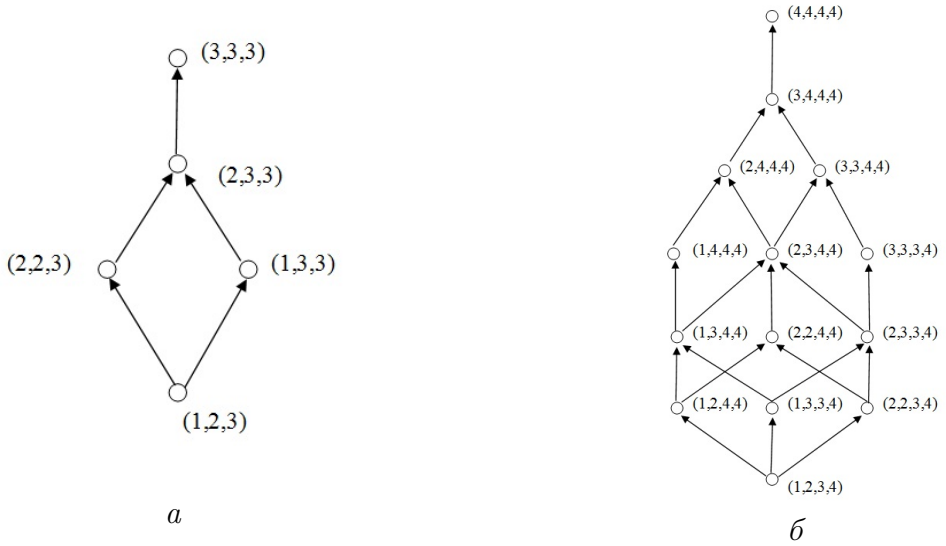
Пусть n — натуральное число. *Допустимыми* будем называть кортежи $(a_1, a_2, \ldots, a_n)$ длины n, состоящие из чисел $1, 2, \ldots, n$ и удовлетворяющие двум условиям:

1) $a_i \geqslant i, i = 1, \ldots, n$;

2) $a_i \leqslant a_{i+1}, i = 1, \ldots, n-1$.

Например, для $n = 3$ получим следующие допустимые последовательности: (1, 2, 3), (2, 2, 3), (1, 3, 3), (2, 3, 3), (3, 3, 3). Для фиксированного n построим ориентированный граф V_n, вершины которого соответствуют допустимым кортежам. Две

вершины соединены ребром тогда и только тогда, когда они отличаются только одной компонентой, причём разница между значениями этой компоненты равна 1. Начало ребра соответствует последовательности с меньшей суммой. На рис. 1 приведены графы V_3 и V_4.



а *б*

Рис. 1. Графы V_3 ($а$) и V_4($б$)

Приведём утверждения относительно структуры графа V_n.

Утверждение 1. Граф V_n содержит число вершин, равное n-му числу Каталана $\dfrac{1}{n+1}C_{2n}^{n}$.

Доказательство. Каждому допустимому кортежу длины n поставим в соответствие правильную скобочную последовательность из n открывающих и n закрывающих скобок. Равенство $a_i = j$ означает, что i-я закрывающая скобка находится после j-й открывающей скобки. Например, кортежу (2, 3, 3) соответствует скобочная последовательность (( ) ( )).

Построенное отображение есть биекция между допустимыми кортежами длины n и правильными $2n$-скобочными последовательностями, количество которых равно n-му числу Каталана. ■

Утверждение 2. Длина любого пути от начальной вершины $(1, 2, 3, \ldots, n)$ до вершины $(n, n, n, \ldots, n)$ в графе v_n равна C_n^2.

Доказательство. Поскольку при движении по ребру сумма элементов кортежа увеличивается на 1, то утверждение следует из тождества $n \cdot n - (1 + 2 + \ldots + n) = = n(n-1)/2 = C_n^2$. ■

Утверждение 3. Число рёбер в графе V_n равно C_{2n-1}^{n-2}.

Доказательство. Из вершины графа N_n, соответствующей последовательности $(a_1, a_2, \ldots, a_n)$, выходят рёбра, количество которых равно количеству пар a_k, a_{k+1} в этой последовательности, таких, что $a_k < a_{k+1}$. Будем называть такие пары правильными. Каждой правильной паре соответствует единственное ребро, ведущее в вершину, соответствующую последовательности $(a_1, a_2, \ldots, a_k + 1, a_{k+1}, \ldots, a_n)$. Таким образом, общее количество рёбер в графе V_n равно количеству правильных пар во всех допустимых последовательностях длины n.

Зафиксируем число n и каждой допустимой последовательности $(1, 2, 3, \ldots, n)$ поставим в соответствие путь Дика (Dyck path) [6] следующим образом. Построим квадрат размером $n \times n$ клеток и в первом столбце выделим горизонтальный отрезок на высоте a_1 клеток, во втором столбце — на высоте a_2 клеток и так далее. Соединим полученные горизонтальные отрезки вертикальными так, чтобы получилась замкнутая ломаная, выходящая из левого нижнего угла. Поскольку, согласно свойствам допустимых последовательностей, полученная ломаная лежит выше диагонали квадрата, выходящей из нижнего угла, то полученная ломаная будет путем Дика. Очевидно, что предлагаемая конструкция устанавливает биекцию между допустимыми последовательностями и путями Дика.

Полученную ломаную можно однозначно описать набором $2n$ символов U и D. Символ $U(D)$ соответствует вертикальному (горизонтальному) звену единичной ломаной при движении из левого нижнего угла. Полученный набор будем называть (U, D)-набором. Каждой правильной паре в допустимой последовательности однозначно соответствует пара DU в соответствующем (U, D)-наборе.

Значит, общее количество рёбер в графе V_n равно количеству пар DU во всех (U, D)-наборах длины $2n$. Согласно результатам Emeric Deutsch [7], количество таких пар равно C_{2n-1}^{n-2}. ■

На рис. 2 изображён путь Дика для последовательности $(1, 3, 3, 4)$, которой соответствует (U, D)-набор вида $U\underline{DU}UD\underline{DU}D$. В этот набор входят две пары DU (они подчеркнуты).

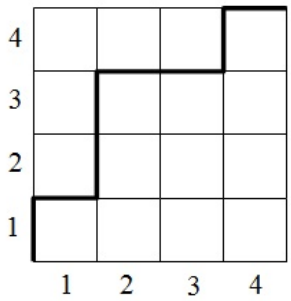


Рис. 2. Путь Дика для (1, 3, 3, 4)

2. W_n-графы

Добавим к графу V_n дополнительную вершину, из которой выходит единственное ребро в вершину $(1, 2, 3, \ldots, n)$. Такую операцию будем называть пополнением графа истоком. Введём понятие W_n-графа.

Определение 1. W_n-графом называется минимальный транзитивный граф, содержащий V_n-граф, пополненный истоком.

На рис. 3 изображён W_3-граф.

Очевидно, что число вершин W_n-графа на единицу превосходит число вершин V_n-графа и равно $\dfrac{1}{n+1}C_{2n}^{n} + 1$.

Утверждение 4. W_n-граф содержит $\dfrac{6(2n)!(2n+2)!}{n!(n+1)!(n+2)!(n+3!)}$ рёбер.

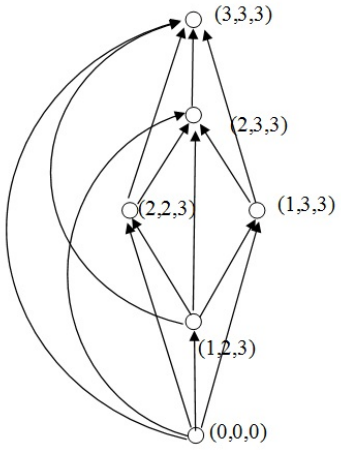


Рис. 3. W_3-граф

Доказательство. В терминах путей Дика число ребер в W_n-графе равно сумме числа пар путей Дика длины $2n$, в которых один из путей лежит не выше другого пути, и числу самих путей Дика.

Как известно [7], элементы последовательности A005700 в OEIS, которые задаются формулой в утверждении, равны количеству замкнутых ломаных длины $2n$, состоящих из единичных отрезков, параллельных осям, начинающихся в начале координат и полностью лежащих внутри угла, образованного прямыми $y = 0$ и $y = x$. Если обозначить через N движение вверх, E — вправо, S — вниз, W — влево, то все допустимые маршруты длины $4 = 2 \cdot 2$ имеют следующий вид: $EWEW$, $EEWW$, $ENSW$.

Установим биекцию между ребрами W_n-графа и указанными маршрутами. Пусть ребро W_n-графа выходит из истока. Тогда рассмотрим (U, D)-набор, соответствующий вершине, в которую это ребро входит. Заменим в этой последовательности U на E и D на W. В результате получим допустимый маршрут, не содержащий N и S. Это означает, что вся ломаная лежит на оси Ox. Пусть теперь ребро W_n-графа выходит не из истока. Тогда рассмотрим (U, D)-наборы $\alpha_1\alpha_2 \dots \alpha_{2n}$ и $\beta_1\beta_2 \dots \beta_{2n}$, соответствующие началу и концу этого ребра. По ним построим допустимый маршрут $\gamma_1\gamma_2 \dots \gamma_{2n}$ по следующему правилу:

$$\begin{gathered}\text{если } \alpha_k = U, \beta_k = U, \text{ то } \gamma_k = E;\\ \text{если } \alpha_k = U, \beta_k = D, \text{ то } \gamma_k = S;\\ \text{если } \alpha_k = D, \beta_k = U, \text{ то } \gamma_k = N;\\ \text{если } \alpha_k = D, \beta_k = D, \text{ то } \gamma_k = W.\end{gathered}$$

В результате получим допустимый маршрут, содержащий точки выше оси Ox. Используя свойства путей Дика, легко убедиться, что построенное отображение есть требуемая биекция. ■

Например, маршруту $ENEESWWW$ (рис. 4) соответствует начальный (U, D)-набор $UDUUUDDD$ и конечный (U, D)-набор $UUUUDDDD$ (рис. 5). Первый из этих наборов отвечает допустимой последовательности $(1, 4, 4, 4)$, а второй — $(4, 4, 4, 4)$; то есть ребро, соответствующее маршруту $ENEESWWW$, в W_4-графе соединяет вершины, соответствующие последовательностям $(1, 4, 4, 4)$ и $(4, 4, 4, 4)$.

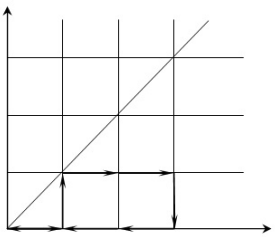


Рис. 4. Маршрут длины 8, лежащий в первом октанте

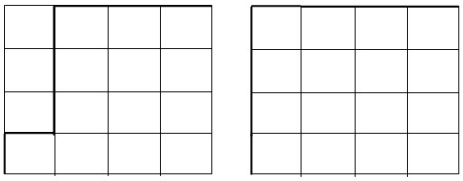


Рис. 5. Пути Дика, соответствующие (U, D)-наборам $UDUUUDDD$ и $UUUUDDDD$

3. Связь V_n- и W_n-графов с задачами размещения

Покажем связь между графами V_n и задачами размещения однотипных фигур на плоскости. Пусть есть фигура F, гомеоморфная открытому кругу, экземпляры которой нужно расположить в один ряд в некоторой полосе. Рассмотрим две соседние фигуры F и F_1. Фигура F_1 получается из F сдвигом на некоторый вектор $\bar{m} = m \cdot \bar{i}$, где $\bar{i}$ — орт. По условию задачи фигуры F и F_1 не должны пересекаться. Все такие сдвиги (при $m > 0$) называются допустимыми трансляциями.

Рассмотрим прямую l, параллельную $\bar{i}$ и имеющую общие точки с F. Она пересекается с фигурой F по некоторой системе интервалов, которая при допустимой трансляции переходит в другую систему интервалов, сдвинутую относительно исходной на вектор $\bar{m}$. Фигуры F и F_1 не имеют общих точек тогда и только тогда, когда для любой прямой из указанного класса исходная и сдвинутая системы интервалов не имеют общих точек. На рис. 6 изображены системы интервалов исходной и сдвинутой непересекающихся фигур.

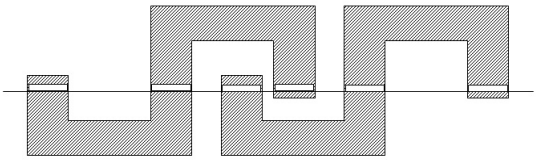


Рис. 6. Пересечение двух конгруэнтных фигур с горизонтальной прямой

Предположим, что фигура $P = l \cap F$ состоит из n интервалов. Поскольку после сдвига исходная система интервалов P и сдвинутая система $P_1 = l \cap F_1$ не пересекаются, то такой сдвиг можно описать кортежем $(a_1, a_2, \ldots, a_n)$ длины n, где a_i есть количество интервалов системы P, которые находятся левее i-го интервала системы P_1. На рис. 7 приведён соответствующий пример.

Рис. 7. Системы интервалов P и P_1, которые соответствуют кортежу $(2, 2, 3, 4)$

Поскольку каждый интервал после сдвига вправо оказывается правее своего исходного положения, то $a_i \geqslant i$. Так как при сдвиге порядок интервалов сохраняется, то $a_{i+1} \geqslant a_i$. Таким образом, допустимым сдвигам систем интервалов на прямой соответствуют допустимые кортежи.

При движении вправо системы P осуществляется переход от одной допустимой трансляции к другой, при этом как минимум один из элементов кортежа увеличивается. Этим объясняется правило построения рёбер в графе V_n.

При переходе от одной допустимой трансляции системы интервалов к соседней может так случиться, что один из элементов кортежа увеличится более чем на единицу или увеличатся сразу несколько элементов этого кортежа. Этим объясняется правило построения рёбер в графе W_n. Нулевой трансляции соответствует исток графа W_n.

Заключение

Введено понятие V_n-графа, где n — натуральное число. Его вершины соответствуют допустимым кортежам длины n, рёбра соединяют вершины, соответствующие кортежам, отличающимся на 1 в одной компоненте. Доказано, что V_n-граф содержит количество вершин, равное n-му числу Каталана, и число рёбер, равное C_{2n-1}^{n-2}. При доказательстве использована биекция между допустимыми кортежами и путями Дика. Введено понятие W_n-графа как минимального транзитивного графа, содержащего V_n-граф, пополненный истоком. Сформулированы и доказаны утверждения о свойствах W_n-графов. Показано, как такие графы возникают в задачах размещения однотипных фигур на плоскости. Вершины W_n-графа соответствуют топологически различным вариантам расположения двух конгруэнтных систем интервалов на прямой, при которых они не имеют общих точек. Ребра W_n-графа соответствуют переходам между двумя вариантами расположения систем интервалов. Подобные конструкции возникают при сечении двух однотипных фигур на плоскости прямой, параллельной вектору трансляции. Изученные свойства V_n- и W_n-графов позволяют оценивать трудоёмкость алгоритмов определения всех возможных перемещений фигуры, при которых она не имеет общих точек со своим исходным положением.

ЛИТЕРАТУРА

1. *Stoyan Yu. G.* Φ-function and its basic properties // Доповіді НАН України. 2001. Т. 1. № 1. С. 112–117.
2. *Burke E. K.* Complete and robust no-fit polygon generation for the irregular stock cutting problem // Eur. J. Operat. Res. 2007. V. 179. No. 1. P. 27–49.
3. *Аввакумов В. Д.* Оптимальное размещение плоских объектов произвольной геометрической формы // Информационные технологии. 2009. № 5. С. 31–35.
4. *Величко І. Г., Зінченко А. І.* Теорема про потяг та її застосування для задач регулярного розкрою // Вісник Харківського національного університету. 2011. Т. 1. № 960. С. 47–54.
5. *Пятницкая Г. Н., Синицын И. Г.* Бесконечные маршруты в графе и оптимальное размещение однотипных фигур в бесконечной узкой полосе // Журн. вычисл. матем. и матем. физ. 1979. Т. 19. № 5. С. 1304–1312.
6. `http://oeis.org/A002054` — Онлайн-энциклопедия целочисленных последовательностей. 2013.
7. `http://oeis.org/A005700` — Онлайн-энциклопедия целочисленных последовательностей. 2013.

DOI 10.17223/20710410/22/5

УДК 519.17

Т-НЕПРИВОДИМЫЕ РАСШИРЕНИЯ ОБЪЕДИНЕНИЙ НЕКОТОРЫХ ТИПОВ ОРГРАФОВ

А. В. Гавриков

Саратовский государственный университет, г. Саратов, Россия

E-mail: alexandergavrikov1989@gmail.com

Приведены алгоритмы построения Т-неприводимых расширений (THP) для объединения некоторых типов орграфов, а именно для объединения ориентированных цепей, объединения орграфа с его THP, а также THP для направленных звезд. Каждый из предложенных алгоритмов имеет полиномиальную асимптотическую сложность. Доказана корректность этих алгоритмов.

Ключевые слова: *Т-неприводимые расширения, минимальные Т-неприводимые расширения, THP, объединения некоторых типов орграфов, объединения ориентированных цепей, направленные звёзды.*

Введение

Под *ориентированным графом* (или *орграфом*) понимается пара $G = (V, \alpha)$, где V — конечное непустое множество (вершины орграфа), а α — отношение на множестве V (дуги орграфа) [1]. Дуга в орграфе $G = (V, \alpha)$ называется *инцидентной* вершине v, если вершина v — конец или начало этой дуги. *Вложение* орграфа $G = (V, \alpha)$ в орграф $H = (W, \beta)$ — взаимно однозначное отображение $\varphi : V \to W$, такое, что для всех $u, v \in V$ из $(u, v) \in \alpha$ следует $(\varphi(u), \varphi(v)) \in \beta$. При этом говорят, что орграф G вкладывается в орграф H. *Часть* орграфа $G = (V, \alpha)$ — орграф $H = (W, \beta)$, такой, что $W \subseteq V$ и $\beta \subseteq (W \times W) \cap \alpha$. Часть орграфа $G = (V, \alpha)$ является *подграфом* орграфа $H = (W, \beta)$, если $\beta = (W \times W) \cap \alpha$. Подграф H *максимален*, если он получается из исходного орграфа G удалением одной вершины и всех инцидентных ей дуг. *Расширение* орграфа $G = (V, \alpha)$ — орграф $H = (W, \beta)$, такой, что $|W| = |V| + 1$ и орграф G вкладывается в каждый максимальный подграф орграфа H. *Соединение* орграфов $G = (V, \alpha)$ и $H = (W, \beta)$, таких, что $V \cap W = \varnothing$, — орграф $G + H = (V \cup W, \alpha \cup \beta \cup V \times W \cup W \times V)$. *Изоморфизм* орграфа $G = (V, \alpha)$ на орграф $H = (W, \beta)$ — взаимно однозначное соответствие $\varphi : V \to W$, сохраняющее отношение смежности. Изоморфность орграфов G и H обозначается через $G \cong H$. Орграфы G и H в этом случае называются *изоморфными*.

Тривиальное расширение (TP) орграфа $G = (V, \alpha)$ — соединение $G + w$ исходного орграфа G с вершиной $w \notin V$ [2]. В силу того, что тривиальное расширение орграфа G единственно с точностью до изоморфизма, возможно ввести функцию $TP(G)$. *Т-неприводимое расширение* (THP) орграфа G — расширение исходного орграфа G, полученное удалением максимального множества дуг из $TP(G)$. *Минимальное Т-неприводимое расширение* орграфа G — расширение орграфа G, полученное удалением максимального количества дуг из $TP(G)$. Другими словами, минимальное THP орграфа G — это THP с минимальным количеством дуг среди всех остальных THP орграфа G.

Т-неприводимые расширения являются одним из видов оптимальных расширений для орграфов. Конструкции оптимальных расширений применяются в диагностике

дискретных систем и криптографии [3]. В общем случае задача поиска THP по заданному орграфу является вычислительно сложной [4].

Путь в орграфе — последовательность дуг вида $(v_1, v_2), (v_2, v_3), \cdots, (v_{n-1}, v_n)$, где $(v_i, v_{i+1}) \in \alpha$, $1 \leqslant i \leqslant n-1$, и никакая дуга не встречается более одного раза. Путь в орграфе является *простым*, если каждая его вершина принадлежит не более чем двум его дугам. *Длина пути* — количество входящих в него дуг. Простой путь в орграфе из n вершин, у которого начальная и конечная вершины не совпадают, является *ориентированной цепью* и обозначается через P_n.

Путь является *циклическим*, если $v_1 = v_n$. *Контур* в орграфе — простой циклический путь. Контур из n вершин обозначается через C_n.

Для THP орграфов известен следующий критерий [5], на который опирается доказательство процедуры их построения.

Теорема 1 (критерий THP для орграфов). Орграф $H = (W, \beta)$ является THP для орграфа $G = (V, \alpha)$ тогда и только тогда, когда одновременно выполнены следующие условия:

1) $|W| = |V| + 1$;

2) в орграфе H существует вершина w, такая, что $H - w \cong G$;

3) орграф G вкладывает в каждый максимальный подграф $H - u$ орграфа H, где $u \neq w$;

4) (свойство неприводимости). При удалении из орграфа H любой дуги, инцидентной вершине w, то есть (u, w) или (w, u), получается орграф, не являющийся расширением для G.

1. THP для объединения ориентированных цепей

Перед тем как рассмотреть задачу поиска THP для объединения ориентированных цепей, покажем, как устроены T-неприводимые расширения некоторой ориентированной цепи. *Степень исхода вершины* v — количество дуг в орграфе $G = (V, \alpha)$, имеющих своим началом вершину v. Степень исхода вершины v обозначают через $d^+(v)$, $d^+(v) = |\alpha(v)|$. *Степень захода вершины* v — количество дуг в орграфе $G = (V, \alpha)$, имеющих своим концом вершину v. Степень захода вершины v обозначают через $d^-(v)$, $d^-(v) = |\alpha^{-1}(v)|$. Вершина v называется *источником*, если её степень захода равна 0, т. е. $d^-(v) = 0$. Вершина v называется *стоком*, если её степень исхода равна 0, т. е. $d^+(v) = 0$. Введём следующие обозначения для вершин, из которых состоит ориентированная цепь P_n. Вершину v ориентированной цепи P_n, являющуюся источником, обозначим через v_0. Остальные вершины пометим номерами от 1 до $n-1$ в порядке их прохождения по дугам цепи из вершины v_0. Очевидным THP для ориентированной цепи P_n является контур C_{n+1} (рис. 1).

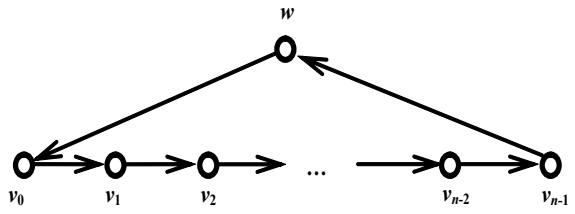
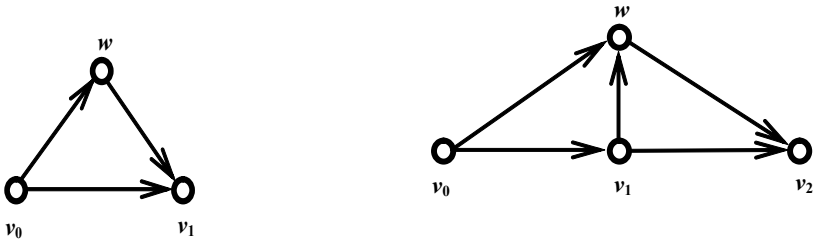


Рис. 1. THP ориентированной цепи

Ясно, что контур C_{n+1} является минимальным ТНР для ориентированной цепи P_n. Однако контур C_{n+1} не является единственным ТНР для P_n. Покажем это. Для ориентированной цепи P_2 ещё одним ТНР является транзитивный турнир, состоящий из трёх вершин; для ориентированной цепи P_3 — орграф, изображенный на рис. 2, *б*.



а *б*

Рис. 2. ТНР ориентированных цепей P_2 (*а*) и P_3 (*б*)

Следующий алгоритм позволяет построить ТНР, не изоморфное контуру C_{n+1}, для ориентированной цепи P_n.

Алгоритм 1

Дана ориентированная цепь P_n, $n \geqslant 4$. Построим одно из её ТНР $H = (W, \beta)$, такое, что $H \ncong C_{n+1}$, следующим образом:

1) добавим к P_n вершину w;

2) добавим дуги (v_0, w) и (v_1, w);

3) добавим дуги (w, v_{n-2}) и (w, v_{n-1});

4) для каждой вершины $v_i \in V$, $2 \leqslant i \leqslant n-3$, добавим дуги (v_i, w) и (w, v_i).

Количество дуг $|\beta|$ в $H = (W, \beta)$ равно $n + 3 + 2(n-4) = 3n - 5$.

Асимптотическая сложность алгоритма 1

Для реализации алгоритма 1 достаточно для каждой вершины $v_i \in V$ исходной ориентированной цепи P_n добавить одну или две (в зависимости от её номера) инцидентные ей и вершине w дуги. Таким образом, асимптотическая сложность алгоритма не превосходит количества вершин и равна $\mathrm{O}(n)$.

Общий вид орграфа, построенного по алгоритму 1, показан на рис. 3.

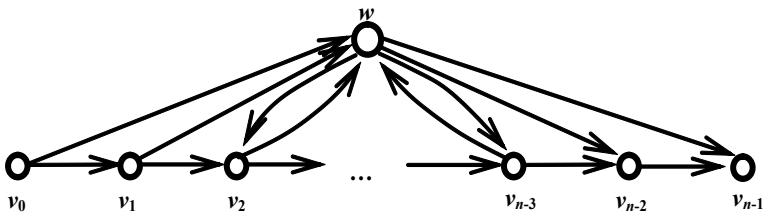


Рис. 3. ТНР ориентированной цепи P_n

Теорема 2. Алгоритм 1 корректен.

Доказательство. Необходимо и достаточно показать выполнение всех пунктов теоремы 1 (критерия ТНР для ориентированных графов).

1. Очевидно, $|W| = |V| + 1$, так как $W = V \cup \{w\}$.

2. Очевидно, что $P_n \cong H - w$ в силу построения в алгоритме.

3. Докажем, что для любой вершины $v_i \in W$, не совпадающей с вершиной w, исходная ориентированная цепь P_n вкладывается в орграф $H - v_i$. При удалении источника v_0 в орграфе $H - v_0$ существует ориентированная цепь $v_1, w, v_2, v_3, \ldots, v_{n-1}$ длины n. При удалении стока v_{n-1} в орграфе $H - v_{n-1}$ существует ориентированная цепь $v_0, w, v_1, v_2, \ldots, v_{n-2}$ длины n. При удалении вершины v_i, которая не является ни источником, ни стоком, в орграфе $H - v_i$ существует ориентированная цепь $v_0, v_1, \ldots, v_{i-1}, w, v_{i+1}, \ldots, v_{n-2}, v_{n-1}$ длины n. Таким образом, ориентированная цепь P_n вкладывается в каждый максимальный подграф орграфа H.

4. Свойство неприводимости. Докажем, что при удалении из орграфа $H = (W, \beta)$ любой дуги, инцидентной вершине w, получится орграф, не являющийся расширением для исходной ориентированной цепи P_n.

В случае удаления дуги (v_0, w) максимальный подграф $H - v_1$ будет содержать изолированную вершину v_0, т. е. вложение ориентированной цепи P_n в один из максимальных подграфов орграфа H невозможно. В случае удаления дуги (v_1, w) максимальный подграф $H - v_2$ будет содержать два стока v_1 и v_{n-1}, в то время как цепь P_n — только один. Таким образом, при удалении дуг, добавленных в п. 2 алгоритма 1, получится орграф, который не является расширением для ориентированной цепи P_n.

В случае удаления дуги (w, v_{n-1}) максимальный подграф $H - v_{n-2}$ будет содержать изолированную вершину v_{n-1}, т. е. вложение ориентированной цепи P_n в один из максимальных подграфов орграфа H невозможно. В случае удаления дуги (w, v_{n-2}) максимальный подграф $H - v_{n-3}$ будет содержать два источника v_0 и v_{n-2}, в то время как цепь P_n — только один. Таким образом, при удалении дуг, добавленных в п. 3 алгоритма 1, получим орграф, не являющийся расширением для P_n.

Рассмотрим ситуацию, возникающую при удалении дуг, добавленных в п. 4 алгоритма 1. Пусть удалена дуга (v_i, w), где $2 \leqslant i \leqslant n - 3$. Тогда максимальный подграф $H - v_{i+1}$ будет содержать два стока v_i и v_{n-1}, в то время как цепь P_n — только один. Если удалена дуга (w, v_i), где $2 \leqslant i \leqslant n - 3$, то максимальный подграф $H - v_{i-1}$ будет содержать два источника v_0 и v_i, в то время как цепь P_n — только один. Свойство неприводимости доказано.

Очевидно, что $H \ncong C_{n+1}$, так как H имеет $n + 3 + 2(n - 4) = 3n - 5$ дуг, а контур C_{n+1} — $n + 1$ дуг. ■

Далее рассмотрим задачу нахождения ТНР для объединения ориентированных цепей. Аналогичная задача, но для случая неориентированных графов, рассматривалась в [2].

Дан орграф $G = (V, \alpha)$, являющийся объединением ориентированных цепей: $G = P_{n_1} \cup P_{n_2} \cup \ldots \cup P_{n_k}$. В орграфе G существует $\sum\limits_{i=1}^{k} n_i$ вершин и $\sum\limits_{i=1}^{k}(n_i - 1) = \sum\limits_{i=1}^{k} n_i - k$ дуг. Обозначим $\sum\limits_{i=1}^{k} n_i$ через n. При этом k вершин являются источниками (начальные вершины каждой из k ориентированных цепей) и k вершин — стоками (конечные вершины каждой из k ориентированных цепей). Остальные $n - 2k$ вершин имеют степени исхода и захода равные 1.

Следующий алгоритм позволяет построить ТНР для объединения ориентированных цепей.

Алгоритм 2

Дан орграф $G = P_{n_1} \cup P_{n_2} \cup \ldots \cup P_{n_k}$. Построим одно из его THP $H = (W, \beta)$ следующим образом:

1) добавим в орграф $G = (V, \alpha)$ вершину w;

2) для каждого источника $v \in V$ добавим дугу (w, v);

3) для каждого стока $v \in V$ добавим дугу (v, w).

Количество дуг $|\beta|$ в $H = (W, \beta)$ равно $n + k$.

