# ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

**6(166) 2010**

**ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ**

**Издается с ноября 1995 г.**

**УЧРЕДИТЕЛЬ**

**Издательство "Новые технологии"**


## СОДЕРЖАНИЕ




Информация о журнале доступна по сети Internet по адресу http://www.**informika.ru/text/magaz/it/** или **http://novtex.ru/IT.**

Журнал включен в систему Российского индекса научного цитирования.

Журнал входит в Перечень научных журналов, в которых по рекомендации ВАК РФ должны быть опубликованы научные результаты диссертаций на соискание ученой степени доктора и кандидата наук.

УДК 621.396.6:621.391.827; 004.056:061.68

**З. М. Гизатуллин,** канд. техн. наук, доц.,
e-mail: gzm_zinnur@mail.ru,
**С. Ф. Чермошенцев,** д-р техн. наук, проф.,
Казанский государственный технический университет им. А. Н. Туполева

# Моделирование электромагнитных помех в неэкранированной витой паре при внешнем гармоническом электромагнитном воздействии*

*Предложены математические модели для анализа электромагнитных помех в неэкранированной витой паре при воздействии гармонического электромагнитного поля в дальней зоне. Приведены результаты моделирования, которые используются для решения задач анализа электромагнитной совместимости и защиты информации в электронных средствах.*

***Ключевые слова:*** *электромагнитная совместимость, защита информации, электромагнитные помехи, неэкранированная витая пара*

## Введение

Современные здания насыщены взаимосвязанными электронными средствами (ЭС) различного назначения. Основной объединяющей средой слаботочных подсистем ЭС является структурированная кабельная система. Основным принципом структурированной кабельной системы является интеграция вычислительных, телефонных и других коммуникационных сетей в едином кабельном пространстве, в результате которой все оборудование в здании объединяется в концепцию "Интеллектуального здания" [1]. Структурированная кабельная система создается в расчете на длительную перспективу и исключает необходимость прокладки дополнительных кабелей при изменении требований к системе коммуникаций, при подключении нового и перемещении существующего эксплуатируемого оборудования. При этом, с точки зрения информационной безопасности (отсутствия физического искажения, блокирования или уничтожения информации [2]), на практике доступ к оборудованию ничем не ограничен. По своим электрофизическим и геометрическим параметрам элементы структурированной кабельной системы зданий являются одними из наиболее опасных приемников и переносчиков электромагнитных помех в ЭС (рис. 1).

Современные структурированные кабельные системы допускают использование следующих типов кабелей:

- неэкранированная витая пара, где витую пару образуют изолированные проводники, попарно перевитые между собой минимально необходимое число раз на определенном отрезке длины;
- экранированная витая пара, где особенностью является наличие двойного экрана — общего для всего кабеля и индивидуального для каждой пары;
- фольгированная витая пара, где витые пары имеют общий экран из алюминиевой фольги;
- оптоволоконный кабель, где для передачи информации используются световые волны.

По некоторым данным, в России процент использования неэкранированной витой пары достигает 90 %. В целом же по всему миру доля кабелей с медной проводкой составляет 84 %, и среди них 82 % составляет неэкранированная витая пара [3].


**Рис. 1. Основные источники и приемники электромагнитных помех в здании**

*Работа выполнена по ФЦП "Научные и научно-педагогические кадры инновационной России" на 2009—2013 годы.

Цель данной работы — прогнозирование электромагнитных помех в неэкранированной витой паре при внешнем гармоническом электромагнитном воздействии в дальней зоне.

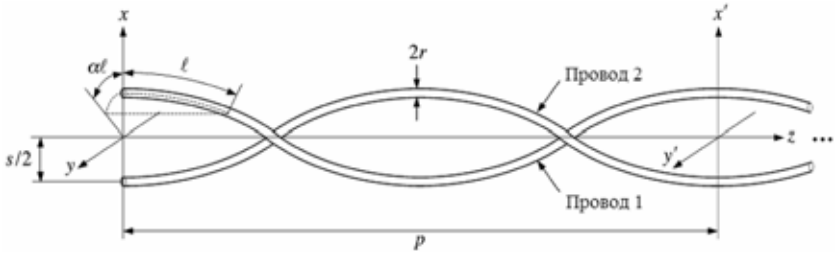



**Рис. 2. Конструкция неэкранированной витой пары**

### Математические модели для анализа электромагнитных помех в неэкранированной витой паре

Для анализа электромагнитных помех в неэкранированной витой паре ее конструкция может быть представлена периодической структурой в виде бифилярной спирали, скрученной относительно средней линии (рис. 2).

Координаты точек $a(l)$, $b(l)$ внутри проводников витой пары определяются следующим образом:

$$x_1(l) = \frac{s}{2}\cos(\alpha l);\ y_1(l) = \frac{s}{2}\sin(\alpha l);\ z_1(l) = \frac{p\alpha l}{2\pi};$$

$$x_2(l) = -\frac{s}{2}\cos(\alpha l);\ y_2(l) = -\frac{s}{2}\sin(\alpha l);\ z_2(l) = \frac{p\alpha l}{2\pi},$$

где $l$ — длина дуги; $s$ — расстояние между осями проводников витой пары (размер и материал проводов 1 и 2 одинаковы); $p$ — шаг скрутки проводников витой пары; $\alpha$ — коэффициент скрутки, который определяется следующим образом:

$$\alpha = \left(\sqrt{\left(\frac{s}{2}\right)^2 + \left(\frac{p}{2\pi}\right)^2}\right)^{-1}.$$

Единичные векторы $\boldsymbol{l_1}(l)$ и $\boldsymbol{l_2}(l)$ направлены по касательной к проводникам 1 и 2 соответственно в точках $a(l)$ и $b(l)$ (рис. 3) и определяются формулами

$$\boldsymbol{l_1}(l) = -\frac{s\alpha}{2}\sin(\alpha l)\boldsymbol{a_x} + \frac{s\alpha}{2}\cos(\alpha l)\boldsymbol{a_y} + \frac{\alpha p}{2\pi}\boldsymbol{a_z};$$

$$\boldsymbol{l_2}(l) = \frac{s\alpha}{2}\sin(\alpha l)\boldsymbol{a_x} - \frac{s\alpha}{2}\cos(\alpha l)\boldsymbol{a_y} + \frac{\alpha p}{2\pi}\boldsymbol{a_z},$$

где $\boldsymbol{a_x}$, $\boldsymbol{a_y}$, $\boldsymbol{a_z}$ — проекции единичных векторов $\boldsymbol{l_1}(l)$и $\boldsymbol{l_2}(l)$ на осях $x$, $y$, $z$ соответственно.

Для данного случая уравнения линии передачи при воздействии внешних неоднородных электромагнитных полей можно записать в следующем виде [4, 5]:

$$\frac{d}{dl}V(l) = -ZI(l) + V_F(l);\ Z = Z_i + j\omega L_e;$$

$$V_F(l) = -\frac{d}{dl}\int_{a(l)}^{b(l)} \boldsymbol{E^i}(p,\ l)\cdot d\boldsymbol{\rho} + \boldsymbol{E^i}(x_1(l),\ y_1(l),$$
$$z_1(l))\cdot \boldsymbol{l_1}(l)) - \boldsymbol{E^i}(x_2(l),\ y_2(l),\ z_2(l)\cdot \boldsymbol{l_2}(l));$$

$$\frac{d}{dl}I(l) = -YV(l) + I_F(l);\ Y = G + j\omega C;$$

$$I_F(l) = -Y\int_{a(l)}^{b(l)} \boldsymbol{E^i}(p,\ l)\cdot d\boldsymbol{\rho};\ d\boldsymbol{\rho} = \boldsymbol{t}(l)d\boldsymbol{\rho};$$

$$\boldsymbol{t}(l) = \cos(\alpha l)\boldsymbol{a_x} + \sin(\alpha l)\boldsymbol{a_y},$$

где $\boldsymbol{\rho}$ — цилиндрическая координата в радиальном направлении; $\boldsymbol{t}(l)$ — единичный вектор, направленный из точки $a(l)$ в точку $b(l)$; $\boldsymbol{E^i}(x(l), y(l), z(l))$ — вектор электрической напряженности внешнего электромагнитного поля; $V(l)$ — напряжение между проводниками витой пары (положительное в точке $b(l)$, отрицательное в точке $a(l)$; $I(l)$ — ток в линии передачи; $Z$ — полное сопротивление линии передачи на единицу длины; $Z_i$ — активное сопротивление линии передачи на единицу длины; $L_e$ — индуктивность линии передачи на единицу длины; $Y$ — полная проводимость линии передачи на единицу длины; $G$ — проводимость изоляции вокруг линии передачи на единицу длины; $C$ — емкость линии передачи на единицу длины; $V_F(l)$ и $I_F(l)$ — распределенные зависимые источники напряжения и тока соответственно; $\omega$ — угловая частота волны воздействующего электромагнитного поля.

Фактически данные уравнения можно рассматривать как модификацию известных уравнений для однородной линии передачи, только в данном случае распределенные источники $V_F(l)$ и $I_F(l)$ учитывают винтовой характер линии передачи неэкранированной витой пары.

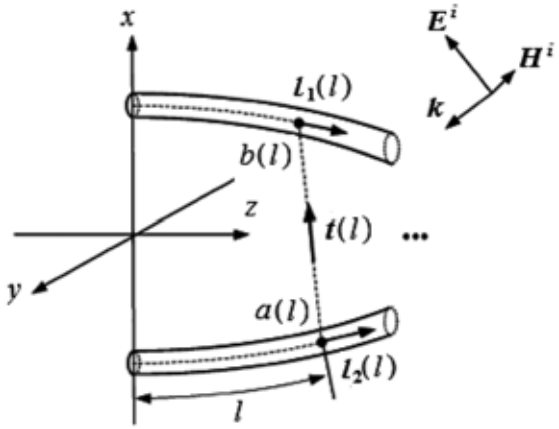



**Рис. 3. Изменение координат проводников витой пары в плоскости *x*—*y* при изменении координат в направлении оси *z***

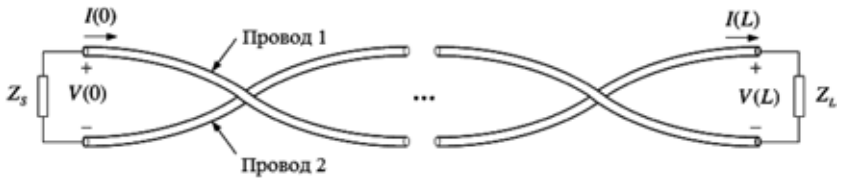



**Рис. 4. Конфигурация неэкранированной витой пары с установленными нагрузочными цепями**

Рассмотрим ограничения данных уравнений.

1. Изоляция вокруг проводников линии передачи считается однородной диэлектрической средой. На практике проводники витой пары покрыты тонким слоем изоляции. При этом его толщина всегда намного меньше наименьшей длины волны воздействующего электромагнитного поля, поэтому можно считать ее однородным диэлектрическим слоем.

2. Индуктивность $L_e$ и емкость $C$ линии передачи на единицу длины считаются независимыми от положения по длине, так как, во-первых, расстояние между осями проводников линии передачи всегда остается неизменным в плоскости $x$—$y$, и, во-вторых, период скрутки линии передачи намного больше, чем расстояние между осями проводников витой пары ($p \gg s$). Данное предположение дает возможность решения задачи для случая однородной линии передачи, а все первичные параметры линии передачи на единицу длины рассчитываются по известным формулам для двухпроводной цилиндрической линии передачи.

3. Минимальная длина волны воздействующего электромагнитного поля намного больше расстояния между осями проводников неэкранированной витой пары ($\lambda \gg s$).

4. Отношение полной длины проводников неэкранированной витой пары $L_Z$ к шагу скрутки намного больше единицы $\left(\frac{L_Z}{p} \gg 1\right)$.

Следующим шагом в решении данной задачи является введение нагрузочных цепей в линии передачи неэкранированной витой пары (рис. 4).

В данном случае токи на нагрузках $I(0)$ и $I(L)$ соответственно при $l = 0$ и $l = L$ можно рассчитать с помощью следующих уравнений:

$$I(0) = \frac{1}{D}\int_0^L (\cosh(\gamma(L-1)) + \frac{Z_L}{Z_C}\sinh(\gamma(L-l))) \times$$
$$\times [\boldsymbol{E^i}(x_1, y_1, z_1)\cdot \boldsymbol{l_1}(l) - \boldsymbol{E^i}(x_2, y_2, z_2)\cdot \boldsymbol{l_2}(l)]dl -$$
$$- \frac{1}{D}\int_{a(l)}^{b(l)} \boldsymbol{E^i}(\rho, l)\cdot d\boldsymbol{\rho}|_{l=L} + \frac{1}{D}\Big[\cos h(\gamma\cdot L) +$$
$$+ \frac{Z_L}{Z_C}\sinh(\gamma L)\Big]\int_{a(l)}^{b(l)} \boldsymbol{E^i}(\rho, l)\cdot d\boldsymbol{\rho}|_{l=0}; \qquad (1)$$

$$I(L) = \frac{1}{D}\int_0^L (\cos h(\gamma l) +$$
$$+ \frac{Z_S}{Z_C}\sinh(\gamma l))[\boldsymbol{E^i}(x_1, y_1, z_1)\cdot \boldsymbol{l_1}(l) -$$
$$- \boldsymbol{E^i}(x_2, y_2, z_2)\cdot \boldsymbol{l_2}(l)]dl -$$
$$- \frac{1}{D}\Big[\cosh(\gamma L) + \frac{Z_S}{Z_C}\sinh(\gamma L)\Big]\times$$
$$\times \int_{a(l)}^{b(l)} \boldsymbol{E^i}(\rho, l)\cdot d\boldsymbol{\rho}|_{l=L} + \frac{1}{D}\int_{a(l)}^{b(l)} \boldsymbol{E^i}(\rho, l)\cdot d\boldsymbol{\rho}|_{l=0}, \qquad (2)$$

где

$$D = \cosh(\gamma L)(Z_S + Z_L) + \sinh(\gamma L)\Big(Z_C + \frac{Z_S Z_L}{Z_C}\Big);$$

$$Z_C = \sqrt{(Z_i + j\omega L_e)/(G + j\omega C)};$$

$$\gamma = \sqrt{(Z_i + j\omega L_e)(G + j\omega C)};$$

$$Z_i = \frac{2Rs}{\pi d},\ Rs = \frac{1}{\sigma\delta_0},\ \delta_0 = \sqrt{\frac{2}{2\pi f\mu\sigma}};\ G = 2\pi C\tan\delta;$$

$\omega$, $\lambda$ — угловая частота и длина волны воздействующего электромагнитного поля соответственно; $\mu$ — относительная магнитная проницаемость изоляции проводника; $\delta_0$ — толщина скин-слоя проводника; $\sigma$ — проводимость проводника на единицу длины; $d$ — диаметр цилиндрического проводника витой пары; $\tan\delta$ — тангенс угла потерь изоляции проводника.