Асимптотическая сложность алгоритма 2

Для реализации алгоритма 2 достаточно знать степени исхода и захода каждой вершины исходного орграфа G. Для данного типа орграфов эту информацию можно вычислить за линейное от количества вершин время. Таким образом, асимптотическая сложность алгоритма оценивается как $\mathrm{O}(n)$.

Теорема 3. Алгоритм 2 корректен.

Доказательство. Необходимо и достаточно показать выполнение всех пунктов теоремы 1 (критерия THP для ориентированных графов).

1. Очевидно, $|W| = |V| + 1$, так как $W = V \cup \{w\}$.

2. Очевидно, что $G \cong H - w$ в силу построения в алгоритме.

3. Докажем, что для любой вершины $v_i \in W$, не совпадающей с вершиной w, исходный орграф $G = P_{n_1} \cup P_{n_2} \cup \ldots \cup P_{n_k}$ вкладывается в орграф $H - v_i$. Удалим вершину v_i из орграфа H. Вершина v_i входит в одну из ориентированных цепей P_{n_j}, $1 \leqslant j \leqslant k$, исходного орграфа G. Вложение орграфа G в орграф $H - v_i$ построим следующим образом. Ориентированную цепь P_{n_j}, состоящую из вершин $v_1, v_2, \ldots, v_{i-1}, v_i, v_{i+1}, \ldots, v_{n_j}$, вложим в ориентированную цепь, образованную вершинами $v_{i+1}, v_{i+2}, \ldots, v_{n_j}, w, v_1, \ldots, v_{i-1}$. Это возможно сделать, так как $(v_{n_j}, w) \in \beta$ и $(w, v_1) \in \beta$ по построению. Остальные вершины при вложении переведём сами в себя.

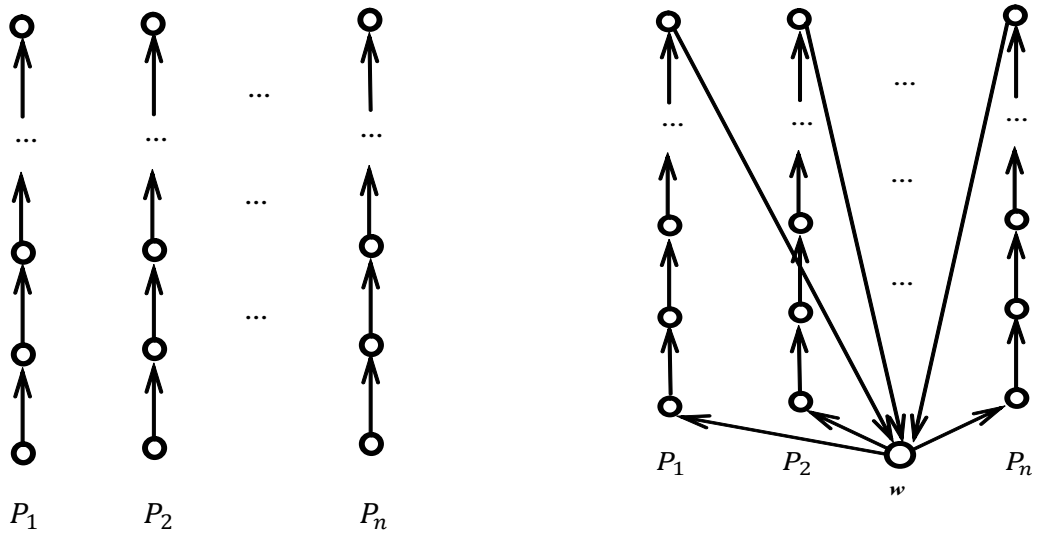
4. Свойство неприводимости. Докажем, что орграф, полученный при удалении из орграфа $H = (W, \beta)$ любой дуги, инцидентной вершине w, не будет расширением для исходного орграфа $G = (V, \alpha)$.

Покажем, что при удалении любой дуги (w, v), которая была добавлена в п. 2 алгоритма 2, орграф $H - (w, v)$ не будет расширением для $G = P_{n_1} \cup P_{n_2} \cup \ldots \cup P_{n_k}$. В этом случае вершина v является источником и началом одной из k ориентированных цепей исходного орграфа G. Из вершины v исходит дуга $(v, u) \in \alpha$. Тогда в максимальном подграфе $H - u$ вершина v будет изолированной вершиной в силу того, что дуги (w, v) нет в орграфе H. Ясно, что в таком случае орграф G не вкладывается в орграф $H - u$, в котором существует изолированная вершина.

Покажем, что при удалении любой дуги (v, w), которая была добавлена в п. 3 алгоритма 2, орграф $H - (v, w)$ не будет расширением для $G = P_{n_1} \cup P_{n_2} \cup \ldots \cup P_{n_k}$. В этом случае вершина v является стоком и концом одной из k ориентированных цепей исходного орграфа G. В вершину v входит дуга $(u, v) \in \alpha$. Тогда в максимальном подграфе $H - u$ вершина v будет изолированной вершиной в силу того, что дуги (v, w) нет в орграфе H. Ясно, что в таком случае орграф G не вкладывается в орграф $H - u$, в котором существует изолированная вершина.

Свойство неприводимости выполнено. ■

На рис. 4 изображены орграф, являющийся объединением ориентированных цепей, и его THP, построенное по алгоритму 2.



а *б*

Рис. 4. Орграф $G = P_1 \cup P_2 \cup \ldots \cup P_n$ (*а*) и его ТНР (*б*)

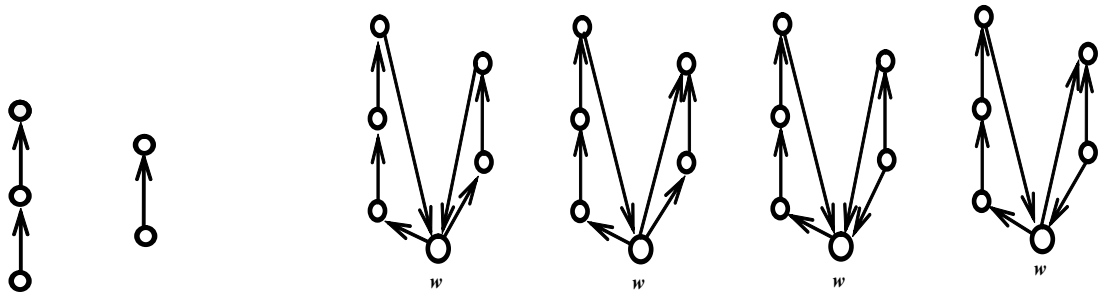
Теорема 4. Для орграфов $G = P_{n_1} \cup P_{n_2} \cup \ldots \cup P_{n_k}$, где $n_i > 2$, $1 \leqslant i \leqslant k$, являющихся объединением ориентированных цепей, состоящих более чем из двух вершин, существует с точностью до изоморфизма только одно минимальное ТНР.

Доказательство. Покажем, что любое минимальное ТНР для орграфа G изоморфно ТНР, построенному по алгоритму 2.

Сначала докажем, что в произвольном минимальном ТНР для рассматриваемого класса орграфов должно быть $n + k$ дуг, где k — количество ориентированных цепей. Ровно столько дуг содержится в минимальном ТНР, построенном по алгоритму 2.

В орграфе $G = P_{n_1} \cup P_{n_2} \cup \ldots \cup P_{n_k}$ существует k вершин, являющихся стоками, и k вершин, являющихся источниками. Количество дуг в G равно $n - k$. Для каждого стока v необходимо добавить исходящую дугу (v, w), иначе в некоторых максимальных подграфах расширения орграфа G будут присутствовать изолированные вершины (для этого будет достаточно удалить вершину u, которая является началом дуги (u, v)). Для каждого источника v необходимо добавить входящую дугу (w, v), иначе в некоторых максимальных подграфах расширения орграфа G будут присутствовать изолированные вершины (для этого будет достаточно удалить вершину u, которая является концом дуги (v, u)). Количество добавленных дуг в этом случае равно $2k$, а общее количество дуг в минимальном ТНР — $n + k$. Итак, необходимо добавить k дуг, входящих в вершину w, и k дуг, исходящих из вершины w. Каждая из $2k$ добавленных дуг будет инцидентна либо одному из k источников, либо одному из k стоков исходного орграфа G. При этом так как $n_i > 2$, $1 \leqslant i \leqslant k$, для выполнения вложения, показанного в п. 3, единственным способом добавления $2k$ дуг, инцидентных вершине w, является алгоритм 2. ■

Примечание. Если одна из ориентированных цепей P_{n_i}, $1 \leqslant i \leqslant n$, орграфа G состоит из двух вершин, то добавление $2k$ дуг в исходный орграф можно выполнить четырьмя способами (рис. 5).

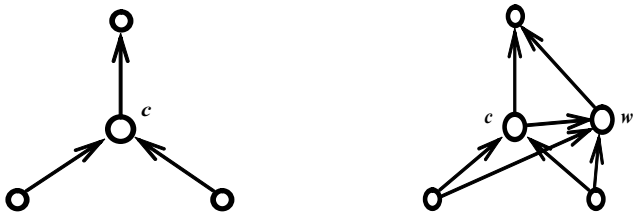


а *б*

Рис. 5. Граф $G = P_3 \cup P_2$ (*а*) и четыре его THP (*б*)

2. THP для направленных звезд

Под направленной звездой с k дугами будем понимать орграф, полученный из полного двудольного графа $K_{1,k}$ некоторой ориентацией его рёбер. Понятия корня и листьев в направленных звёздах аналогичны случаю неориентированных звёзд. Направленная звезда S_3 с тремя листьями изображена на рис. 6, *а*.



а *б*

Рис. 6. Направленная звезда S_3 (*а*) и её THP (б)

Прежде чем дать алгоритм построения одного из THP для произвольной направленной звезды, докажем следующую лемму.

Лемма 1. Пусть S_k — направленная звезда, а $H = (W, \beta)$ — одно из её THP. При любом вложении $\varphi : V \to W$ направленной звезды S_k в максимальный подграф $H - c$, полученный удалением корня c, имеет место $\varphi(c) = w$, т. е. корень c звезды S_k отображается в вершину w орграфа $H - c$.

Доказательство. Корень c направленной звезды S_k имеет k инцидентных дуг, при этом вершина c соединена дугой с каждым листом v_i, $0 \leqslant i \leqslant k-1$. В любом максимальном подграфе любого THP для направленной звезды лист v_i может быть соединён дугой не более чем с двумя вершинами: с вершиной c и с вершиной w. Следовательно, так как корень c должен быть соединен дугами не менее чем с k вершинами, где $k > 2$, то единственной вершиной, в которую может отображаться центр c, в максимальном подграфе $H - c$ является вершина w. Это возможно, разумеется, при достаточном количестве добавленных дуг в THP, инцидентных вершине w. Таким образом, при любом вложении $\varphi : V \to W$ направленной звезды S_k в максимальный подграф $H - c$ имеет место $\varphi(c) = w$. ■

Следствие 1. В любом THP для направленной звезды S_k выполняется $d^+(c) \leqslant d^+(w)$, $d^-(c) \leqslant d^-(w)$, и вершина w соединена хотя бы одной дугой с каждым листом v_i, $0 \leqslant i \leqslant k-1$.

Следующий алгоритм позволяет построить THP для направленной звезды S_k.

Алгоритм 3

1) Добавим к направленной звезде S_k вершину w;
2) для каждого листа v_i, такого, что $d^+(v_i) = 1$, добавим дугу (v_i, w);
3) для каждого листа v_i, такого, что $d^-(v_i) = 1$, добавим дугу (w, v_i);
4) добавим дугу (c, w).

Асимптотическая сложность алгоритма 3 равна $O(k)$, так как за $O(1)$ можно проанализировать каждую из $k+1$ вершин.

THP, которое строит алгоритм 3, содержит $2k+1$ дуг, так как к направленной звезде S_k добавляется $k+1$ дуга.

THP для направленной звезды S_3, построенное по алгоритму 3, показано на рис. 6.

Теорема 5. Алгоритм 3 корректен.

Доказательство. Рассмотрим орграф $H = (W, \beta)$, полученный алгоритмом 3. Для доказательства корректности предложенного алгоритма необходимо и достаточно показать выполнение всех пунктов теоремы 1 (критерия THP для ориентированных графов).

1. $|W| = |V| + 1$, так как $V = \{v_1, v_2, \ldots, v_n\}, W = \{v_1, v_2, \ldots, v_n, w\}$.

2. Очевидно, что $S_k \cong H - w$ в силу построения в алгоритме 3.

3. Покажем, что направленная звезда S_k вкладывается в каждый максимальный подграф $H = (W, \beta)$.

Рассмотрим максимальный подграф $H - c$, полученный удалением корня c. Тогда существует вложение $\varphi : V \to W$, такое, что $\varphi(c) = w$ и $\varphi(v_i) = v_i$, $0 \leqslant i \leqslant k-1$. Действительно, в орграфе $H - c$ по построению $d^+(c) = d^+(w)$ и $d^-(c) = d^-(w)$, орграф $H - c$ является направленной звездой и $H - c \cong S_k$.

Рассмотрим максимальный подграф $H - l$, полученный удалением одного листа l. Если $d^-(l) = 1$, то существует вложение $\varphi : V \to W$, такое, что $\varphi(c) = c$, $\varphi(l) = w$ и $\varphi(v_i) = v_i$, $0 \leqslant i \leqslant k-1$, $v_i \neq l$; если же $d^+(l) = 1$, то существует вложение $\varphi : V \to W$, такое, что $\varphi(c) = w$, $\varphi(l) = c$ и $\varphi(v_i) = v_i$, $0 \leqslant i \leqslant k-1$, $v_i \neq l$.

4. Свойство неприводимости. Докажем, что при удалении из орграфа $H = (W, \beta)$ любой дуги, инцидентной вершине w, получится орграф, не являющийся расширением для направленной звезды S_k.

Пусть из H удалена дуга между вершиной w и некоторым листом v_i, $0 \leqslant i \leqslant k-1$. Тогда в орграфе $H - c$ вершина v_i является изолированной. Вложение невозможно.

Пусть из H удалена дуга (c, w), добавленная в п. 4 алгоритма 3. Тогда в орграфе $H - v_i$, где v_i — некоторый лист, имеет место $d^+(w) + d^-(w) < k$ и $d^+(c) + d^-(c) < k$, то есть корень c направленной звезды S_k также не отображается в вершину w. ■

THP для направленной звезды S_k, построенное по алгоритму 3, не единственное. Например, для направленной звезды S_3 (см. рис. 6) существует другое THP (рис. 7).

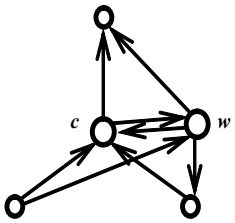


Рис. 7. THP для направленной звезды S_3

3. Т-неприводимое расширение для объединения орграфа и его Т-неприводимого расширения

Аналогичная задача, но для неориентированных графов, рассмотрена в [2].

Теорема 6 (THP для объединения орграфа и его THP). Пусть $G = (V, \alpha)$ — орграф и $H = (W, \beta)$ — одно из его THP. Тогда одним из THP для орграфа $G \cup H$ является орграф $H \cup H'$, где $H' = (W', \beta')$ — изоморфная копия орграфа $H = (W, \beta)$.

Доказательство. Необходимо показать выполнение всех пунктов критерия THP для орграфов.

1. $|W \cup W'| = |W| + |W'| = |W| + (|V| + 1)| = (|W| + |V|) + 1 = |W \cup V| + 1$.

2. Так как орграф $H' = (W', \beta')$ является изоморфной копией одного из THP для орграфа $G = (V, \alpha)$, то в нём по критерию THP существует вершина w, такая, что $H' - w \cong G$. Следовательно, $(H \cup H') - w \cong H \cup (H' - w) \cong H \cup G$.

3. Так как $H = (W, \beta)$ является одним из THP для орграфа $G = (V, \alpha)$, то $G = (V, \alpha)$ вкладывается в любой максимальный подграф $H - u$ орграфа $H = (W, \beta)$, где $u \neq w$. При удалении любой вершины u, такой, что $u \neq w$, получим орграф $H \cup (H' - u)$. Тогда исходный орграф $G \cup H$ вкладывается в орграф $H \cup (H' - u)$ следующим образом: орграф G вкладывается в орграф $H' - u$, а орграф H вкладывается сам в себя.

4. Удалим произвольную дугу (u, w) из орграфа $H \cup H'$. Получим орграф $H \cup H' - (u, w) \cong H \cup (H' - (u, w))$. Рассмотрим его максимальный подграф $(H - w) \cup (H' - (u, w)) \cong G \cup (H' - (u, w))$. Очевидно, что часть H исходного орграфа $G \cup H$ не вкладывается ни в орграф G, ни в орграф $H' - (u, w)$. Таким образом, существует максимальный подграф орграфа $H \cup H' - (u, w)$, в который не вкладывается исходный орграф $G \cup H$. ■

ЛИТЕРАТУРА

1. *Богомолов А. М., Салий В. Н.* Алгебраические основы теории дискретных систем. М.: Наука, Физматлит, 1997. 368 с.
2. *Курносова С. Г.* Т-неприводимые расширения для некоторых классов графов // Теоретические проблемы информатики и её приложений: сб. науч. тр. / под ред. проф. А. А. Сытника. Саратов: Изд-во Сарат. ун-та, 2004. С. 113–125.
3. *Салий В. Н.* Доказательства с нулевым разглашением в задачах о расширении графов // Вестник Томского государственного университета. Приложение. 2003. № 6. С. 63–65.
4. *Абросимов М. Б.* О сложности некоторых задач, связанных с расширениями графов // Матем. заметки. 2010. Т. 88. № 5. С. 643–650.
5. *Курносова С. Г.* Т-неприводимое расширение для симметричных ориентаций цепей // Теоретические проблемы информатики и ее приложений: сб. науч. тр. / под ред. проф. А. А. Сытника. Саратов: Изд-во Сарат. ун-та, 2006. С. 76–81.

ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ В ДИСКРЕТНОЙ МАТЕМАТИКЕ

DOI 10.17223/20710410/22/6

УДК 519.712

ОБ АСИМПТОТИКЕ РЕШЕНИЙ РЕКУРРЕНТНЫХ СООТНОШЕНИЙ СПЕЦИАЛЬНОГО ВИДА И ТЕХНИКЕ КУЛЬМАНА — ЛЮКХАРДТА

В. В. Быкова

Институт математики и фундаментальной информатики Сибирского федерального университета, г. Красноярск, Россия

E-mail: bykvalen@mail.ru

Обсуждаются проблемы решения рекуррентных соотношений, возникающих в анализе вычислительной сложности рекурсивных алгоритмов. Класс рассматриваемых алгоритмов ограничен алгоритмами расщепления, а именно DPLL-алгоритмами, предназначенными для решения задачи пропозициональной выполнимости. Исследована техника Кульмана — Люкхардта, традиционно применяемая при исследовании вычислительной сложности алгоритмов расщепления. Предложена теорема, устанавливающая верхние оценки времени выполнения DPLL-алгоритма в случае сбалансированного расщепления. Теорема расширяет возможности техники Кульмана — Люкхардта, так как учитывает неоднородность рекуррентного соотношения, описывающего время работы DPLL-алгоритма.

Ключевые слова: *вычислительная сложность рекурсивных алгоритмов, алгоритмы расщепления, решение рекуррентных соотношений.*

Введение

Задача пропозициональной выполнимости (англ. вариант — SATISFIABILITY, или коротко SAT) является одной из известных задач дискретной математики и теории сложности вычислений. Это задача о выполнимости формулы логики высказываний. Для формул, записанных в конъюнктивной нормальной форме, задача SAT является NP-полной (С. А. Кук, 1971) [1]. Многие проблемы из класса NP естественным образом сводятся к данной задаче [2].

Задача SAT формулируется следующим образом. Пусть X — конечное множество переменных, принимающих значения `false` и `true`. Формула F в конъюнктивной нормальной форме (КНФ) представляет собой конъюнкцию конечного числа элементарных дизъюнкций. Требуется установить, существует ли для входной формулы F, заданной в КНФ над множеством переменных X, набор значений переменных, при котором каждая дизъюнкция формулы F принимает значение `true` (такой набор принято называть выполняющим). Формула считается выполнимой, если для неё существует хотя бы один выполняющий набор, и невыполнимой, если выполняющего набора не существует.

Входной формуле F обычно приписывают числовую величину, которая принимает неотрицательные целые значения (принадлежит $\mathbb{Z}_+$) и характеризует размерность этой формулы. Такие величины называют мерами сложности формулы логики высказываний. В качестве основных мер сложности рассматривают: $N = |X|$ — число

переменных; K — число дизъюнкций; $L = |F|$ — длину формулы F, как количество входящих в неё литералов (переменных или их отрицаний). Относительно этих мер сложности традиционно определяют вычислительную сложность (время работы) исследуемого алгоритма решения задачи SAT.

Задача пропозициональной выполнимости NP-полная, поэтому маловероятно, что она может быть решена за полиномиальное время. Так, алгоритм полного перебора, анализирующий все 2^N различных наборов значений N переменных, устанавливает тривиальную верхнюю оценку вычислительной сложности задачи SAT в худшем случае. Поиск алгоритмов, работающих быстрее полного перебора, активно ведётся с начала 60-х годов прошлого века и до настоящего времени. Среди них наиболее известными являются алгоритмы расщепления [3]. Алгоритмы расщепления — это декомпозиционные алгоритмы. Они сводят задачу SAT для входной формулы F к конечному числу подзадач, каждая из которых — это SAT для формул меньшей сложности, чем сложность F. Причём это сведение может быть детерминированным (алгоритм производит рекурсивные вызовы самого себя для формул меньшей сложности, полученных при расщеплении F) или вероятностным (случайный выбор одной из образованных после расщепления формул). Детерминированные алгоритмы расщепления принято назвать DPLL-алгоритмами (по первым буквам фамилий их авторов: M. Davis, H. Putman, G. Logemann, D. Loveland) [4, 5]. Значительная часть алгоритмов, разработанных за последние пятьдесят лет для задачи SAT, базируется на идеях, которые заложены в DPLL-алгоритмах. Далее везде под алгоритмами расщепления будем иметь в виду DPLL-алгоритмы.

В данной работе алгоритмы расщепления рассматриваются с точки зрения вычислительной сложности. Будем использовать традиционные для анализа алгоритмов асимптотические обозначения [1]. Например, для вещественнозначных функций $q(n)$ и $h(n)$ от $n \in \mathbb{Z}_+$ запись $q(n) = \Theta[h(n)]$ означает, что существуют такие константы $c_1, c_2 > 0$ и такое $n_0 \in \mathbb{Z}_+$, что $c_1 h(n) \leqslant q(n) \leqslant c_2 h(n)$ при $n \geqslant n_0$. Аналогично, пишут $q(n) = \mathrm{O}[h(n)]$, если найдётся константа $c > 0$ и такое $n_0 \in \mathbb{Z}_+$, что $q(n) \leqslant c \cdot h(n)$ при всех $n \geqslant n_0$. Точно так же $q(n) = \mathrm{o}[h(n)]$ тогда и только тогда, когда для любой положительной константы c существует такое $n_0 \in \mathbb{Z}_+$, что всегда $q(n) < c \cdot h(n)$ при $n \geqslant n_0$. Все эти обозначения предполагают, что функции $q(n)$ и $h(n)$ неотрицательные асимптотически, т. е., по крайней мере, начиная с некоторого значения n.

В п. 1 работы приводится общая схема функционирования алгоритма расщепления; в п. 2 определяется вид рекуррентного соотношения, описывающего время работы DPLL-алгоритма, исследуются вычислительные возможности получения для него решения в явном виде или в виде асимптотических оценок, а также анализируется техника Кульмана — Люккхардта, традиционно применяемая при рассмотрении вычислительной сложности алгоритмов расщепления; в п. 3 представляется основной результат — теорема, которая устанавливает верхние оценки времени выполнения DPLL-алгоритма в случае сбалансированного расщепления. В отличие от техники Кульмана — Люккхардта, данная теорема определяет асимптотические оценки в явном виде и позволяет учитывать временные затраты алгоритма, которые не связаны с его рекурсивными вызовами.

1. Схема работы алгоритма расщепления

В алгоритмах расщепления снижение меры сложности входной формулы осуществляется с помощью редуцирования и расщепления этой формулы. В общем виде схему работы DPLL-алгоритма можно представить следующим образом.

Вход: формула F в КНФ над X.
Ответ: «Выполнима» / «Невыполнима».

1. Редуцировать F, т. е. преобразовать формулу F в формулу F_0 меньшей сложности с помощью конечного набора правил редукции. Эти правила применять к F до тех пор, пока хотя бы одно из них применимо.
2. Если задача SAT тривиально решается для F_0, то выдать соответствующий ответ.
3. Выбрать переменную $x \in X$ по определённому правилу. Таких переменных может быть несколько. Их число определяется правилом расщепления, употребляемом на следующем шаге.
4. Выполнить расщепление формулы F_0 на конечное число формул меньшей сложности по заданному закону — правилу расщепления, используя различные значения переменных, выбранных на шаге 3.
5. Осуществить рекурсивные вызовы данного алгоритма применительно к полученным формулам. Выдать ответ, основываясь на результатах, возвращённых рекурсивными вызовами: если хотя бы один из рекурсивных вызовов вернул выполняющий набор, то ответ — «Выполнима», в противном случае — «Невыполнима».

Таким образом, DPLL-алгоритм — рекурсивный алгоритм, вызывающий сам себя на формулах меньшей сложности. Рекурсивные вызовы прекращаются, как только задача SAT решается тривиально, т. е. если в результате преобразований получается формула, для которой решение задачи может быть найдено за полиномиальное время. В частности, к полиномиально разрешимым случаям приводят следующие классы формул: пустые формулы (они не содержат дизъюнкций вообще, всякая такая формула интерпретируется как `true`); формулы в 2-КНФ (каждая их дизъюнкция содержит не более двух литералов); хорновские формулы (в них каждая дизъюнкция содержит переменные с отрицанием, кроме одной переменной) и ряд других классов формул [6, 7]. К трудоёмкости шагов DPLL-алгоритма предъявляют ряд естественных требований. Суть их в том, что все действия, не связанные с рекурсивными вызовами (это шаги 1–4 алгоритма), должны выполняться за полиномиальное время относительно выбранной меры сложности формул.

Различные расщепляющие алгоритмы отличаются друг от друга в основном правилами редукции, стратегией выбора переменных на шаге 3, правилами расщепления и используемой мерой сложности формул. Уже предложено и доказано значительное число правил редукции, наиболее известные из них можно найти в работах [3 – 5]. Среди них: «единичная дизъюнкция», «чистый литерал», «резолюция», «поглощение» и др. Типичные стратегии по выбору переменных для расщепления — это, чаще всего, эвристики вида: «взять любую переменную из самой короткой дизъюнкции», «предпочтение отдать переменной, входящей в наибольшее число дизъюнкций». Правила расщепления также могут быть различными. Например, простейшее правило расщепления сводится к замене формулы F_0 на две формулы

$$F_1 = F_0[x] \text{ и } F_2 = F_0[\neg x]. \tag{1}$$

При этом формула F_1 получается из формулы F_0 присваиванием значения `true` переменной x. Это влечёт удаление всех дизъюнкций, содержащих x без отрицания, и удаление литерала $\neg x$ в оставшихся дизъюнкциях. Формула F_2 получается аналогичным образом — присваиванием переменной x значения `false` и дальнейшим соответствующим упрощением полученной формулы. Таким образом, при данном правиле расщепления DPLL-алгоритм выполняет два рекурсивных вызова для формул F_1 и F_2

соответственно. Своеобразные правила расщепления, например такие, как

$$F_1 = F_0[x, y], \quad F_2 = F_0[x, \neg y], \quad F_3 = F_0[\neg x]; \tag{2}$$

$$F_1 = F_0[x, y], \quad F_2 = F_0[x, \neg y], \quad F_3 = F_0[\neg x, y], \quad F_4 = F_0[\neg x, \neg y], \tag{3}$$

порождают три и более рекурсивных вызова алгоритма. Во всех случаях уменьшение меры сложности образованных формул в сравнении с F достигается за счёт редуцирования F и расщепления F_0 по выбранным переменным.

Важно отметить следующие факты. Если расщепление выполняется по правилу $F_1 = F_0[x]$ и $F_2 = F_0[\neg x]$, то DPLL-алгоритм в худшем случае (например, для невыполнимой формулы, к которой неприменимы рассматриваемые правила редукции) перебирает все 2^N различных наборов значений N переменных. Как будет показано ниже, подобное наблюдается и при других правилах расщепления, когда отсутствует этап редуцирования и в качестве меры сложности входной формулы выступает число переменных. Это вполне согласуется с тем, что задача SAT является NP-полной. Для данной задачи (когда отсутствуют какие-либо ограничения на длину и структуру дизъюнкций во входной формуле) до сих пор не найдено методов решения, в том числе и DPLL-алгоритмов, со временем работы $\mathrm{O}(c^N)$, где $c < 2$. Между тем известны алгоритмы расщепления с оценками вычислительной сложности $\mathrm{O}(1{,}238823^K)$, $\mathrm{O}(1{,}073997^L)$ относительно числа K дизъюнкций и длины L входной формулы соответственно [8]. Область применения DPLL-алгоритмов не ограничивается задачей SAT. Алгоритмы расщепления успешно применяются при решении многих NP-полных задач, например MAX-SAT, MAX-2-SAT [9]. Существует большое количество научных публикаций, посвященных этому классу алгоритмов. Более того, имеется много работ, описывающих DPLL-алгоритмы для SAT, где каждая следующая улучшает результат предыдущей за счёт введения иной меры сложности формул, оригинальных правил редукции и расщепления, более тщательного анализа алгоритмов. При этом предлагаемые алгоритмы сравниваются между собой преимущественно с точки зрения O-оценки (верхней асимптотической оценки) времени их работы в худшем случае.

2. Рекуррентные соотношения в анализе алгоритмов расщепления: асимптотика решений и техника Кульмана — Люкхардта

В общем случае анализ вычислительной сложности DPLL-алгоритмов представляет собой весьма непростую задачу ввиду их рекурсивности. В настоящее время основным математическим инструментом исследования вычислительной сложности рекурсивных алгоритмов является метод рекуррентных соотношений [1, 10]. Идея данного метода состоит в построении и решении рекуррентного соотношения, которому удовлетворяет функция, описывающая время работы рассматриваемого алгоритма. Однако данный метод не универсален. Существует несколько серьёзных препятствий для его практического воплощения. Во-первых, для построения рекуррентного соотношения необходимо, чтобы преобразование, уменьшающее значение параметра рекурсии, было линейным относительно этого параметра [11]. Во-вторых, если рекуррентное соотношение всё же найдено, то нет никакой гарантии, что удастся установить явную форму или асимптотическую оценку его решения. Причина в том, что общих методов решения рекуррентных соотношений до сих пор не найдено. Известны лишь методы решения некоторых классов рекуррентных соотношений [12, 13]. Интересно выяснить, какой вид имеет рекуррентное соотношение, описывающее время работы DPLL-алгоритма, и какие вычислительные возможности имеются для нахождения его решения? Дадим ответы на поставленные вопросы.

Пусть неотрицательная целая величина n — некоторая мера сложности входной формулы F (например, количество переменных или число дизъюнкций). Предположим, что DPLL-алгоритм вначале редуцирует, а затем расщепляет формулу F на $m > 1$ формул $F_1, F_2, \ldots, F_m$ сложности $n_1, n_2, \ldots, n_m$ соответственно, $n_m \leqslant \ldots \leqslant n_2 \leqslant n_1 < n$. Пусть $T(n)$ — монотонно неубывающая функция от n, представляющая общее время работы алгоритма расщепления для формулы F сложности n; её областью значений является множество неотрицательных вещественных чисел, а областью определения — множество неотрицательных целых чисел. Из схемы работы DPLL-алгоритма следует, что значение $T(n)$ складывается из времени выполнения рекурсивных вызовов для формул $F_1, F_2, \ldots, F_m$, а также из временных затрат, не связанных с рекурсивными вызовами:

$$T(n) = T(n_1) + T(n_2) + \cdots + T(n_m) + f(n).$$

Здесь $f(n)$ — функция полиномиального порядка роста [14], которая как раз и учитывает нерекурсивные затраты DPLL-алгоритма, т. е. время, затрачиваемое им на редуцирование F, выбор переменных для расщепления, построение решения для F из решений для $F_0, F_1, F_2, \ldots, F_m$ и на прочие подобные действия. Выражение для $T(n)$ можно представить так:

$$T(n) = T(n - t_1) + T(n - t_2) + \cdots + T(n - t_m) + f(n), \tag{4}$$

где $t_1 = n - n_1$, $t_2 = n - n_2$, $\ldots$, $t_m = n - n_m$, $1 \leqslant t_1 \leqslant t_2 \leqslant \ldots \leqslant t_m < n$. В публикациях по DPLL-алгоритмам вектор $(t_1, t_2, \ldots, t_m)$ обычно называют *вектором расщепления*. Далее будем придерживаться этой терминологии. Вектор расщепления назовём *сбалансированным*, если все его элементы равны, и *максимально несбалансированным*, если в нём нет ни одной пары равных элементов.

Поскольку в векторе расщепления возможны равные элементы, то равенство (4) можно преобразовать к следующему неоднородному линейному рекуррентному соотношению порядка $k \geqslant 1$ с постоянными коэффициентами:

$$T(n) = a_1 T(n-1) + a_2 T(n-2) + \cdots + a_k T(n-k) + f(n), \quad n \geqslant k; \tag{5}$$

$$T(n) = \Theta(1), \quad 0 \leqslant n \leqslant k-1. \tag{6}$$

Начальные условия (6) свидетельствуют о том, что время работы алгоритма расщепления ограничено сверху и снизу константами для формулы сложности не более $k-1$. В соотношении (5) коэффициенты $a_1, a_2, \ldots, a_k$ — положительные целые константы, не равные нулю одновременно. Для рекуррентного соотношения порядка k естественными являются условия $a_1, a_2, \ldots, a_{k-1} \geqslant 0$, $a_k \geqslant 1$.

Общих методов нахождения явного вида решения рекуррентного соотношения (5) неизвестно. Между тем при $f(n) \equiv 0$ к нему возможно применение классической теоремы Пуанкаре — Перрона [15, 16]. Согласно теореме Пуанкаре — Перрона, общий вид решения однородного линейного рекуррентного соотношения с постоянными коэффициентами устанавливается через корни соответствующего характеристического многочлена. Так, если $\alpha_1, \alpha_2, \ldots, \alpha_k$ — простые (некратные) корни уравнения

$$x^k - a_1 x^{k-1} - a_2 x^{k-2} - \cdots - a_k = 0, \tag{7}$$

то решение рекуррентного соотношения

$$T(n) = a_1 T(n-1) + a_2 T(n-2) + \cdots + a_k T(n-k)$$

определяется формулой

$$T(n) = c_1\alpha_1^n + c_2\alpha_2^n + \cdots + c_k\alpha_k^n, \tag{8}$$

где c_1, c_2, …, c_k — постоянные величины, значения которых вычисляются, исходя из начальных условий (6). Таким образом, если α — наибольший положительный вещественный корень уравнения (7), то, согласно (8), верна асимптотическая оценка

$$T(n) = O(\alpha^n). \tag{9}$$

Следует заметить, что (7) по правилу перемены знаков в последовательности коэффициентов алгебраического уравнения [17] имеет ровно один положительный вещественный корень. В силу единственности он простой. Именно этот корень и выступает в роли α. В работах, посвящённых DPLL-алгоритмам, величину α называют числом расщепления и пишут $\alpha = \alpha(t_1, t_2, \ldots, t_m)$, подчёркивая тем самым зависимость значения α от вектора расщепления $(t_1, t_2, \ldots, t_m)$. Применение к (7) метода Лагранжа для определения границ положительных вещественных корней алгебраического уравнения [17] приводит к оценке $\alpha \leqslant 2$, в которой равенство достижимо в ряде случаев. В частности,

$$\alpha(1,1) = \alpha(1,2,2) = \alpha(2,2,2,2) = 2,$$

в чём нетрудно убедиться, решив надлежащие характеристические уравнения. Примечательно, что векторы расщепления $(1,1)$, $(1,2,2)$, $(2,2,2,2)$ отвечают DPLL-алгоритмам, в которых отсутствует этап редуцирования, в роли n выступает число N переменных, а расщепление выполняется по правилам (1)–(3) соответственно. В данном случае оценка (9) принимает вид $O(2^N)$. Это подтверждает приведённый в п. 1 факт, что DPLL-алгоритмы не способны улучшить время работы алгоритма полного перебора, если сложность формул измерять числом входящих в них пропозициональных переменных.

Для нахождения чисел расщепления нет общих вычислительных формул, поэтому они находятся преимущественно с помощью численных методов [17], исходя из вектора расщепления. В ряде частных случаев можно воспользоваться свойствами чисел расщепления, которые легко доказываются. Некоторые из них приведены в работе [9]. Например, для сбалансированного вектора расщепления с $m > 1$ равными элементами справедливо равенство

$$\alpha(\underbrace{k, k, \ldots, k}_{m \text{ раз}}) = m^{1/k} = 2^{(\log_2 m)/k},$$

а для максимально несбалансированного вектора расщепления $(1, 2, \ldots, m)$ верно строгое неравенство $\alpha(1, 2, \ldots, m) < 2$. Кроме того, всегда

$$\alpha(t_1, t_2, \ldots, t_m) \leqslant \alpha(p_1, p_2, \ldots, p_m),$$

если $t_i \geqslant p_i$ для всех i. Отсюда при $m > 1$

$$m^{1/m} = \alpha(\underbrace{m, m, \ldots, m}_{m \text{ раз}}) < \alpha(1, 2, \ldots, m) < 2,$$

т. е. сбалансированный вектор расщепления приводит к меньшему значению α, чем максимально несбалансированный с тем же числом элементов.

Разработчики DPLL-алгоритмов для анализа вычислительной сложности этих алгоритмов традиционно применяют технику Кульмана — Люкхардта [18, 19]. Многие оценки для задачи SAT и родственных с ней NP-полных задач получены с помощью данной техники. Её подробное описание дано в работе [3]. Исследование техники Кульмана — Люкхардта показало, что она полностью базируется на теореме Пуанкаре — Перрона и приведённых выше особенностях алгебраического уравнения (7), хотя её описание не содержит явных ссылок на эти известные математические факты [18, 19]. Неоднородность рекуррентного соотношения (5) в технике Кульмана — Люкхардта игнорируется. Чтобы как-то обозначить существование этой неоднородности, разработчиками DPLL-алгоритмов всегда делается оговорка, что оценка времени работы алгоритма получена с точностью до полиномиального сомножителя, для чего используется запись

$$T(n) = \mathrm{poly}(n) \cdot \mathrm{O}(\alpha^n). \tag{10}$$

Степень полинома в (10) не уточняется (в общем случае это сделать невозможно, ведь число расщепления α, как правило, приходится находить численно).

Техника Кульмана — Люкхардта имеет свои достоинства, которые способствуют её широкому практическому применению. Во-первых, данная техника служит единым инструментом, с помощью которого оценивается время работы алгоритмов расщепления. Это даёт возможность сопоставлять такие алгоритмы между собой. Во-вторых, численный характер техники позволяет её автоматизировать. Известны программы для автоматического доказательства верхних оценок DPLL-алгоритмов [20]. В-третьих, техника Кульмана — Люкхардта максимально аккумулирует математические возможности вычисления оценок для решений рекуррентных соотношений вида (5). Одним из возможных направлений развития данной техники является рассмотрение случаев, когда $f(n) \not\equiv 0$. Игнорирование неоднородности в (5) — это основной недостаток техники Кульмана — Люкхардта. Между тем учёт этой неоднородности чрезвычайно важен с точки зрения развития идей, заложенных в DPLL-алгоритмах. В самом деле, постоянное расширение списка правил редукции и расщепления направлено на уменьшение числа расщепления α, а значит, и второго сомножителя правой части (10). Однако это неизбежно приводит к увеличению нерекурсивных расходов DPLL-алгоритма и первого сомножителя в правой части (10). Таким образом, принимая во внимание допустимые значения n, целесообразно сопоставлять вклад обоих сомножителей в значение функции $T(n)$ (ведь при $1 \leqslant n \leqslant 100$ значение $10n^3$ гораздо больше, чем $1{,}1^n$). Предлагаемая ниже теорема даёт возможность учитывать неоднородность в (5) для одного частного случая, когда расщепление является сбалансированным и $f(n) = \Theta(n^\tau)$, где τ — неотрицательная вещественная константа.

3. Теорема для сбалансированного расщепления

Расщепление формулы F является сбалансированным относительно меры n, если ему отвечает сбалансированный вектор расщепления

$$(\underbrace{k, k, \ldots, k}_{a \text{ раз}}),$$

где a — число формул, получаемых при расщеплении, и каждая из этих формул характеризуется мерой сложности $(n - k)$, где $a > 1$. Заметим, что здесь число формул, получаемых в процессе расщепления, обозначено через a (вместо m, используемого ранее).

Рекуррентное соотношение (5) при сбалансированном расщеплении значительно упрощается, поскольку $a_1 = a_2 = \cdots = a_{k-1} = 0$, $a_k = a$. В данном случае (5), (6) можно записать так:

$$T(n) = aT(n-k) + f(n), \text{если } n \geqslant k; \tag{11}$$

$$T(n) = \Theta(1), \text{если } 0 \leqslant n \leqslant k-1. \tag{12}$$

Для такого неоднородного линейного рекуррентного соотношения возможно получение асимптотических оценок в явном виде, что подтверждает следующая теорема.

Теорема 1. Пусть задано рекуррентное соотношение (11) с начальными условиями (12), где $a > 1$, $k \geqslant 1$ — целые константы. Пусть $\tau \geqslant 0$ — вещественная константа. Тогда при любом достаточно большом значении n верны оценки

$$T(n) = \mathrm{O}(n^\tau a^{n/k}), \text{если } f(n) = \Theta(n^\tau); \tag{13}$$

$$T(n) = \mathrm{O}(a^{n/k}), \text{если } f(n) \equiv 0. \tag{14}$$

Доказательство. Покажем справедливость данной теоремы путём рассмотрения ряда возможных случаев.

С л у ч а й 1. Пусть $f(n) \equiv 0$. Оценка (14) прямо следует из (9). В данном случае $\alpha = a^{1/k}$.

С л у ч а й 2. Предположим, что n кратно k, т. е. представимо в виде $n = kp$, $p \in \mathbb{Z}_+$. Пусть также $f(n) = bn^\tau$, где $b > 0$, $\tau \geqslant 0$ — вещественные константы.

Введём функцию $Q(p) = T(kp)$. Тогда $T(kp-k) = T(k(p-1)) = Q(p-1)$ и (11),(12) можно представить следующим образом:

$$T(kp) = aT(kp-k) + bk^\tau p^\tau, \quad \text{если } p > 0,$$
$$T(kp) = \Theta(1), \quad \text{если } p = 0,$$

или

$$Q(p) = aQ(p-1) + bk^\tau p^\tau, \quad \text{если } p > 0, \tag{15}$$
$$Q(p) = \Theta(1), \quad \text{если } p = 0.$$

Подстановка соотношения (15) самого в себя при условии, что $Q(0) = \Theta(1)$, даёт

$$Q(p) = a^p\Theta(1) + bk^\tau S(a,p,\tau), \tag{16}$$

где $S(a,0,\tau) = 0$, а при $p > 0$

$$S(a,p,\tau) = a^{p-1}\cdot 1^\tau + a^{p-2}\cdot 2^\tau + \cdots + a^0\cdot p^\tau = \sum_{i=1}^{p} a^{p-i}i^\tau.$$

Найдём значение конечной суммы $S(a,p,\tau)$. При $\tau = 0$ и $a > 1$ справедливо равенство

$$S(a,p,0) = \sum_{i=1}^{p} a^{p-i}i^0 = a^{p-1} + a^{p-2} + \cdots + a^0 = \frac{a^p - 1}{a-1}.$$

Таким образом, при $\tau = 0$ для рекуррентного соотношения (15) верна оценка

$$Q(p) = a^p\Theta(1) + bk^\tau S(a,p,\tau) = a^p\Theta(1) + b\frac{a^p-1}{a-1} = \Theta(a^p).$$

Поскольку $p = n/k$, то

$$Q(p) = T(kp) = T(n) = \Theta(a^{n/k}),$$

что отвечает оценке (13), ведь Θ-оценка функции всегда влечет справедливость соответствующей O-оценки той же функции.

Для произвольного вещественного $\tau \geqslant 0$ значение суммы $S(a, p, \tau)$ установить крайне сложно. Между тем значение $S(a, p, \tau)$ можно оценить сверху. В работе автора [10] доказано неравенство

$$S(a, p, \tau) \leqslant p^\tau \frac{a^p - 1}{a - 1},$$

из которого следует, что

$$Q(p) = a^p\Theta(1) + bk^\tau S(a, p, \tau) \leqslant a^p\Theta(1) + bk^\tau p^\tau \frac{a^p - 1}{a - 1},$$

$$T(kp) \leqslant a^p\Theta(1) + bk^\tau p^\tau \frac{a^p - 1}{a - 1}. \tag{17}$$

Принимая во внимание, что $k^\tau p^\tau = n^\tau$ и $p = n/k$, из (17) вытекает оценка

$$T(n) = \mathrm{O}(n^\tau a^{n/k}), \tag{18}$$

которая совпадает с (13).

С л у ч а й 3. Пусть по-прежнему $n = kp$, $p \in \mathbb{Z}_+$. Предположим теперь, что $f(n) = \Theta(n^\tau)$, $\tau \geqslant 0$. Функцию $f(n) = \Theta(n^\tau)$ можно записать в виде $f(n) = bn^\tau + \mathrm{o}(n^\tau)$, где $b > 0$ — некоторая вещественная константа. Тогда в правой части равенства (16) появляется дополнительное слагаемое

$$\mathrm{o}(k^\tau p^\tau \sum_{i=1}^{p} a^{p-i}) = \mathrm{o}\left(k^\tau p^\tau \frac{a^p - 1}{a - 1}\right) = \mathrm{o}\left(k^\tau p^\tau a^p\right). \tag{19}$$

С учётом этого неравенство (17) преобразуется к виду

$$T(kp) \leqslant a^p\Theta(1) + bk^\tau p^\tau \frac{a^p - 1}{a - 1} + \mathrm{o}\left(k^\tau p^\tau a^p\right),$$

из которого следует, что дополнительное слагаемое в (16) не изменяет оценку (18).

С л у ч а й 4. В предыдущих двух случаях предполагалось, что n кратно k. Однако это предположение можно снять, исходя из монотонности функции $T(n)$, поскольку если n не кратно k, то всегда найдется ближайшее сверху n^*, такое, что $n^* = kp$, $p \in \mathbb{Z}_+$, и $T(n) \leqslant T(n^*)$. Очевидно, что в этой ситуации порядок роста функции $T(n)$, выраженный верхней асимптотической оценкой (13), остаётся прежним для любого целого неотрицательного n. ■

Доказанная теорема расширяет возможности техники Кульмана — Люккхардта, так как учитывает неоднородность рекуррентного соотношения, описывающего время работы DPLL-алгоритма в случае сбалансированного расщепления. Теорема может быть применена и тогда, когда расщепление не является сбалансированным. Для этого необходимо сбалансировать вектор расщепления и воспользоваться монотонностью функции $T(n)$. Конечно, полученная оценка может оказаться хуже реальной, но зато она

определяется в явном виде и без привлечения процедуры нахождения числа расщепления. Покажем это на примере. Пусть задано рекуррентное соотношение

$$T(n) = 2T(n-6) + 2T(n-7) + n^3. \tag{20}$$

В данном случае вектор расщепления равен (6, 7). Техника Кульмана — Люкхардта приводит к оценке

$$T(n) = \mathrm{poly}(n) \cdot \mathrm{O}(1{,}24^n).$$

Поскольку $T(n-7) \leqslant T(n-6)$, то возможен переход к сбалансированному вектору расщепления (6, 6) и соответствующему рекуррентному соотношению

$$T(n) \leqslant 4T(n-6) + n^3.$$

Применение теоремы 1 (здесь $a = 4$, $k = 6$) даёт оценку

$$T(n) = \mathrm{O}(n^3 1, 26^n),$$

которая учитывает неоднородность соотношения (20). Какой из полученных оценок отдать предпочтение, зависит от специфики исследуемого алгоритма расщепления и входных данных решаемой задачи.

ЛИТЕРАТУРА

1. *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы: построение и анализ. М.: Вильямс, 2012.
2. *Отпущенников И. В., Семёнов А. А.* Технология трансляции комбинаторных проблем в булевы уравнения // Прикладная дискретная математика. 2011. № 1(11). С. 96–115.
3. *Всемирнов М. А., Гирш Э. А., Данцин Е. Я., Иванов С. В.* Алгоритмы для пропозициональной выполнимости и верхние оценки их сложности // Теория сложности вычислений. VI. Зап. научн. сем. ПОМИ. СПб., 2001. Т. 277. С. 14–46.
4. *Davis M., Logemann G., and Loveland D.* A machine program for theorem-proving // Comm. ACM. 1962. No. 5(7). P. 394–397.
5. *Davis M. and Putman H.* A computing procedure for quantification theory // J. ACM. 1960. No. 7(3). P. 201–215.
6. *Franco J. and Gelder A. V.* A perspective on certain polynomial-time solvable classes of Satisfiability // Discr. Appl. Math. 2003. V. 125. Iss. 2–3. P. 177–214.
7. *Алексеев В. Б., Носов В. А.* NP-полные задачи и их полиномиальные варианты. Обзор // Обозрение прикладной и промышленной математики. 1997. Т. 4. Вып. 2. С. 165–193.
8. *Hirsch E. A.* New worst-case upper bounds for SAT // J. Automated Reasoning. 2000. No. 24(4). P. 397–420.
9. *Куликов А. С., Куцков К.* Новые верхние оценки для задачи максимальной выполнимости // Дискретная математика. 2009. № 21(1). С. 139–157.
10. *Быкова В. В.* Математические методы анализа рекурсивных алгоритмов // Журнал Сибирского федерального университета. Сер. Математика и физика. 2008. № 1(3). С. 236–246.
11. *Головешкин В. А., Ульянов М. В.* Теория рекурсии для программистов. М.: Физматлит, 2006.
12. *Гельфонд А. О.* Исчисление конечных разностей. М.: Физматлит, 1959.
13. *Грэхем Р., Кнут Д., Паташник О.* Конкретная математика. Основание информатики. М.: Бином. Лаб. знаний, 2006.
14. *Быкова В. В.* Эластичность алгоритмов // Прикладная дискретная математика. 2010. № 2(8). С. 87–95.

15. *Poincare H.* Sur les equations lineaires aux differentielles ordinaires et aux differences finies // Amer. J. Math. 1885. V. 7. P. 203–258.
16. *Perron O.* Uber einen Satz des Herrn Poincare // J. Reine Angewand. Math. 1909. B. 136. S. 17–34.
17. *Бахвалов Н. С.* Численные методы. М.: Бином. Лаб. знаний, 2006.
18. *Kullmann O.* New methods for 3-SAT decision and worst-case analysis // Theor. Computer Sci. 1999. No. 223. P. 1–72.
19. *Kullmann O. and Luckhardt H.* Deciding propositional tautologies: Algorithms and their complexity // Preprint. 1997. `http://www.cs.swan.ac.uk/~csoliver`.
20. *Куликов А. С., Федин С. С.* Автоматические доказательства верхних оценок на время работы алгоритмов расщепления // Теория сложности вычислений. IX. Зап. научн. сем. ПОМИ. СПб., 2004. Т. 316. С. 111–128.

2013 Вычислительные методы в дискретной математике №4(22)

DOI 10.17223/20710410/22/7

УДК 519.1

КОМБИНАТОРНЫЕ ЧИСЛА ДЛЯ ПОДСЧЁТА РАЗБИЕНИЙ КОНЕЧНЫХ МУЛЬТИМНОЖЕСТВ

В. В. Гоцуленко

Институт технической теплофизики НАН Украины, г. Киев, Украина

E-mail: gosul@ukr.net

Рассматриваются комбинаторные числа для подсчёта всех разбиений произвольного конечного мультимножества как в упорядоченную, так и в неупорядоченную сумму его подмультимножеств. Найдены производящие функции для введённых комбинаторных чисел и исследованы некоторые свойства этих чисел.

Ключевые слова: *разбиения мультимножества, диофантовы уравнения, производящие функции.*

Введение

В классическом смысле под разбиением [1, 2] произвольного множества понимают его представление в виде объединения произвольного числа попарно непересекающихся подмножеств. Всякое представление натурального числа суммой натуральных чисел также называется разбиением этого числа. Разбиения конечных множеств изучаются в комбинаторике и теории чисел.

К основным комбинаторным задачам относятся задачи подсчёта и перечисления разбиений данного типа [1, 2]. Разбиения конечных множеств, а также подсчёт числа различных разбиений, удовлетворяющих тем или иным условиям, представляет особый интерес в комбинаторике. В частности, некоторые комбинаторные числа естественно возникают как количества разбиений того или иного вида. Например, число Стирлинга второго рода $S(m,n)$ [1, 2] представляет собой число неупорядоченных разбиений m-элементного множества на n частей. При этом мультиномиальный коэффициент $\binom{m}{k_1 k_2 \dots k_n}$ [1–3] выражает число упорядоченных разбиений m-элементного множества на n частей фиксированных размеров $k_1, k_2, \dots, k_n$. Отметим, что многие комбинаторные задачи приводят к вычислению числа способов разбиения m-элементного множества на n подмножеств, среди которых допускаются и пустые подмножества. Однако в этом случае задача сводится к определению числа всех (неупорядоченных) разбиений заданного m-элементного множества на не более чем n непустых его подмножеств. Число всех неупорядоченных разбиений (без пустых подмножеств) m-элементного множества называется числом Белла B_m [1, 2].

Одно из возможных обобщений приведённых выше комбинаторных конструкций связано с допущением дублирования во множествах некоторых элементов, т. е. переход к мультимножествам.

1. Формализация комбинаторных чисел для подсчёта неупорядоченных разбиений конечных мультимножеств

Обозначим через $X = \{m_1 a_1, m_2 a_2, \dots, m_N a_N\}$ исходное мультимножество, которое необходимо представить в виде неупорядоченной суммы n подмультимножеств X_k,

$k = 1, \ldots, n$. Отметим также, что вектор первичной спецификации [4] каждого из подмультимножеств X_k имеет вид

$$[k_1, k_2, \ldots, k_N]^T, \quad \text{где} \quad 0 \leqslant k_i \leqslant m_i, \ i = 1, \ldots, N.$$

Для произвольного представления $X = \bigcup\limits_{k=1}^{n} X_k$ обозначим через $x_{k_1,k_2,\ldots,k_N}$ число подмультимножеств X_k с вектором первичной спецификации равным $[k_1, k_2, \ldots, k_N]^T$.

Набор целых неотрицательных чисел $x_{k_1,k_2,\ldots,k_N}$, как несложно проверить, является решением системы линейных диофантовых уравнений:

$$\begin{cases} \sum\limits_{k_1=0}^{m_1} \sum\limits_{k_2=0}^{m_2} \ldots \sum\limits_{k_N=0}^{m_N} x_{k_1,k_2,\ldots,k_N} \begin{bmatrix} k_1 \\ k_2 \\ \ldots \\ k_N \end{bmatrix} = \begin{bmatrix} m_1 \\ m_2 \\ \ldots \\ m_N \end{bmatrix}, \\ \sum\limits_{k_1=0}^{m_1} \sum\limits_{k_2=0}^{m_2} \ldots \sum\limits_{k_N=0}^{m_N} x_{k_1,k_2,\ldots,k_N} = n. \end{cases} \tag{1}$$

Данная система содержит $(1 + m_1)(1 + m_2) \ldots (1 + m_N)$ неизвестных $x_{k_1,k_2,\ldots,k_N}$ и $N + 1$ уравнений.

Таким образом, задача сводится к определению числа всех целых неотрицательных решений системы уравнений (1). Обозначим данное число через $S(m_1, m_2, \ldots, m_N; n)$.

Получим производящую функцию для введённых комбинаторных чисел $S(m_1, m_2, \ldots, m_N; n)$. Рассмотрим для этого следующую промежуточную задачу. Обозначим через $A = (a_{ij})_{n \times m}$ произвольную матрицу с целыми неотрицательными элементами, через $b = (b_i)_n$ — произвольный вектор-столбец с целыми неотрицательными элементами. Определим число всех целых неотрицательных решений системы линейных диофантовых уравнений

$$a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{im}x_m = b_i, \ i = 1, \ldots, n. \tag{2}$$

Данное число равно коэффициенту при $t_1^{b_1} t_2^{b_2} \ldots t_n^{b_n}$ в разложении в степенной ряд производящей функции

$$\prod_{j=1}^{m} \left(1 - t_1^{a_{1j}} t_2^{a_{2j}} \ldots t_n^{a_{nj}}\right)^{-1}. \tag{3}$$

Для применения формулы (3) к определению числа целочисленных неотрицательных решений системы диофантовых уравнений (1) перенумеруем неизвестные $x_{k_1,k_2,\ldots,k_N}$, $k_i = 0, 1, \ldots, m_i$, $i = 1, \ldots, N$, с помощью одного индекса. Одно из возможных таких взаимно однозначных соответствий, как несложно проверить, задаётся следующей последовательностью рекурсивно определяемых функций:

$$\begin{gathered} \forall\, (0 \leqslant i_1 \leqslant m_1, 0 \leqslant i_2 \leqslant m_2, \ldots, 0 \leqslant i_N \leqslant m_N) \\ f_2(i_1, i_2) = i_1(m_2 + 1) + i_2 + 1, \\ f_3(i_1, i_2, i_3) = f_2(i_1, i_2) + i_3 f_2(m_1, m_2), \\ \ldots \\ f_N(i_1, i_2, \ldots, i_N) = f_{N-1}(i_1, i_2, \ldots, i_{N-1}) + i_N f_{N-1}(m_1, m_2, \ldots, m_{N-1}). \end{gathered}$$

Следовательно, задача (1) допускает представление в виде (2), если положить

$$A = (a_{ij})_{(N+1)\times f_N(m_1,m_2,\ldots,m_N)}\,, \quad b = (b_i)_{N+1}\,, \tag{4}$$

где

$$\forall p \in \{0,\ldots,m_1\}(a_{1k} = p), k = f_N\left(p, i_2, i_3, \ldots, i_{N-1}, i_N\right), 0 \leqslant i_s \leqslant m_s, s \in \{1,\ldots,N\} \setminus \{1\};$$
$$\forall p \in \{0,\ldots,m_2\}(a_{2k} = p), k = f_N\left(i_1, p, i_3, \ldots, i_{N-1}, i_N\right), 0 \leqslant i_s \leqslant m_s, s \in \{1,\ldots,N\} \setminus \{2\};$$
$$\ldots$$
$$\forall p \in \{0,\ldots,m_N\}(a_{N,k} = p), k = f_N\left(i_1, i_2, \ldots, i_{N-1}, p\right), 0 \leqslant i_s \leqslant m_s, s \in \{1,\ldots,N-1\};$$
$$a_{N+1,k} = 1, k = 1,\ldots,f_N\left(m_1, m_2, \ldots, m_N\right);$$
$$b_k = m_k, k = 1,\ldots,N; \quad b_{N+1} = n.$$

Таким образом, полагая

$$\Psi_{m_1,m_2,\ldots,m_N}\left(t_1, t_2, \ldots, t_{N+1}\right) = \prod_{j=1}^{f_N(m_1,m_2,\ldots,m_N)} \left(1 - t_1^{a_{1j}} t_2^{a_{2j}} \ldots t_{N+1}^{a_{N+1,j}}\right)^{-1},$$

где элементы матрицы A определяются формулами (4), получаем, что число $S\left(m_1, m_2, \ldots, m_N; n\right)$ равно коэффициенту при $t_1^{m_1}\ldots t_N^{m_N} t_{N+1}^n$ в разложении функции $\Psi_{m_1,m_2,\ldots,m_N}\left(t_1, t_2, \ldots, t_{N+1}\right)$ в степенной ряд, т. е. имеет место равенство

$$S\left(m_1, m_2, \ldots, m_N; n\right) = \psi^{m_1,m_2,\ldots,m_N;n}_{m_1,m_2,\ldots,m_N,n},$$

где

$$\Psi_{m_1,m_2,\ldots,m_N}\left(t_1, t_2, \ldots, t_{N+1}\right) = \sum_{r_1,\ldots,r_{N+1}\geqslant 0} \psi^{m_1,m_2,\ldots,m_N;n}_{r_1,r_2,\ldots,r_N,r_{N+1}} t_1^{r_1} t_2^{r_2} \ldots t_N^{r_N} t_{N+1}^{r_{N+1}}.$$

Для введённых комбинаторных чисел справедлива формула

$$S\left(m_1, m_2, \ldots, m_N; m\right) = S\left(m_1, m_2, \ldots, m_N; n\right) \quad \text{для всех} \quad m \geqslant n = \sum_{i=1}^{N} m_i. \tag{5}$$

Действительно, если $n = \sum\limits_{i=1}^{N} m_i$, то в этом случае лишь одно разбиение, состоящее из одноэлементных множеств, не содержит пустых множеств. Поэтому при $n > \sum\limits_{i=1}^{N} m_i$ все разбиения мультимножества $X = \{m_1 a_1, m_2 a_2, \ldots, m_N a_N\}$ получаются путём добавления $n - \sum\limits_{i=1}^{N} m_i$ пустых множеств в каждое разбиение при $n = \sum\limits_{i=1}^{N} m_i$, что устанавливает взаимно однозначное соответствие между рассматриваемыми разбиениями и, таким образом, доказывает формулу (5).