Действительные напряжения на нагрузках $V_d(0)$ и $V_d(L)$ вычисляются по формулам

$$V_d(0) = \frac{|V(0)|}{\sqrt{2}} = \frac{|I(0)Z_S|}{\sqrt{2}};$$

$$V_d(L) = \frac{|V(L)|}{\sqrt{2}} = \frac{|I(L)Z_L|}{\sqrt{2}},$$

где $V(0)$ и $V(L)$ — соответственно расчетные комплексные напряжения на нагрузках $Z_S$ и $Z_L$.

Уравнения (1), (2) учитывают воздействующее электромагнитное поле в общем виде, т. е. оно может быть и неоднородным. Однако для оценки влияния гармонического электромагнитного поля на линию передачи в виде неэкранированной витой пары ограничимся случаем дальней зоны, где характерно постоянное соотношение вектора напряженности электрического и магнитного полей, т. е. электромагнитное поле рассматривается в виде плоской электромагнитной волны, распространяющейся в свободном пространстве (рис. 5).

Для данного случая воздействующее электромагнитное поле описывается следующим образом:

$$\boldsymbol{E^i}(x, y, z) = \boldsymbol{E^i}(e_x\boldsymbol{a_x} + e_y\boldsymbol{a_y} + e_z\boldsymbol{a_z})\mathbf{e}^{-j(k_x x + k_y y + k_z z)}. \qquad (3)$$

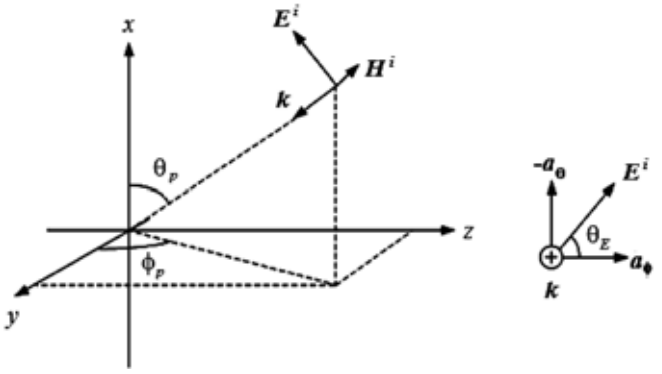



**Рис- 5. Описание ориентации воздействующего электромагнитного поля**

Здесь

$$e_x = \sin(\theta_E)\sin(\theta_p);$$

$$e_y = -\sin(\theta_E)\cos(\theta_p)\cos(\phi_p) - \cos(\theta_E)\sin(\phi_p);$$

$$e_z = -\sin(\theta_E)\cos(\theta_p)\sin(\phi_p) + \cos(\theta_E)\cos(\phi_p);$$

$$k_x = -k\cos(\theta_p);$$

$$k_y = -k\sin(\theta_p)\cos(\phi_p);\ k_z = -k\sin(\theta_p)\sin(\phi_p);$$

$$k = \frac{\omega}{c_0} = \frac{2\pi}{\lambda},$$

где $c_0$ — скорость света; $\theta_p$, $\phi_p$, $\theta_E$ — углы падения и поляризации воздействующего электромагнитного поля по отношению к линии передачи. Подставляя уравнение (3) в уравнения (1), (2) и учитывая ограничения, получаем более простые уравнения (4, (5) для решения задачи анализа электромагнитного воздействия в дальней зоне на неэкранированную витую пару:

$$I(0) \approx -\frac{E^i_S}{2D}[(\alpha e_x + j\kappa_y e_z)(F_+(L) + \frac{Z_L}{Z_C}F_-(L)) -$$

$$- (\alpha e_y - j\kappa_x e_z)(K_+(L) + \frac{Z_L}{Z_C}K_-(L)) +$$

$$+ 2(e_x\cos(\alpha L) + e_y\sin(\alpha L))\mathbf{e}^{-j\kappa_z L} -$$

$$- 2e_x(\cos h(\gamma L) + \frac{Z_L}{Z_C}\sin h(\gamma L))]; \qquad (4)$$

$$I(L) \approx -\frac{E^i_S}{2D}\Big[(\alpha e_x + j\kappa_y e_z)(F^*_+(L) + \frac{Z_S}{Z_C}F^*_-(L)) - \ldots$$

$$\ldots - (\alpha e_y - j\kappa_x e_z)(K^*_+(L) + \frac{Z_S}{Z_C}K^*_-(L)) +$$

$$+ 2(\cos h(\gamma L) + \frac{Z_S}{Z_C}\sin h(\gamma L))(e_x\cos(\alpha L) +$$

$$e_y\sin(\alpha L))\mathbf{e}^{-j\kappa_z L} - 2e_x\Big], \qquad (5)$$

где

$$F_\pm(L) = F_1(L) \pm F_2(L);$$

$$K_\pm(L) = K_1(L) \pm K_2(L);$$

$$F^*_\pm(L) = \pm F_1(L)\mathbf{e}^{-\gamma L} + F_2(L)\mathbf{e}^{\gamma L};$$

$$K^*_\pm(L) = \pm K_1(L)\mathbf{e}^{-\gamma L} + K_2(L)\mathbf{e}^{\gamma L};$$

$$F_1(L) =$$

$$= \frac{(\alpha\mathbf{e}^{\gamma L} - (\alpha\cos(\alpha L) + (\gamma + j\kappa_z)))\sin(\alpha L)\mathbf{e}^{-j\kappa_z L}}{(\gamma + j\kappa_z)^2 + \alpha^2};$$

$$F_2(L) =$$

$$= \frac{(\alpha\mathbf{e}^{-\gamma L} - (\alpha\cos(\alpha L) + (-\gamma + j\kappa_z)))\sin(\alpha L)\mathbf{e}^{-j\kappa_z L}}{(\gamma + j\kappa_z)^2 + \alpha^2};$$

$$K_1(L) =$$

$$= \frac{(\gamma + j\kappa_z)\mathbf{e}^{\gamma L} + (\alpha\sin(\alpha L) - (\gamma + j\kappa_z)\cos(\alpha L))\mathbf{e}^{-j\kappa_z L}}{(\gamma + j\kappa_z)^2 + \alpha^2};$$

$$K_2(L) =$$

$$= \frac{(-\gamma + j\kappa_z)\mathbf{e}^{-\gamma L} + (\alpha\sin(\alpha L) + (\gamma - j\kappa_z)\cos(\alpha L))\mathbf{e}^{-j\kappa_z L}}{(\gamma - j\kappa_z)^2 + \alpha^2};$$

$$\kappa_x = \frac{k_x\alpha p}{2\pi};\ \kappa_y = \frac{k_y\alpha p}{2\pi};\ \kappa_z = \frac{k_z\alpha p}{2\pi}.$$

Мощность рассеивания $P$ на нагрузках $Z_S$, $Z_L$ неэкранированной витой пары может быть вычислена с помощью следующих выражений:

$$P_S = \frac{1}{2}\mathrm{Re}(Z_S|I(0)|^2);\ P_L = \frac{1}{2}\mathrm{Re}(Z_L|I(L)|^2). \qquad (6)$$

### Сравнение результатов моделирования и экспериментальных данных

Для проверки предложенных математических моделей проведем сравнение расчетных результатов с приведенными в литературе экспериментальными данными [6]. На рис. 6—8 представлены сравнительные результаты расчета мощности рассеивания на нагрузке $Z_S$ (6) при внешнем гармоническом воздействии с различных сторон неэкранированной витой пары и при различном положении вектора напряженности электрического поля (горизонтально, вертикально относительно плоскости земли).

### Моделирование электромагнитных помех в неэкранированной витой паре

Рассмотрим электромагнитные помехи в неэкранированной витой паре 5-й категории (стандарт EIA/TIA 568) при внешнем гармоническом электромагнитном воздействии в дальней зоне. Пара-

метры неэкранированной витой пары 5-й категории:

- волновое сопротивление $Z_C$ = 100 Ом;
- сопротивление нагрузки $Z_C = Z_L$ = 100 Ом;
- шаг скрутки $p$ = 16 мм;
- расстояние между осями проводников $s$ = 0,9 мм;
- диаметр проводника (медь) $d$ = 0,54 мм;
- толщина изоляции 0,18 мм;
- диэлектрическая проницаемость материала изоляции 1,74.

На рис. 9 представлены результаты моделирования электромагнитных помех (максимальные значения) на нагрузке $Z_C$ неэкранированной витой пары при гармоническом электромагнитном воздействии (частота электромагнитного воздействия 2 ГГц; максимальная напряженность электрического поля 30 В/м; направление распространения электромагнитного поля — по оси $y$, вектор напряженности электрического поля направлен вдоль оси $z$ (рис. 9, *а*); направление распростра-

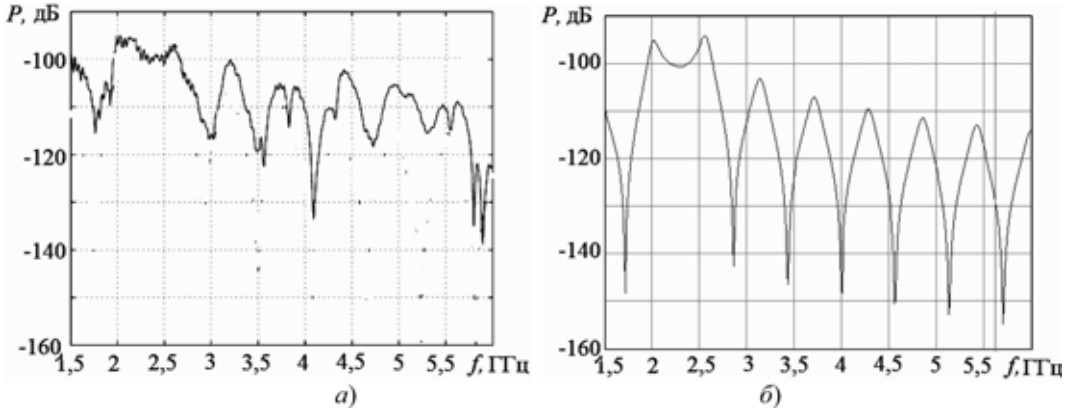



**Рис. 6. Мощность рассеивания на нагрузке $Z_S$ (*а* — эксперимент, *б* — моделирование) при внешнем воздействии сбоку витой пары (вектор напряженности электрического поля направлен по оси *z*)**

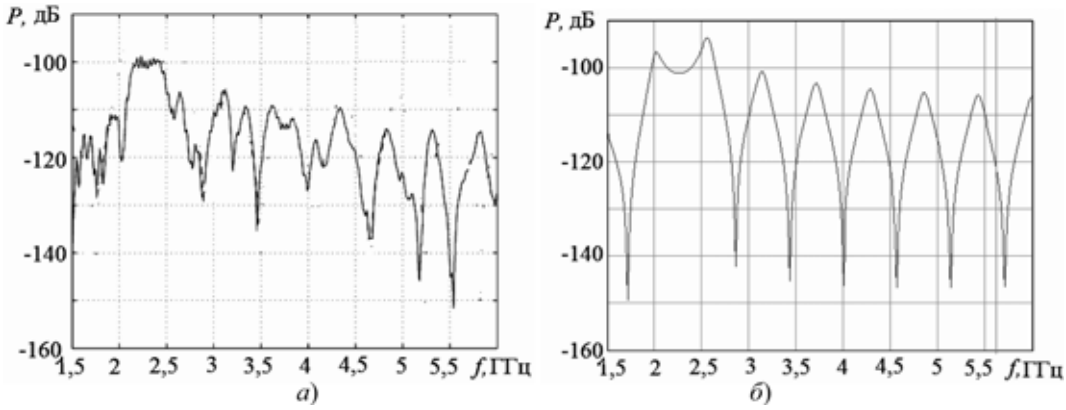



**Рис. 7. Мощность рассеивания на нагрузке $Z_S$ (*а* — эксперимент, *б* — моделирование) при внешнем воздействии сбоку витой пары (вектор напряженности электрического поля направлен по оси *x*)**

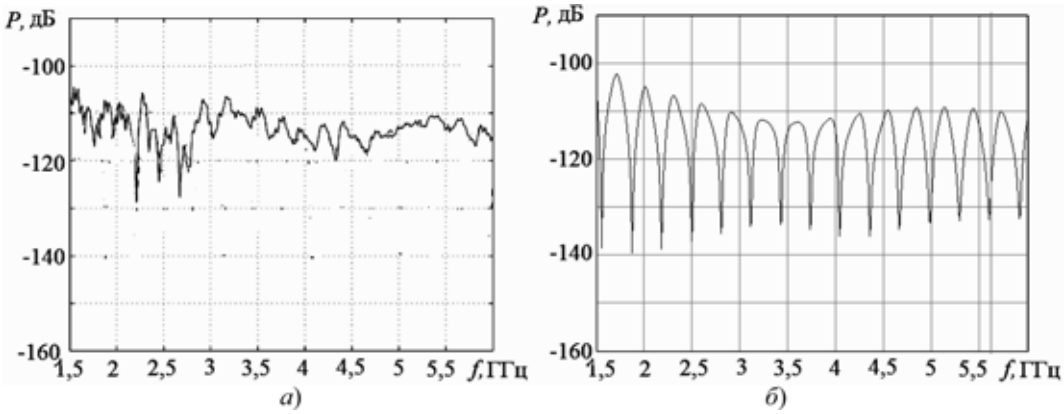



**Рис. 8. Мощность рассеивания на нагрузке $Z_S$ (*а* — эксперимент, *б* — моделирование) при внешнем воздействии с торца витой пары (вектор напряженности электрического поля направлен по оси *x*)**