Далее рассматриваются некоторые частные случаи и примеры вычисления рассмотренных комбинаторных чисел.

Пример 1. Полагая $m_i = 1$ для всех $i = 1,\ldots,N$, получаем, что комбинаторное число $S\left(m_1, m_2, \ldots, m_N; n\right)$ определяет число всех неупорядоченных n-элементных разбиений N-элементного множества. Поэтому при $n = N$ получаем, что $S\left(1, 1, \ldots, 1; n\right) = B_n$, где B_n — число Белла.

Как частный случай формулы (5), получаем

$$S\left(1, 1, \ldots, 1; m\right) = S\left(1, 1, \ldots, 1; n\right) \text{ для всех } m \geqslant n, n \in \mathbb{N}.$$

В данном случае элементы матрицы (4) лежат в булеане $\{0,1\}$ и легко вычисляются, так как рекурсивная функция $f_N(i_1,i_2,i_3,\ldots,i_N)$ допускает представление в явной форме

$$f_N(i_1,i_2,i_3,\ldots,i_N)=1+2i_1+i_2+\sum_{k=3}^{N}2^{k-1}i_k,\ i_s\in\{0,1\},\ s=1,\ldots,N.$$

Например, для случая $N=3$ матрица (4) имеет вид

$$A=\begin{bmatrix}0&0&1&1&0&0&1&1\\0&1&0&1&0&1&0&1\\0&0&0&0&1&1&1&1\\1&1&1&1&1&1&1&1\end{bmatrix}.$$

Следовательно, коэффициент при $t_1t_2t_3t_4^n$ в разложении в степенной ряд производящей функции

$$\frac{(1-t_3t_4)^{-1}(1-t_2t_3t_4)^{-1}(1-t_1t_3t_4)^{-1}(1-t_1t_2t_3t_4)^{-1}}{(1-t_4)(1-t_2t_4)(1-t_1t_4)(1-t_1t_2t_4)}$$

определяет комбинаторное число

$$S(1,1,1;n)=\begin{cases}1,\ \text{если } n=1,\\2,\ \text{если } n=2,\\5,\ \text{если } n\geqslant 3.\end{cases}$$

Пример 2. Предположим, что имеется m_1 предметов вида a_1 и m_2 предметов вида a_2. Рассмотрим задачу определения числа способов разложения этих предметов по n одинаковым коробкам без каких-либо ограничений.

Таким образом, необходимо найти число способов разложения мультимножества $X=\{m_1a_1,m_2a_2\}$ в неупорядоченную сумму n его подмультимножеств $X=\bigcup\limits_{k=1}^{n}\{X_k\}$.

В этом случае формулы (4) (при $N=2$) приводят к следующему результату:

$$A=(a_{ij})_{3\times(m_1+1)(m_2+1)},\ b=(b_i)_3,$$

где

$$\forall p\in\{0,\ldots,m_1\}(a_{1k}=p)\ \text{при}\ p(m_2+1)+1\leqslant k\leqslant(p+1)(m_2+1);$$
$$\forall p\in\{0,\ldots,m_2\}(a_{2k}=p)\ \text{при}\ k=i(m_2+1)+p+1, i=0,\ldots,m_1;$$
$$a_{3k}=1, k=1,\ldots,(m_1+1)(m_2+1);\ b_1=m_1, b_2=m_2, b_3=n.$$

Например, если $m_1=1$ и $m_2=3$, то матрица A имеет вид

$$A=\begin{bmatrix}0&0&0&0&1&1&1&1\\0&1&2&3&0&1&2&3\\1&1&1&1&1&1&1&1\end{bmatrix}$$

и, следовательно, комбинаторное число $S(1,3;n)$ определяется как коэффициент при $t_1t_2^3t_3^n$ в разложении в степенной ряд производящей функции

$$\frac{(1-t_1t_3)^{-1}(1-t_1t_2t_3)^{-1}(1-t_1t_2^2t_3)^{-1}(1-t_1t_2^3t_3)^{-1}}{(1-t_3)(1-t_2t_3)(1-t_2^2t_3)(1-t_2^3t_3)}.$$

Элементарные подсчёты показывают, что

$$S(1,3;n)=\begin{cases}1, \text{ если } n=1,\\ 4, \text{ если } n=2,\\ 6, \text{ если } n=3,\\ 7, \text{ если } n\geqslant 4.\end{cases}$$

2. Формализация комбинаторных чисел для подсчёта упорядоченных разбиений конечных мультимножеств

Обозначим $\widetilde{S}(m_1,m_2,\ldots,m_N;n)$ число всех упорядоченных n-элементных разбиений мультимножества $X=\{m_1a_1,m_2a_2,\ldots,m_Na_N\}$, предполагая, что допускаются повторения подмультимножеств, а также пустые подмультимножества. Зафиксируем произвольное упорядоченное разбиение $\{X_i : i=1,\ldots,n\}$ мультимножества X. Обозначим через $[k_{1i},k_{2i},\ldots,k_{Ni}]$, $0\leqslant k_{si}\leqslant m_s$, $s=1,\ldots,N$, вектор первичной спецификации подмультимножества X_i.

Тогда каждое упорядоченное разбиение мультимножества X однозначно задаётся следующей матрицей:

$$\begin{bmatrix} k_{11} & k_{12} & \ldots & k_{1n}\\ k_{21} & k_{22} & \ldots & k_{2n}\\ \ldots & \ldots & \ldots & \ldots\\ k_{N1} & k_{N2} & \ldots & k_{Nn}\end{bmatrix}.$$

При этом из равенства $\bigcup\limits_{i=1}^{n}X_i=X$ следует, что элементы данной матрицы должны быть решениями в целых неотрицательных числах системы линейных уравнений

$$\sum_{j=1}^{n}k_{ij}=m_i,\ i=1,\ldots,N. \tag{6}$$

Переименуем неизвестные k_{ij} одним индексом с помощью отображения $(i,j)\to$ $\to(i-1)\,n+j$ [5] и запишем систему (6) в стандартной форме

$$\sum_{j=1}^{Nn}a_{ij}x_j=m_i,\ i=1,\ldots,N,$$

где

$$a_{ij}=\begin{cases}1, \text{ если } (i-1)\,n+1\leqslant j\leqslant in,\\ 0 \text{ в остальных случаях,}\end{cases}\quad i=1,\ldots,N,\ j=1,\ldots,Nn. \tag{7}$$

Таким образом, из соотношений (2)–(3), (7) следует, что комбинаторное число $\widetilde{S}(m_1,m_2,\ldots,m_N;n)$ равно коэффициенту при $t_1^{m_1}t_2^{m_2}\ldots t_N^{m_N}$ в разложении в степенной ряд производящей функции

$$\left((1-t_1)^n(1-t_2)^n\ldots(1-t_N)^n\right)^{-1}.$$

Наконец, используя разложение

$$\frac{1}{(1-t)^r}=\sum_{k=0}^{\infty}(-1)^k\begin{pmatrix}-r\\ k\end{pmatrix}t^k,$$

получим

$$\widetilde{S}(m_1,m_2,\ldots,m_N;n)=(-1)^{m_1+m_2+\ldots+m_N}\begin{pmatrix}-n\\ m_1\end{pmatrix}\begin{pmatrix}-n\\ m_2\end{pmatrix}\ldots\begin{pmatrix}-n\\ m_N\end{pmatrix}.$$

Замечание. Отметим, что формула для подсчёта упорядоченных разбиений конечных мультимножеств может быть получена без применения формул (2)–(3). Действительно, окончательно задача сводится к определению числа всех целых неотрицательных решений системы линейных диофантовых уравнений (6). Несложно проверить, что данное число равно коэффициенту при $t_1^{m_1} t_2^{m_2} \dots t_N^{m_N}$ в стандартном разложении по возрастающим степеням $t_1^{a_1} t_2^{a_2} \dots t_N^{a_N}$ производящей функции

$$\left(\frac{1}{1-t_1}\right)^n \left(\frac{1}{1-t_2}\right)^n \dots \left(\frac{1}{1-t_N}\right)^n.$$

Отсюда, воспользовавшись тождеством

$$\left(\frac{1}{1-t}\right)^n = \sum_{k=0}^{\infty} \left(\begin{array}{c} n+k-1 \\ k \end{array} \right) t^k,$$

получаем

$$\widetilde{S}\,(m_1, m_2, \dots, m_N; n) = \left(\begin{array}{c} n+m_1-1 \\ m_1 \end{array} \right) \left(\begin{array}{c} n+m_2-1 \\ m_2 \end{array} \right) \dots \left(\begin{array}{c} n+m_N-1 \\ m_N \end{array} \right).$$

Заключение

В данной работе задача о подсчёте числа различных разбиений конечного мультимножества сведена к определению числа всех целых неотрицательных решений соответствующей системы линейных диофантовых уравнений с многоиндексной нумерацией неизвестных. Полученные соотношения (4) позволяют представить данную систему в стандартной форме (2), для которой известен явный вид производящей функции (3). Это позволило получить производящие функции для рассматриваемых комбинаторных чисел и исследовать некоторые их свойства. При этом для числа всех разбиений произвольного конечного мультимножества в упорядоченную сумму его подмультимножеств получены явные формулы.

Отметим, что для практики, безусловно, важны не только формулы (или алгоритмы) вычисления тех или иных комбинаторных чисел, но и оценки и асимптотики этих чисел [6]. Однако вопросы, связанные с эффективностью реализации вычислений с помощью полученных соотношений, требуют дальнейшего исследования.

ЛИТЕРАТУРА

1. *Стенли Р.* Перечислительная комбинаторика. М.: Мир, 1990. 400 с.
2. *Новиков Ф. А.* Дискретная математика для программистов. СПб.: Питер, 2009. 384 с.
3. *Виленкин Н. Я.* Комбинаторика. М.: Наука, 1969. 323 с.
4. *Заторский Р. А.* Подсчет m-подмультимножеств через их вторичные спецификации / под ред. К. А. Рыбникова // Комбинаторный анализ. М.: МГУ, 1986. Вып. 7. С. 136–145.
5. *Гоцуленко В. В.* Формула для числа сочетаний с повторениями при ограничениях и её применение // Прикладная дискретная математика. 2013. № 2(20). С. 71–77.
6. *Яблонский С. В.* Введение в дискретную математику. М.: Наука, 1986. 384 с.

2013 Вычислительные методы в дискретной математике №4(22)

DOI 10.17223/20710410/22/8

УДК 519.7

ЗАДАЧА КОММИВОЯЖЁРА: УЛУЧШЕННАЯ НИЖНЯЯ ГРАНИЦА В МЕТОДЕ ВЕТВЕЙ И ГРАНИЦ

Ю. Л. Костюк

Национальный исследовательский Томский государственный университет, г. Томск, Россия

E-mail: kostyuk_y_l@sibmail.com

Рассматривается решение задачи коммивояжёра методом ветвей и границ. Предлагается способ дополнительного (по сравнению с алгоритмом Литтла) уточнения нижней границы, особенно эффективный для случая симметричной матрицы, и на его основе строится новый алгоритм. С помощью вычислительного эксперимента получены оценки констант в формулах трудоёмкости для трёх модификаций алгоритма Литтла на трёх видах случайных матриц расстояний: 1) несимметричных матрицах со случайными расстояниями; 2) матрицах с евклидовыми расстояниями между случайными точками внутри квадрата; 3) несимметричных матрицах со случайными расстояниями, удовлетворяющими неравенству треугольника.

Ключевые слова: *задача коммивояжёра, метод ветвей и границ, вычислительный эксперимент.*

Введение

В известном алгоритме Литтла [1], реализующем метод ветвей и границ для задачи коммивояжёра (ЗК), на каждом шаге обработки матрицы расстояний из каждой строки и столбца вычитается величина наименьшего в строке или столбце элемента, в результате чего в каждой строке и каждом столбце появляется не менее чем по одному нулю. Сумма величин всех вычтенных элементов даёт нижнюю границу для оптимального решения, на основе которой производится отсечение бесперспективных вариантов решения.

В [2] описано дополнительное преобразование матрицы, являющееся частным случаем преобразования в венгерском алгоритме решения задачи о назначениях [3]. В результате в матрице появляются дополнительные нули и увеличивается нижняя граница оптимального маршрута, что улучшает процесс отсечения. Как показал вычислительный эксперимент [2], алгоритм с таким преобразованием работает гораздо быстрее в случае несимметричной матрицы, однако для симметричной матрицы это не так.

1. Вычисление улучшенной нижней границы оптимального решения задачи коммивояжера

При решении ЗК и задачи о назначениях применяется одно и то же преобразование матрицы. Преобразование основано на том факте, что если из любого столбца или строки матрицы вычесть (или прибавить) константу, то стоимость оптимального маршрута коммивояжёра и стоимость решения задачи о назначениях уменьшается (увеличивается) на величину этой константы, а само решение остается прежним.

В алгоритме Литтла [1] из каждой строки и столбца выполняется вычитание наименьших в строках и столбцах элементов с целью получения в каждой строке и каждом столбце не менее чем по одному нулю. В модифицированном алгоритме [2] после

этого находятся такие группы строк (столбцов), в которых имеется по одному нулю в одном и том же столбце (строке), и для каждой такой группы вычитается минимальный ненулевой элемент, а к столбцу (строке), где находятся нули, добавляется такое же значение. В результате увеличивается количество нулей в матрице и растёт нижняя граница стоимости оптимального маршрута. Трудоёмкость такого преобразования остаётся в пределах $O(n^2)$, как и для преобразования в алгоритме Литтла, но с увеличенным множителем.

В венгерском алгоритме [3] похожий процесс получения дополнительных нулей в матрице продолжается ещё дальше, до тех пор, пока из полученных нулей не удаётся построить подмножество независимых нулей, когда в каждой строке и каждом столбце имеется ровно по одному независимому нулю. В результате нижняя граница оптимального маршрута может дополнительно увеличиться. Однако трудоёмкость этого действия имеет порядок $O(n^3)$.

Все эти преобразования матрицы позволяют в лучшем случае получить нижнюю границу стоимости оптимального маршрута коммивояжёра, равную стоимости решения задачи о назначениях. Рассмотрим дальнейшее уточнение нижней границы стоимости оптимального маршрута коммивояжёра.

Рассмотрим ориентированный невзвешенный граф, в котором вершины те же, что и в исходной ЗК, а рёбра соответствуют нулевым элементам преобразованной матрицы. В этом графе просмотром вглубь выделим сильносвязные компоненты [4], трудоёмкость такого действия — $O(n^2)$. Для сильносвязных компонент справедливо следующее свойство: из любой вершины можно перейти в любую другую вершину внутри той же самой компоненты по рёбрам с нулевым весом. Если нулевые элементы в матрице получены венгерским алгоритмом, то каждая из компонент будет состоять не менее чем из двух вершин. После преобразования матрицы модифицированным алгоритмом большая часть компонент будет содержать по две или больше вершин, но в отдельных компонентах может содержаться одна вершина. В любом случае цикл по всем вершинам исходной матрицы, т. е. маршрут коммивояжёра, будет, кроме нулевых рёбер, содержать также рёбра перехода между компонентами, часть этих рёбер перехода будут нулевыми, а часть — ненулевыми.

Чтобы получить более точную нижнюю границу маршрута коммивояжёра, к ранее вычисленной границе необходимо добавить нижнюю границу суммарной длины рёбер перехода между компонентами, чтобы получился единый маршрут. Для этого каждую из выделенных сильносвязных компонент будем считать супервершиной, т. е. вершиной другого графа — графа компонент. Для каждой пары (i, j) супервершин вычислим ориентированное расстояние как длину минимального ребра между парами вершин (p, q) матрицы ЗК, где вершина p принадлежит i-й компоненте, а вершина q — j-й компоненте. Трудоёмкость такого действия не превышает $O(n^2)$ при любом количестве компонент. Однако рёбра полученной матрицы расстояний графа компонент между супервершинами ещё не являются кратчайшими расстояниями перехода из одной компоненты в другую, так как в оптимальном маршруте вершины исходной матрицы, принадлежащие одной и той же компоненте, могут идти не подряд, а перемежаться с вершинами из других компонент. Поэтому надо из матрицы расстояний графа компонент получить матрицу кратчайших расстояний. Это можно сделать либо алгоритмом Флойда, либо k-кратным применением алгоритма Дейкстры [4], где k — количество компонент. В обоих случаях трудоёмкость такого действия — $O(k^3)$.

После этого полученную матрицу кратчайших расстояний подвергнем такому же преобразованию, как исходную матрицу, тогда сумма вычтенных элементов даст до-

полнительную положительную поправку к нижней границе стоимости оптимального маршрута коммивояжёра.

На втором этапе выполним такие же действия с преобразованной матрицей кратчайших расстояний между супервершинами. Для этого выделим сильносвязные компоненты супервершин, построим по ним матрицу расстояний для суперсупервершин и вычислим на ней кратчайшие расстояния. После преобразования последней матрицы получим ещё одну дополнительную поправку к нижней границе.

Аналогичные действия будем продолжать на третьем, четвертом и т. д. этапах до тех пор, пока очередная дополнительная поправка к нижней границе не станет равной нулю.

Оценим трудоёмкость всего этого процесса. На каждом этапе уточнения нижней границы при обработке матрицы размером n выполняются следующие действия:

1) получение нулей в каждой строке и каждом столбце матрицы: трудоёмкость $\mathrm{O}(n^2)$;
2) выделение сильно связных компонент: трудоёмкость $\mathrm{O}(n^2)$;
3) построение матрицы расстояний супервершин: трудоёмкость $\mathrm{O}(n^2)$;
4) вычисление матрицы кратчайших расстояний для супервершин: трудоёмкость $\mathrm{O}(k^3)$, где k — количество компонент.

Если каждая компонента связности содержит не менее двух вершин, то $k \leqslant n/2$, поэтому в худшем случае общая трудоёмкость вычисления улучшенной нижней границы на всех этапах не превысит суммы

$$T(n) \leqslant a\left(n^2 + (n/2)^2 + (n/4)^2 + \ldots\right) + b\left((n/2)^3 + (n/4)^3 + \ldots\right) \leqslant \frac{4}{3}an^2 + \frac{1}{7}bn^3, \quad (1)$$

где a и b — константы.

В алгоритме, реализующем метод ветвей и границ для ЗК, описанный процесс вычисления нижней границы производится на каждом шаге, как только выбрано очередное ребро для маршрута, т. е. нижняя граница вычисляется с учётом того, что часть рёбер уже включена в маршрут.

2. Дополнительные усовершенствования в реализации алгоритмов

В [2] подробно изложена реализация алгоритма Литтла и модифицированного алгоритма. В данной работе рассмотрим некоторые дополнительные усовершенствования в реализации обоих этих алгоритмов при поиске вглубь на дереве решений. Эти усовершенствования применимы также в случае вычисления на каждом этапе алгоритма улучшенной нижней границы, так, как это описано выше. Предлагаются следующие усовершенствования.

1. В обоих алгоритмах на очередном шаге, после преобразования матрицы, выбирается нулевой элемент M_{ij} в матрице для включения соответствующего ребра в маршрут. Критерий выбора: максимальное значение второго элемента в строке или столбце. После этого из обработанной создается новая матрица, в которой удалена i-я строка и j-й столбец, а в старой матрице элемент M_{ij} заменяется на ∞. Дополнительным улучшением здесь является то, что попутно для обеих матриц оценивается изменение нижней границы всех вариантов маршрута, которые можно построить по этим матрицам.

2. Когда при построении варианта маршрута на очередном шаге матрица уже имеет размер 4×4, то после выбора еще одного ребра новая матрица будет иметь размер

3×3. Для такой матрицы возможно лишь два варианта замыкания полного маршрута. В этом случае можно легко оценить, какой из этих вариантов лучше и имеет ли он меньшую длину, чем ранее найденный маршрут, и для этого не требуется делать полного преобразования новой матрицы.

3. Длина списка всех узлов дерева решений, т. е. матриц, на любом промежуточном этапе просмотра вглубь равна $n - 2$, где n — размер исходной матрицы, так как обработка заканчивается на матрице размером 3×3. При этом размер матрицы в первом узле — n, во втором — $(n - 1)$, в третьем — $(n - 2)$ и т. д. Поэтому целесообразно выделить память для такого списка из $(n - 2)$-х узлов в самом начале, а в процессе выполнения алгоритма копировать новые матрицы в уже существующие узлы.

4. Если первоначальная матрица была симметричной, то для оптимального маршрута коммивояжёра всегда существует равный ему по длине маршрут в обратном направлении. Поэтому, когда для такой матрицы выбирается самое первое ребро M_{ij} для включения в маршрут, то в старой матрице необходимо заменить на ∞ не только элемент M_{ij}, но и элемент M_{ji}, чтобы сразу предотвратить попытку формирования маршрута в противоположном направлении. При этом надо учесть, что при выборе второго и последующих рёбер матрица перестаёт быть симметричной.

Как показал вычислительный эксперимент, в совокупности все эти усовершенствования позволяют уменьшить время выполнения алгоритмов на 20–30 %, а для симметричной матрицы — до 60 %.

3. Реализация алгоритмов

Все три алгоритма (алгоритм Литтла, модифицированный алгоритм, а также модифицированный алгоритм с вычислением улучшенной нижней границы) реализованы со всеми рассмотренными выше усовершенствованиями. В алгоритмах используются следующие глобальные переменные:

- стоимость $S_{\min}$ наилучшего на текущий момент построенного маршрута;
- размер n_0 исходной матрицы расстояний;
- массив P_{opt} размером n_0, в котором записан наилучший на текущий момент построенный маршрут.
- массив P_t размером n_0, в котором записан частично построенный маршрут: $P_t[i] = j$ для каждого входящего в маршрут ребра (i, j).

Каждый узел дерева решений представляется экземпляром класса со следующей структурой:

- указатели на соседние узлы, на левый и правый узел;
- размер матрицы в узле n;
- первая нижняя граница S_0, равная сумме вычтенных из строк и столбцов матрицы величин;
- вторая нижняя граница S_m, более точная, чем S_0 ($S_m \geqslant S_0$);
- матрица расстояний в узле M, её размер $n \times n$;
- признак симметричности матрицы;
- два массива размером n, в которых записаны начальные и конечные вершины отрезков частично построенного маршрута;
- четыре массива размером n, в которых записаны номера нулевых элементов и вторых наименьших элементов в строках и столбцах.

В алгоритме узлы дерева решений сцепляются в линейный список, представляющий собой обход частично построенного дерева решений по висячим узлам слева

направо. Узлы в списке обрабатываются, начиная с корневого узла. После обработки текущего узла производится переход к левому или правому узлу в списке.

Далее приведён общий вид модифицированного алгоритма с вычислением улучшенной нижней границы. Алгоритм реализует поиск вглубь на дереве решений.

1. Выделение памяти для начального элемента списка — корневого узла дерева решений — размером $n = n_0$. Присваивание начальных значений для матрицы в корневом узле, вспомогательным массивам и глобальным переменным.
2. Проверка матрицы на симметричность и задание значения признаку симметричности.
3. Выделение памяти для списка узлов дерева решений с присоединением узлов слева от корневого узла и заданием другим узлам размеров соответственно: $n-1$, $n-2$, $\ldots$, 3.
4. Цикл, пока список узлов дерева решений не просмотрен до конца:

4.1. Если $S < S_{\min}$, то для текущего узла в списке:

4.1.1. Вычитание из строк и столбцов матрицы минимальных значений с коррекцией S_0 и S_m.

4.1.2. Если $S < S_{\min}$, то для текущего узла в списке:

4.1.2.1. Дополнительное преобразование матрицы с коррекцией S_0 и S_m.

4.1.2.2. Если $S < S_{\min}$, то для текущего узла в списке:

4.1.2.2.1. Если $n > 4$, то вычисление улучшенной нижней границы с коррекцией S_m.

4.1.2.2.2. Если $S < S_{\min}$, то для текущего узла в списке:

4.1.2.2.2.1. Выбор наилучшего ребра для включения в маршрут.

4.1.2.2.2.2. Формирование матрицы размером $n-1$ в левом узле списка.

4.1.2.2.2.3. Коррекция матрицы в старом узле с учетом симметричности.

4.1.2.2.2.4. Если $n = 4$, то:

4.1.2.2.2.4.1. Выбор лучшего варианта замыкания маршрута в новой матрице, и если его длина меньше $S_{\min}$, то запоминание нового маршрута.

4.1.2.2.2.4. Иначе текущий узел слева.

4.1.2.2.2. Иначе текущий узел справа.

4.1.2.2. Иначе текущий узел справа.

4.1.2. Иначе текущий узел справа.

4.1. Иначе текущий узел справа.

5. Уничтожение списка узлов дерева решений.

4. Вычислительный эксперимент

В вычислительном эксперименте на трёх видах модельных данных проверялась сравнительная трудоёмкость следующих трех алгоритмов:

1) алгоритма Литтла с усовершенствованиями;

2) модифицированного алгоритма с усовершенствованиями;

3) модифицированного алгоритма с усовершенствованиями и с вычислением улучшенной нижней границы.

Все алгоритмы написаны на языке Паскаль в системе Delphi, вычисления производились на компьютере с процессором Atlon, работающем на частоте 2,9 ГГц.

В табл. 1–3 представлены среднее количество обрабатываемых узлов дерева решений и среднее время вычисления маршрута, полученные по одним и тем же сгенерированным экземплярам матриц для всех трёх алгоритмов для следующих трёх видов распределений элементов матриц:

1) случайная матрица, элементы в которой — равномерно распределённые независимые случайные целые числа из диапазона от 0 до 1000, матрица несимметричная;
2) матрица евклидовых расстояний для n равномерно распределённых независимых случайных точек на плоскости с координатами из диапазона от 0 до 1000; вычисляются расстояния между ними, матрица симметричная;
3) случайная матрица с неравенством треугольника, на которой вычислены кратчайшие расстояния алгоритмом Флойда, матрица несимметричная.

Для каждого размера матриц n генерировалось от 4000 (для минимальных n) до 400 (для максимальных n) вариантов матриц. В таблицах отсутствуют те результаты вычислений, для получения которых потребовалось бы слишком много времени.

Из табл. 1 видно, что для случайных матриц модифицированный алгоритм имеет существенное преимущество над алгоритмом Литтла, при $n = 70$ его время работы в 100 раз меньше. Но вычисление в нём улучшенной нижней границы не даёт дополнительного уменьшения времени. Это происходит из-за того, что у таких матриц часто получается всего одна сильносвязная компонента и нижняя граница не улучшается.

Таблица 1

Результаты эксперимента для случайных матриц

Размер матрицы n	Алгоритм 1		Алгоритм 2		Алгоритм 3	
	Среднее кол-во узлов	Среднее время, с	Среднее кол-во узлов	Среднее время, с	Среднее кол-во узлов	Среднее время, с
30	926	0,0078	152	0,0023	143	0,0026
40	5610	0,079	386	0,0100	361	0,0112
50	29224	0,619	977	0,0351	904	0,0376
60	156106	4,638	2252	0,113	2093	0,119
70	966825	38,61	5573	0,374	5189	0,398
80	–	–	15056	1,326	14176	1,360
90	–	–	27276	3,06	25300	3,11
100	–	–	85623	12,06	79236	12,19
110	–	–	181540	31,47	165508	31,09

Из табл. 2 видно, что для матриц евклидовых расстояний модифицированный алгоритм уступает алгоритму Литтла, его время работы больше в 2–4 раза. На таких матрицах вычисление улучшенной нижней границы даёт наибольший эффект: при $n = 30$ время работы модифицированного алгоритма с вычислением улучшенной нижней границы в 15 раз меньше, чем у алгоритма Литтла. Здесь количество сильносвязных компонент нередко оказывается близким к $n/2$, в результате уточнённая нижняя граница становится заметно больше, чем в алгоритме Литтла, что приводит к улучшению процесса отсечения. Следует также отметить, что матрицы евклидовых расстояний оказались самыми трудными для всех рассматриваемых алгоритмов.

Из табл. 3 видно, что для случайных матриц с неравенством треугольника модифицированный алгоритм имеет существенное преимущество над алгоритмом Литтла, при $n = 45$ его время работы в 20 раз меньше. Но вычисление в нём улучшенной нижней границы хотя и приводит к некоторому уменьшению количества обрабатываемых узлов, но в целом на 10–20 % увеличивает общее время работы.

Следует также заметить, что разброс количества обрабатываемых узлов дерева решений и времени вычисления для матриц одинакового размера и одного и того же вида распределения оказался очень большим, разница нередко достигала многих сотен раз.

Таблица 2

Результаты эксперимента для матриц евклидовых расстояний

Размер матрицы n	Алгоритм 1		Алгоритм 2		Алгоритм 3	
	Среднее кол-во узлов	Среднее время, с	Среднее кол-во узлов	Среднее время, с	Среднее кол-во узлов	Среднее время, с
15	793	0,0017	1008	0,0039	174	0,0023
20	14744	0,0404	25846	0,1220	949	0,0183
25	165027	0,578	407836	2,563	5010	0,133
30	4734542	13,0	3152506	22,8	23401	0,842
35	–	–	–	–	117408	5,51
40	–	–	–	–	888505	71,9

Таблица 3

Результаты эксперимента для случайных матриц с неравенством треугольника

Размер матрицы n	Алгоритм 1		Алгоритм 2		Алгоритм 3	
	Среднее кол-во узлов	Среднее время, с	Среднее кол-во узлов	Среднее время, с	Среднее кол-во узлов	Среднее время, с
25	5043	0,0168	873	0,0042	484	0,0040
30	35809	0,132	2624	0,0134	1924	0,0144
35	351811	1,23	7348	0,0458	5976	0,0510
40	631431	3,03	33528	0,180	28153	0,236
45	4221193	21,3	135320	0,86	131190	1,08
50	–	–	2056272	12,8	1803337	15,0

Это является следствием того, что ЗК относится к NP-трудным задачам. Из-за этого, к сожалению, для произвольной матрицы невозможно гарантировать, что время решения на ней ЗК не превысит заранее рассчитанной величины. Поэтому данные в таблицах следует рассматривать как ориентировочные.

Трудоёмкость по количеству обрабатываемых узлов дерева решений у всех алгоритмов приближённо имеет порядок $O(c^n)$, где параметр c имеет разную величину для различных алгоритмов и распределений. При этом трудоёмкость по времени имеет порядок $O(n^2c^n)$ для первого и второго алгоритмов. Согласно формуле (1), у третьего алгоритма (с вычислением улучшенной нижней границы) для различных распределений трудоёмкость по времени в общем случае имеет порядок $O((an^2 + bn^3)c^n)$, где a, b — постоянные параметры. Однако, как показал вычислительный эксперимент, для рассмотренных распределений при относительно небольших n параметр b во много раз меньше a, и им можно пренебречь. Поэтому для всех алгоритмов и распределений можно считать, что трудоёмкость по времени имеет порядок $O(n^2c^n)$.

Оценим величины параметров в формулах трудоёмкости по методу наименьших квадратов. Количество обрабатываемых узлов U как функция от n:

$$U(n) = a \cdot c^n, \tag{2}$$

где a и c — параметры.

В работе [2] выведены следующие формулы для оценивания параметров c и a:

$$c = \exp\left(\sum_i \ln\frac{U_i}{U_{i-1}}(n_i - n_{i-1}) / \sum_i (n_i - n_{i-1})^2\right); \tag{3}$$

$$a = \exp\left(\frac{1}{K}\sum_{i=1}^{K}(\ln U_i - n_i \ln c)\right). \tag{4}$$

Суммирование в этих формулах ведётся по количеству измерений K, помещённых в табл. 1–3. Вычисленные по формулам (3) и (4) оценки параметров a и c для формулы (2) у трёх рассмотренных алгоритмов для трёх видов распределений приведены в табл. 4.

Таблица 4

Оценки параметров a и c в формуле (2) для количества узлов

Вид распределения	Алгоритм 1		Алгоритм 2		Алгоритм 3	
	Параметр a	Параметр c	Параметр a	Параметр c	Параметр a	Параметр c
1	4,9957	1,1898	11,162	1,0926	10,667	1,0922
2	0,11917	1,7854	0,43497	1,7101	0,93746	1,4071
3	1,3519	1,4000	0,20319	1,3642	0,08029	1,3895

Время вычисления (в секундах) как функция от n:

$$U(n) = b \cdot c^n. \tag{5}$$

Формулы для оценивания параметров c и b, вывод которых аналогичен выводу формул (3) и (4), следующие:

$$c = \exp\left(\left(\sum_i \ln\frac{U_i}{U_{i-1}}(n_i - n_{i-1}) - \sum_i \ln\frac{n_i}{n_{i-1}}(n_i - n_{i-1})\right) / \sum_i (n_i - n_{i-1})^2\right); \tag{6}$$

$$b = \exp\left(\frac{1}{K}\sum_{i=1}^{K}(\ln U_i - n_i \ln c - 2\ln n_i)\right). \tag{7}$$

Вычисленные по формулам (6) и (7) оценки параметров b и c для формулы (5) у трёх рассмотренных алгоритмов для трёх видов распределений приведены в табл. 5.

Таблица 5

Оценки параметров b и c в формуле (5) для времени

Вид распределения	Алгоритм 1		Алгоритм 2		Алгоритм 3	
	Параметр b	Параметр c	Параметр b	Параметр c	Параметр b	Параметр c
1	$5{,}1052 \cdot 10^{-8}$	1,1857	$18{,}646 \cdot 10^{-8}$	1,0904	$22{,}076 \cdot 10^{-8}$	1,0885
2	$0{,}38013 \cdot 10^{-8}$	1,6549	$1{,}5407 \cdot 10^{-8}$	1,6255	$3{,}6702 \cdot 10^{-8}$	1,4113
3	$1{,}7476 \cdot 10^{-8}$	1,3478	$0{,}14256 \cdot 10^{-8}$	1,3391	$0{,}14172 \cdot 10^{-8}$	1,3449

Рассмотренные алгоритмы были также протестированы на известной ЗК с 48 городами [5] с симметричной матрицей. Алгоритм 1 получил результат за 1,5 мин, алгоритм 2 — за 45 мин, а алгоритм 3 — за 11 с. Все они вычислили один и тот же оптимальный маршрут длиной 11461, такой же, как в работе [2]: 1 2 48 15 43 21 33 30 23 9 10 40 36 34 6 8 47 7 38 14 18 12 22 13 28 32 25 3 5 29 26 41 24 35 17 31 20 11 16 42 4 46 45 39 44 27 37 19.

Программы, реализующие алгоритмы 2 и 3, включая модуль с описанием класса для решения ЗК, написаны на Паскале в системе Delphi. Исходные тексты программ в виде архивов доступны в сети интернет по ссылкам [6, 7]. Кроме того, в архив [7] помещен файл с матрицей расстояний для упомянутой ЗК с 48 городами [5].

Заключение

Как показал вычислительный эксперимент, время выполнения рассмотренных алгоритмов существенно различается для матриц расстояний с распределениями разных видов. Модифицированный алгоритм по быстродействию существенно превосходит алгоритм Литтла на случайных матрицах, однако уступает ему на матрицах евклидовых расстояний. Предложенный способ вычисления более точной нижней границы наилучшего решения ЗК на промежуточных шагах работы алгоритма, когда часть рёбер включена в маршрут, позволил создать новый алгоритм, который существенно превосходит алгоритм Литтла, в том числе на матрицах евклидовых расстояний. Применение нового алгоритма позволяет заметно увеличить размер матриц с различными распределениями, для которых можно за относительно небольшое время получить точное решение ЗК.

ЛИТЕРАТУРА

1. *Little J. D. C., Murty K. G., Sweeney D. W., and Karel C.* An algorithm for the Traveling Salesman Problem // Operations Research. 1963. No. 11. P. 972–989.
2. *Костюк Ю. Л.* Эффективная реализация алгоритма решения задачи коммивояжера методом ветвей и границ // Прикладная дискретная математика. 2013. № 2 (20). С. 78–90.
3. *Данциг Дж.* Линейное программирование, его применения и обобщения: пер. с англ. М.: Прогресс, 1966.
4. *Ахо А., Хопкрофт Дж., Ульман Дж.* Построение и анализ вычислительных алгоритмов: пер. с англ. М.: Мир, 1979. 536 с.
5. *Хелд М., Карп Р. М.* Применения динамического программирования к задачам упорядочивания // Киб. сборник, старая серия. М.: Мир, 1964. Вып. 9. С. 202–218.
6. *Костюк Ю. Л.* Модифицированный алгоритм решения задачи коммивояжера методом ветвей и границ. Версия 2. Программа и модуль с описанием класса: язык Паскаль в системе Delphi. 2013, июль. Электронный ресурс: `www.inf.tsu.ru/Decanat/Staff.nsf/people/KostjukJuL`.
7. *Костюк Ю. Л.* Модифицированный алгоритм решения задачи коммивояжера методом ветвей и границ с улучшенной нижней границей. Программа и модуль с описанием класса: язык Паскаль в системе Delphi. 2013, август. Электронный ресурс: `www.inf.tsu.ru/Decanat/Staff.nsf/people/KostjukJuL`.

2013 Вычислительные методы в дискретной математике №4(22)

DOI 10.17223/20710410/22/9

УДК 519.7

О РЕАЛИЗАЦИИ ОСНОВНЫХ ЭТАПОВ БЛОЧНОГО АЛГОРИТМА ВИДЕМАНА — КОППЕРСМИТА ДЛЯ ДВОИЧНЫХ СИСТЕМ ЛИНЕЙНЫХ УРАВНЕНИЙ НА ВЫЧИСЛИТЕЛЯХ КЛАСТЕРНОГО ТИПА

А. С. Рыжов

Лаборатория ТВП, г. Москва, Россия

E-mail: ryzhovalexander@mail.ru

Рассматриваются вопросы реализации наиболее трудоёмких этапов алгоритма Видемана — Копперсмита поиска решений сильно разреженных систем линейных уравнений на современных ЭВМ. Исследуются вопросы эффективной реализации отдельных операций алгоритма. Отдельно рассмотрены проблемы, возникающие при реализации алгоритма на вычислителях кластерного типа.

Ключевые слова: *системы линейных уравнений, алгоритм Видемана — Копперсмита.*

Введение

Необходимость нахождения решений системы однородных линейных уравнений (СОЛУ) с сильно разреженной матрицей возникает в задачах вычислительной алгебры достаточно часто. Не являются исключением и теоретико-числовые задачи, имеющие непосредственное отношение к криптографии. Например, при нахождении дискретного логарифма в поле $\mathrm{GF}(2^n)$ (при простом $p = 2^n - 1$) методом Копперсмита [1] необходимо найти решения системы линейных уравнений над простым полем $\mathbb{Z}_p$. При факторизации чисел методом решета числового поля одним из этапов является нахождение нескольких решений большой разреженной системы линейных уравнений над полем $\mathrm{GF}(2)$.

В настоящий момент для решения данной задачи обычно используется один из двух алгоритмов: Ланцоша — Монтгомери [2] и Видемана — Копперсмита [3]. Оба алгоритма имеют свои преимущества и недостатки, а также ограничения на используемые вычислители. Алгоритм Ланцоша — Монтгомери реализован, например, в пакете GGNFS. Он ориентирован на выполнение на одном вычислителе кластерного типа с высокоскоростным внутренним соединением узлов. Алгоритм Видемана — Копперсмита, напротив, позволяет наиболее трудоёмкие этапы выполнять на нескольких не связанных друг с другом кластерах. Именно этот алгоритм был выбран авторами работы [4] для реализации линейного этапа при факторизации 768-битного числа.

При реализации математических алгоритмов на современных вычислительных системах возникает ряд вопросов, связанных со способом представления данных, выполнением отдельных шагов и координацией работы узлов вычислителя. Знание и грамотное использование особенностей вычислителя, на котором предполагается выполнение алгоритма, помогают существенно снизить время его работы по сравнению с «лобовой» реализацией. В то же время в работах, содержащих результаты экспериментальных исследований математических алгоритмов, эти вопросы, как правило, не освещаются. Например, в [4] приводится время выполнения отдельных этапов факторизации 768-битного числа и характеристики используемых вычислительных кла-

стеров, однако не затрагиваются вопросы, связанные с реализацией этих этапов на указанных вычислителях.

В данной работе рассматриваются вопросы реализации наиболее трудоёмких этапов алгоритма Видемана — Копперсмита для двоичных систем линейных уравнений на современных вычислителях. В п. 1 излагается сам алгоритм Видемана — Копперсмита; п. 2 посвящён вопросам реализации ряда операций, выполняющихся на отдельных шагах алгоритма. В п. 3 рассмотрены нюансы выполнения алгоритма Видемана — Копперсмита на вычислителях кластерного типа; п. 4 посвящён вопросам применения алгоритма в рамках факторизации больших целых чисел методом решета числового поля.

Работу можно рассматривать как обзор подходов к программированию отдельных процедур алгоритма Видемана — Копперсмита и реализации его в целом на вычислителях кластерного типа. Автору не удалось найти публикаций, в которых бы детально анализировались вопросы эффективной реализации приводимых методов на современных многопроцессорных системах, и настоящая работа преследует цель восполнить этот пробел.

1. Алгоритм Видемана — Копперсмита

Блочный алгоритм Видемана — Копперсмита является обобщением алгоритма Видемана поиска решения разреженной СОЛУ [5], ориентированным на вычислительную технику, работающую со словами в несколько байтов. Он подробно описан в [3, 6]. Приведём краткое описание основных шагов алгоритма.

Пусть A — матрица размера $M \times N$ над конечным полем P и необходимо найти ненулевое решение СОЛУ

$$Ax = 0. \tag{1}$$

Будем считать, что $P = \mathrm{GF}(2)$, параметры алгоритма $m, n \in \mathbb{N}$ $(m \geqslant n)$ кратны длине машинного слова в битах: $m = 64m'$, $n = 64n'$, и что выполнено неравенство $N \geqslant M + m$.

Выберем случайную матрицу $Z \in P_{m,M}$ размера $m \times M$ над полем P. Пусть $B \in P_{M,M}$ — матрица из первых M столбцов матрицы A СОЛУ (1), $Y \in P_{M,n}$ — матрица, составленная из столбцов матрицы A с номерами $M{+}1, M{+}2, \ldots, M{+}n$. Выполняем следующие три этапа:

1) Вычислим матрицы $a_i = ZB^iY$, $i = 0, 1, \ldots, L$ (здесь L — параметр метода; см. далее).
2) Найдём такие векторы $f_0, \ldots, f_d \in P^{(n)}$, что $\sum\limits_{j=0}^{d} a_{i+j} f_j = 0 \in P^{(m)}$ для всех $i \geqslant 0$.

   Формальная сумма $\sum\limits_{j=0}^{d} f_j X^j$ называется *векторным аннулирующим многочленом последовательности* $(a_i : i = 0, \ldots)$.
3) Вычислим вектор w, составленный из векторов $\sum\limits_{j=0}^{d-1} B^j Y f_{j+1}$ и f_0 и дополненный нулями до длины N. Можно показать [3, 6], что w с большой вероятностью является решением системы (1).

Для нахождения векторного аннулирующего многочлена Копперсмитом предложен отдельный алгоритм [3, 6], основанный на последовательном вычислении приближений Паде. При этом в результате работы алгоритма может быть получено до n различных линейно независимых аннулирующих многочленов, что в результате при-

водит к получению до n линейно независимых решений системы (1). Из теоретико-вероятностных соображений [3] достаточно иметь $L = \lfloor M/m \rfloor + \lfloor M/n \rfloor + \Delta$ элементов последовательности (a_i), где $\Delta \approx 100$.

Заметим, что на этапах 1 и 3 алгоритма при вычислении матриц $W_i = B^i Y$ и $a_i = ZB^i Y$ вместо вычисления степеней матрицы B достаточно выполнять последовательное умножение матрицы Y на матрицу B: $W_{i+1} = BW_i$. При этом трудоёмкость этого умножения, равно как и вся трудоёмкость этапов 1 и 3, полностью определяется размером матрицы B и её весом, т. е. числом ненулевых элементов. Если ω — среднее число ненулевых элементов в строке матрицы B, то трудоёмкость вычисления очередной матрицы W_{i+1} равна $\mathrm{O}(M\omega)$; соответственно трудоёмкость этапов 1 и 3 равна $\mathrm{O}(M^2\omega)$.

Важно отметить, что на 64-разрядных ЭВМ умножение одного двоичного вектора на разреженную матрицу B выполняется с такой же трудоёмкостью, что и умножение на матрицу B одновременно 64 векторов. За счёт этого в блочном алгоритме Видемана — Копперсмита достигается существенное снижение трудоёмкости, так как сокращается число шагов на 1-м и 3-м этапах по сравнению с исходным алгоритмом Видемана (частный случай при $m = n = 1$). Например, при $m = n = 64$ число шагов L уменьшится в 64 раза! Если $n = 64n'$, то вычисления матриц W_i можно выполнять параллельно на n' несвязанных вычислителях, умножая по 64 столбца матрицы W_i на матрицу B независимо на каждом вычислителе.

Замечание 1. Не стоит забывать о том, что вместо алгоритма Берлекэмпа — Мэсси, используемого на втором этапе при $m = n = 1$, возникает необходимость применения алгоритма Копперсмита для построения векторного аннулирующего многочлена. Трудоёмкость этого алгоритма равна $\mathrm{O}(m^3L^2) = O\left(mM^2\right)$ (при $m \geqslant n$), и при сокращении числа шагов на первом и третьем этапах алгоритма растет трудоёмкость второго этапа. Однако при небольших значениях m и n трудоёмкость второго этапа в разы меньше, чем трудоёмкость этапов 1 и 3. Кроме того, в работе [7] предложена рекурсивная версия алгоритма Копперсмита, имеющая трудоёмкость порядка $\mathrm{O}\left(m^3 L \log^2 L\right)$. При факторизации 768-битного числа [4] для нахождения решений большой разреженной СОЛУ с помощью алгоритма Видемана — Копперсмита были выбраны параметры $m = 16 \cdot 64$, $n = 8 \cdot 64$. Размер матрицы системы был равен $N \approx 193 \cdot 10^6$. Время работы алгоритма на трёх мощных вычислительных кластерах составило 119 сут, причём на выполнение второго этапа с использованием лишь одного из кластеров ушло менее 18 ч.

Замечание 2. Использование параметров m и n, больших 64, позволяет выполнять этапы 1 и 3 на независимых вычислителях и снижает число шагов на этих этапах, за счёт чего снижается общее время выполнения первого и третьего этапов алгоритма. Но трудоёмкость этих этапов не снижается: при $m = n = 128$ число шагов L в 2 раза меньше, чем при $m = n = 64$, но эти шаги необходимо выполнить для каждой из двух частей матрицы Y (хотя их и можно выполнять параллельно на двух независимых вычислителях).

2. Вопросы реализации первого и третьего этапов алгоритма на современных ЭВМ

Итак, на каждом шаге первого этапа алгоритма Видемана — Копперсмита выполняются следующие операции:

— умножение матрицы W_i размера $M \times n$ на сильно разреженную матрицу B размера $M \times M$: $W_{i+1} = BW_i$;

— умножение матрицы W_{i+1} на матрицу Z размера $m \times M$: $a_{i+1} = ZW_{i+1}$.

На третьем этапе выполняется вычисление вектора $v = \sum_{j=0}^{d-1} B^j Y f_{j+1}$, где f_j — векторы длины n (коэффициенты аннулирующего многочлена). Данные вычисления можно также выполнять параллельно на независимых вычислителях, разбивая B^jY на n' блоков по 64 столбца, а каждый вектор f_j — на n' составляющих длины 64. Поскольку на втором этапе может быть получено до n различных векторных аннулирующих многочленов $f_i(x) = \sum_{j=0}^{d_i} f_j^{(i)} x^j$, $i = 1, \ldots, n$, на третьем этапе можно одновременно вычислять n векторов $v_i = \sum_{j=0}^{d_i-1} B^j Y f_{j+1}^{(i)}$. Соответственно на каждом шаге третьего этапа выполняются следующие операции:

— умножение матрицы $W_{t-1} = B^{t-1}Y$ размера $M \times n$ на сильно разреженную матрицу B размера $M \times M$: $W_t = BW_{t-1}$;

— умножение матрицы W_t на набор векторов-коэффициентов $f_{t+1}^{(i)}, i = 1, \ldots, n$, каждый длины n, и побитное сложение с результатами предыдущих шагов, т. е. с суммами $\sum_{j=0}^{t-1} B^j Y f_{j+1}^{(i)}$.

Очевидно, что наиболее трудоёмкой операцией каждого шага первого и третьего этапов алгоритма является умножение на разреженную матрицу B: $W_{i+1} = BW_i$. Тем не менее неэффективная реализация других операций может существенно повлиять на время работы алгоритма, поэтому необходимо оптимизировать каждую операцию, выполняемую на шагах различных этапов алгоритма.

2.1. Умножение вектора на разреженную матрицу

Двоичная разреженная матрица обычно представляется в виде последовательности строк (или столбцов); каждая строка определяется числом ненулевых элементов и их номерами. Например, матрица

$$B = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix}$$

записывается в виде массива $(\mathbf{2}, 1, 3, \mathbf{1}, 2, \mathbf{2}, 1, 2)$ (жирным выделены веса строк, т. е. число ненулевых элементов). При построчном хранении матрицы B (размера $M \times M$) умножение $W_{i+1} = BW_i$, где W_i — матрица $M \times 64$, выполняется следующим образом (здесь и далее приводится пример реализации на языке C++).

Алгоритм 1. Умножение (слева) на разреженную матрицу, представленную построчно в виде списка ненулевых элементов.

Вход: массив `B` (представление разреженной матрицы), массив `W` (матрица W_i, представленная в виде массива 64-разрядных чисел), число M.

Выход: массив `W2` (представление матрицы $W_{i+1} = BW_i$).

```
1 int st, i, len;    //st - номер строки матрицы B
2                    //len - вес очередной строки
3 __int64 tmp;       //результат умножения на очередную строку
4                    //(64-разрядное число)
5 i = 0;             //индекс в массиве B
6 for (st=0; st<M; st++)  //проходим все строки матрицы
```

```
 7 {
 8     len=B[i];           //вес очередной строки
 9     i++;
10     tmp = 0;
11     while (len--)       //проходим все ненулевые элементы
12     {
13         tmp = tmp ^ W[B[i]];    // ^ - побитное сложение
14         i++;
15     }
16     W2[st] = tmp;       //очередной элемент результата
17 }
```

Конец алгоритма.

Хранение матрицы B можно реализовать и в виде набора весов и номеров ненулевых элементов её столбцов. Для приведённой выше матрицы B её представление по столбцам — $(\mathbf{2}, 1, 3, \mathbf{2}, 2, 3, \mathbf{1}, 1)$. В этом случае умножение $W_{i+1} = BW_i$ можно выполнить с помощью следующего алгоритма.

Алгоритм 2. Умножение (слева) на разреженную матрицу, представленную по столбцам в виде списка ненулевых элементов.

Вход: массив `B` (представление разреженной матрицы по столбцам), массив `W` (матрица W_i, представленная в виде массива 64-разрядных чисел), число M.

Выход: массив `W2` (представление матрицы $W_{i+1} = BW_i$).

```
 1 int st,i,len;   //st - номер столбца матрицы B
 2                 //len - вес очередного столбца
 3 __int64 tmp;    //очередной элемент массива W
 4 for (st=0; st<M; st++) W2[st] = 0;     //обнуляем результат
 5 i = 0;          //индекс в массиве B
 6 for (st=0; st<M; st++)  //проходим все столбцы матрицы
 7 {
 8     len=B[i];           //вес очередного столбца
 9     tmp = W[st];        //следующий элемент массива W
10     i++;
11     while (len--)       //проходим все ненулевые элементы
12     {
13         W2[B[i]] = W2[B[i]] ^ tmp;  //побитное сложение
14         i++;
15     }
16     W2[str] = tmp;      //очередной элемент результата
17 }
```

Конец алгоритма.

Если ω — среднее число ненулевых элементов в строке (столбце) матрицы B, то трудоёмкость обоих алгоритмов равна $\mathrm{O}(M\omega)$. Проведены исследования по сравнению времени работы алгоритмов 1 и 2 для различных размеров входных данных. Результаты экспериментов приведены в табл. 1 (время работы указано в миллисекундах).

Из табл. 1 видно, что алгоритм 1 выигрывает по скорости работы. Это объясняется тем, что внутри основного цикла алгоритма 2 выполняется долгая (по сравнению с остальными) операция записи в память (прибавление элемента массива `W` к соответствующему элементу результата). В алгоритме 1 промежуточное значение очередного элемента выходного массива `W2` хранится в переменной `tmp`, и если в машинном коде

Таблица 1
Время работы алгоритмов 1 и 2

M	ω	Алгоритм 1	Алгоритм 2
100 000	100	24	29
200 000	100	51	62
500 000	100	137	164
1 000 000	100	484	648
2 000 000	100	1445	2269
5 000 000	100	4537	6949
10 000 000	100	11677	17135
200 000	500	251	303
500 000	200	272	321
1 000 000	100	484	648
2 000 000	50	712	1050

эта переменная реализована с помощью одного из регистров процессора, то запись в неё происходит во много раз быстрее, чем запись в ячейку оперативной памяти.

Следует также отметить разницу во времени работы алгоритмов в четырёх последних экспериментах. Несмотря на то, что теоретическая трудоёмкость алгоритмов $O(M\omega)$ для выбранных параметров одинакова, умножение на более плотные матрицы работает быстрее за счёт того, что реже выполняется операция записи в память. Этим также объясняется нелинейный рост времени работы алгоритмов относительно размера матрицы при одинаковом среднем числе ненулевых элементов в строке (столбце).

Алгоритм 1 допускает эффективное распараллеливание по строкам матрицы B: умножение на строки с номерами $n_1+1, n_1+2, \ldots, n_1+k$ можно выполнять независимо от умножения на другие строки; при этом будут получены k элементов выходного массива `W2` с номерами $n_1+1, n_1+2, \ldots, n_1+k$. Тем не менее из-за особенностей работы потоков с общей оперативной памятью не рекомендуется распараллеливать данный алгоритм на большое число потоков.

2.2. Умножение двух векторов

В исходном алгоритме Видемана (т. е. при $m = n = 1$) на каждом шаге первого этапа после умножения вектора на матрицу вычисляется скалярное произведение двух векторов длины M: $a_i = ZW_i$. Трудоёмкость вычисления скалярного произведения равна $2M$ операций в поле GF(2). В общем случае в алгоритме Видемана — Копперсмита требуется перемножить две матрицы размеров $m \times M$ и $M \times n$. При фиксированных m и n это также можно сделать за $O(M)$ операций, используя тривиальный алгоритм. Однако с увеличением параметров m и n трудоёмкость тривиального алгоритма быстро растет, поэтому для ускорения данной процедуры целесообразно использовать более быстрые алгоритмы, например LUT-алгоритм на основе таблиц поиска (Look-Up Tables). Описание LUT-алгоритма можно найти, например, в [8, гл. 6].

Идея метода заключается в следующем. Пусть, для начала, $m = n = 8$, а матрицы Z и W размеров $8 \times M$ и $M \times 8$ представлены в виде массивов байтов `Z` и `W` длины M. Результат умножения $a = ZW$ есть матрица 8×8, представляемая массивом из 8 байтов. Каждая строка результата есть линейная комбинация строк второго множителя: $a_i = \sum\limits_j z_{ij} W_j$. Выделение отдельных элементов z_{ij}, т. е. отдельных битов элементов массива `Z`, является достаточно долгой операцией. Вместо этого удобнее сохранять промежуточный результат вычисления линейных комбинаций $\sum\limits_j z_{ij} W_j$ сра-

зу для всех $i = 1, \ldots, 8$, а именно: в дополнительный массив LUT из $2^8 = 256$ байтов в ячейку с номером $k \in \{0, \ldots, 255\}$ запишем сумму строк второго множителя с такими номерами $j \in \{1, \ldots, M\}$, для которых значение столбца первого множителя (т. е. элемента массива Z) равно k:

$$\mathtt{LUT}_k = \sum_{\substack{j \in \{1,\ldots,M\}: \\ Z_j = k}} W_j.$$

Создание массива LUT выполняется за один проход по массивам Z и W: для $j = 1, \ldots, M$ выполняем побитное сложение $\mathtt{LUT}_{Z_j} := \mathtt{LUT}_{Z_j} \oplus W_j$. Получение результата умножения из массива LUT тривиально. Например, первая строка результата есть линейная комбинация элементов массива LUT, первый (младший) бит номеров которых равен 1.

Для случая $m = n = 64$ невозможно выделить дополнительную память LUT объёмом 2^{64} 64-разрядных элементов. Вместо этого будем использовать 8 различных таблиц LUT, разбив вычисления по номерам байтов в строке результата. При больших значениях m и n удобнее решать задачу, разбивая на подзадачи с $m = n = 64$.

Итоговый алгоритм на языке C++ можно записать следующим образом.

Алгоритм 3. Умножение матриц размеров $64 \times M$ и $M \times 64$: $a = ZW$.

Вход: массивы Z и W 64-разрядных чисел, представляющие матрицы Z и W соответственно, число M.

Выход: массив A из 64-х 64-разрядных чисел, представляющий результат (матрицу a размера 64×64).

```
 1 __int64 LUT[256 * 8];
 2 __int64 zj, wj;
 3 int i, j, k;
 4 for (j = 0; j < 256*8; j++) LUT[j] = 0;
 5 for (j = 0; j < M; j++)
 6 {
 7     zj = Z[j];
 8     wj = W[j];
 9     for (i=0; i<8; i++)
10         LUT[i*256 + (zj >> 8*i) & 0xFF] ^= wj;
11     //при m=n=8 здесь вместо цикла по i было бы только
12     //одно побитное сложение: LUT[Z[j]] ^= W[j];
13 }
14 for (i = 0; i < 64; i++) a[i] = 0;
15 for (i = 0; i < 8; i++)
16 {
17     for (j = 0; j < 256; j++)
18         if ((j >> i) & 1)
19             for (k = 0; k < 8; k++)
20                 a[8*k + i] ^= LUT[k*256 + j];
21 }
```

Конец алгоритма.

Алгоритм 3 работает на ЭВМ существенно быстрее тривиального алгоритма (см. табл. 2). Выигрыш достигается за счёт того, что результатом является небольшая двоичная матрица размера 64×64, а также за счёт эффективного использования дополнительной памяти и многоразрядности ЭВМ.

Таблица 2

Время работы алгоритма умножения матриц $64 \times M$ и $M \times 64$

M	Тривиальный алгоритм	Алгоритм 3
100 000	4,69	0,42
1 000 000	47,02	4,27
10 000 000	470,03	42,59
100 000 000	4715,9	436,8

2.3. Умножение на коэффициенты аннулирующего многочлена

При вычислении вектора $v = \sum\limits_{j=0}^{d-1} B^j Y f_{j+1}$ на каждом шаге третьего этапа необходимо, помимо вычисления матрицы $W_j = B^j Y$, реализовать умножение этой матрицы размера $M \times n$ на вектор f_{j+1} длины n. Если $n = 64$ и W_j представляется в виде массива длины M, состоящего из 64-разрядных чисел, а вектор f_{j+1} записан как одно 64-разрядное число, то умножение W_j на f_{j+1} сводится к операциям побитного умножения и вычисления чётности двоичного веса Хэмминга 64-разрядных чисел. С помощью сдвига и побитного сложения в 2 раза уменьшается размер задачи вычисления чётности двоичного веса вектора, а для 16-разрядных чисел соответствующие значения можно вычислить заранее и хранить в памяти. В результате умножение W_j на f_{j+1} и сложение с результатом предыдущих шагов выполняется за $6M$ операций с 64-разрядными числами и M операций обращения к массиву двоичных весов 16-разрядных чисел. Однако при наличии достаточного объёма оперативной памяти можно выполнить требуемое умножение и сложение более эффективно. Поскольку отображение $\chi : V_{64} \to V_1$, реализующее вычисление чётности двоичного веса Хэмминга 64-разрядного вектора, является линейным, можно на каждом шаге вычислять только побитное произведение f_{j+1} и элементов матрицы W_j и складывать их (побитно) с соответствующими результатами предыдущих шагов, а отображение χ вычислить после выполнения всех d шагов третьего этапа. При этом необходимый объём памяти для хранения промежуточных сумм $\sum\limits_j B^j Y f_{j+1}$ возрастёт в 64 раза.