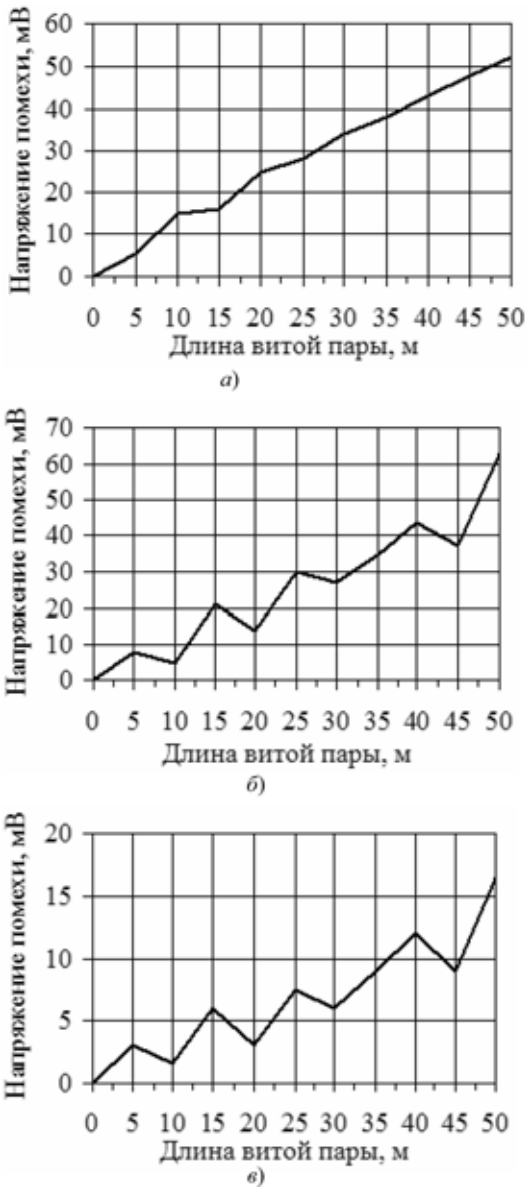



**Рис 9. Максимальное значение напряжения $V_d(0)$ электромагнитной помехи в неэкранированной витой паре в зависимости от ее длины**

нения электромагнитного поля — по оси *y*, вектор напряженности электрического поля направлен вдоль оси *x* (рис. 9, *б*); направление распространения электромагнитного поля — по оси *z*, вектор напряженности электрического поля направлен вдоль оси *x* (рис. 9, *в*)).

## Заключение

По результатам данной работы можно сделать следующие выводы.

1. Неэкранированная витая пара является основой для построения большинства слаботочных систем зданий, а по своим электрофизическим и геометрическим параметрам может выступать хорошим приемником электромагнитных полей и преобразователем их в электромагнитные помехи.

2. Предложены математические модели для анализа электромагнитных помех в неэкранированной витой паре при внешнем гармоническом электромагнитном воздействии в дальней зоне.

3. Сравнение результатов моделирования и экспериментальных данных [6] позволяет говорить о том, что они достаточно хорошо согласуются (расхождение в среднем не более ±1,5 дБ), что подтверждает адекватность математических моделей.

4. Результаты анализа показывают, что электромагнитные помехи в неэкранированной витой паре (длина до 50 м) при различных вариантах ориентации векторов электромагнитного воздействия могут достигать нескольких десятков милливольт. При этом можно предположить, что ломаный характер зависимости электромагнитных помех (максимальные значения) в неэкранированной витой паре от ее длины связан с резонансными эффектами, возникающими при внешнем гармоническом электромагнитном воздействии.

5. Прогнозирование электромагнитных помех, наводимых в неэкранированной витой паре при воздействии разнообразных источников гармонических электромагнитных полей в дальней зоне, позволит заранее принять необходимые меры по обеспечению информационной безопасности и электромагнитной совместимости современных ЭС.

### Список литературы

1. **Кечиев Л. Н., Степанов П. В.** ЭМС и информационная безопасность в системах телекоммуникаций. М.: Технологии, 2005. 320 с.

2. **ГОСТ Р 50922—2006.** Защита информации. Основные термины и определения. М.: Стандартинформ, 2006. 18 с.

3. **Сергеев Р.** Шестая степень совершенства СКС // BYTE/Россия. 2002. № 11. С. 52—54.

4. **Taylor C. D., Castillo J. P.** On the response of a terminated twisted-wire cable excited by a plane-wave electromagnetic field // IEEE Transaction on Electromagnetic Compatibility. 1980. N 1. P. 16—19.

5. **Smith A.** Coupling of external electromagnetic fields to transmission lines. New York: Inter science, 1977. 125 p.

6. **Armenta R. B., Sarris C. D.** Efficient evaluation of the terminal response of a twisted-wire pair excited by a plane-wave electromagnetic field // IEEE Transaction on Electromagnetic Compatibility. 2007. N 3. P. 698—707.

УДК 004.738.5:002.53:002.55

**Г. А. Тарнавский,** д-р физ.-мат. наук,
Институт вычислительной математики
и математической геофизики СО РАН,
e-mail: Gennady.Tarnavsky@gmail.com

# Облачные вычисления: технология "Data Files Cruise" организации информационных потоков на портале Sci.Shop.ru

*Рассматривается новая технология DFC ("Data Files Cruise", "круиз файлов данных") организации информационных потоков в Центре компьютерного моделирования, дополняющая технологию SaaS ("Software as a Service", "программное обеспечение как услуга") парадигмы "Облачные вычисления".*

***Ключевые слова:*** *информационные технологии, облачные вычисления, программное обеспечение как услуга, круиз файлов данных, компьютерное моделирование, дистанционное обучение*

## Введение

Cloud Computing ("облачные вычисления") — одно из перспективных направлений современных информационных технологий. Концепция Cloud Computing основана на уверенности в том, что сеть Интернет (облако — символ Интернета) в состоянии удовлетворить потребности пользователей в генерировании и обработке данных в широких диапазонах их запросов.

Так, система Google Apps обеспечивает приложения для бизнеса в режиме он-лайн, доступ к которым происходит с помощью Интернет-браузеров, при этом программное обеспечение и данные хранятся на серверах Google. Кроме того, операционная система Google Chrome OS целиком основана на облачных вычислениях.

Cloud Computing в настоящее время включает в себя три основные группы технологий:

- "инфраструктура как услуга";
- "платформа как услуга";
- "программное обеспечение как услуга".

В предлагаемой статье рассматривается последняя из перечисленных технологий — технология SaaS ("Software as a Service"). На примере портала SciShop.ru, полностью базирующегося на Cloud Computing, анализируются позитивные и негативные аспекты технологии SaaS, а также технологии DFC ("Data Files Cruise", "круиз файлов данных"), которая специально разработана для обеспечения облачных вычислений в Центре компьютерного моделирования.

## Центр компьютерного моделирования SciShop.ru

Примером применения технологий "Облачных вычислений" является Web-ресурс SciShop.ru, Центр компьютерного моделирования в Интернете.

Центр компьютерного моделирования (рис. 1, подробнее см. [1]) предназначен для реализации современных потребностей в дистрибьюции научных продуктов и направлен на решение фундаментальных проблем, связанных с разнообразными научными, техническими, социальными и психологическими аспектами разработки и продвижения специализированного Web-ресурса, особой точки обмена произведенным научным продуктом, в том числе и на платной основе, в системе Интернет.

Центр ориентирован на три аспекта использования: как электронная книга (в т. ч. учебник), как электронный справочник (база данных) и как инструментарий для научных исследований (процессорная система), и предназначен для хранения, пополнения и систематизации накопленной информации, обеспечения непрерывности научного про-


**Рис. 1. Главная страница (фрагмент) Центра компьютерного моделирования**

гресса и его ускорения в данной области за счет преемственности осуществленных разработок.

Правовые, экономические, социальные и психологические проблемы, связанные с использованием контента Центра, рассмотрены в работе [2]. Заметим, что развитие дистанционного обучения на базе Центра с применением программных комплексов для тренинга специалистов, аспирантов и студентов в декларированных предметных областях (например, в области "Нанотехнологии и наноматериалы", см. [3]) выявляет дополнительные проблемы, часть которых анализируется ниже.

### SaaS-технологии направления Cloud Computing

Разработка комплексов компьютерного моделирования, как правило, завершается (возможно, с дальнейшим развитием) коммерциализацией проекта.

Обычно передача вычислительного комплекса заключается в приобретении лицензии, документации и кодов компьютерной программы. После этого покупатель устанавливает приобретенный продукт на собственном компьютерном оборудовании.

Если инсталляция комплекса делается силами организации пользователя, то, как правило, это происходит с большими затруднениями, которые могут быть вызваны разнообразными причинами — от использования разных версий операционной системы до особенностей установленных у продавца и покупателя поддерживающих систем. Поэтому многие организации вынуждены содержать дополнительный штат профессиональных и дорогостоящих администраторов для поддержки корпоративного программного обеспечения.

Одним из выходов в данной ситуации является переход на так называемые SaaS-технологии, которые естественно вписываются в Cloud Computing. Эта концепция была реализована на портале SciShop.ru.

Главная сущность SaaS-технологии ("Программное обеспечение как услуга") заключается в следующем. SaaS-технология (иногда называемая SoD, "Software on Demand", "Программное обеспечение по требованию") — модель продажи или лизинга программных ресурсов, при которой поставщик разрабатывает продукт и самостоятельно управляет им, предоставляя клиенту доступ к нему через Интернет. Укажем, что подобная модель замены продажи услугой без отчуждения товара от продавца к покупателю используется для распространения не только программных приложений, но и, например, кинофильмов (только просмотр, без продажи собственно фильма) и т. п.

Основное преимущество SaaS для потребителя состоит в отсутствии затрат, связанных с установкой, обновлением и поддержкой работоспособности оборудования и программного обеспечения, функционирующего на нем. В рамках модели SaaS заказчики платят не за владение продуктом как таковым, а за его аренду, т. е. использование через Web-интерфейс (рис. 2).


**Рис. 2. Страница "Оплатите вашу заявку с помощью SMS" раздела "Прием абонентской платы" Центра компьютерного моделирования SciShop.ru**

Таким образом, в отличие от классической схемы лицензионной (т. е. санкционированной обеими сторонами) покупки, клиент несет сравнительно небольшие периодические затраты. Ему не требуется инвестировать существенные средства для приобретения программного продукта и аппаратной платформы для его развертывания, а затем поддерживать его работоспособность.

Схема периодической оплаты предполагает, что в случае, если в данный период нет необходимости в продукте, то заказчик может приостановить его использование и заморозить выплаты разработчику.

### Доступ к программным ресурсам и организация вычислений

Рассмотрим организацию "Облачных вычислений" в Центре компьютерного моделирования SciShop.ru. Поясним маршрутизацию действий клиента на примере одного из информационно-вычислительных комплексов Центра. Допустим, что клиент работает в области кремниевой элек-

троники, например, с необходимостью изучения нанотехнологического процесса легирования подложки кремния методом ионной имплантации, и намерен использовать для проведения расчетов своей задачи процессорную систему "IMPL" информационно-вычислительного комплекса "Нано" (подробнее см. [1, 3, 4]).

Для работы с программным сегментом "IMPL" в режиме дистанционного доступа по сети Интернет необходимо:

- зайти на сайт Центра компьютерного моделирования SciShop.ru (см. рис. 1);
- нажатием кнопки "Центр-1" перейти в соответствующий сегмент ресурса, корневую страницу группы программных комплексов "Удар", "Поток", "Астра" и "Нано";
- нажатием кнопки "Нано" перейти на базовую страницу этого раздела (рис. 3, см. вторую сторону обложки);
- по гиперссылке "Компьютерные вычисления-1" перейти на эту линию раздела "Нано";
- нажатием кнопки "IMPL" перейти на страницу "Программный комплекс "IMPL" (рис. 4, см. вторую сторону обложки).

Эта страница содержит сценарий задания на проведение расчета, окна ввода цифровых параметров и клавишу "Запустить программу". Нажатием клавиши осуществляется формирование вычислительного задания и пересылка его в Суперкомпьютерный центр СО РАН.

Далее проводится инициализация размещенных там процессорных систем комплекса "IMPL", исполнение задания (проведение расчета), запись решения в транспортный файл и его размещение на одном из ресурсов Интернета.

Адрес этого ресурса, доступный только клиенту, указывается ему на особой странице (рис. 5) сайта, которая генерируется автоматически, без участия пользователя, после нажатия им клавиши "Запустить программу". По окончании решения и завершения системных операций клиент, нажав этот адрес, являющийся гиперссылкой, может получить решение в виде цифровых таблиц на странице "Вывод информации".

Гиперссылка "Вывести график", размещенная справа на плашке этой страницы, обеспечивает инициализацию графической системы GnuPlot и визуализацию (рис. 6) полученного числового решения.

Таким образом, последовательность действий клиента по формированию собственной расчетной задачи в Центре компьютерного моделирования является абсолютно прозрачной и ясной (даже для неспециалиста в области вычислительных технологий) и строго детерминирована от начала (вход на сайт SciShop.ru) и до конца (получение в свое распоряжение результатов расчетов в таблично-цифровом и графическом формате). При этом клиент даже "не видит" ни сценария задания, ни кодов программного комплекса. Он освобожден от необходимости организовывать маршрутизацию информационных потоков к вычисляющему компьютеру и обратно. Пользователю необходимо только ввести исходные цифровые данные в специализированные окна ввода и нажать клавишу "Запустить программу".