Заметим, что требуемый объём оперативной памяти можно уменьшить в 2 раза с помощью M сдвигов и M сложений на каждом шаге.

3. Реализация первого и третьего этапов алгоритма на вычислителях кластерного типа

Рассмотрим особенности, возникающие при реализации алгоритма Видемана — Копперсмита на кластерах. Под кластером, или вычислителем кластерного типа, будем понимать некоторый набор ЭВМ (узлов), соединённых между собой высокоскоростной сетью. Поскольку в этом случае отсутствует возможность использования быстрой общей памяти, необходимо решать вопросы о распределении обрабатываемых данных по узлам кластера, координации работы узлов, а также о способе обмена данными между узлами.

3.1. Распределение данных и распараллеливание вычислений

При выборе способа представления данных и распределения их между узлами кластера необходимо исходить из соображений эффективной реализации наиболее трудо-

ёмких операций алгоритма. В случае алгоритма Видемана — Копперсмита наиболее трудоёмкой операцией, выполняемой на каждом шаге первого и третьего этапов, является умножение векторов на разреженную матрицу. Если использовать алгоритм 1, то данная операция легко распараллеливается между узлами кластера путём разбиения разреженной матрицы на блоки строк примерно одинакового веса. Каждый узел хранит свою порцию строк разреженной матрицы B, а также всю матрицу W_j. После умножения на каждом узле находится соответствующая часть результата, т. е. матрицы $W_{j+1} = BW_j$. Перед следующим умножением необходимо выполнить обмен между узлами, чтобы на каждом из них вновь была полная копия правого множителя.

Строки матрицы B необходимо распределять по узлам кластера таким образом, чтобы каждый узел тратил примерно одинаковое время на умножение матрицы W_j на свою порцию строк матрицы B. Как правило, используются узлы кластера с одинаковой производительностью, поэтому время умножения определяется весом строк разреженной матрицы B, доставшихся данному узлу, и их количеством. Важно учитывать время записи результата умножения W_j на одну строку матрицы B, поэтому умножение на несколько «тяжелых» строк матрицы B выполняется быстрее, чем умножение на большее количество более «легких» строк с таким же суммарным количеством ненулевых элементов (см. п. 2.1).

Умножение двух матриц Z и W_j для получения очередного элемента последовательности Крылова $a_j = ZW_j = ZB^jY$, особенно с помощью алгоритма 3, выполняется намного быстрее, чем умножение W_j на разреженную матрицу B, поэтому данную операцию можно не распараллеливать и выполнять на одном из узлов после формирования на нем полной копии матрицы W_j. При этом на выбранном узле необходимо хранить всю матрицу Z.

Вычисление сумм $\sum\limits_j B^jYf_{j+1}$ на шагах третьего этапа также является достаточно быстрой операцией, к тому же её легко распараллелить, так как одновременно может вычисляться до n таких сумм.

3.2. Передача данных и координация работы узлов

Для выполнения алгоритма на кластере необходимо выделить один из узлов для координации действий. На этом управляющем узле хранится результат вычислений первого этапа — матричная последовательность (a_i). Этот же узел выполняет умножение матриц Z и W_j на каждом шаге первого этапа.

Управляющий узел должен следить за ходом выполнения шагов алгоритма. Он даёт команду остальным узлам на выполнение умножения на разреженную матрицу, ждёт отчёта узлов о завершении умножения, затем инициирует процедуру обмена данными между узлами. Как отмечалось выше, после умножения матрицы W_j на разреженную матрицу B на каждом узле будет получена часть матрицы $W_{j+1} = BW_j$, соответствующая доли матрицы B на этом узле. Перед следующим умножением, а также перед вычислением матрицы a_{j+1} необходимо получить полную копию W_{j+1} на каждом узле кластера.

Наиболее эффективным способом обмена данными между узлами в этом случае представляется передача данных по кольцу. Кратко изложим идею этого способа. Пусть имеется k узлов и на i-м узле хранится соответствующая часть X_i некоторого массива X, $i = 1, \ldots, k$. Заметим, что размер части массива X на некоторых узлах может быть равен нулю. Передача осуществляется следующим образом. Сначала каждый узел передаёт предыдущему узлу свою часть массива X, т. е. узел с номером i передает узлу с номером $i - 1$ часть X_i (первый узел передаёт X_1 на узел с номером k). По-

сле этого каждый узел передаёт предыдущему только что полученную от следующего узла часть: i-й узел передает часть X_{i+1}. И так далее. На j-м шаге узел с номером i передает часть X_{i+j-1} и получает часть X_{i+j}, $j = 1, \ldots, k$. Через $k-1$ шагов каждый узел будет иметь весь массив X. Применительно к алгоритму Видемана — Копперсмита, при равномерном распределении данных между k узлами каждый узел отправляет и получает $\dfrac{k-1}{k}64M$ битов. Заметим, что нет необходимости координировать данную процедуру на различных шагах: каждый узел самостоятельно продолжает передачу и приём данных в необходимом объёме, если перед началом процедуры ему известен размер части на предыдущем узле и общий размер массива X.

При использовании технологии `MPI` выполнить подобный обмен между процессами одной группы можно с помощью готовой процедуры `MPI_ALLGATHER` (см., например, [9]). Реализация обмена была подготовлена путём использования системных функций для установления соединения и пересылки данных между двумя узлами (библиотека `Winsock` под ОС Windows и `sys/socket` под ОС Linux). Заметим, что если сетевые карты узлов поддерживают режим полного дуплекса (что типично для современных вычислителей), то удаётся достичь максимально возможной скорости передачи для данной сети. Например, при $M = 10\,000\,000$ и равномерном распределении матрицы между $k = 10$ узлами, соединёнными сетью `gigabit Ethernet` (передача до 10^9 бит/с), сетевой обмен между узлами на каждом шаге выполняется примерно за 585 мс (каждый узел отправляет и получает $576 \cdot 10^6$ битов).

В случае, когда умножение узлами матрицы W_j на их порции разреженной матрицы B выполняется за время, сравнимое с временем обмена результатами умножения, имеет смысл выполнять умножение и передачу одновременно. При этом каждый узел начинает выполнять умножение и сразу же передаёт либо вычисленную им часть матрицы W_{j+1}, либо полученную от соседнего узла. Из-за нагрузки на оперативную память снижение времени работы алгоритма в 2 раза невозможно, однако при оптимальном балансе между временем умножения и передачи удаётся получить выигрыш примерно в 1,5 раза. Заметим, что подобный подход исключает возможность обмена результатами вычисления W_{j+1} между узлами с помощью готовых решений технологии `MPI`.

Результаты работы третьего этапа алгоритма Видемана — Копперсмита могут занимать большой объём памяти, и в этом случае их можно хранить на отдельно выделенных для этого узлах кластера. По окончании выполнения третьего этапа эти узлы передают результаты вычислений на управляющий узел.

3.3. Реализация проверки и сохранения промежуточных результатов

При решении задач больших размеров важными аспектами реализации алгоритма являются проверка и сохранение промежуточных результатов. Наибольшей вычислительной и ёмкостной трудоёмкостью обладает операция умножения на разреженную матрицу. Наибольший объём данных, передаваемых между узлами кластера, также связан с этой операцией. Поэтому проверка правильности вычисления матриц W_i, $i = 1, 2, \ldots$, представляет наиболее важную задачу с точки зрения контроля правильности работы алгоритма.

Проверка правильности вычисления матриц $W_i = B^i Y$ может быть реализована следующим образом. Пусть $i_{\text{test}} \in \mathbb{N}$ — параметр метода, Z_{test} — случайная матрица размера $m_{\text{test}} \times M$. Вычислим значение $Z_{\text{test}} B^{i_{\text{test}}}$. Далее выполняем шаги первого или третьего этапов алгоритма и на шаге $i = i_{\text{test}}$ выполняем проверку: $(Z_{\text{test}} B^{i_{\text{test}}})\, W_0 \stackrel{?}{=}$

$\stackrel{?}{=} Z_{\text{test}} W_{i_{\text{test}}}$. Если равенство не выполняется, то в процессе выполнения данного этапа алгоритма была допущена ошибка и необходимо начать выполнение этого этапа сначала. В случае выполнения сохраняем значение $W_{i_{\text{test}}}$ и выполняем следующие i_{test} шагов, после чего (на шаге $i = 2i_{\text{test}}$) снова выполняем проверку: $(Z_{\text{test}} B^{i_{\text{test}}})\, W_{i_{\text{test}}} \stackrel{?}{=} \stackrel{?}{=} Z_{\text{test}} W_{2i_{\text{test}}}$. Если равенство не выполняется, возвращаемся к шагу $i = i_{\text{test}}$, иначе сохраняем $W_{2i_{\text{test}}}$ и продолжаем выполнение шагов алгоритма. И так далее. На шаге $i = k \cdot i_{\text{test}}$, $k = 1, 2, \ldots$, проверяем равенство $(Z_{\text{test}} B^{i_{\text{test}}})\, W_{(k-1)i_{\text{test}}} \stackrel{?}{=} Z_{\text{test}} W_{k \cdot i_{\text{test}}}$ и сохраняем значение $W_{k \cdot i_{\text{test}}}$ либо возвращаемся на шаг $i = (k-1)i_{\text{test}}$.

Для обнаружения ошибок достаточно использовать матрицу Z_{test} размеров $64 \times M$. В качестве Z_{test} можно взять первые 64 столбца случайной матрицы Z. Однако иногда вместо случайной матрицы Z выбирают матрицу, состоящую из первых m столбцов единичной матрицы (см. далее), и в этом случае Z_{test} лучше выбирать случайно. Параметр i_{test} выбирают исходя из размеров задачи и надёжности вычислителя. При факторизации 768-битного числа, описанной в [4], проверка выполнялась каждые $i_{\text{test}} = 2^{14}$ шагов (всего на первом этапе было вычислено $L \approx 565000$ элементов последовательности (a_i), причём в работе не сказано о наличии сбоев при вычислении).

При выполнении первого этапа алгоритма Видемана — Копперсмита в качестве промежуточных результатов выступают элементы последовательности (a_i) и матрицы $W_i = B^i Y$, поэтому их сохранение реализуется автоматически при выполнении проверки правильности вычислений. В случае сбоя на шаге i (например, при отключении питания вычислителя и т. п.) достаточно вернуться к шагу $k \cdot i_{\text{test}}$, где $k = \lfloor i / i_{\text{test}} \rfloor$, и продолжить выполнение алгоритма. На третьем же этапе необходимо выполнять сохранение промежуточных сумм $\sum\limits_j B^j Y f_{j+1}$. Эта процедура также выполняется каждые i_{test} шагов, и промежуточные суммы можно хранить либо на узлах, которые эти суммы вычисляют, либо на управляющем узле кластера. В последнем случае необходимо выполнить передачу этих значений с соответствующих узлов кластера на управляющий узел. Заметим, что если применяется метод ускорения вычислений сумм $\sum\limits_j B^j Y f_{j+1}$, описанный в п. 2.3, то объём передаваемых и сохраняемых данных не возрастает, так как перед передачей и сохранением промежуточных результатов можно выполнить вычисление чётности двоичных весов Хэмминга и хранить собственно значения $\sum\limits_j B^j Y f_{j+1}$.

3.4. Некоторые способы снижения вычислительной и ёмкостной сложности алгоритма

Приведём некоторые незначительные изменения алгоритма Видемана — Копперсмита, которые позволяют снизить требуемый объём оперативной памяти вычислителя и сократить трудоёмкость отдельных операций.

Первое изменение связано с выбором матрицы Z. В исходном описании алгоритма сказано, что матрица Z должна выбираться случайным образом. При этом появляются затраты на её хранение и умножение на матрицы W_i. Действительно, если матрицу Y можно разбить на подматрицы по 64 столбца и выполнять первый и третий этапы алгоритма для каждой из этих подматриц на несвязанных между собой вычислителях, то матрица Z должна храниться на каждом вычислителе целиком, что при большом значении m приводит к существенным затратам оперативной памяти. Чтобы этого избежать, в качестве Z можно использовать первые m строк единичной матрицы. В этом случае матрицу Z хранить не нужно, а результат умножения $a_i = Z W_i$ — это первые

m строк матрицы W_i. Следует отметить, что в этом случае нарушается обоснование надёжности алгоритма [3], основанное на вероятностных соображениях, однако многочисленные эксперименты показывают, что для невырожденных матриц A алгоритм работает нормально. В качестве компромисса можно использовать матрицу Z, столбцы которой имеют вес 1. Если $m \leqslant 256$, то для хранения такой матрицы достаточно M байтов вместо $mM/8$, а умножение на неё матрицы W_i выполняется, очевидно, более эффективно, чем в общем случае.

Более важным является снижение трудоёмкости умножения матрицы W_i на разреженную матрицу B. При изложении алгоритма в п. 1 было сказано, что B состоит из первых M столбцов матрицы A. Тем не менее при неравномерном распределении веса столбцов имеет смысл составлять матрицу B из столбцов меньшего веса; иными словами, перед выполнением алгоритма выполнить перестановку столбцов матрицы A (перенумерацию неизвестных), чтобы снизить вес матрицы B и тем самым снизить трудоёмкость умножения на эту матрицу.

Вес строк матрицы A также может быть распределён неравномерно. В результате работы алгоритма Видемана — Копперсмита получается не одно, а сразу n решений системы. Если перед выполнением алгоритма отбросить одно уравнение системы, т. е. одну строку матрицы A, и найти n решений для такой системы, то можно получить не менее $n-1$ решений исходной системы. Действительно, пусть $w_1, \ldots, w_{k-1}, w_k$ — решения урезанной системы, не удовлетворяющие отброшенному уравнению, $1 \leqslant k \leqslant n$. Тогда $k-1$ векторов $w_1 \oplus w_k, \ldots, w_{k-1} \oplus w_k$ будут удовлетворять и отброшенному уравнению, и, очевидно, всем остальным. Следовательно, если требуется получить небольшое число решений (например, одно), наиболее тяжёлые строки матрицы A (т. е. строки большого веса) можно изначально исключить; при исключении t строк из полученных n решений можно сформировать не менее $n-t$ решений исходной системы уравнений.

4. Особенности применения алгоритма Видемана — Копперсмита в рамках метода решета числового поля факторизации целых чисел

Как уже было отмечено, нахождение решений большой разреженной системы однородных линейных уравнений над полем GF(2) является одним из этапов метода решета числового поля, применяемого для факторизации больших целых чисел. При этом матрица решаемой системы не является случайной и имеет ряд ярко выраженных особенностей, что необходимо учитывать при выполнении алгоритма Видемана — Копперсмита.

Не вдаваясь в подробности, можно сказать, что каждая строка матрицы соответствует очередному простому числу, а элементы строки представляют чётность показателя степени, с которой данное простое число входит в разложение некоторых выбранных целых чисел. Для случайного натурального числа n и простого p вероятность того, что p^k делит n, а p^{k+1} не делит n, примерно равна $(p-1)/p^{k+1}, k = 0, 1, \ldots$ (понятно, что при $k > \log_p n$ эта вероятность равна нулю). Соответственно вероятность того, что элемент строки, соответствующей простому числу p, равен 1, примерно равна $\sum\limits_{k=1}^{\infty} \dfrac{p-1}{p^{2k}} = \dfrac{1}{p+1}$. Поэтому первые строки матрицы имеют достаточно большой вес (для первой строки, соответствующей $p = 2$, вес равен примерно $N/3$), а затем с ростом номера строки вес строки быстро снижается. Используя идею п. 3.4, можно отбросить несколько первых строк (т. е. уравнений), выполнить алгоритм, а затем из полученных решений урезанной системы составить несколько решений исходной системы.

Ещё один важный момент касается предварительной обработки матрицы решаемой разреженной СОЛУ. Например, если вес строки равен 1, то можно сразу выполнить исключение соответствующей переменной из решаемой системы. Аналогично, если некоторая переменная встречается только в одном уравнении (т. е. вес столбца равен 1), то соответствующие строку и столбец можно удалить перед выполнением алгоритма поиска решений. Применяются и другие операции (см., например, [10]), приводящие к снижению размеров M и N системы, но увеличивающие средний вес строки ω матрицы системы. Такая предварительная обработка матрицы называется *фильтрацией* и выполняется до того момента, пока эти операции приводят к снижению числа $M^2\omega$, определяющего трудоёмкость первого и третьего этапов алгоритма Видемана — Копперсмита (при реализации алгоритма на кластере необходимо также учитывать общее время передачи данных по сети, которое пропорционально M^2). При факторизации 768-битного числа [4] исходная матрица $35\,288\,334\,017 \times 47\,762\,243\,404$ была за 15 дней ужата до размеров $192\,795\,550 \times 192\,796\,550$ со средним весом строки $\omega = 144$.

При выполнении фильтрации решение о её прекращении принимается, исходя из оптимальных соотношений между размерами матрицы и её весом, приводящих к минимальной трудоёмкости алгоритма нахождения решений СОЛУ. Для одной и той же матрицы данные соотношения могут быть различными при использовании разных вычислителей для выполнения алгоритма Видемана — Копперсмита. Поэтому при решении сильно разреженной СОЛУ фильтрация является неотъемлемым этапом и должна согласовываться с выбранным вычислителем и параметрами алгоритма Видемана — Копперсмита.

Заключение

Рассмотрены вопросы, связанные с реализацией первого и третьего этапов алгоритма Видемана — Копперсмита. Наиболее трудоёмкой операцией этих этапов является умножение блока векторов на разреженную матрицу. Эта операция может быть эффективно распараллелена путем независимого умножения на различные строки разреженной матрицы, однако при реализации на вычислителях кластерного типа требуется дополнительный обмен между узлами для получения результата умножения.

Уделено внимание и другим операциям первого и третьего этапов алгоритма Видемана — Копперсмита, поскольку их неэффективная реализация может существенно увеличить время работы алгоритма. В рамках исследования особенностей реализации алгоритма на кластерах рассмотрены вопросы проверки и сохранения промежуточных результатов вычислений для защиты от сбоев в работе вычислителя. Приведены также некоторые особенности, возникающие при использовании алгоритма Видемана — Копперсмита в рамках факторизации больших целых чисел методом решета числового поля.

ЛИТЕРАТУРА

1. *Coppersmith D.* Fast evaluation of logarithms in fields of characteristic two // IEEE Trans. Inform. Theory. 1984. V. IT-30(4). P. 587–594.
2. *Montgomery P. L.* A block Lanczos algorithm for finding dependencies over GF(2) // EUROCRYPT'95. LNCS. 1995. V. 921. P. 106–120.
3. *Coppersmith D.* Solving linear equations over GF(2) via block Wiedemann algorithm // Math. Comp. 1994. V. 62(205). P. 333–350.
4. *Kleinjung T., Aoki K., Franke J., et al.* Factorization of a 768-bit RSA modulus // CRYPTO-2010. LNCS. 2010. V. 6223. P. 333–350.

5. *Wiedemann D. H.* Solving sparse linear equations over finite fields // IEEE Trans. Inform. Theory. 1986. V. IT-32(1). P. 54–62.
6. *Рыжов А. С.* О реализации алгоритма Копперсмита для двоичных матричных последовательностей на вычислителях кластерного типа // Прикладная дискретная математика. 2013. № 3. С. 112–122.
7. *Thomé E.* Subquadratic computation of vector generating polynomials and improvement of the block Wiedemann algorithm // J. Symb. Comput. 2002. No. 33. P. 757–775.
8. *Ахо А., Хопкрофт Д., Ульман Дж.* Построение и анализ вычислительных алгоритмов. М.: Мир, 1979. 536 с.
9. `www.open-mpi.org/doc/v1.6` — Open MPI v1.6.4 documentation. 2013.
10. *Cavallar S.* Strategies for filtering in the number field sieve // Proc. ANTS IV. LNCS. 2000. V. 1838. P. 209–231.

2013 Вычислительные методы в дискретной математике №4(22)

DOI 10.17223/20710410/22/10

УДК 519.863

АЛГОРИТМ ТОЧНОГО РЕШЕНИЯ ДИСКРЕТНОЙ ЗАДАЧИ ВЕБЕРА ДЛЯ ПРОСТОГО ЦИКЛА

Р. Э. Шангин

Южно-Уральский государственный университет, г. Челябинск, Россия

E-mail: shanginre@gmail.com

Предлагается полиномиальный алгоритм, находящий точное решение задачи Вебера в дискретной постановке для простого цикла и конечного множества позиций размещения, основанный на динамическом программировании. Проведен вычислительный эксперимент по анализу эффективности предложенного алгоритма в сравнении с пакетом IBM ILOG CPLEX.

Ключевые слова: *задача Вебера, простой цикл, динамическое программирование, точный алгоритм.*

Введение

Рассматривается дискретная задача Вебера [1, 2] для случая, когда размещаемый граф имеет вид простого цикла. Приведём математическую формулировку задачи.

Пусть $G = (J, E)$ — простой цикл, где J — множество вершин (размещаемые объекты); E — множество рёбер графа G (связи между размещаемыми объектами). Пусть V — конечное множество позиций (точек), предназначенных для размещения вершин графа G. Размещением вершин графа G назовём однозначное отображение $\pi : J \to V$, то есть вершина $i \in J$ размещается в позицию $\vartheta_i \in V$. Множество всех однозначных отображений множества вершин J в множество позиций V обозначим через $\Pi = \{\pi : J \to V\}$.

Стоимость размещения вершины $i \in J$ в множестве позиций V задаётся функцией $p : J \times V \to \mathbb{R}^+$, где $p(i, \vartheta_i)$ — стоимость размещения вершины $i \in J$ в позиции $\vartheta_i \in V$. Стоимость размещения ребра $[i, j] \in E$ на V^2 определяется функцией $c : E \times V^2 \to \mathbb{R}^+$, где $c([i, j], \vartheta_i, \vartheta_j)$ — стоимость размещения ребра $[i, j] \in E$ на V^2 при размещении его концевых вершин $i, j \in J$ в позициях $\vartheta_i, \vartheta_j \in V$ соответственно. Отметим, что зависимость между стоимостью размещения ребра $[i, j]$ и стоимостью размещения его концевых вершин i, j отсутствует.

Необходимо разместить вершины графа G в позициях множества V таким образом, чтобы суммарная стоимость размещения вершин и рёбер графа G была минимальной:

$$F(\pi) = \sum_{[i,j]\in E} c([i, j], \pi(i), \pi(j)) + \sum_{i\in J} p(i, \pi(i)) \to \min_{\pi\in\Pi}. \quad (1)$$

Заметим, что в дискретной постановке задачи Вебера размещаемые объекты рассматриваются как точки, а структура области, в которой производится размещение, является дискретной, то есть для размещения объектов указывается конечное количество позиций, причём ограничения на размещение объектов отсутствуют.

В общем случае дискретная задача Вебера является NP-трудной [3] и представляет собой релаксацию квадратичной задачи о назначениях [4], где условие инъективности

отображения множества вершин J графа G в конечное множество позиций размещения V снимается, то есть в дискретной задаче Вебера в одну позицию возможно размещение нескольких вершин графа [5 – 7].

Задача Вебера исследовалась в различных постановках, в том числе для непрерывной области размещения [8], в многокритериальной постановке [9] и др. Известны полиномиально разрешимые частные случаи. Для решения задачи Вебера на древовидной сети в непрерывной постановке, то есть когда допускается размещение объектов на дугах, разработаны полиномиальные алгоритмы [10]. В [11] предложен полиномиальный алгоритм решения минимаксной задачи Вебера на дереве. Предложен полиномиальный алгоритм для решения задачи Вебера для корневого дерева и конечного множества позиций размещения [6].

1. Точный алгоритм для решения дискретной задачи Вебера для цикла

Обозначим тройкой (G, V, F) рассматриваемую дискретную задачу Вебера (1), где $G = (J, E)$ — простой цикл, V — конечное множество позиций размещения и F — функция стоимости размещения графа G. Предлагается полиномиальный алгоритм CyWPA (Cycle Weber Problem Algorithm), основанный на динамическом программировании (ДП) и находящий оптимальное решение задачи (G, V, F).

Введём следующие обозначения. Пусть $s \in J$ — произвольная вершина цикла G и $T = (I, W)$ — подграф графа G, индуцированный множеством вершин $J \setminus \{s\}$. Пусть $N = |I|$ — мощность множества вершин цепи T и $i_N \in I$ — висячая вершина цепи T. Выбор вершины i_N в качестве корня дерева $T = (I, W)$ индуцирует на множестве вершин I отношение частичного порядка

$$L = \{(i, j) : i, j \in I, j \text{ принадлежит цепи в } T \text{ между } i \text{ и } i_N\}. \tag{2}$$

В дальнейшем будем считать, что $I = \{1, 2, \ldots, N\}$ и выполняется условие $(l, m) \in L \Rightarrow$ $\Rightarrow l < m$, где $l, m \in I$. Обозначим G_i — подграф графа G, индуцированный множеством вершин $\{s, 1, 2, \ldots, i-1, i\}$. На рис. 1 представлены цепь T и подграф G_3 в цикле G.

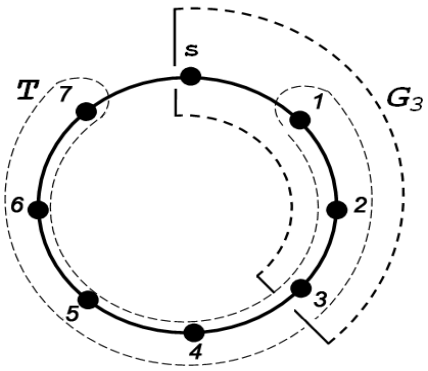


Рис. 1. Цепь T и подграф G_3 в простом цикле G

Положим, что $G(\vartheta_s)$, $\vartheta_s \in V$, есть граф G, в котором вершина s размещена в позицию ϑ_s. Исходную задачу (G, V, F) разобьём на ряд подзадач $(G(\vartheta_s), V, F)$ для любых $\vartheta_s \in V$, где тройка $(G(\vartheta_s), V, F)$ есть дискретная задача Вебера. Решение каждой подзадачи $(G(\vartheta_s), V, F)$, в свою очередь, разбивается на $N + 1$ шагов процесса ДП. Значение функции Беллмана $f_i(\vartheta_i)$, вычисленное на шаге i процесса ДП решения подзадачи $(G(\vartheta_s), V, F)$ для некоторого состояния $\vartheta_i \in V$, есть стоимость оптимального размещения подграфа G_i в множестве позиций V, когда размещение вершин s и i в множестве V

равно ϑ_s и ϑ_i соответственно. На шаге i процесса ДП решения подзадачи $(G(\vartheta_s), V, F)$ для каждого состояния $\vartheta_i \in V$ определяется также множество $V(\vartheta_i)$ оптимального размещения вершин подграфа G_i в множестве позиций V, когда размещение вершин s и i в множестве V равно ϑ_s и ϑ_i соответственно.

Алгоритм CyWPA

Э т а п 0. Выбрать произвольную вершину $s \in J$. Определить подграф $T = (I, W)$ и на множестве его вершин задать отношение порядка (2). Для каждого $\vartheta_s \in V$ определить граф $G(\vartheta_s)$.

Э т а п 1. Для каждого $\vartheta_s \in V$ решить подзадачу $(G(\vartheta_s), V, F)$.

Шаг 1 процесса ДП. Для каждого состояния $\vartheta_1 \in V$ вычислить значение функции Беллмана

$$f_1(\vartheta_1) = p(1, \vartheta_1) + c([1, s], \vartheta_1, \vartheta_s) + p(s, \vartheta_s)$$

и определить множество размещения

$$V(\vartheta_1) = \{\vartheta_1\} \cup \{\vartheta_s\}.$$

Шаг i, $i = 2, 3, \ldots, N-1$, процесса ДП. Для каждого $\vartheta_i \in V$ и любого $\vartheta_{i-1} \in V$ вычислить значение функции $R(\vartheta_i, \vartheta_{i-1})$ стоимости оптимального размещения подграфа G_i в множестве позиций V, когда вершины s, i и $i-1$ размещены в позициях ϑ_s, ϑ_i и ϑ_{i-1} соответственно:

$$R(\vartheta_i, \vartheta_{i-1}) = p(i, \vartheta_i) + c([i, i-1], \vartheta_i, \vartheta_{i-1}) + f_{i-1}(\vartheta_{i-1}).$$

Для каждого состояния $\vartheta_i \in V$ вычислить значение функции Беллмана

$$f_i(\vartheta_i) = \min_{\vartheta_{i-1} \in V} \{R(\vartheta_i, \vartheta_{i-1})\} \tag{3}$$

и определить множество размещения

$$V(\vartheta_i) = \{\vartheta_i\} \cup V(\vartheta^*_{i-1}),\ \vartheta^*_{i-1} = \arg \min_{\vartheta_{i-1} \in V} \{R(\vartheta_i, \vartheta_{i-1})\}. \tag{4}$$

Шаг N процесса ДП. Для каждого $\vartheta_{i_N} \in V$ и любого $\vartheta_{i_N - 1} \in V$ вычислить значение функции $R(\vartheta_{i_N}, \vartheta_{i_N - 1})$ по формуле

$$R(\vartheta_{i_N}, \vartheta_{i_N - 1}) = p(i_N, \vartheta_{i_N}) + c([i_N, i_N - 1], \vartheta_{i_N}, \vartheta_{i_N - 1}) + f_{N-1}(\vartheta_{i_N - 1}) + c([i_N, s], \vartheta_{i_N}, \vartheta_s).$$

Для каждого состояния $\vartheta_{i_N} \in V_N$ вычислить значение функции Беллмана $f_N(\vartheta_{i_N})$ и определить множество $V(\vartheta_{i_N})$ по формулам (3) и (4) соответственно.

Шаг $N+1$ процесса ДП. Определить оптимальное размещение $\pi_{G(\vartheta_s)}$ вершин графа $G(\vartheta_s)$ и его стоимость $F_{G(\vartheta_s)}$ по формулам

$$\pi_{G(\vartheta_s)} = V(\vartheta^*_{i_N}),\ \vartheta^*_{i_N} = \arg \min_{\vartheta_{i_N} \in V} \{f_N(\vartheta_{i_N})\}, \tag{5}$$

$$F_{G(\vartheta_s)} = \min_{\vartheta_{i_N} \in V} \{f_N(\vartheta_{i_N})\}.$$

Э т а п 2. После того, как оптимальное решение подзадач $(G(\vartheta_s), V, F)$ для каждого $\vartheta_s \in V$ найдено, определить оптимальное решение π^*_G исходной задачи (G, V, F):

$$\pi^*_G = \pi_{G(\vartheta^*_s)},\ \vartheta^*_s = \arg \min_{\vartheta_s \in V} \{F_{G(\vartheta_s)}\}. \tag{6}$$

Стоп.