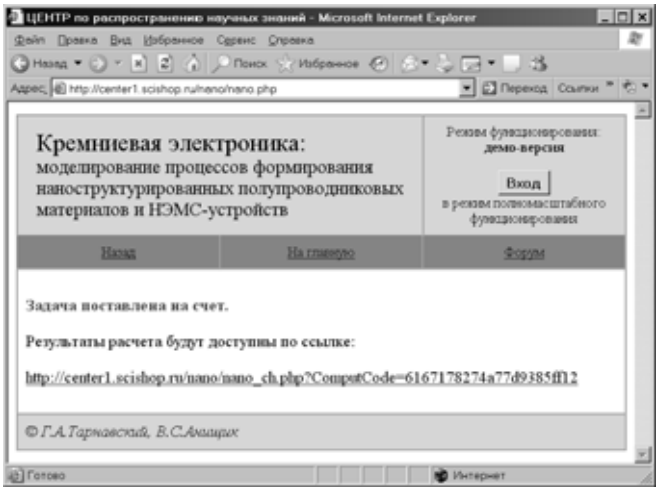



**Рис. 5. Страница "Постановка задачи на счет программного сегмента IMPL" линии "Компьютерные вычисления-1" раздела "Нано"**

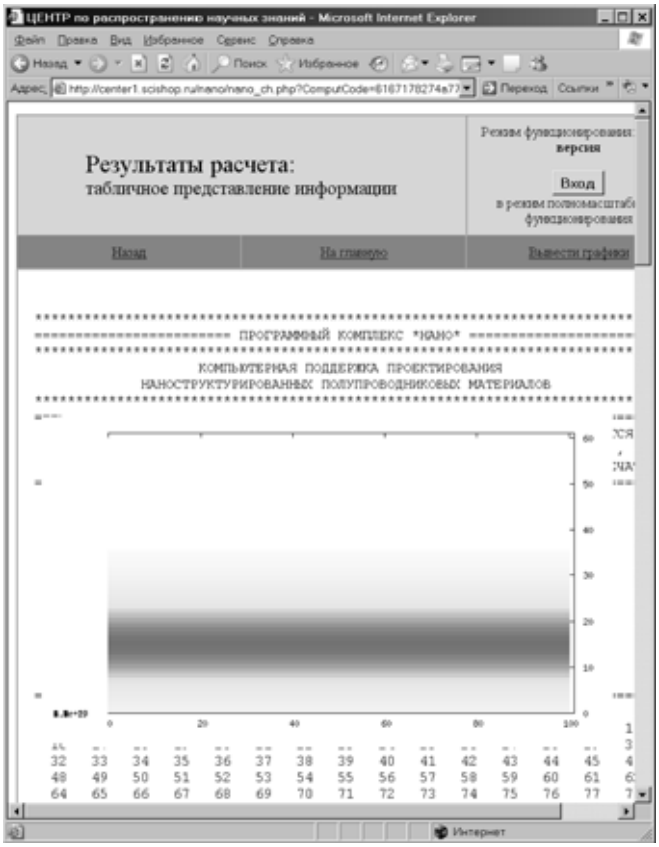



**Рис. 6. Таблично-графическое представление информации в разделе "Нано"**

### DFC-технология организации информационных потоков в Центре SciShop.ru — новая технология направления Cloud Computing

Заметим, что портал SciShop.ru является пионером непосредственного проведения компьютерного моделирования в Интернете. Эскизное проектирование этого ресурса было проведено в 1999—2002 гг. (термин Cloud Computing появился в 1998 г., а SaaS — в 2001 г.). В 2003 г. была начата практическая разработка Центра компьютерного моделирования [5], а в 2004 г. — опытная эксплуатация первой линии [6, 7] Центра. К настоящему времени в Сети пока нет другого аналогичного портала, функционирующего действительно в режиме непосредственного расчета научных и прикладных задач [8].

При создании Web-ресурса SciShop.ru была существенно развита методология Cloud Computing. Эта методология в настоящий момент предполагает, что информационный поток движется только в двух направлениях: "браузер клиента — сервер Интернета" и обратно, "сервер Интернета — браузер клиента". Разработанную и функционирующую новую организацию информационных потоков в SciShop.ru можно назвать технологией DFC (Data Files Cruise) — "круиз файлов данных". Эта технология по сравнению с технологией SaaS имеет существенно больше направлений движения потоков информации: "браузер клиента — сервер Интернета SciShop.ru — вход в суперкомпьютерный центр ($IP_1$-адрес) — вычислительный процессор — выход из суперкомпьютерного центра ($IP_2$-адрес) — сервер Интернета SciShop.ru — браузер клиента" (рис. 7).

Полученный файл решения клиент может взять непосредственно на портале SciShop.ru, или по указанному $IP_2$-адресу. В дальнейшем предполагается запустить третий вариант — отсылку решения на адрес, указанный клиентом.

DFC-технология является огромным преимуществом Центра компьютерного моделирования, поскольку клиент может вообще не знать системного программирования, являясь научным исследователем или прикладным расчетчиком, и сосредоточиться на физическом или техническом смысле задачи, без неоправданных и излишних затрат времени и интеллектуальных усилий на формирование вычислительного задания.

Заметим во избежание недоразумений, что процедура использования суперкомпьютеров в режиме удаленного доступа достаточно хорошо известна, однако здесь имеется в виду совсем иное — создание файлов доступа к суперкомпьютеру, проведение расчетов и пересылка решения по указанному адресу осуществляется системами Центра SciShop.ru автоматически, без участия клиента. В дальнейшем возможно использование и других суперкомпьютерных центров на альтернативной и/или совмещенной основе.

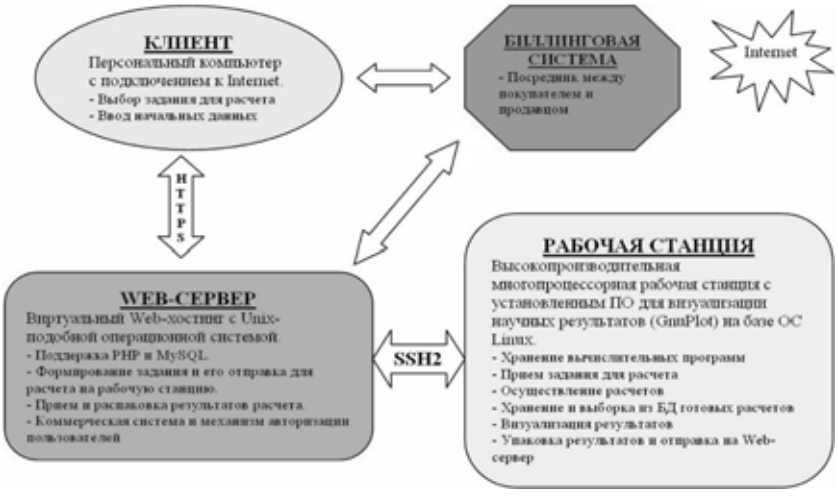



**Рис. 7. Схема организации информационных потоков в Центре компьютерного моделирования SciShop.ru**

### Позитивные и негативные аспекты технологий SaaS и DFC

К характеристикам pro и contra модели Cloud Computing (и реализующим ее технологиям SaaS и DFC) можно отнести следующие факторы.

Позитивные факторы для разработчиков:

- эффективная борьба с нелицензионным использованием программного продукта, поскольку сам продукт не попадает к заказчику;
- несанкционированное использование доступа нескольких пользователей под одним логином относительно легко обнаруживается и пресекается;
- существенное уменьшение затрат на развертывание и внедрение технической и консалтинговой поддержки для каждого заказчика.

Позитивные факторы для потребителей:

- отсутствие необходимости установки программного обеспечения на рабочих местах пользователей, поскольку доступ к нему осуществляется через обычный браузер;
- радикальное сокращение затрат на развертывание системы в организации;
- сокращение затрат на техническую поддержку и обновление развернутых систем, вплоть до их полного отсутствия;
- быстрота внедрения, обусловленная отсутствием затрат времени на развертывание системы;

- понятный интерфейс;
- ясность и предсказуемость платежей;
- возможность получения более высокого уровня обслуживания программного обеспечения.

Негативные факторы для разработчиков:

- концепция SaaS применима далеко не для всех функциональных задач;
- поскольку основная экономия ресурсов провайдера достигается за счет масштаба, технология SaaS оказывается неэффективной для малого числа клиентов;
- модель неэффективна при необходимости глубокой индивидуальной адаптации под каждого заказчика.

Негативные факторы для заказчиков:

- привязка заказчиков к единственному разработчику и его хостинг-площадке;
- нестабильность работы провайдера может приводить к невозможности долгосрочного планирования и даже срыву сроков разработки проектов;
- нежелательность использования модели SaaS (тем более с расширением DFC) для проектов строгой конфиденциальности, вследствие высокой возможности утечки информации со стороны поставщика услуг и невозможности контролировать этот процесс;
- затруднительность повышения качества сервисов в текущем режиме работы;
- необходимость наличия постоянно действующего подключения к Интернету с достаточно высокой скоростью передачи данных.

Проанализируем негативные факторы SaaS, в частности, с точки зрения развития в России дистанционного образования и проведения НИОКР в режиме удаленного доступа.

Разумеется, вся концепция дистанционного обучения в современных условиях должна базироваться на SaaS-технологиях Cloud Computing. Это касается не только проведения компьютерного моделирования как учебного тренинга или полномасштабного пользования программными средствами, но и просмотра в режиме реального времени учебных фильмов и лекций по различным областям знания.

Рассмотрим один из негативных факторов — стабильность Интернета. С развитием Сети значение этого фактора уменьшается. В развитых странах и крупных городах России он неактуален уже сейчас. Однако в российских регионах подобные проблемы по-прежнему возникают и с ними приходится считаться.

Заострим внимание на проблеме конфиденциальности. Исследовательский процесс в любой стране и, естественно, в России, имеет определенные "ниши" закрытых тематик, по которым утечка информации крайне нежелательна. Постороннему квалифицированному специалисту не составит труда даже по ряду признаков определить направленность тематики НИОКР. Это может зачеркнуть применение Cloud Computing для данных целей в принципе, несмотря на большой спектр позитивных факторов. В этом вопросе не может дать гарантий даже юридический договор по защите информации.

Со стороны разработчика одним из главных недостатков модели SaaS является высокая стоимость входа на рынок.

Чтобы предоставить конкурентную стоимость клиенту, разработчику требуется "эффект масштаба", т. е. большое число потенциальных клиентов. Однако небольшие проекты с очень невысокой и удобной формой оплаты услуг могут быть весьма эффективными, с большими нормами прибыли в зависимости от тематики, определяющей число клиентов.

## Заключение

Направление информационных технологий Cloud Computing является весьма перспективным направлением дистрибьюции знаний. В статье кратко рассмотрены и проанализированы некоторые позитивные и негативные факторы использования SaaS- и DFC-технологий, входящих в Cloud Computing.

Более полное рассмотрение и анализ этих факторов высвечивают дополнительные юридические и экономические проблемы контента таких Web-ресурсов, как Центр компьютерного моделирования SciShop.ru. Кроме того, вполне возможны и даже сейчас просматриваются предпосылки юридических коллизий взаимоотношений "покупатель—продавец программного продукта", точнее, "потребитель—поставщик интеллектуальных услуг", поскольку идеология Cloud Computing базируется на замене продажи услугой (объем данной статьи не позволяет остановиться на этом вопросе подробнее).

Поскольку вступил в силу закон РФ о возможности создания инновационных структур на базе университетов и научных учреждений, ожидается активизация на рынке предложений интеллектуальных продуктов, поэтому необходим своевременный анализ возникающих новых проблем.

*Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (проект № 08-07-12001-офи).*

### Список литературы

1. **Тарнавский Г. А., Алиев А. В., Анищик В. С., Тарнавский А. Г., Жибинов С. Б., Чесноков С. С.** Информационные технологии и проблемы создания Центра компьютерного моделирования в Интернете // Информационные технологии. 2009. № 8. С. 68—73.

2. **Жибинов С. Б., Тарнавский Г. А.** Научно-образовательный Интернет-центр компьютерного моделирования: проблемы интеллектуальной собственности контента // Право и образование. 2009. № 8. С. 86—98.

3. **Тарнавский Г. А., Жибинов С. Б., Тарнавский А. Г., Чесноков С. С., Алиев А. В., Анищик В. С.** Дистанционное обучение. Курс лекций "Нанотехнологии и наноматериалы: программный комплекс NanoMod компьютерного моделирования процессов формирования наноструктурированных полупроводниковых материалов для электроники". Лекция 1. Центр компьютерного моделирования в Интернете. Лекция 2. Общее описание программного комплекса NanoMod. Лекция 3. Оксидирование кремния // Инфосфера. 2009. № 43. С. 15—26.

4. **Тарнавский Г. А., Анищик В. С.** Решатели процессорной системы программного комплекса NanoMod // Нано- и микросистемная техника. 2009. № 4. С. 6—13.

5. **Тарнавский Г. А.** Создание в среде Интернет информационно-вычислительного центра по распространению научных знаний в области мультипроцессорного компьютерного моделирования задач аэродинамики и физической газовой динамики. Проект РФФИ № 04-07-90002-в, 2003. URL: http://grant.rfbr.ru/person_proj.asp

6. **Тарнавский Г. А., Тарнавский А. Г., Гилев К. В., Алиев А. В., Вавилин А. Е. Вайнер Д. А. Шварц О. Я., Пиун А. В., Пиун П. В., Хакимзянов Г. С., Малыхин С. М.** Создание в среде Интернет информационно-вычислительного Центра по распространению научных знаний в области аэромеханики и физической газовой динамики // Труды 14-й Зимней школы по механике сплошных сред. Пермь: Изд-во ИМСС УрО РАН. 2005. С. 171.

7. **Тарнавский Г. А., Тарнавский А. Г., Гилев К. В.** Информационно-вычислительный Интернет-центр "Аэромеханика". Первая линия: программный комплекс "Удар" // Вычислительные методы и программирование. 2005. Т. 6. № 1. С. 27—48.

8. **Тарнавский Г. А., Чесноков С. С.** Краткий обзор Web-ресурсов компьютерного моделирования в Интернете // Труды ИВМиМГ СО РАН. Серия: Информатика. 2009. Т. 9. С. 125—131.

---

УДК 519.87

**М. А. Месягутов,** аспирант,
Уфимский государственный авиационный технический университет,
e-mail: marat.mesyagutov@googlemail.com

# Задача двухмерной ортогональной упаковки: поиск нижней границы на базе решения одномерной продолженной упаковки

*Рассматривается задача одномерной продолженной упаковки. Для ее решения предлагается метод ветвей и границ с использованием правила "следующий подходящий" и оценок, получаемых с помощью решения задачи линейного программирования. Предлагается специальная процедура для сокращения размерности решаемой задачи. Решение применяется для построения улучшенных нижних границ в задаче двухмерной упаковки.*

***Ключевые слова:*** *одномерная продолженная упаковка, двухмерная ортогональная упаковка, метод ветвей и границ, линейное программирование*

## Введение

Рассмотрим следующую задачу двухмерной упаковки в полосу (2D Strip Packing Problem, 2SPP). Имеется прямоугольная полоса фиксированной ширины $W$ и полубесконечной длины. Требуется разместить $m$ прямоугольных предметов ширины $w_i$ и длины $l_i$, $i \in I := \{1, ..., m\}$, в полосе так, чтобы стороны прямоугольников были параллельны сторонам полосы и предметы не перекрывались между собой и со сторонами полосы, при этом длина занятой части полосы достигала минимума. Все величины — целые.