Теорема 1. Алгоритм CyWPA находит точное решение задачи Вебера (G, V, F), где $G = (J, E)$ — простой цикл; V — конечное множество позиций размещения.

Доказательство. Докажем, что подзадача $(G(\vartheta_s), V, F)$ при каждом фиксированном размещении $\vartheta_s \in V$ решается алгоритмом CyWPA оптимально. Очевидно, на шаге 2 процесса ДП для любых $\vartheta_s, \vartheta_2 \in V$ алгоритм находит оптимальное размещение вершины 1 относительно заданного размещения ϑ_s, ϑ_2 вершин s и 2, причём в данном случае это достигается путем полного перебора всех способов размещения ϑ_1 вершины 1 в множестве позиций V. На шаге 3 процесса ДП для любых $\vartheta_s, \vartheta_3 \in V$ алгоритм находит оптимальное размещение вершин 1 и 2 относительно заданного размещения ϑ_s, ϑ_3 вершин s и 3, поскольку для вершины 1 оптимальное размещение определено на предыдущем шаге 1 процесса ДП для каждого заданного размещения ϑ_2 вершины 2, а оптимальное размещение вершины 2, согласно формуле (4), находится перебором способов размещения ϑ_2 в множестве позиций V с целью минимизации стоимости размещения подграфа G_3, когда размещение вершин s и 3 в V равно ϑ_s и ϑ_3 соответственно.

Аналогично, на последующих шагах $i = 4, 5, \ldots, N$ динамического процесса для любых $\vartheta_s, \vartheta_i \in V$ алгоритм находит оптимальное размещение вершин $1, 2, \ldots, i-1$ относительно заданного размещения ϑ_s, ϑ_i вершин s и i.

Таким образом, рекуррентно определяя оптимальное размещение вершин подграфа G_i на шагах $i = 2, 3, \ldots, N$, алгоритм на шаге $N+1$ процесса ДП, согласно формуле (5), зная оптимальное размещение вершин $1, 2, \ldots, N-1$ относительно любых заданных размещений ϑ_s и ϑ_{i_N}, находит оптимальное размещение графа $G(\vartheta_s)$.

Поскольку размещение графа $G(\vartheta_s)$ для любых $\vartheta_s \in V$ найдено оптимально, то алгоритм на этапе 2, осуществив полный перебор способов размещения ϑ_s вершины s в множестве позиций V согласно формуле (6), находит оптимальное решение исходной задачи (G, V, F). ■

Теорема 2. Вычислительная сложность алгоритма CyWPA не превосходит $\mathrm{O}(|V|^3(|J|-2))$ операций. Пространственная сложность равна $\mathrm{O}(|V|^2)$ памяти.

Доказательство. Определим оценку числа операций, требуемых для решения одной подзадачи $(G(\vartheta_s), V, F)$, $\vartheta_s \in V$. На шаге 1 процесса ДП необходимо выполнить $\mathrm{O}(|V|)$ операций, так как $\mathrm{O}(|V|)$ операций требуется для вычисления значений функции Беллмана $f_1(\cdot)$ и $\mathrm{O}(|V|)$ операций — для определения множеств размещения $V(\cdot)$ вершин подграфа G_1. На каждом из последующих шагов $i = 2, 3, \ldots, N$ процесса ДП требуется выполнить $\mathrm{O}(|V|^2)$ операций, поскольку $\mathrm{O}(|V|^2)$ операций необходимо для вычисления значений функции $R(\cdot)$, столько же — для вычисления значений функции Беллмана $f_i(\cdot)$ и столько же — для определения множеств размещения $V(\cdot)$ вершин подграфа G_i. На шаге $N+1$ требуется выполнить $\mathrm{O}(|V|)$ операций, так как столько операций необходимо для определения множества $\pi_{G(\vartheta_s)}$ оптимального размещения вершин графа $G(\vartheta_s)$ и для вычисления значения функции $F_{G(\vartheta_s)}$ стоимости такого оптимального размещения. В соответствии с этим для решения одной подзадачи $(G(\vartheta_s), V, F)$ требуется $\mathrm{O}(|V|^2(|J|-2))$ вычислительных операций, где

$$\mathrm{O}(|V|^2(|J|-2)) = \mathrm{O}(|V|) + \mathrm{O}(|V|^2)(|J|-2) + \mathrm{O}(|V|).$$

Заметим, что для определения множества π^*_G оптимального размещения вершин графа G требуется $\mathrm{O}(|V|)$ операций. Исходя из того, что количество решаемых подзадач $(G(\vartheta_s), V, F)$ равно $|V|$, а для решения одной подзадачи требуется $\mathrm{O}(|V|^2 \cdot (|J|-2))$

операций, оценка вычислительной сложности алгоритма CyWPA равна $O(|V|^3 \cdot (|J|-2))$ операций.

Определим оценку пространственной сложности алгоритма. Положим, что для хранения множества $V(\cdot)$ требуется $O(|V|)$ памяти. При решении любой подзадачи $(G(\vartheta_s), V, F)$ на шаге 1 процесса ДП требуется $O(|V|^2)$ памяти, так как $O(|V|^2)$ памяти требуется для хранения множеств $V(\vartheta_1)$, $\vartheta_1 \in V$, и $O(|V|)$ памяти — для хранения массива $f_1(\cdot)$, где для хранения одного значения функции $f_i(\cdot)$ требуется $O(1)$ памяти. На каждом из последующих шагов $i = 2, 3, \ldots, N$ процесса ДП требуется $O(|V|^2)$ памяти, поскольку $O(|V|^2)$ памяти требуется для хранения множеств $V(\vartheta_{i-1})$, $\vartheta_{i-1} \in V$, и массива $f_{i-1}(\cdot)$, полученных на предыдущем шаге $i-1$ процесса ДП; $O(|V|^2)$ памяти требуется для хранения массива $R(\vartheta_i, \vartheta_{i-1})$, $\vartheta_i, \vartheta_{i-1} \in V$, и $O(|V|^2)$ памяти — для хранения множеств $V(\vartheta_i)$, $\vartheta_i \in V$, и массива $f_i(\cdot)$. На шаге $N+1$ требуется $O(|V|^2)$ памяти, так как столько памяти требуется для хранения множеств $V(\vartheta_{i_N})$, $\vartheta_{i_N} \in V$, и массива $f_N(\cdot)$, полученных на предыдущем шаге N процесса ДП, и $O(|V|)$ памяти требуется для хранения множества $\pi_{G(\vartheta_s)}$ оптимального размещения вершин графа $G(\vartheta_s)$. Заметим, что на каждом шаге $i = 2, 3, \ldots, N$ процесса ДП для определения множеств $V(\vartheta_i)$, $\vartheta_i \in V$, и массива $f_i(\cdot)$ требуются только соответствующие множества $V(\vartheta_{i-1})$, $\vartheta_{i-1} \in V$, и массив $f_{i-1}(\cdot)$, полученные на предыдущем шаге $i-1$, поэтому необходимость хранения в памяти на шаге $i = 2, 3, \ldots, N$ массивов $V(\vartheta_j)$, $\vartheta_j \in V$, и $f_j(\cdot)$, определённых на всех предшествующих шагах $j = i-2, i-3, \ldots$ процесса ДП, отпадает.

На этапе 2 для определения множества π_G^* оптимального размещения вершин графа G требуется $O(|V|^2)$ памяти, поскольку $O(|V|)$ памяти требуется для хранения одного множества размещения $\pi_{G(\vartheta_s)}$, а количество хранимых в памяти множеств $\pi_{G(\vartheta_s)}$ на этапе 2 равно $|V|$. В соответствии с этим оценка пространственной сложности предложенного алгоритма CyWPA равна $O(|V|^2)$ памяти. ■

2. Экспериментальное исследование эффективности алгоритма CyWPA

Алгоритм CyWPA реализован на ЭВМ в среде MATLAB. Проведён вычислительный эксперимент по анализу его эффективности. Для оценки эффективности алгоритма использовался программный пакет IBM ILOG CPLEX Optimization Studio 12.2 (решение модели целочисленного линейного программирования (ЦЛП) дискретной задачи Вебера алгоритмом ветвей и границ с ограничением по времени работы).

Для проведения эксперимента был случайным образом с равномерным распределением сгенерирован класс тестовых задач, состоящий из серий, каждая из которых включает 30 задач одинаковой размерности. Вычисления проводились на компьютере с процессором Intel Pentium 1.86 GHz.

Результаты вычислительного эксперимента приведены в таблице, где $\bar{t}_{\text{alg}}$ — среднее время работы предложенного алгоритма, с; $\bar{t}_{\text{ILP}}$ — среднее время работы модели ЦЛП, реализованной в среде IBM ILOG CPLEX Optimization Studio 12.2, с.

Результаты эксперимента

$\lvert J\rvert$	$\lvert V\rvert$	$\bar{t}_{\text{alg}}$	$\bar{t}_{\text{ILP}}$
5	5	0,0167	0,0071
10	10	0,0854	0,6927
20	20	0,9139	15,3793
40	40	9,2132	—
100	100	386,8167	—

«—» — решение не удалось получить за приемлемое время (1000 с)

Для дискретной задачи Вебера для простого цикла $G = (J, E)$ и конечного множества позиций размещения V размерности $|J| = 40$, $|V| = 40$ и выше не удалось получить решение с помощью пакета IBM ILOG CPLEX за приемлемое время (1000 с); среднее время решения задачи Вебера такой размерности с помощью предлагаемого алгоритма CyWPA не превысило 10 с. Для задачи Вебера размерности $|J| = 20$, $|V| = 20$ среднее время работы предлагаемого алгоритма не превысило 1 с, тогда как среднее время работы пакета IBM ILOG CPLEX составило 16 с. Стоит отметить, что среднее время решения задачи Вебера размерности $|J| = 5$, $|V| = 5$ и ниже с помощью предложенного алгоритма CyWPA значительно превзошло среднее время работы пакета IBM ILOG CPLEX.

Заключение

Предложен полиномиальный алгоритм, основанный на динамическом программировании, находящий оптимальное решение дискретной задачи Вебера, когда размещаемый граф имеет вид простого цикла. Определены оценки вычислительной и пространственной сложности предложенного алгоритма.

Проведён вычислительный эксперимент по анализу эффективности предложенного алгоритма в сравнении с пакетом IBM ILOG CPLEX Optimization Studio 12.2. Среднее время решения дискретной задачи Вебера размерности $|J| = 100$, $|V| = 100$ для простого цикла и конечного множества позиций размещения с помощью предложенного алгоритма CyWPA не превысило 400 с, в то время как с помощью пакета IBM ILOG CPLEX 12.2 не удалось получить решение задачи размерности $|J| = 40$, $|V| = 40$ за 1000 с. Из результатов эксперимента следует, что применение предложенного алгоритма перспективно для решения дискретной задачи Вебера для простого цикла средней и большой размерности.

ЛИТЕРАТУРА

1. *Шангин Р. Э.* Исследование эффективности приближенных алгоритмов решения одного частного случая задачи Вебера // Экономика, статистика и информатика. Вестник УМО. 2012. № 1. С. 163–168.
2. *Панюков А. В., Пельцвергер Б Ф.* Оптимальное размещение дерева в конечном множестве // Журн. вычисл. математики и матем. физики. 1988. Т. 28. № 2. С. 618–620.
3. *Панюков А. В.* Модели и методы решения задач построения и идентификации геометрического размещения: дис. ... д-ра физ.-мат. наук. Челябинск, 1999. 260 с.
4. *Сергеев С. А.* Квадратичная задача о назначениях I // Автоматика и телемеханика. 1999. № 8. С. 127–147.
5. *Панюков А. В., Пельцвергер Б. Ф., Шафир А. Ю.* Оптимальное размещение точек ветвления транспортной сети на цифровой модели местности // Автоматика и телемеханика. 1990. № 9. С. 153–162.
6. *Panyukov A. V. and Pelzwerger B. V.* Polynomial algorithms to finite Weber problem for a tree network // J. Comput. Appl. Math. 1991. V. 35. P. 291–296.
7. *Шангин Р. Э.* Разработка и анализ алгоритмов для задачи Вебера // Проблемы оптимизации и экономические приложения. Омск, 2012. С. 121–122.
8. *Трубин В. А.* Эффективный алгоритм для задачи Вебера с прямоугольной метрикой // Кибернетика. 1978. № 6. С. 67–70.
9. *Zabudsky G. G. and Filimonov D. V.* An algorithm for minimax location problem on tree with maximal distances // Proc. Second Intern. Workshop “Discrete Optimization Methods in Production and Logistics” (DOM2004). Omsk, 2004. P. 81–85.

10. *Забудский Г. Г., Филимонов Д. В.* О минимаксной и минисуммной задачах размещения на сетях // Труды XII Байкальской Междунар. конф. «Методы оптимизации и их приложения». Омск, 2001. С. 150–155.

11. *Филимонов Д. В.* Решение дискретной минимаксной задачи размещения на древовидной сети // Материалы ежегодного научного семинара аспирантов. Омск, 2003. С. 58–61.

ДИСКРЕТНЫЕ МОДЕЛИ РЕАЛЬНЫХ ПРОЦЕССОВ

DOI 10.17223/20710410/22/11

УДК 519.710.22

МОДЕЛИРОВАНИЕ РОСТА ТКАНИ С УЧЁТОМ ВОЗМОЖНОСТИ ВНЕШНЕГО ВОЗДЕЙСТВИЯ НА ЕЁ ФОРМУ

М. Н. Назаров

Национальный исследовательский университет «МИЭТ», г. Москва, Россия

E-mail: Nazarov-Maximilian@yandex.ru

Построена имитационная модель морфогенеза, которая включает в свой формализм внешнее управляющее воздействие на форму скоплений клеток в виде потока сигнальных молекул через границу этих скоплений. Данная модель использует только универсальные параметры, для которых гарантируется наличие биологического смысла, а также возможность использования их значений в моделях морфогенеза на основе дифференциальных уравнений.

Ключевые слова: *многоклеточные организмы, морфогенез, дискретные модели, управляемый синтез ткани.*

Введение

Благодаря прогрессу в технологии трёхмерной печати на текущий момент стало возможным печатать трёхмерные конфигурации из живых клеток с помощью специализированного 3D-принтера. Данная технология открывает большие возможности для таких дисциплин, как регенеративная медицина и трансплантология, а также может послужить источником живых моделей тканей и органов для фармакологии. С примером реализации технологии трёхмерной печати для эмбриональных стволовых клеток можно ознакомиться в работе [1].

Однако на пути широкого внедрения данной технологии для массового производства органов и тканей встаёт сразу несколько серьёзных проблем. Во-первых, для любых более или менее развитых органов высших животных характерно использование клеток разных типов, организованных в комплексные трёхмерные структуры, которые исключительно сложно воспроизвести с помощью 3D-печати (например, любые из перечисленных органов пронзает множество тонких капиллярных сосудов, и воспроизвести их с помощью принтера практически невозможно). Во-вторых, клетки в настоящих организмах при формировании тканей и органов поддерживают механические связи с соседями, тем самым препятствуя распаду скопления и стабилизируя его форму. Однако клеточная конфигурация, которая получается в результате 3D-печати, будет сразу после окончания печати практически полностью лишена подобных связей, и на создание устойчивых связей друг с другом клеткам потребуется время, за которое распечатанное скопление может потерять свою форму.

Одно из возможных решений обозначенных проблем заключается в печати «заготовки для органа» из менее специализированных клеток, которые имеют гораздо более простую трёхмерную структуру, а также не сильно чувствительны к небольшим смещениям клеток сразу после печати. При этом, чтобы данные заготовки выросли

в полноценный орган, для них нужно будет воспроизвести условия, в которых они развиваются в настоящем организме.

Ключевым процессом, который позволяет регулировать развитие тканей, является межклеточный сигнальный обмен; таким образом, воспроизведение условий роста тканей и органов сводится в первую очередь к внесению внешнего сигнального воздействия на скопления клеток (подробнее об этом см. [2–4]).

Нужно отметить, что сама задача воспроизведения условий, в которых существуют отдельные органы и ткани в организме, не ограничивается только 3D-печатью заготовок для органов, но и может быть напрямую использована в фармакологии для создания моделей определённых тканей для последующего тестирования лекарств. Так, например, в работе [5] рассматривается построение «клеточных популяций на чипе» — специальной программируемой подложке, которая способна воспроизводить механические напряжения для клеток.

В работе [6] рассмотрен набор универсальных параметров для моделей морфогенеза и на его основе построены две демонстрационные модели: имитационная клеточно-автоматная и количественная модель на базе дифференциальных уравнений. Напомним, что под универсальными параметрами в [6] подразумевается такой набор параметров, для которого требуется:

1) наличие биологического смысла для всех параметров;
2) возможность оценки значений параметров на базе биохимических моделей;
3) переносимость значений параметров между моделями разных классов.

Для решения задачи о воспроизведении условий развития ткани в искусственных условиях построим новую имитационную модель на базе демонстрационной модели из работы [6], а именно обобщим имитационную модель из [6] на случай асимметричного деления и миграции клеток, а также внешнего управляющего воздействия на скопление клеток.

1. Описание модели

В качестве основного механизма, управляющего активностью клеток, в [6] выбран обмен химическими сигналами. При этом эффекты от сигналов для упрощения были ограничены запуском программируемой смерти, ускорением или замедлением скорости наступления фазы деления, а также выбором типов для новых клеток. Для этого случая в работе [6] были выбраны следующие параметры:

— $T = \{\tau_1, \ldots, \tau_n\}$ — типы клеток;
— $I_S(\tau_1), \ldots, I_S(\tau_n)$ — интенсивности синтеза сигналов;
— $Pe(\tau_1, \tau_1), \ldots, Pe(\tau_n, \tau_n)$ — вероятности передачи сигнального воздействия внутрь клетки при столкновении с сигналом ($0 \leqslant Pe \leqslant 1$);
— $Pr(\tau_1, \tau_1), \ldots, Pr(\tau_n, \tau_n)$ — приоритеты сигналов ($Pr \geqslant 0$);
— $D(\tau_1, \tau_1), \ldots, D(\tau_n, \tau_n)$ — новые типы для дочерних клеток;
— $K(\tau_1, \tau_1), \ldots, K(\tau_n, \tau_n)$ — элементарные приращения скорости наступления деления или программируемой смерти;
— $V(\tau_1), \ldots, V(\tau_n)$ — объёмы клеток ($V_{\max} = \max\limits_{\tau} V(\tau)$).

Пусть клетка типа τ способна создавать k видов различных химических сигнальных веществ $S_1(\tau), \ldots, S_k(\tau)$. Для определённости положим, что в единицу времени клетка выделяет в среднем $I_j(\tau)$ химических веществ $S_j(\tau)$, где $j = 1, \ldots, k$. Тогда в качестве параметра интенсивности $I_S(\tau)$, согласно [6], будет выбрано $I_S(\tau) = \min\limits_{j} I_j(\tau)$. В результате усреднённые сигналы S в моделях из [6] однозначно описываются типом

клетки τ, которая этот сигнал создаёт ($S = S(\tau)$):

$$S(\tau) = \left(\frac{I_1(\tau)}{I_S(\tau)} S_1(\tau), \ldots, \frac{I_k(\tau)}{I_S(\tau)} S_k(\tau) \right). \tag{1}$$

В данной работе также будем использовать усреднённые сигналы $S(\tau)$ и набор параметров из [6]. Для искомой имитационной модели требуется дополнительно описывать асимметричное деление и миграцию клеток. Поэтому, учитывая, что пространство и время у имитационной модели дискретны, получаем, что базовый набор параметров из [6] необходимо дополнить следующими величинами:

- a — ребро куба идеализированной клетки;
- L, H, W — размеры пространства в элементарных кубах;
- Δt — время, для которого рассчитываются радиусы миграции, и одновременно элементарный такт времени в модели;
- $O(\tau_1), \ldots, O(\tau_n)$ — тип деления. При $O(\tau) = 1$ деление асимметричное и клетка сохраняет свой тип после деления с образованием дочерней клетки типа τ_{new}. При $O(\tau) = 0$ деление симметричное и клетка после него меняет свой тип на τ_{new};
- $K_M(\tau_1, \tau_1), \ldots, K_M(\tau_n, \tau_n)$ — элементарные приращения радиуса r_c свободной миграции для клеток под действием сигналов от других клеток;
- $r_C^{\max}(\tau_1), \ldots, r_C^{\max}(\tau_n)$ — максимальные радиусы миграции для клеток, которые достижимы за время Δt; потребуем, чтобы выполнялось $r_C(\tau) \leqslant a$ для всех τ;
- $v_d^{\max}(\tau_1), \ldots, v_d^{\max}(\tau_n)$ — максимальные интенсивности деления/смерти для клеток. Для определённости потребуем $v_d^{\max}(\tau) \leqslant 1$ для всех типов τ.

Замечание 1. Дополнительно к этому введём в множество типов клеток T специальный тип τ_0, который будет отвечать за моделирование мёртвых клеток. Для τ_0 условимся считать объём $V(\tau_0) = a^3$ и $Pe(\tau_0, \tau) = 0$ для любых $\tau \in T$, а все остальные параметры $I_S, Pr, K, \ldots$ для τ_0 определять не нужно, так как мёртвые клетки не делятся и не синтезируют сигналов.

Переменными обобщённой имитационной модели являются:

1) $Injection_{(z,x,y)}(t)$ — семейство сигналов, которые поступают извне модели в момент времени t в отдельную элементарную ячейку (z, x, y) пространства;
2) $\text{Cell}_{(z,x,y)}(t) = (\tau, v_d, r_c, \varphi_d, \varphi_m, \varphi_s, \text{Sin}, \text{Sig}, \text{Sout}, \text{Plan})$ — идеализированная клетка, которая расположена в ячейке (z, x, y) в момент времени t, где
   - τ — тип клетки (включая τ_0 для мёртвых клеток);
   - v_d — интенсивность наступления деления/смерти;
   - r_c — радиус свободной миграции клетки;
   - φ_d — фаза деления/смерти;
   - φ_m — фаза миграции;
   - φ_s — фаза синтеза сигналов;
   - Sin — семейство сигналов из ближайшего окружения клетки;
   - Sig — семейство сигналов, которые были приняты клеткой;
   - Sout — семейство сигналов, ожидающих ретрансляции;
   - Plan $= (\tau_{\text{new}}, Pr_{\text{new}})$ — план деления, который задаёт, какой тип (τ_{new}) получат дочерние клетки после деления и насколько приоритетным (Pr_{new}) будет такое деление.

При таком подходе *пространство* модели является дискретным и его можно однозначно описать с помощью следующего отображения:

$$\text{Cell}(t) : \{1, \ldots, H\} \times \{1, \ldots, L\} \times \{1, \ldots, W\} \longrightarrow C \cup \{\varnothing\},$$

где C — это множество всех клеток. Отображение $\mathrm{Cell}(t)$ ставит в однозначное соответствие тройке дискретных координат (z, x, y), $z = 1, \ldots, H$, $x = 1, \ldots, L$, $y = 1, \ldots, W$, содержимое ячейки в момент времени t; для пустой ячейки $\mathrm{Cell}_{(z,x,y)}(t) = \varnothing$ (см. пример на рис. 1).

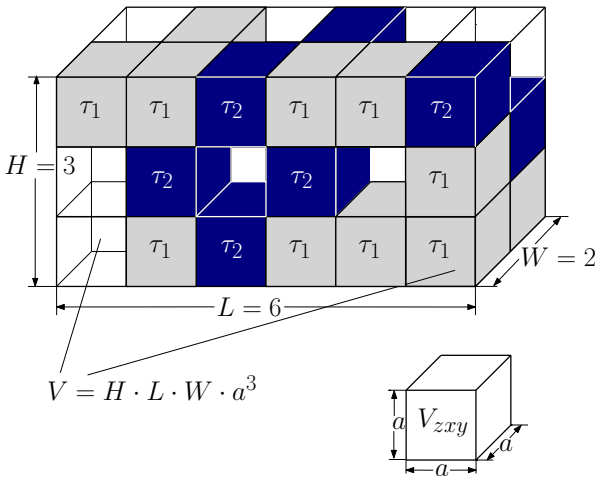


Рис. 1. Пример пространства модели для случая $L = 6$, $W = 2$, $H = 3$

Для упрощения работы с дискретным пространством кубов введём специальные обозначения:

— $d \in \mathrm{Directions} = \{\uparrow, \downarrow, \rightarrow, \leftarrow, \odot, \otimes\}$ — направления;
— $\overline{d} \in \mathrm{Directions}$ — противоположное направление для d;
— $d(z, x, y)$ — соседний куб с (z, x, y) по направлению d;
— $\mathrm{Dir}_{\mathrm{sig}}(z, x, y)(t)$ — допустимые направления для передачи сигнала. В множество $\mathrm{Dir}_{\mathrm{sig}}(z, x, y)$ входят все направления d, по которым с клеткой (z, x, y) соседствует любая непустая ячейка $d(z, x, y)$; ячейки с мёртвыми клетками τ_0 также считаются заполненными;
— $\mathrm{Dir}_{\mathrm{div}}(z, x, y)(t)$ — допустимые направления для деления клетки. В это множество входят все направления d, по которым с клеткой (z, x, y) соседствуют либо пустые, либо заполненные мёртвыми клетками ячейки $d(z, x, y)$.

Динамику обобщённой модели будем описывать с помощью набора правил для изменения содержимого $\mathrm{Cell}_{(z,x,y)}(t) = (\tau, v_d, r_c, \varphi_d, \varphi_m, \varphi_s, \mathrm{Sin}, \mathrm{Sig}, \mathrm{Sout}, \mathrm{Plan})$. Выделим семь этапов, которые будут исполняться последовательно для вычисления нового состояния модели для момента времени $t + 1$.

Э т а п 1. Приём сигналов.

Для всех сигналов из входных семейств $S \in \mathrm{Sin}(t) \cup Injection_{(z,x,y)}(t)$ разыгрывается случайное событие приёма сигнала. С вероятностью $Pe(\tau, \tau_S)$ сигнал $S = (\tau_S)$ будет перемещён в семейство $\mathrm{Sig}(t)$, и с вероятностью $1 - Pe(\tau, \tau_S)$ сигнал считается прошедшим мимо клетки и попадает в семейство $\mathrm{Sout}(t)$.

Э т а п 2. Корректировка интенсивности деления/смерти v_d.

Сначала рассчитывается суммарный приоритет всех поступивших сигналов:

$$SumInf(t) = \sum_{S \in \mathrm{Sig}(t)} Pr(\tau, \tau_S). \tag{2}$$

Приращение интенсивности v_d вычисляется с учётом насыщения при стремлении v_d к максимальной интенсивности деления/смерти $v_d^{\max}(\tau)$:

$$\begin{aligned}\Delta v_d(t) = \left(1 - \frac{v_d(t)}{v_d^{\max}(\tau)}\right) \sum_{\substack{S\in \mathrm{Sig}(t):\\ K(\tau,\tau_S)\geqslant 0}} \frac{Pr(\tau,\tau_S)}{SumInf(t)}\, K(\tau,\tau_S)\,\Delta t\,\frac{V(\tau)}{a^3} + \\ + \left(\frac{v_d(t)}{v_d^{\max}(\tau)}\right) \sum_{\substack{S\in \mathrm{Sig}(t):\\ K(\tau,\tau_S)<0}} \frac{Pr(\tau,\tau_S)}{SumInf(t)}\, K(\tau,\tau_S)\,\Delta t\,\frac{V(\tau)}{a^3}.\end{aligned} \tag{3}$$

Корректировка интенсивности v_d осуществляется с учётом того, что v_d не должно становиться отрицательным, а также большим, чем $v_d^{\max}(\tau)$:

$$v_d(t+1) = \begin{cases} v_d^{\max}(\tau), & \text{если} \quad v_d(t) + \Delta v_d(t) \geqslant v_d^{\max}(\tau), \\ v_d(t) + \Delta v_d(t), & \text{если} \quad 0 \leqslant v_d(t) + \Delta v_d(t) < v_d^{\max}(\tau), \\ 0, & \text{если} \quad v_d(t) + \Delta v_d(t) < 0. \end{cases} \tag{4}$$

Э т а п 3. Корректировка радиуса r_c.

Потенциальное изменение радиуса r_c можно посчитать по формуле, которая полностью аналогична формуле для приращения v_d:

$$\begin{aligned}\Delta r_c(t) = \left(1 - \frac{r_c(t)}{r_C^{\max}(\tau)}\right) \sum_{\substack{S\in \mathrm{Sig}(t):\\ K_M(\tau,\tau_S)\geqslant 0}} \frac{Pr(\tau,\tau_S)}{SumInf(t)}\, K_M(\tau,\tau_S)\,\Delta t\,\frac{V(\tau)}{a^3} + \\ + \left(\frac{r_c(t)}{r_C^{\max}(\tau)}\right) \sum_{\substack{S\in \mathrm{Sig}(t):\\ K_M(\tau,\tau_S)<0}} \frac{Pr(\tau,\tau_S)}{SumInf(t)}\, K_M(\tau,\tau_S)\,\Delta t\,\frac{V(\tau)}{a^3}.\end{aligned} \tag{5}$$

Изменение радиуса r_c осуществляется с учётом того, что радиус r_c не должен становиться отрицательным, а также большим, чем $r_C^{\max}(\tau)$:

$$r_c(t+1) = \begin{cases} r_C^{\max}(\tau), & \text{если} \quad r_c(t) + \Delta r_c(t) \geqslant r_C^{\max}(\tau), \\ r_c(t) + \Delta r_c(t), & \text{если} \quad 0 \leqslant r_c(t) + \Delta r_c(t) < r_C^{\max}(\tau), \\ 0, & \text{если} \quad r_c(t) + \Delta r_c(t) < 0. \end{cases} \tag{6}$$

Э т а п 4. Корректировка плана деления.