На рис. 1, *а* представлено допустимое решение задачи 2SPP. Если провести вертикальные линии по правой и левой сторонам предметов, то упаковку можно условно разделить на блоки. Далее, выписав номера предметов, которые пересекают каждый блок, можно построить соответствующую упаковке блок-структуру (Slice Structure, SS) в виде списка кортежей [1, 2]. Для упаковки, изображенной на рис. 1, *а*, имеем блок-структуру SS = $= \langle(1, 2)2; (1, 3, 4)1; (5, 6, 3, 4)2; (5, 4)1; (7, 8)2; (8)1\rangle$ с указанием длины каждого блока (кортежа). Каждой блок-структуре отвечает одномерная прямоугольно ориентированная упаковка (1D Rectangular Oriented Stock Cutting, 1ROSC), см. рис. 1, *б*). Нетрудно видеть, что эта упаковка удовлетворяет условию разнородности (т. е. в контейнер пакуются предметы только с различными номерами) и условию продолженности упаковки предмета с одним номером в соседние контейнеры. Одномерные упаковки, обладающие только этими свойствами, назовем *одномерными продолженными упаковками* (рис. 1, *в*). Заметим, что упаковка, представленная на рис. 1, *б*, является одной из допустимых одномерных продолженных упаковок. Однако множество допустимых одномерных упаковок шире множества решений, получаемых из допустимых двухмерных упаковок.

Рассмотрим следующую задачу одномерной продолженной упаковки (1D Contiguous Bin Packing Problem, 1CBPP): в одномерные контейнеры, длина которых совпадает с шириной $W$ двухмерной полосы, требуется упаковать предметы заданной длины $w_i$ в количестве $b_i$ (потребности), $i \in I := \{1, ..., m\}$, соблюдая при этом следующие условия:

1) в один и тот же контейнер можно упаковать предметы только с различными номерами (*разнородность*);

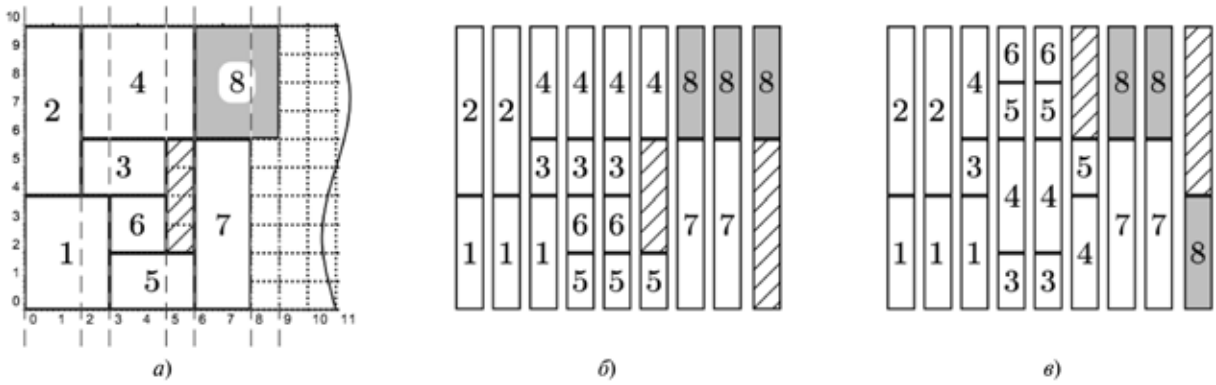


**Рис. 1. Допустимые решения задач:**
*а* — двухмерной упаковки; *б* — прямоугольно ориентированной одномерной упаковки; *в* — одномерной продолженной упаковки

2) если при упаковке предмета с номером $i \in I$ в текущий контейнер потребность в нем не удовлетворена, то этот предмет упаковывается в следующий контейнер (*продолженность выбора*).

Требуется *минимизировать* число упакованных контейнеров. При этом все величины — целые.

Эта задача рассматривалась независимо различными авторами [3], а также в виде задачи одномерной прямоугольно ориентированной упаковки (1D Rectangular Oriented Cutting Stock Problem, 1ROCSP) в связи с ее применением в блочной технологии для решения 2SPP [1, 2]. В работах [4, 5] показано, что решение задачи 1CBPP является нижней границей для решений задачи 2SPP. Задача 1CBPP имеет также самостоятельные применения, например, в сфере задач расписания как задача расписания ресурсов [6].

В разделе 1 описана процедура *препроцессинга*, которая позволяет до начала решения задачи сократить ее размерность. В связи с тем, что решение задачи 1CBPP может использоваться для оценки нижней границы решений задачи 2SPP, а эффективных точных методов для решения 1CBPP в литературе нет, то авторами предлагается метод ветвей и границ для решения 1CBPP. Способ организации ветвлений описан в разделе 2. Оценка частичных решений обсуждается в разделе 3. В этом же разделе обсуждается стратегия поиска верхней границы. В заключение, в разделе 4 приводятся результаты численного эксперимента.

## 1. Препроцессинг

На множестве предметов задачи 1CBPP иногда удается найти такое подмножество, определенная комбинация предметов которого обязательно появится в одном из оптимальных решений. Если некоторые части оптимального решения будут распознаны на начальном этапе, то они могут быть удалены из рассмотрения, тем самым размерность задачи будет понижена. Эту процедуру назовем *препроцессингом*.

Рассмотрим первую часть упаковки, представленной на рис. 2. Возникла ситуация, когда общая потребность всех предметов с длиной, равной $(W - 1)$, превосходит общую потребность всех предметов с длиной, равной 1. Очевидно, что комбинация этих предметов, приведенная на рис. 2, не хуже любой другой и войдет в оптимальное решение именно таким образом. Ее можно удалить перед решением задачи. Для второй части упаковки возникает аналогичная ситуация для предметов $W - 3$, 3, когда общая потребность всех "длинных" предметов превосходит общую потребность всех "коротких" предметов. Комбинации из предметов по типу приведенной на рис. 2 войдут в оптимальное решение и могут быть удалены из рассмотрения. Имеет место следующее утверждение.

**Утверждение 1.** *Пусть дана задача 1CBPP с исходной информацией* $\langle W, m, w, b\rangle$. *Если существует* $\overline{W} < W/2$, *для которого справедлива система неравенств*

$$\sum_{w=1}^{\tilde{w}} \sum_{i \in I(w)} b_i \geqslant \sum_{w=1}^{\tilde{w}} \sum_{i \in I(W-w)} b_i,\ 1 \leqslant \tilde{w} \leqslant \overline{W} - 1; \quad (1)$$

$$\sum_{w=1}^{\overline{W}} \sum_{i \in I(w)} b_i \geqslant \sum_{w=1}^{\overline{W}} \sum_{i \in I(W-w)} b_i, \quad (2)$$

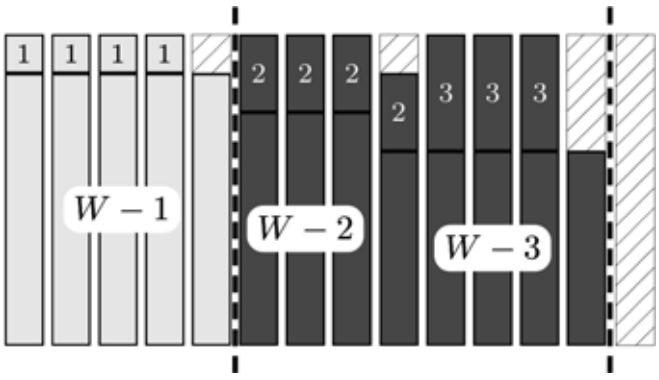



**Рис. 2. Препроцессинг: удаление комбинаций предметов, заведомо входящих в оптимальное решение**

*где* $I(w) = \{i \in I : w_i = w\}$, *то множество предметов* $\tilde{I} := \bigcup_{w=1}^{\overline{W}} (I(w) \cup I(W - w))$ *может быть удалено из рассмотрения, при этом значение оптимального решения начальной задачи* $OPT(I) = \tilde{H} + OPT(I \setminus \tilde{I})$, *где* $\tilde{H} = \sum_{w=1}^{\overline{W}} \sum_{i \in I(W-w)} b_i$; $OPT(I \setminus \tilde{I})$ — *значение оптимального решения остаточной задачи, состоящей из предметов* $I \setminus \tilde{I}$.

После сокращения множества $I := I \setminus \tilde{I}$ в нем все еще могут оставаться предметы, которые удовлетворяют системе неравенств (1)—(2). Поэтому процедура препроцессинга должна повторяться до тех пор, пока существует такое $\overline{W} < W/2$, при котором эти неравенства выполнены.

Отметим, что процедура препроцессинга также справедлива для задачи 2SPP.

## 2. Ветвления

В процессе комбинаторного поиска множество кодов задает пространство решений, и исходная задача нахождения оптимального решения сводится к поиску наилучшего элемента (кода) в этом пространстве.

Рассмотрим приоритетный список предметов $u := \langle i_1, \ldots i_m \rangle$, $i_1 \in I$, $l = 1, \ldots, m$, который определяет порядок последовательного размещения каждого элемента из $I$ в одномерный контейнер. Пара $i_l$ и $i_{l+1}$, где $i_l, i_{l+1} \in I$, означает, что предмет $i_{l+1}$ будет размещен следующим по порядку после $i_l$. Он размещается в самой нижней среди левых допустимых позиций в одномерном контейнере, но не раньше уже размещенного предмета $i_l$. Если предмет $i_{l+1}$ не умещается в контейнер вместе с предметом $i_l$, тогда предмет $i_{l+1}$ размещается в последующих контейнерах. При этом, если потребность в предмете в текущем контейнере не удовлетворена, то в следующем контейнере предмет смещается по возможности вниз. Приведенное правило упаковки предметов в одномерные контейнеры называется алгоритмом (правилом) следующий подходящий (Next Fit, NF).

**Определение 1.** *Пространство представлений решений называется P-допустимым, если оно удовлетворяет следующим требованиям: 1) пространство кодов конечно; 2) каждый элемент кода соответствует допустимому решению; 3) решение задачи может быть получено по его коду за полиномиальное время; 4) наилучший элемент в пространстве кодов соответствует оптимальному решению исходной задачи.*

**Утверждение 2.** *Множество перестановок элементов списка* $\langle i_1, \ldots, i_m \rangle$ *определяет P-допустимое пространство решений задачи 1CBPP размерности m, которое состоит из m! перестановок, каждая из которых может быть трансформирована в допустимую упаковку за время* $O(m \cdot \log m)$ *и по меньшей мере одна из них соответствует оптимальному решению задачи 1CBPP.*

Обозначим $\mathrm{NF}(\langle i_1, \ldots, i_m \rangle)$ факт получения упаковки контейнера применением правила NF к приоритетному списку $\langle i_1, \ldots, i_m \rangle$, $i_l \in I$, $l = \{1, \ldots, m\}$.

Идея ветвлений состоит в последовательном и направленном размещении каждого элемента множества $I$. Посредством этого строится дерево ветвлений $T = (V, E)$, листья которого представляют собой множество всех перестановок элементов из $I$. С каждым узлом $u \in V$ связана упорядоченная последовательность $u := \langle i_1, \ldots, i_m \rangle$ размещенных предметов $i_l \in I$, $l = 1, \ldots, d$; $d \leqslant m$. При этом каждому узлу $u \in V$ дерева $T$ соответствует однозначно получаемая с помощью алгоритма $\mathrm{NF}(u)$ частичная упаковка (блок-структура [1, 7]) $\langle (\alpha^1, \chi_1), \ldots, (\alpha^{\sigma(u)}, \chi_{\sigma(u)}) \rangle$, в которой предметы $\bar{I}(u) := I \setminus \{i_1, \ldots, i_d\}$ должны быть размещены; $\alpha^j$ — вектор упаковки блока $j$, а $\chi_j$ — интенсивность (длина) блока $j$. Ребро $(u, v)$, где $u, v \in V$, $u \neq v$, принадлежит множеству $E$, если подзадача $v$ получается из $u$ размещением предмета $i \in \bar{I}(u)$, т. е. $v = \langle u, i \rangle$. Следующий выбранный предмет $i \in \bar{I}(u)$ для узла $u \in V$ размещается в соответствии с правилом NF в блоках:

$$\langle (\alpha^{j(u)}, \chi_{j(u)}, \ldots, (\alpha^{\sigma(u)}, \chi_{\sigma(u)}) \rangle, \qquad (3)$$

где последний размещенный элемент $i_d$ в $u$ помещен в блоке $(\alpha^{j(u)}, \chi_{j(u)})$, т. е. $\alpha_{i_d}^{j(u)} = 1 \wedge \alpha_{i_d}^{j(u)-1} = 0$.

Если $i \in \bar{I}(u)$ не может быть размещен в блоках (3), то для размещения этого предмета инициализируется новый блок $\sigma(u) + 1$.

## 3. Нижние и верхние границы

**Нижние границы.** На рис. 3 изображена полученная с помощью алгоритма $\mathrm{NF}(u)$ блок-структура $\mathrm{SS}(u)$ узла $u = \langle i_1, \ldots, i_d \rangle \in V$, в которой предметы $i_1, \ldots, i_d$ уже размещены и предмет $i_d$ находится в блоке $(\alpha^{j(u)}, \chi_{j(u)})$. Эти предметы создают пустые неиспользованные остатки в контейнерах, в которые оставшиеся предметы $\bar{I}(u) := I \setminus \{i_1, \ldots, i_d\}$ могут быть размещены. Неиспользованные остатки могут быть рассмотрены как множество одномерных прутков материала длины $\lambda_k$, имеющихся в количестве $u_k$.

Сформулируем для узла $u \in V$ следующую задачу. Имеются контейнеры материала длины $\lambda_k$ и комплектности $u_k$, в которые необходимо упаковать предметы $w_i$ потребности $b_i$, $i \in \bar{I}(u)$, соблюдая при этом два условия: каждый предмет $w_i$ пакуется только один раз в один контейнер и ком-

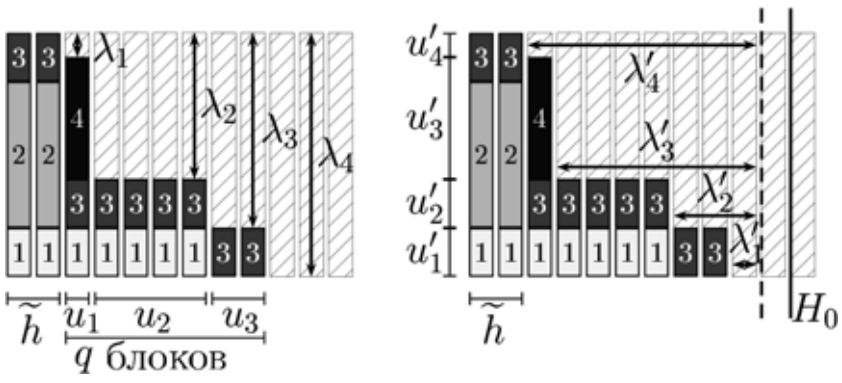



**Рис. 3. Частичная упаковка: горизонтальная (*а*) и вертикальная (*б*) релаксации**

плектность $u_k$ каждого типа контейнера не должна быть превышена. При этом число упакованных контейнеров последнего типа $\lambda_{q(u)}$ должно быть минимально. Эта задача известна как проблема одномерного раскроя с различными типами прутков [8, 9] (1D Cutting Stock Problem with Multiple Stock Lengths, 1CSPM). Для ее решения предложена релаксация на основе линейного программирования L1CSPM($u$), которая решается с использованием метода генерации столбцов. Обозначим $z^*_{LP}$ значение оптимального решения задачи L1CSPM($u$). Справедливо следующее утверждение.

**Лемма 1.** *Пусть дан узел* $u = \langle i_1, ..., i_d \rangle \in V$, $i_l \in I$, $l = 1, ..., d$, $d \leqslant m$. Число

$$lb_{LP}(u) := \sum_{j=1}^{\sigma(u)} \chi_j + \lceil z^*_{LP}(u) \rceil \qquad (4)$$

*является нижней границей для значений решений, которые могут быть получены ветвлением узла* $u$.

**Верхняя граница.** Вместе с тем, если известна верхняя граница $H_0$ (например, полученная приближенно любым эвристическим методом), тогда можно проверить, умещаются ли оставшиеся предметы в контейнеры до $H_0 - 1$. Если нет, то рассматриваемый узел может быть удален из рассмотрения. Аналогично L1CSPM($u$) можно сформулировать задачу вертикальной релаксации L1CSPM′ ($u$, $H_0$), где предметы $b_i$ потребности $w_i$ упаковываются в контейнеры длины $\lambda'_k$, комплектности $u'_k$ (рис. 3, *б*). При этом контейнеры последнего типа доступны уже в ограниченном количестве. Обозначим $z'^*_{LP}(u, H_0)$ значение оптимального решения задачи L1CSPM′ ($u$, $H_0$). Справедливо следующее утверждение.

**Лемма 2.** *Пусть дан узел* $u = \langle i_1, ..., i_d \rangle \in V$, $i_l \in I$, $l = 1, ..., d$, $d \leqslant m$. *Если*

$$z'^*_{LP}(u, H_0) \geqslant \varepsilon, \quad \varepsilon > 0, \qquad (5)$$

*то решение, которое будет получено ветвлением узла* $u$, *будет иметь значение не лучше, чем верхняя граница* $H_0$ *для значения целевой функции.*

Все узлы дерева ветвлений $T$ делятся на узлы с приоритетом и без него. В первую очередь рассматриваются узлы с приоритетом.

**Определение 2.** *Пусть дан узел* $u = \langle i_1, ..., i_d \rangle \in V$ *дерева ветвлений* $T$. *Тогда потомок* $v = \langle u, i \rangle$ *узла* $u$, $i \in \bar{I}(u)$, *называется приоритетным для заданного параметра* $r \in Z_+$, *если предмет* $i$ *находится в релаксации L1CSPM(*$u$*) в контейнере типа* $k$, *где* $k \leqslant r$.

Число приоритетных узлов в процессе ветвлений в дереве $T$ не постоянно. В процессе деления приоритетных и неприоритетных узлов могут образовываться новые приоритетные узлы. Они обрабатываются в первую очередь.

В процессе ветвления узла $u \in V$ создается узел $v$, при этом $(u, v) \in E$, только если нижняя граница (4) меньше, чем верхняя граница для дерева $T$, и не выполнено условие (5). Имеет место следующая теорема.

**Теорема 1.** *Алгоритм, основанный на методе ветвей и границ на приоритетных списках с учетом нижней границы (4) и условия (5), является точным.*

## 4. Численный эксперимент

Ввиду того, что решение задачи 1CBPP является нижней границей для 2SPP, численный эксперимент проводился на задачах 2SPP библиотеки задач Bortfeld [10]. Результаты сравнивались с известными нижними границами. Программное обеспечение было реализовано на C++ и скомпилировано gcc 4.1.2. Вычислительные эксперименты проводились на AMD Opteron 250 (2,4 GHz).

Библиотека состоит из 500 примеров. Эти примеры были случайно сгенерированы и поделены на 10 классов. Задачи из первых шести классов были предоставлены Berkey и Wang. Остальные четыре класса предоставили Martello и Vigo. Ширина полосы для каждого класса $W = 100$. Число предметов в каждом классе 20, 40, 60, 80 и 100. Для каждого числа предметов сгенерировано по 10 задач.

Для каждого примера Bortfeld также предложил нижнюю границу, среднее значение которой для каждого класса приведено в табл. 1 в столбце Bortfeld(B). Для сравнения приводится значение оптимального решения непрерывной релаксации задачи одномерного раскроя прутков (столбец LP), округленное сверху. В столбце 1CBPP(C) приведено среднее значение решений 1CBPP, из которых в шестом столбце указано число задач, решенных оптимально. Для остальных задач, которые не были решены оптимально, приведено ми-

Таблица 1

**Среднее значение нижних границ для задач Berkey & Wang и Martello & Vigo**

| Класс | N | Bortfeld(B) | LP | 1CBPP(C) | Число задач, решенных оптимально |
|---|---|---|---|---|---|
| c101 | 50 | 187,18 | 187,68 | 187,74 | 33 |
| c102 | 50 | 60,52 | 60,52 | 60,52 | 33 |
| c103 | 50 | 504,12 | 507,62 | 507,84 | 12 |
| c104 | 50 | 193,50 | 193,50 | 193,50 | 0 |
| c105 | 50 | 1613,16 | 1630,66 | 1631,72 | 26 |
| c106 | 50 | 506,40 | 506,40 | 506,40 | 0 |
| c107 | 50 | 1577,82 | 1588,76 | 1591,24 | 46 |
| c108 | 50 | 1397,92 | 1399,58 | 1399,84 | 0 |
| c109 | 50 | 3343,10 | 3344,56 | 3346,16 | 50 |
| c110 | 50 | 909,16 | 917,58 | 917,70 | 6 |
| | 500 | 1029,29 | 1033,69 | 1034,27 | 206 |

Таблица 2

**Сравнение абсолютных значений нижних границ Bortfeld(B), LP и 1CBPP(C)**

| Класс | B > LP | B > C | LP > B | LP > C | C > B | C > LP |
|---|---|---|---|---|---|---|
| c101 | 1 | 0 | 15 | 0 | 16 | 3 |
| c102 | 0 | 0 | 0 | 0 | 0 | 0 |
| c103 | 0 | 0 | 33 | 0 | 36 | 6 |
| c104 | 0 | 0 | 0 | 0 | 0 | 0 |
| c105 | 1 | 0 | 44 | 0 | 45 | 10 |
| c106 | 0 | 0 | 0 | 0 | 0 | 0 |
| c107 | 9 | 0 | 27 | 0 | 27 | 16 |
| c108 | 0 | 0 | 15 | 0 | 16 | 6 |
| c109 | 3 | 0 | 5 | 0 | 5 | 4 |
| c110 | 0 | 0 | 39 | 0 | 39 | 1 |
| | 14 | 0 | 178 | 0 | 184 | 46 |

нимальное значение нижней границы среди всех необработанных узлов. На каждый пример налагалось ограничение времени счета в 3 мин.

Случаи, когда одна из границ превышает другие, приведены в табл. 2, из которой видно, что в 178 и 184 случаях нижние границы LP (столбец LP > B) и точное решение задачи 1CBPP (столбец C > B) соответственно больше, чем нижняя граница, приведенная Bortfeld. В 46 случаях 1CBPP оказалась лучше, чем нижняя граница LP (столбец C > LP). Из столбцов B > C и LP > C видно, что нижняя граница, основанная на точном решении 1CBPP, на данных задачах строго лучше, чем две другие известные из литературы нижние границы.

Результаты численного эксперимента:

- На предварительном этапе для каждой задачи выполнен препроцессинг. Это позволило во многих случаях сократить размерность решаемых задач.
- На классах c104, c106, c108 ни одного примера не удалось решить оптимально. Эти классы можно обозначить как наиболее сложные для разработанного алгоритма. И напротив, практически все задачи из классов c107, c109 были решены оптимально. В целом, чуть меньше половины задач было решено оптимально, что говорит об удовлетворительной производительности алгоритма.
- Нижняя граница на основе решения задачи 1CBPP всегда не меньше границ Bortfeld, LP.

## Заключение

В данной статье предложена модификация на основе метода ветвей и границ с использованием линейного программирования. Основываясь на том факте, что решение одномерной продолженной задачи является нижней границей для задачи двухмерной упаковки, исследование эффективности проводилось на двухмерных задачах упаковки. Несмотря на большую размерность задач (20—100), 40 % из них были решены оптимально. Для сравнения качества полученных решений использовались известные границы для двухмерной упаковки. Лучшие результаты показал предложенный метод.

### Список литературы


1. **Мухачева Э. А., Мухачева А. С.** Задача прямоугольной упаковки: методы локального поиска оптимума на базе блочных структур // Автоматика и телемеханика. 2004. № 2. С. 10—15.
2. **Мухачева Э. А., Мухачева А. С., Чиглинцев А. В.** Генетический алгоритм блочной структуры в задачах двухмерной упаковки // Информационные технологии. 1999. № 11. С. 13—18.
3. **Martello S., Monachi M. and Vigo D.** An exact approach to the strip-packing problem // INFORMS Journal on Computing. 2003. N 15(3). P. 310—319.
4. **Картак В. М., Месягутов М. А., Мухачева Э. А., Филиппова А. С.** Локальный поиск ортогональных упаковок с использованием нижних границ // Автоматика и телемеханика. 2009. № 6. С. 167—180.
5. **Житников В. П., Филиппова А. С.** Задача прямоугольной упаковки в полубесконечную полосу: поиск решения в окрестности локальной нижней границы // Информационные технологии. 2007. № 5. С. 55—62.
6. **Hartmarm S.** Project Scheduling under limited resources. Models, methods and applications. Berlin: Springer, 1999.
7. **Belov G., Scheithauer G., and Mukhacheva E. A.** One-dimensional heuristics adapted for two-dimensional rectangular strip packing // The Journal of the Operational Research Society. 2008. N 59(6). P. 823—832.
8. **Belov G. and Scheithauer G.** A cutting plane algorithm for the one-dimensional cutting stock problem with multiple stock lengths // European Journal of Operational Research. 2002. N 141(2). P. 274—294.
9. **Шайтхауэр Г., Мухачева А. С., Белов Г. Н., Мухачева Э. А.** Проектирование одномерного раскроя материала различной длины на базе непрерывной релаксации и метода отсекающих плоскостей // Информационные технологии. 2004. № 1. С. 28—36.
10. **Bortfeld A.** A genetic algorithm for the two-dimensional strip packing problem with rectangular prices // European Journal of Operational Research. 2006. N 172(3). P. 814—837.

УДК 519.863

**Э. А. Мухачева,** д-р техн. наук, проф.,
**Р. С. Валеев,** аспирант,
Уфимский государственный авиационный технический университет,
e-mail: valeevrus@inbox.ru

# Локальный поиск размещения товарных позиций на базе анализа их номенклатурной принадлежности

*Рассматривается задача прямоугольного размещения в условиях складирования грузов в многомодульных комплексах. Для конструирования допустимых размещений используется алгоритм следующий подходящий (NF) с поиском лучших решений в окрестности с нижней локальной границей. Она может быть найдена путем решения задачи одномерного раскроя с дополнительными ограничениями. Для поиска исходного допустимого решения предложена модификация алгоритма (0—1)-рюкзак.*

***Ключевые слова:*** *прямоугольное размещение, складирование, (0—1)-рюкзак, ABC-анализ*

## Введение

С ростом номенклатуры товарных единиц на многомодульных комплексах (складах) появилась потребность в применении методов, систематизирующих их по различным критериям. В логистике широкое распространение получил метод *ABC*. Он позволил сократить время, затрачиваемое на складирование грузов, а также на формирование заказов. Применение метода *ABC* к решению задач оптимального размещения товарных позиций на складах рассматривалось различными авторами [1, 2]. При этом в качестве критерия оптимальности принимается суммарный пробег транспортного средства, доставляющего товары на склады. В этой статье критерием является плотность размещения товаров в боксах и на стеллажах склада. При этом нас интересует метод размещения и программа управления им при одно- и двухмерном расположении предметов. Для простоты изложения будем считать, что суммарная площадь размещаемых грузов не более свободной площади стеллажа. Это позволит нам рассматривать задачу размещения предметов на полубесконечной полосе.