Для всех возможных новых типов τ_d вычисляется суммарный приоритет (в том числе и для $\tau_d = \tau_0$, который моделирует стандартную программируемую смерть, а также $\tau_d = \varnothing$, который моделирует смерть с поглощением остатков клетки):

$$\mathrm{Sum}_{Pr}(\tau_d, t) = \sum_{\substack{S\in \mathrm{Sig}(t),\\ D(\tau,\tau_S)=\tau_d}} Pr(\tau,\tau_S). \tag{7}$$

Осуществляется поиск типа $\overline{\tau_d}$, для которого суммарный приоритет максимальный:

$$\mathrm{Max}_{Pr}(t) = \max_{\tau_d}\left(\mathrm{Sum}_{Pr}(\tau_d,t)\right), \quad \overline{\tau_d}(t) : \mathrm{Sum}_{Pr}(\overline{\tau_d},t) = \mathrm{Max}_{Pr}(t). \tag{8}$$

В случае, если максимальный приоритет $\mathrm{Max}_{Pr}(t)$ превосходит $Pr_{\mathrm{new}}(t)$, то происходит обновление плана деления:

$$\begin{cases} \tau_{\mathrm{new}}(t+1) = \overline{\tau_d}(t), \\ Pr_{new}(t+1) = \mathrm{Max}_{Pr}(t), \end{cases} \quad \text{если } \mathrm{Max}_{Pr}(t) > Pr_{\mathrm{new}}(t). \tag{9}$$

В противном же случае сохраняется старый план деления $\mathrm{Plan}(t+1) = \mathrm{Plan}(t)$. При этом переход к новому этапу для отдельной клетки $\mathrm{Cell}_{(z,x,y)}(t)$ возможен только в том случае, если для всех клеток в модели закончена работа по реализации этапа 4.

Э т а п 5. Синтез и передача сигналов.

Добавляем в семейство $\mathrm{Sout}(t)$ сигналы $S(\tau)$, количество которых задаётся параметром $I = I_S(\tau) \cdot \Delta t \cdot a^3 / V(\tau)$. Пусть для определённости $I = m + k/n$, где $m, k, n \in \mathbb{N}$. Тогда количество сигналов $S(\tau)$ может быть вычислено по формуле

$$\overline{m}(t) = m + \sigma(\varphi_s(t) - 1 + k/n),$$

где σ — функция Хевисайда. Далее для каждого сигнала из семейства $\mathrm{Sout}(t)$ разыгрывается случайное событие передачи: с равными вероятностями сигнал передаётся любой из клеток ближайшего окружения и помещается в семейство Sin этой клетки (ближайшим окружением клетки $\mathrm{Cell}_{(z,x,y)}$ будем считать саму клетку $\mathrm{Cell}_{(z,x,y)}$, а также все клетки из множества $\mathrm{Dir}_{\mathrm{sig}}(z,x,y)(t)$). После завершения передачи всех сигналов пересчитываем фазу синтеза сигналов по формуле

$$\varphi_s(t+1) = \begin{cases} \varphi_s(t) + k/n, & \text{если } \varphi_s(t) + k/n < 1, \\ \varphi_s(t) - 1 + k/n, & \text{если } \varphi_s(t) + k/n \geqslant 1. \end{cases} \tag{10}$$

Переход к следующему этапу для отдельной клетки $\mathrm{Cell}_{(z,x,y)}(t)$ возможен только в том случае, если для всех клеток в модели закончена работа по реализации этапа 5.

Э т а п 6. Миграция.

Перед началом данного этапа все клетки упорядочиваются в случайную последовательность $\alpha = (\mathrm{Cell}_1, \mathrm{Cell}_2, \ldots)$. Реализация этапа миграции проводится по очереди для всех клеток в соответствии с их номерами в случайной последовательности α. В случае $(\varphi_m(t) + r_c(t)/a) \geqslant 1$ разыгрывается случайное событие миграции — обмен значений $\mathrm{Cell}_{(z,x,y)}(t)$ с произвольной соседней клеткой из $\mathrm{Dir}_{\mathrm{sig}}(z,x,y)(t)$, которая пока не подвергалась миграции на текущем этапе. При этом обмену не подвержены семейства сигналов Sin, которые считаются привязанными к ячейке пространства (z,x,y). Это обусловлено тем фактом, что для описания миграции сигналов в модели используется этап 5. Можно считать, что два других семейства сигналов Sig, Sout также не подвержены обмену, так как они будут пустыми для всех клеток на начало этапа миграции.

Когда обмен завершён, то обе переставленные клетки считаются мигрировавшими на данный момент времени. Отметим, что если клетка окружена уже мигрировавшими клетками, то она не меняет своего положения. После завершения этапа осуществляем пересчёт фазы миграции φ_m по правилу

$$\varphi_m(t+1) = \begin{cases} \varphi_m(t) + \dfrac{r_c(t)}{a}, & \text{если } \varphi_m(t) + \dfrac{r_c(t)}{a} < 1, \\ \varphi_m(t) - 1 + \dfrac{r_c(t)}{a}, & \text{если } \varphi_m(t) + \dfrac{r_c(t)}{a} \geqslant 1. \end{cases} \tag{11}$$

При этом переход к новому этапу для отдельной клетки $\mathrm{Cell}_{(z,x,y)}(t)$ возможен только в том случае, если для всех клеток в модели закончена работа по реализации этапа 6.

Э т а п 7. Деление/программируемая смерть.

Перед началом данного этапа все клетки упорядочиваются в случайную последовательность $\beta = (\mathrm{Cell}_1, \mathrm{Cell}_2, \ldots)$. Реализация этапа деления проводится по очереди для всех клеток в соответствии с их номерами в последовательности β.

В случае, если запланировано деление ($\tau_{\text{new}} \neq \varnothing$ и $\tau_{\text{new}} \neq \tau_0$), то клетка создаёт новую клетку в одной из соседних ячеек по направлениям $\text{Dir}_{\text{div}}(z, x, y)(t)$. (Под пустой соседней ячейкой подразумевается такая ячейка окружения, в которую не разделились клетки, предшествующие Cell в последовательности β.) Дополнительно к этому, если деление симметричное ($O(\tau) = 0$), то клетка меняет тип на τ_{new}.

1) Если в качестве соседней ячейки выбрана мёртвая клетка τ_0, то все сигналы, которые были в этой ячейке, переносятся в семейство Sin новой клетки.
2) Если в окружении клетки отсутствуют свободные ячейки, то для деления выбирается такое направление d, вдоль которого минимально расстояние ρ от делящейся клетки до пустой ячейки либо ячейки с клетками типа τ_0. Данное направление должно рассчитываться с учётом новых клеток, которые создали предшествующие в последовательности β клетки.
3) Если такое направление найти удалось, то клетка делится в этом направлении, сдвигая параллельным переносом клетки по направлению d на ρ с замещением граничной пустой ячейки (либо граничной ячейки с клетками τ_0).

При этом во время сдвига клеток обмену не подлежат семейства сигналов Sin, которые считаются привязанными к своим ячейкам пространства. Если два и больше направления являются минимальными, то среди них выбирается произвольное. Если по всем потенциальным направлениям деления клетки плотно упираются в границу модели, то фаза деления принудительно обнуляется.

Пусть для определённости $v_d(\tau) = k/n$, где $k, n \in \mathbb{N}$. Тогда количество актов деления можно вычислить как $\overline{m}(t) = \sigma(\varphi_d(t) - 1 + k/n)$. Для новых клеток используются параметры $(\tau_{\text{new}}, v_d, 0, 0, 0, 0, \varnothing, \varnothing, \varnothing, (\tau_{\text{new}}, 0))$, где v_d берётся от материнской клетки. После завершения этапа деления осуществляем пересчёт фазы деления по формуле

$$\varphi_d(t+1) = \begin{cases} \varphi_d(t) + k/n, & \text{если } \varphi_d(t) + k/n < 1, \\ \varphi_d(t) - 1 + k/n, & \text{если } \varphi_d(t) + k/n \geqslant 1. \end{cases} \tag{12}$$

В случае, если клетка была запрограммирована на апоптоз ($\tau_{\text{new}} = \tau_0$), то её ячейка получает этот тип после конца фазы. Если же клетка была настроена на смерть с поглощением ($\tau_{\text{new}} = \varnothing$), то её ячейка считается пустой, начиная со следующей итерации.

Переход от этапа 7 к новому отсчёту времени $t + 1$ и соответственно к этапу 1 проводится строго синхронно для всех клеток $\text{Cell}_{(z,x,y)}(t)$ в скоплении.

2. Специфика практического применения модели

Введение внешнего управляющего воздействия в рамках рассмотренной имитационной модели осуществляется за счёт усреднённого потока сигналов $Injection_S^{(z,x,y)}(t)$ к отдельной идеализированной клетке (z, x, y). На практике вместо идеализированных сигналов $S(\tau)$ в скопление клеток нужно будет вводить реальные химические вещества $S_1(\tau), \ldots, S_k(\tau)$ в пропорции, которая задаётся формулой (1).

Замечание 2. Реализовать подобный поток сигналов $S(\tau)$ для внутренних клеток скопления будет достаточно сложно, поскольку потребует введения шприцев глубоко внутрь скопления, что потенциально может привести к повреждению клеток. Поэтому основной вариант сводится к реализации потока сигналов $Injection_S^{(z,x,y)}(t)$ через границу всего пространства модели: $z = 1 \vee z = H; x = 1 \vee x = L; y = 1 \vee y = W$.

Найти искомые потоки сигналов $Injection_S^{(z,x,y)}(t)$ для исследуемого скопления клеток $\mathcal{C}_o$ можно, если предварительно формализовать модель для более крупного скопления $\mathcal{C}$, которое содержит в себе скопление $\mathcal{C}_o$ (см. рис. 2). Затем для более крупного

скопления $\mathcal{C}$ по ходу эмуляции динамики клеток вычислить потоки сигналов через границу $\mathcal{C}_o$ как функции от времени t и потом применить их в качестве начальных приближений для потоков сигналов $Injection_S^{(z,x,y)}(t)$.

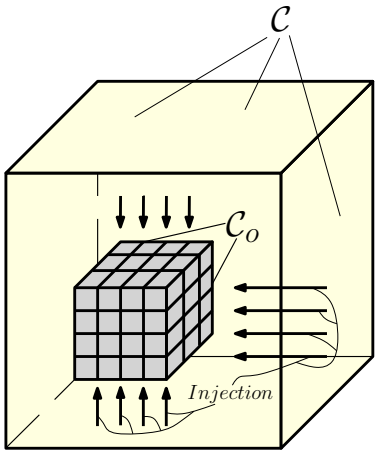


Рис. 2. Выделение потоков сигналов внутри $\mathcal{C}$ через границу меньшего скопления $\mathcal{C}_o$

Замечание 3. Если на практике ставится задача поиска $Injection_S^{(z,x,y)}(t)$ — оптимальных потоков для некоторого скопления клеток, то совсем не обязательно решать её сразу с помощью обобщённой имитационной модели (2)–(12). Более эффективным представляется сначала формализовать базовые модели из работы [6], а затем найти с их помощью первые приближения для универсальных параметров и экспортировать их значения в обобщённую имитационную модель.

В [6] рассмотрены общие алгоритмы для оценки значений универсальных параметров $Pe, Pr, D, K, \ldots$ на основе данных о внутриклеточной биохимии, а также о строении мембран клеток. Однако одним из главных ограничивающих факторов для данной процедуры является определение множества типов клеток T. Так, если для поиска множества T выбрать в качестве основного критерия различие форм клеток, то это может привести к объединению большого количества похожих, но функционально различных типов клеток в крупные фиктивные типы.

Таким образом, для получения начальных приближений для типов клеток T и универсальных параметров $Pe, D, \ldots$ при работе с реальными многоклеточными организмами потребуется прибегнуть к помощи модели внутриклеточной биохимии.

На данный момент построение модели внутриклеточной биохимии для поиска множества типов T и оценки универсальных параметров является актуальной задачей. Поэтому для демонстрации работы модели (2)–(12) прибегнем к абстрактным клеточным скоплениям, лишённым непосредственного биологического прототипа.

Рассмотрим работу алгоритма, который предложен в замечании 3, на примере абстрактного скопления клеток четырёх типов $T = \{\tau_0, \tau_1, \tau_2, \tau_3\}$, где τ_0 — это мёртвые клетки. Пусть на рис. 3 представлена динамика развития данного скопления для некоторых тестовых отсчётов времени $t = 0, 1, 2, 4, 8$.

Элементарный такт времени в модели положим равным $\Delta t = 0{,}01$. Остальные базовые параметры H, L, W, a можно рассчитать на основе рис. 3. В частности, ребро элементарного куба $a = \sqrt[3]{V(\tau)} = 0{,}01$, а размеры пространства $L = W = 101$ и $H = 1$, так как скопление задано на плоскости.

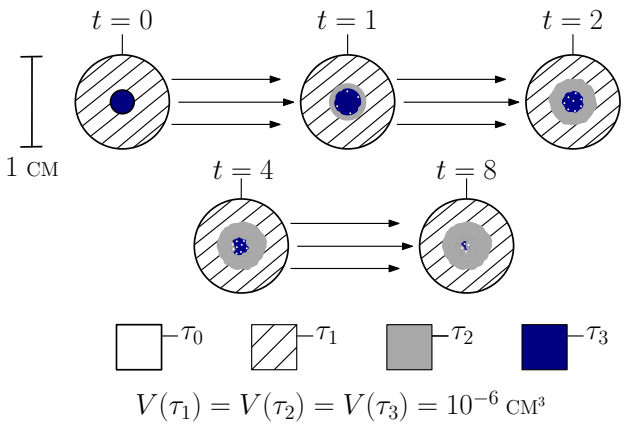


Рис. 3. Развитие абстрактного скопления клеток $T = \{\tau_0, \tau_1, \tau_2, \tau_3\}$

Первичная задача состоит в том, чтобы найти значения основных параметров модели $I_S, Pr, Pe, D, K, O, K_M, r_C^{\max}, v_d^{\max}$ для описания динамики скопления на рис. 3. Часть из этих параметров можно оценить с помощью базовой количественной модели из работы [6], которая описывает усреднённую динамику численностей клеток $N(\tau)$ с помощью дифференциальных уравнений. Для того чтобы воспользоваться количественной моделью, осуществим преобразование данных о конфигурациях клеток с учётом масштаба и объёмов $V(\tau)$ в графики для численностей $N(\tau)$ (рис. 4).

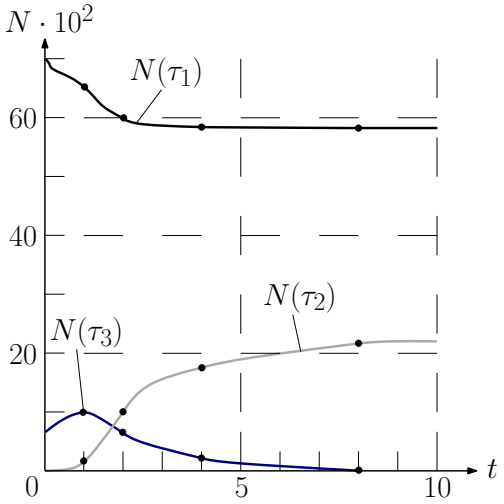


Рис. 4. Зависимости численностей клеток $N(\tau_1), N(\tau_2), N(\tau_3)$ от времени

После того как установлены зависимости $N(\tau, t)$, можно найти первое приближение для параметров I_S, Pr, Pe, D, K с помощью количественной модели из [6]. Затем итоговые значения параметров находятся уже с привлечением имитационной модели (2)–(12). Полученные для данного примера значения приведены на рис. 5.

Остальные значения параметров модели сохранили значения по умолчанию при формализации: $O(\tau) = 1$, $K_M(\tau, \tau^*) = 0$ и $r_C^{\max}(\tau) = 0$ для всех типов $\tau, \tau^* \in T$. После того как параметры модели (2)–(12) найдены, можно воспроизвести развитие исследуемого скопления в ограниченном объёме и с меньшим числом клеток. Для этого по ходу реализации динамики модели для бóльшего скопления на рис. 3 регистрируем потоки сигналов через интересующую нас границу. Затем данные потоки используются

Pe	τ_1	τ_2	τ_3
τ_1	0.001	0.01	0.2
τ_2	0.01	0.001	0.01
τ_3	0.01	0.04	0.001

Pr	τ_1	τ_2	τ_3
τ_1	20	1	2
τ_2	2	30	2
τ_3	4	90	30

D	τ_1	τ_2	τ_3
τ_1	τ_2	τ_1	τ_0
τ_2	τ_3	τ_2	τ_2
τ_3	τ_3	τ_0	τ_3

K	τ_1	τ_2	τ_3
τ_1	-0.002	-0.84	0.55
τ_2	0.4	-1.8	1.3
τ_3	950	600	850

$$I_S(\tau_1) = I_S(\tau_2) = I_S(\tau_3) = 150$$
$$V(\tau_1) = V(\tau_2) = V(\tau_3) = 10^{-6}\ \text{см}^3$$
$$H = 1,\ L = W = 101,\ a = 10^{-2}$$
$$v_d^{\max}(\tau_1) = v_d^{\max}(\tau_2) = v_d^{\max}(\tau_3) = 1$$

Рис. 5. Значения универсальных параметров для примера абстрактного скопления клеток

для воспроизведения динамики клеток в замкнутом объёме за счёт введения этих сигналов через его границу (рис. 6).

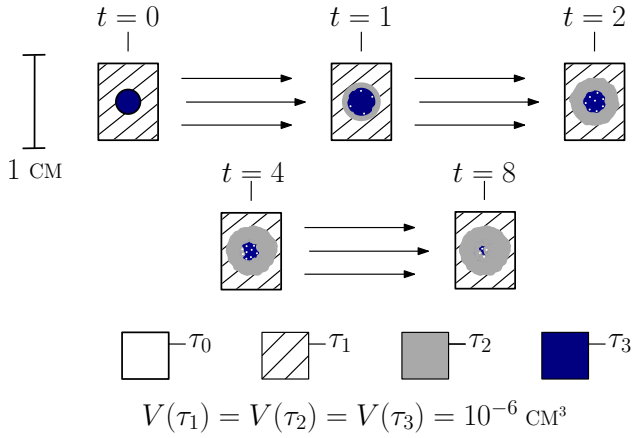


Рис. 6. Развитие абстрактного скопления клеток $T = \{\tau_0, \tau_1, \tau_2, \tau_3\}$

Заключение

Основная задача работы — построение модели, которая позволит воспроизводить условия развития ткани вне исходного организма. Данная задача решена для случая обмена химическими межклеточными сигналами, а искомое воспроизведение условий реализовано в имитационной модели в виде потока сигнальных молекул $Injection_S^{(z,x,y)}(t)$ через границу скоплений. Полученную имитационную модель можно потенциально использовать на практике для производства органов и тканей с помощью клеточной печати. Фактически алгоритм воспроизведения нужной ткани сведётся к выполнению четырёх шагов:

1) экспериментальное установление распределения слабо специализированных клеток на начальный момент развития ткани;
2) вычисление с помощью имитационной модели (2)–(12) потоков $Injection_S^{(z,x,y)}(t)$, которые необходимы для формирования требуемого сегмента ткани;
3) распечатка начальной заготовки ткани из стволовых клеток на 3D-принтере;
4) воспроизведение во времени потоков $Injection_S^{(z,x,y)}(t)$ через границу скопления.

Нужно отметить, что построенная имитационная модель (2)–(12) имеет существенные ограничения для применения на практике. Как уже было отмечено, для облегче-

ния поиска первых приближений значений универсальных параметров необходимо построить модель внутриклеточной биохимии. Искомая биохимическая модель должна иметь возможность оценить, как воздействие химических сигналов будет проявляться на макроуровне: ускоряется или замедляется наступление деления/смерти, происходит ли принудительный запуск программируемой смерти и будет ли влияние оказано на выбор новых типов для дочерних клеток. Дополнительно к этому требуется реализовать алгоритм распознавания типов клеток на трёхмерных томограммах. Без этих инструментов использование имитационной модели (2)–(12) для описания реальных скоплений клеток будет существенно затруднено.

Для повышения предсказательных возможностей построенной имитационной модели (2)–(12) можно предложить как минимум три варианта её обобщения.

В первую очередь, можно получить более точную модель, если отказаться от усреднённых сигналов $S(\tau)$ и ввести реальные химические сигналы в формализм модели.

Второй вариант обобщения сводится к более точному описанию миграции клеток, в частности, к учёту вынужденной миграции клеток и хемотаксиса (движения клеток вдоль градиента концентрации определённого вещества) [7].

Третье обобщение заключается в моделировании альтернативных путей передачи данных между клетками. Например, можно учесть два типа механических напряжений между клетками: пассивные и активные напряжения. При этом под пассивным напряжением обычно подразумевается межклеточное давление и растяжение ткани под действием гравитации, а под активным — растяжение ткани в результате роста, деления и миграции клеток (подробнее о роли этих напряжений см. в [8]).

Автор выражает благодарность Н. В. Суворовой и А. В. Решетникову за критические замечания и помощь в корректировке статьи.

ЛИТЕРАТУРА

1. *Xu F., Sridharan B., et al.* Embryonic stem cell bioprinting for uniform and controlled size embryoid body formation // Biomicrofluidics. 2011. V. 5. No. 2. P. 022207.
2. *Белоусов Л. В.* Основы общей эмбриологии. М.: Изд-во Моск. ун-та, 2005. 368 с.
3. *Корочкин Л. И.* Биология индивидуального развития (генетический аспект). М.: Изд-во Моск. ун-та, 2003. 400 с.
4. *Быков В. Л.* Цитология и общая гистология (функциональная морфология клеток и тканей человека). СПб.: СОТИС, 2002. 520 с.
5. *Baker M.* Tissue models: A living system on a chip // Nature. 2011. V. 471. No. 7340. P. 661–665.
6. *Назаров М. Н.* О поиске универсальных параметров для моделей морфогенеза // Владикавказский математический журнал. 2013. Т. 15. № 3. С. 41–49.
7. *Bagorda A. and Parent C. A.* Eukaryotic chemotaxis at a glance // J. Cell Sci. 2008. V. 121. P. 2621–2624.
8. *Belousov L. V.* Morphomechanics: goals, basic experiments and models // Int. J. Dev. Biol. 2006. V. 50. No. 2. P. 81–92.

СВЕДЕНИЯ ОБ АВТОРАХ

АБОРНЕВ Александр Викторович — ООО «Центр сертификационных исследований», г. Москва. E-mail: **abconf.c@gmail.com**

БЫКОВА Валентина Владимировна — доцент, доктор физико-математических наук, профессор Института математики и фундаментальной информатики Сибирского федерального университета, г. Красноярск. E-mail: **bykvalen@mail.ru**

ВЕЛИЧКО Игорь Георгиевич — доцент, кандидат физико-математических наук, профессор кафедры высшей математики Запорожского национального технического университета, г. Запорожье. E-mail: **wig64@mail.ru**

ГАВРИКОВ Александр Владимирович — аспирант Саратовского государственного университета, г. Саратов. E-mail: **alexandergavrikov1989@gmail.com**

ГОЦУЛЕНКО Владимир Владимирович — старший науный сотрудник, кандидат технических наук, старший научный сотрудник отдела теплофизических основ энергосберегающих теплотехнологий Института технической теплофизики НАН Украины, г. Киев. E-mail: **gosul@ukr.net**

ДЕВЯНИН Петр Николаевич — доктор технических наук, доцент, председатель УМС Учебно-методического объединения по образованию в области информационной безопасности, г. Москва. E-mail: **peter_devyanin@hotmail.com**

ЗИНЧЕНКО Андрей Иванович — старший преподаватель кафедры информационных технологий Запорожского национального университета, г. Запорожье.
E-mail: **andriver@znu.edu.ua**

КАРПОВ Артем Валерьевич — аспирант кафедры защиты информации и криптографии Национального исследовательского Томского государственного университета, г. Томск. E-mail: **karpov@isc.tsu.ru**

КОСТЮК Юрий Леонидович — профессор, доктор технических наук, Национальный исследовательский Томский государственный университет, г. Томск.
E-mail: **kostyuk_y_l@sibmail.com**

НАЗАРОВ Максим Николаевич — ассистент кафедры ВМ-1 Национального исследовательского университета «МИЭТ», г. Москва.
E-mail: **Nazarov-Maximilian@yandex.ru**

РЫЖОВ Александр Сергеевич — сотрудник Лаборатории ТВП, г. Москва.
E-mail: **ryzhovalexander@mail.ru**

ШАНГИН Роман Эдуардович — аспирант Южно-Уральского государственного университета, г. Челябинск. E-mail: **shanginre@gmail.com**

АННОТАЦИИ СТАТЕЙ НА АНГЛИЙСКОМ ЯЗЫКЕ

Abornev A. V. **PERMUTATIONS INDUCED BY DIGIT-INJECTIVE TRANSFORMATIONS OF A MODULE OVER A GALOIS RING.** New classes of nonlinear permutations that can be represented by linear transformations of a module over a Galois ring are constructed. Some applications in cryptography are discussed.
Keywords: *digit-injective matrix, DI-matrix, permutation, Galois ring.*

Karpov A. V. **PERMUTATION POLYNOMIALS OVER RESIDUE CLASS RINGS.** Problems of finding inverse for a permutation polynomial over the ring $\mathbb{Z}_{p^k}$ for prime p and any $k > 1$ are studied. Necessary and sufficient conditions for two permutation polynomials to be inverse polynomials modulo prime power are found. Given a known inverse polynomial modulo p^2, a formula for inverse polynomial modulo p^k is pointed. Given a pair of inverse polynomials modulo p^k, a method for constructing other such pairs is proposed.
Keywords: *permutation polynomials, residue class rings, polynomial permutations.*

Devyanin P. N. **SYSTEM ADMINISTRATION IN MROSL DP-MODEL.** The de-jure administration rules are considered in mandatory entity-role DP-model for security of access and information flows control in OS of Linux set (MROSL DP-model). These rules are intended for administrating registration records of users, access permissions, sets of roles and administration roles, hierarchies of roles and administration roles, mandatory and integrity labels of entities, subjects (sessions), roles and administration roles. They allow to define detailed specifications of access control functions in secure OS Astra Linux Special Edition which, in turn, makes it possible to verify MROSL DP-model realization in software.
Keywords: *mandatory entity-role DP-model, OS Linux.*

Velichko I. G., Zinchenko A. I. **V-GRAPHS AND THEIR RELATION TO THE PROBLEM OF LOCATING OBJECTS IN A PLANE.** For two congruent figures with no common interior points, the locations in a plane are studied. A line being parallel to a shift vector intersects these pieces in two identical systems of intervals shifted by this vector. An oriented V_n-graph is constructed, its vertices correspond to the topologically different variants of relative position of two systems of n intervals, and the edges correspond to the allowable transitions between vertices. The term of W_n-graph is introduced as a minimal transitive graph which contains V_n-graph augmented with an incident vertex. The properties of V_n-graphs and W_n-graphs are proved.
Keywords: *placement of figures in a plane oriented graph, W-graph, the Catalan numbers, Dyck path, system slots, congruent figures.*

Gavrikov A. V. **T-IRREDUCIBLE EXTENSION OF UNIONS OF SOME TYPES ORGRAPHS.** Algorithms constructing T-irreducible extensions (TIE) for unions of paths, for the union of an oriented graph and its TIE, and for directed stars are proposed.
Keywords: *T-irreducible extension, TIE, minimal T-irreducible extension, union of some types digraphs, union of paths, TIE for directed stars.*

Bykova V. V. **ON THE ASYMPTOTIC SOLUTION OF A SPECIAL TYPE RECURRENCE RELATIONS AND THE KULLMANN — LUCKHARDT'S TECHNOLOGY.** The problem of solving recurrence relations that arise in the analysis of the computational complexity of recursive algorithms is discussed. The research is only related to splitting algorithms, namely, DPLL-algorithms, which are named after the author's names by the first letters (Davis, Putman, Logemann, Loveland), and are designed to solve the problem SAT. The technology by Kullmann — Luckhardt, which is traditionally used in the analysis of the computational complexity of splitting algorithms, is explored. A theorem containing an upper estimate for the algorithm time is proved in the case of balanced splitting. The theorem extends the capabilities of the Kullmann — Luckhardt's technology.

Keywords: *computational complexity of recursive algorithms, splitting algorithms, solving recurrence relations.*

Gotsulenko V. V. **COMBINATORIAL NUMBERS OF THE FINITE MULTISET PARTITIONS.** The problem of determining the number of different partitions of any finite multiset is considered. The generating functions for these combinatorial numbers are found and some of its properties are stated.

Keywords: *multiset partitions, Diophantine equations, generating functions.*

Kostyuk Yu. L. **THE TRAVELLING SALESMAN PROBLEM: IMPROVED LOWER BOUND IN THE BRANCH-AND-BOUND METHOD.** In a previous article, the author describes the implementation of Little's algorithm for solving the travelling salesman problem by the branch-and-bound method and gives a new modified algorithm where an additional transformation of the distance matrix is used at each step of the solution. This improves the lower bound for the exact solution and speeds up its work, but only in the case of non-symmetric matrices. In the present article, a method is described to further improve the lower bound, especially in the case of symmetric matrices. A new algorithm based on it is constructed. With the help of a computer simulation, some estimates of the constants in the formulas for the algorithm complexity are obtained for three types of distance matrices: 1) non-symmetric matrices with random distances, 2) matrices with the Euclidean distances between random points in the square, and 3) non-symmetric matrices with random distances satisfying the triangle inequality.

Keywords: *travelling salesman problem, branch-and-bound method, computing experiment.*

Ryzhov A. S. **IMPLEMENTING MAIN STEPS OF WIEDEMANN — COPPERSMITH ALGORITHM FOR BINARY SYSTEMS OF LINEAR EQUATIONS ON CLUSTERS.** This paper concerns implementation of the main steps of block Wiedemann's algorithm for solving systems of linear equations. Effective implementation of particular operations is provided. Specific problems of implementing the algorithm for running on clusters are studied.

Keywords: *systems of linear equations, Wiedemann — Coppersmith's algorithm.*

Shangin R. E. **EXACT ALGORITHM FOR SOLVING DISCRETE WEBER PROBLEM FOR A CYCLE.** A polynomial algorithm solving discrete Weber problem for cycle and for finite set of location positions is presented. The algorithm is based on the dynamic programming idea. The comparison of action times of the given algorithm and of an integer linear programming model, which was realized in IBM ILOG CPLEX, is carried out.

Keywords: *Weber problem, cycle, dynamic programming, exact algorithm.*

Nazarov M. N. **MODELLING THE TISSUE GROWTH WITH THE POSSIBILITY OF EXTERNAL INFLUENCE ON TISSUE SHAPE.** The imitation model of morphogenesis is constructed; its formalism includes the external influence on the shape of cell clusters in the form of the stream of signalling molecules across the border of these clusters. This model uses only the universal parameters, which are guaranteed to have biological meaning, as well as the possibility of the use in the models of morphogenesis based on differential equations.

Keywords: *multicellular organisms, morphogenesis, discrete models, controllable tissue formation.*

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*