Рассматривается задача ортогонального размещения множества $\{w,\ l\}$, $w = (w_1,\ w_2,\ ...,\ w_m)$, $l = (l_1,\ l_2,\ ...,\ l_m)$, прямоугольных предметов ширины $w_i$ и длины $l_i$, $i = \overline{1, m}$, в полосу ширины $W$ и полубесконечной длины (2 Dimensional Strip Packing, 2DSP). При этом будем считать, что величины $W$, $w_i$, $l_i$, $i = \overline{1, m}$, — целые числа. Ортогональное размещение прямоугольников в полосе без перекрытия друг с другом и гранями полосы называется допустимой прямоугольной упаковкой (Rectangular Packing, RP). На множестве допустимых RP требуется минимизировать длину $L$ занятой части полосы. Задача является NP-трудной. Поэтому для ее решения часто применяют эвристические и приближенные алгоритмы. Среди них отметим конструктивные эвристики и вероятностные методы локального поиска. Конструктивные эвристики находят допустимое решение за один проход. Это могут быть простые эвристические алгоритмы типа *подходящий* [3]. Их применяют в качестве декодеров алгоритмов, воспроизводящих упаковку по ее коду. Другой тип конструктивных эвристик — однопроходные сложные алгоритмы, с их помощью удается найти допустимое решение, близкое к оптимальному [4]. Примерами методов локального поиска являются метаэвристики, в том числе эволюционные алгоритмы. Это одноточечные алгоритмы случайного поиска, $(1 + 1) - EA$ [5]; алгоритмы имитации отжига [6]; генетические алгоритмы [7, 8]. Вычислительная схема эволюционных алгоритмов состоит в следующем: находят начальное допустимое решение и его окрестность (множество допустимых решений, близких по некоторому критерию к начальному); далее в окрестности ищется локально-оптимальное решение. Процедура повторяется для нового начального решения, которое находят с помощью *оператора мутации*. Процесс продолжается пока не будет найден глобальный оптимум. Поиск последнего представляет нетривиальный процесс. Ограничиваются нахождением допустимой упаковки и оценивают качество решения. Что касается локального оптимума, то удалось построить окрестность с локальной нижней границей, приближенной к локальному оптимуму [9]. При этом поиск в окрестности осуществляется путем решения одномерной задачи раскроя с дополнительными ограничениями. Эти понятия и соответствующие утверждения введены А. С. Филипповой. В данной статье вначале ищется близкое к оптимальному локально-оптимальное решение задачи линейного раскроя с дополнительными ограничениями и отвечающие ему перестановки элементов. Затем применяют алгоритм "следующий подходящий" (*NF*) к найденным перестановкам и оценивают полученное решение.

### 1. Λ-окрестность допустимой упаковки

Рассмотрим индивидуальную задачу ортогональной упаковки ($P$, $L$), где $P$ — конечное множество упаковок $p$ с исходными данными $\langle W; m; w; l\rangle$ и упаковку $p^* \in P$, доставляющую минимум функции цели $L = \max_i(x_i + l_i)$. Упаковка $p^*$ длины $L^* = \min L$ — глобальный минимум. Для каждой допустимой упаковки $p \in P$ определим функцию окрестности Λ, которая задает для $p$ множество соседних решений, в некотором смысле близких к данному. С каждой индивидуальной задачей ($P$, $L$) связана следующая задача одномерного сквозного размещения. Ее принято называть задачей раскроя (One Dimensional Cutting Stock Problem, 1DCSP). Положим $Z = W$ — длина одномерного поддона; $\lambda = w$ — вектор длин предметов; $\beta = l$ — вектор требуемого числа предметов (ассортиментный вектор). Требуется минимизировать Λ — число затраченных поддонов. Рассмотрим задачу 1DCS с дополнительными ограничениями на допустимое решение:

$1^0$. В поддон можно укладывать только различные предметы (*разнородность*).

$2^0$. Если не все предметы типа $i$ умещаются в поддон, то оставшиеся размещаются в следующих за ним поддонах (*продолженность*).

Задачу 1DCS с ограничениями $1^0$ и $2^0$ называют *прямоугольно ориентированной* (Rectangular Oriented Cutting Stock Problem, ROCSP), а ее допустимое решение — прямоугольно ориентированным линейным раскроем (Rectangular Oriented Linear Cutting, ROLC) [10]. Функцией цели в задаче ROCS является число Λ затраченных контейнеров. Если при решении задачи ROCS множества предметов $\{w, l\}$ с очередностью размещения, заданной списком π, применяется алгоритм *NF*, то прямоугольно ориентированный линейный раскрой запишем как $\mathrm{ROLC}_{NF}((\lambda, \beta), \pi)$.

**Пассивные перестановки.** Интересующая нас Λ-окрестность определяется перестановками элементов в списке π, в результате которых получают прямоугольно ориентированные одномерные раскрои, различающиеся только порядком следования предметов в одном и том же поддоне. Будем именовать такие раскрои и отвечающие им списки π-*эквивалентными*. Пусть имеется одномерный линейный раскрой $\mathrm{ROLC}_{NF}((\lambda, \beta), \pi)$.

**Определение 1.** *Перестановка элементов в списке π называется пассивной, если применение NF к полученному списку π′ приведет к раскрою* $ROLC_{NF}((\lambda, \beta), \pi')$, *эквивалентному исходному* $ROLC_{NF}((\lambda, \beta), \pi)$.

Число занятых поддонов в обоих раскроях одинаковое. Очевидным является следующее утверждение.

**Лемма 1.** *(Критерий пассивной перестановки). Пусть в блоке $S_j$ блок-структуры ROLC выбраны два элемента $i_1(\pi)$ и $i_2(\pi)$, удовлетворяющие следующим условиям*:

- *оба элемента $i_1$ и $i_2$ начинаются в блоке j, и элемент $i_1$ предшествует элементу $i_2$ в списке π*;
- *длина $\lambda_{i2}$ элемента $i_2$ превосходит остаточную емкость $r_{j-1}$ предыдущего блока $S_{j-1}$.*

*Тогда пара $(i_1(\pi), i_2(\pi))$ является пассивной в перестановке π.*

Пусть имеется ортогональная упаковка $p \in P$, отвечающий ей список $\pi(p)$ и прямоугольно ориентированный линейный раскрой $\mathrm{ROLC}_{NF}((\lambda, \beta), \pi(p))$ длины Λ. Тогда построение соседнего с $p$ решения состоит из выполнения следующих процедур.

Λ1. *Построение последовательности π′, эквивалентной π*: случайный выбор подходящего раскроя и случайный выбор в соответствующем столбце пары элементов, удовлетворяющий условиям леммы 1. Перестановка пассивных элементов в отмеченном блоке — новая последовательность π′, эквивалентная π.

Λ2. *Конструирование соседнего с p решения*: при входных данных $\langle W; m; w; l\rangle$ применяем алгоритм *NF* к списку π′, получаем $p' = RP_{NF}((w, l), \pi')$ длины $L'$ и вычисляем координаты прямоугольников.

Повторяем процесс построения соседних решений, получаем окрестность решения ортогональной упаковки $p$, которую будем именовать Λ-окрестностью и обозначать $\Lambda(p)$.

### 2. Нижняя граница в Λ-окрестности

**NF-активные упаковки.** Частным случаем задачи 1DCS одномерного раскроя является одномерная задача упаковки One Dimensional Bin Packing, 1DBP), в которой все предметы различные, т. е. $\beta_i = 1$, $i = \overline{1, m}$. Допустимое решение 1DBP можно представить в виде списка $S = (S_1; S_2; ...; S_j, ...; S_N)$, где $S_j$ — кортеж (последовательность номеров предметов, помещенных в $j$-й поддон). Кортежу $S_j$ отвечает блок $j$ с интенсивностью применения $\chi_j = 1$. Порядок π размещения элементов можно получить конкатенацией редуцированных кортежей $S_j$, $j = \overline{1, \Lambda}$.

При исходных данных задачи 1DBP воспользуемся для ее решения алгоритмом *NF*, считая при этом известной перестановку π, устанавливающую порядок размещения предметов. Полученную упаковку множества предметов $\{\lambda\}$ обозначим $P_{NF}(\lambda, \pi)$.

**Определение 2.** *Допустимая одномерная упаковка p множества прямоугольников из m предметов называется NF-активной при фиксированной перестановке π, если* $P_{NF}(\lambda, \pi)$ *совпадает с p.*

Это определение введено В. В. Залюбовским в работе [12], посвященной исследованию одномерных упаковок. Пусть $p$ — допустимая упаковка множества предметов $\{\lambda\}$ и $|p|$ — число занятых поддонов. Справедливо следующее утверждение.

**Лемма 2.** (В. В. Залюбовский) *Для любой допустимой упаковки $p$ множества предметов $\{\lambda\}$ существует NF-активная упаковка $p'$ такая, что*

$$|p'| \leqslant |p|. \qquad (1)$$

В то же время задача 1DCS является простым обобщением проблемы упаковки. А именно: при $\beta_i > 1$, $\lambda_i$ повторяется $\beta_i$ раз, т. е. 1DCSP — задача упаковки 1DBP с повторяющимися предметами. Тогда модификация *NF* с перестановкой $\pi$ приведет к *NF*-активному линейному раскрою. Что касается задачи ROCS, то учет условий $1^0$ и $2^0$ легко реализуется в рамках указанных модификаций.

**Локальная нижняя граница Λ-окрестности прямоугольной упаковки.** Пусть известна допустимая упаковка $p \in P$ множества предметов $\{w, l\}$ и ей отвечает перестановка $\pi(p)$ и $\mathrm{ROLC}_{NF}((\lambda, \beta), \pi(p))$ длины $\Lambda$.

Следствием из леммы 2 является следующее утверждение.

**Лемма 3.** *Для любой прямоугольной упаковки $p$ множества предметов $\{w, l\}$ и NF-активного прямоугольно ориентированного линейного раскроя $ROLC_{NF}((\lambda, \beta), \pi(p))$, $\lambda = w$, $\beta = l$, справедливо неравенство*

$$|ROLC_{NF}((\lambda, \beta), \pi(p))| \leqslant |p(w, l)|, \qquad (2)$$

*какой бы алгоритм для воспроизведения $p$ ранее не был применен.*

**Определение 3.** *Нижней локальной границей в $\Lambda$-окрестности упаковки $p \in P$ назовем число $h$ такое, что для любой упаковки $p$ из этой окрестности*

$$h \leqslant |p|. \qquad (3)$$

Из леммы 3 согласно определению 3 следует основное утверждение:

**Теорема 4.** (А. С. Филиппова) *Решение $\Lambda = ROCL_{NF}((\lambda, \beta), \pi(p))$ задачи ROCS с исходной информацией $\langle W; m; \lambda = w; \beta = l\rangle$, $\pi(p)$, является локальной нижней границей в $\Lambda$-окрестности упаковки $p$ множества предметов $\{w, l\}$.*

**Определение 4.** *Если для некоторой упаковки $p^*$ из $\Lambda$-окрестности $p$ имеем $|p^*| = \Lambda$, то $\Lambda$ — локально-оптимальный минимум.*

Возможны несколько стратегий построения и использования $\Lambda$-окрестности для поиска локально-оптимального решения. Остановимся на одной из них. Стратегия $1D$—$\Lambda$ основана на том, что $\Lambda$-границу можно вычислить до построения текущего решения. Для этого решается задача ROCS. Ее решение — начальная локальная $\Lambda$-граница. Для списка $\pi$, отвечающего ROLC, применяя алгоритм *NF*, находят начальную упаковку $p = RP$. Выполняя пассивные перестановки в списке $\pi$, строят окрестность и находят в ней упаковку с рекордным значением длины $L$. Далее, если $L$ не подходит, то применяя одноточечный эволюционный алгоритм $(1 + 1) - EA$ [5], выполняется генерация новой окрестности с меньшим значением $\Lambda$ и поиском в ней упаковки с рекордным значением $L' < L$. Эта стратегия без применения метаэвристики $(1 + 1) - EA$ применяется для размещения предметов из группы $C$ с применением метаэвристики — для группы $B$.

### 3. Использование алгоритма (0—1)-рюкзак для построения исходной Λ-границы

В качестве нижней границы при размещении предметов из группы $A$ может использоваться любое число, не превосходящее оптимального решения задачи 2DBP. Воспользуемся локальными границами, полученными как хорошие допустимые решения. Подходы для их получения таковы: при исходных данных 2DBP формируется задача ROCS; для нее находят оптимальное или близкое к нему (рациональное) решение $\Lambda$. Это число и принимается в качестве локальной границы 2DBP. Далее осуществляется поиск решений $L$ в $\Lambda$-окрестности. Если $L = \Lambda$, то это означает достижение локальной границы; если нет, то подсчитывается отклонение *gap* от нее. Остается решить вопрос об алгоритме, с помощью которого следует определять $\Lambda$-границу. В качестве таковых воспользуемся модификацией эффективных алгоритмов для решения задачи 1DCS с учетом условий $1^0$ и $2^0$. Одним из таких алгоритмов является (0—1)-рюкзак.

Вначале приведем краткую характеристику алгоритма (0—1)-рюкзак для решения задачи 1DCS. Используется следующая формулировка задачи 0—1-рюкзак [13]: даны целые числа $\lambda_i$, $i = 1, 2, ..., m$ и $W$. Требуется определить, существует ли такое подмножество $I$ множества $\{1, 2, ..., m\}$, что $\sum\limits_{i \in I} \lambda_i = W$.

Известно, что задача "0—1-рюкзак" NP-полная, и существует псевдополиномиальный алгоритм (*Pseudo-polynomial Algorithm*, PA) [13], корректно решающий эту задачу за время $O(m \cdot W)$. Он строит множества $M_i$, $i = 1, 2, ..., m$, содержащие всевозможные суммы целых чисел $\lambda_i$, $i = 1, 2, ..., m$, не превышающие $W$.

*Алгоритм PA.*

*Шаг 1.* $M_0 = \{0\}$.

*Шаг 2.* Для $i = 1, 2, ..., m$ выполнить:

$M_i = \varnothing$;

для каждого целого числа $\lambda$ из $M_{i-1}$ выполнить: добавить к $M_i$ целые числа $w$ и $\lambda + \lambda_i$, если $\lambda + \lambda_i \leqslant W$ и $\lambda + \lambda_i \notin M_i$.

*Шаг 3.* Заключить, что данная задача имеет решение только в том случае, если $W \in M_m$.

*Конец алгоритма PA.*

В работе [4] приведена модификация алгоритма *PA*, которая состоит в следующем: между числами $\lambda_i$, $i = 1, 2, ..., m$, и множеством натуральных чисел $\{1, 2, ..., m\}$ устанавливается взаимно однозначное соответствие $\lambda_1 \to 1$, $\lambda_2 \to 2$, ..., $\lambda_m \to m$ и вводятся в рассмотрение кортежи вида $(i_1, ..., i_r)$, где $1, ..., r$ — номера позиций чисел $\lambda_i$, $i = 1, 2, ..., m$. Пусть $K$ — множество всех кортежей. Рассматривается задача поиска кортежей (*Tuple Search Problem*, *TSP*).

*Задача TSP.* Даны целые числа $\lambda_i$, $i = 1, 2, ..., m$ и $W$. Требуется сформировать такое множество $K$ кортежей, чтобы выполнялись условия:

1. $\sum\limits_{i \in I} \lambda_i \leqslant W$, где $I$ — подмножество множества $\{1, 2, ..., m\}$;

2. Добавление любого элемента из $I$, не входящего в кортеж, нарушает условие 1.

Для решения *TSP* применяется модифицированный алгоритм *PA* (*Modified Pseudo-polynomial Algorithm, MPA*) [4], который заключается в том, что для каждой суммы $\lambda + \lambda_i \leqslant W$, $i = 1, 2, ..., m$, $\lambda \in M_{i-1}$, находится кортеж из номеров прямоугольников, ширины которых вошли в сумму $\lambda + \lambda_i$. При этом кортеж находится даже в том случае, если число $\lambda + \lambda_i \leqslant W$ уже есть в множестве $M_i$. Это позволяет находить не одну, а несколько комбинаций чисел $\lambda_i$, $i = 1, 2, ..., m$, которые в сумме будут давать число $\lambda \in M_i$, $i = 1, 2, ..., m$. Совокупность найденных кортежей образует множество $K$.

Ниже представлены основные шаги алгоритма *MPA*.

*Модифицированный алгоритм MPA*

*Шаг 1.* $M_0 = \{0\}$, $K = \varnothing$.

*Шаг 2.* Для $i = 1, 2, ..., m$ выполнить:

построение множеств $M_i$:

$M_i = \varnothing$;

для каждого целого числа $\lambda$ из $M_{i-1}$ выполнить:

добавить к $M_i$ целые числа $\lambda$ и $\lambda + \lambda_i$, если $\lambda + \lambda_i \leqslant W$ и $\lambda + \lambda_i \in M_i$.

Формирование множества кортежей $K$:

для каждого целого числа $\lambda$ из $M_{i-1}$ выполнить:

найти кортежи, состоящие из номеров тех чисел, которые вошли в сумму $\lambda + \lambda_i$, если $\lambda + \lambda_i \leqslant W$ (кортежи находятся и в том случае, если число $\lambda + \lambda_i \leqslant W$ уже есть в $M_i$);

*Шаг 3.* Найти $\lambda_{\max} = \max\limits_{\lambda \in M_m} \lambda$ и соответствующие $\lambda_{\max}$ кортежи из $K$.

*Конец алгоритма MPA*

Для решения задачи ROCS предлагается использовать алгоритм *MPA* с учетом ограничений $1^0$ и $2^0$. Назовем его прямоугольно ориентированным (*ROMPA*). Приведем схему этого алгоритма.

*Схема алгоритма ROMPA*

1. Создание множества кортежей $K$ на базе алгоритма *MPA*.

2. До тех пор, пока все $b_i$, $i = 1, 2, ..., m$, не равны 0, выполнить:

найти $\max b_i$ и $\arg\max b_i$, $i = 1, 2, ..., m$

выбрать из множества $K$ кортежи $r_j$, содержащие в качестве элемента номер $\arg\max b_i$;

выбрать из множества $\{r_j\}$ кортеж наибольшего веса, т. е. кортеж, соответствующий $\lambda_{\max}$, $\lambda_{\max} = \max\limits_{\lambda \in M_m} \lambda$.

Заполнить поддон выбранным кортежем:

2.4.1. Если $\lambda_{\max} = W$, то поддон заполняется без остатка.

2.4.2. Если $\lambda_{\max} \neq W$ (есть остаток в поддоне), то остаток поддона заполнить подходящим кортежем с учетом условий $1^0$ и $2^0$.

Уменьшить на 1 количество $b_i$ для всех элементов кортежа.

Удалить из множества кортежей все кортежи с $b_i = 0$.

Конец_пока

*Конец алгоритма*

Пример работы алгоритма *ROMPA* для задачи 1DCS.

*Вход*: $W = 14$; $\lambda = (10, 2, 4, 12, 8)$; $m = 16$; $b = (2, 4, 5, 2)$

*Выход*: Минимальное число $N$ контейнеров с учетом условий $1^0$ и $2^0$.

Рассмотрим работу алгоритма *ROMPA* на данном примере. В табл. 1 приведены кортежи, полученные с помощью алгоритма *MPA*, а в табл. 2 показано заполнение контейнеров полученными кортежами с учетом условий $1^0$ и $2^0$.

После того, как задача ROCS решена и найдено число $\Lambda$ затраченных поддонов, число $\Lambda$ объявляется нижней локальной границей. Далее вос-

Таблица 1

**Создание множества кортежей с помощью MPA**

| $w_i$ | 2 | 4 | 6 | 8 | 10 | 12 | 14 |
|---|---|---|---|---|---|---|---|
| Кортежи | (3) | (3) | (2, 3) | (5) | (1); (2, 5) | (1, 2); (4); (3, 5) | (1, 3); (2, 4); (2, 3, 5) |

Таблица 2

**Заполнение поддонов с учетом условий $1^0$ и $2^0$**

| Номера поддонов | № 1 | № 2 | № 3 | № 4 | № 5 | № 6 | № 7 |
|---|---|---|---|---|---|---|---|
| Кортежи | (1, 3) | (2, 4) | (1, 3) | (2, 3, 5) | (2, 3, 5) | (3, 5, 4) | (4, 2) |

Таблица 3

**Значения локальных нижних границ для классов C01—C10**

| Пример | C01 | C02 | C03 | C04 | C05 | C06 | C07 | C08 | C09 | C10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1—10 | 59,34 | 21,1 | 161,8 | 62,8 | 543,2 | 164,4 | 502,7 | 416,3 | 1108 | 322,5 |
| 11—20 | 120,57 | 40,54 | 335 | 131,8 | 1078,8 | 346,3 | 1033,7 | 961,2 | 2190,9 | 707,7 |
| 21—30 | 184,34 | 60,1 | 513,2 | 193,8 | 1658,6 | 541,4 | 1530,54 | 1414,1 | 3410,1 | 1032,8 |
| 31—40 | 260,61 | 83,7 | 705,9 | 275,7 | 2245,9 | 693,9 | 2237,3 | 1972,1 | 4588,1 | 1206,7 |
| 41—50 | 315,1 | 99,2 | 853,9 | 327,4 | 2689,6 | 883,5 | 2656 | 2460,4 | 5434,9 | 1580,3 |
| Среднее | 187,99 | 60,9 | 513,96 | 198,3 | 1643,22 | 525,9 | 1592,04 | 1444,82 | 3346,4 | 935,7 |
| Среднее SPGAL | 187,9 | 60,7 | 513,4 | 197,9 | 1647,0 | 524,7 | 1592,7 | 1442,1 | 3347,7 | 933,3 |

станавливаются исходные данные задачи 2DSP и список $\pi$, устанавливающий порядок включения деталей в одномерный раскрой. При указанных данных применяется алгоритм следующий подходящий (*NF*) для конструирования прямоугольной упаковки. Пусть получена упаковка длины $L$. Возможны следующие два случая: а) $L = \Lambda$, локальная граница достигнута, конец; б) $L > \Lambda$, рекорд сохраняется и переходят на перестановку пассивных элементов в $\pi$. Получен новый список $\pi'$, с ним вновь решается задача конструирования прямоугольной упаковки в полосу и т. д. Поиск лучшей упаковки в окрестности завершается достижением $\Lambda$ или выполнением заданного числа итераций.

Были проведены численные эксперименты на примерах, описанных в работах J. Berkey и P. Wang (1987), а также S. Martello и D. Vigo (1998) [13, 14]. Результаты численных экспериментов (табл. 3) подтверждают достаточную эффективность метода расчета оптимума с использованием одномерной задачи (0—1)-рюкзак. Кроме того, для сравнения приведены значения $L$, полученные алгоритмом SPGAL Bortfeld [16]. Решения с использованием задачи (0—1)-рюкзак сопоставимы или оказываются лучше результатов Bortfeld.

## 3. Применение локального поиска в Λ-окрестности для решения задач размещения товаров в складской логистике

Рассмотренный в предыдущем разделе метод решения задачи прямоугольной упаковки предлагается применять при размещении различных товарных единиц в логистических хабах — крупных мультимодальных комплексах (складах), предназначенных для обработки большого количества крупногабаритных грузов, упакованных в ящики или поддоны. При этом комплексы условно разделяют на две основные зоны: зону приемки товара и зону хранения товара. Последняя зона разбивается на две подзоны: "холодную", в которой хранятся редко востребованные товары; "горячую" — для складирования товаров с высокой оборачиваемостью. При подготовке товаров к складированию обычно выполняются следующие операции [17]:

- разгрузка товарных единиц в зоне приемки товара;
- формирование различных групп товаров по *ABC*-методу. Этот метод был предложен Парето и основан на следующем эмпирическом правиле: основная доля продаж формируется за счет относительно небольшого ассортимента товаров. В связи с этим товары разделяются на три группы: группа *A* — товары с максимальным объемом продаж; группа *B* — со средними показателями продаж; группа *C* — с наименьшим объемом продаж, требующих длительного хранения. Выделим дополнительную группу товаров — группу *D*, которая может быть востребована клиентами в порядке, заданном расписанием;
- перемещение товарных единиц к местам хранения с помощью подъемно-транспортных средств;
- размещение товарных единиц каждой группы на полках высотных стеллажей в зонах хранения с максимальным использованием мест хранения. При этом для сокращения времени размещения разработанный метод упаковки позволяет получить карту размещения товаров, которая предусматривает постоянные места их хранения. Для сокращения времени поиска нужного товара при размещении товаров группы *A* используются, как правило, нижние полки стеллажей, близкие к выходу; для товарных единиц группы *B* — вторые ярусы стеллажей, и, наконец, для товаров группы *C* — верхние полки стеллажей. Поскольку товары группы *A* являются наиболее востребованными, для их размещения достаточно применять простые методы типа "следующий подходящий" (*NF*), либо эволюционную метаэвристику (1 + 1) − *EA* декодером *NF*;
- отборка товаров из мест хранения;
- комплектование и упаковка товаров;
- погрузка товаров в транспортное средство.


**Рис. 2. Логистическая цепочка размещения товарных единиц на складе**

В связи со сказанным общая схема размещения товарных единиц на стеллажах состоит в следующем:

1. Формирование структуры складского комплекса, включая зоны хранения.

2. Распределение товара на группы по *ABC*-методу.

3. Выбор зоны хранения — "горячей" или "холодной" в соответствии с группой товара.

4. Выбор области хранения — соответствующей полки стеллажа для размещения товара определенной группы.

5. Если размещается товар группы *A*, то размещение осуществляется с помощью стратегии *NF*, либо применяется метаэвристика $(1 + 1) - EA$.

6. Если размещаются товары групп *B* или *C*, то применяется алгоритм *ROMPA*.

7. Конец размещения.

На рис. 1 (см. третью сторону обложки) приведена карта размещения товарных единиц, распределенных по группам *A*, *B*, *C* на полках стеллажей. При этом товары группы *A* (на рис. 1 выделены красным цветом) размещаются ближе к зоне приемки товара, в "горячей" зоне; товары группы *B* (на рис. 1 выделены зеленым цветом) и товары группы *C* (на рис. 1 выделены желтым цветом) — менее востребованные — размещены в "холодной" зоне склада. На рис. 2 приведена общая схема организации логистической цепочки на складе.

## Заключение

Предложен метод локального поиска решения одномерной задачи раскроя с дополнительными ограничениями. Ее исходные данные определяются входной информацией двумерного размещения. Полученное решение $\Lambda$ является локальной нижней границей в одноточечной метаэвристике поиска рационального решения двухмерной упаковки. Построена схема поиска, которая включает следующие процедуры: решение ROCS и определение $\Lambda$, $\pi$; решение задачи 2DSP с $\pi$ и определение $L$; сравнение $\Lambda$ и $L$ и принятие альтернативы: продолжить или закончить поиск. Качество полученного решения зависит от близости $\Lambda$ к глобальной границе, от способа организации поиска и от исходных данных (геометрии) задачи. Разработанный метод локального поиска применяется при размещении товарных единиц на стеллажах в складских помещениях. Полученные решения 2DSP с оценкой $L$ вполне могут использоваться в качестве верхней границы с простой структурой упаковки.

### Список литературы


1. **Гаджинский А. М.** Логистика: учеб. пособие для высших учебных заведений. М.: ИВЦ "Маркетинг", 2000. 375 с.
2. **Лукинский В. С.** и др. Модели и методы теории логистики: учеб. пособие, 2-е изд. / Под ред. В. С. Лукинского. СПб: Питер, 2008. 448 с.
3. **Lodi A., Martello S., Vigo D.** Recent advance on two-dimensional bin packing problems // Discrete Applied Mathematics. 2002. P. 379—396.
4. **Мухачева Э. А., Валеева А. Ф.** Метод динамического перебора в задаче двухмерной упаковки // Информационные технологии. 2000. № 5. С. 30—37.
5. **Борисовский П. А., Еремеев А. В.** О сравнении некоторых эволюционных алгоритмов // Автоматика и телемеханика. 2004. № 3. С. 3—9.
6. **Loris F.** An application of simulated annealing to the cutting stock problem // European journal of Operational Research. 1999. N 114. P. 814—837.
7. **Bortfeld A.** A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces // European journal of Operational Research. 2006. N 172(3). P. 814—837.
8. **Мухачева Э. А., Мухачева А. С., Чиглинцев А. В.** Генетический алгоритм блочной структуры в задачах двухмерной упаковки // Информационные технологии. 1999. № 11. С. 13—18.
9. **Житников В. П., Филиппова А. С.** Задача прямоугольной упаковки в полубесконечную полосу: поиск решения в окрестности локальной нижней границы // Информационные технологии. 2007. № 5. С. 55—62.
10. **Мухачева Э. А., Мухачева А. С.** Задача прямоугольной упаковки: методы локального поиска оптимума на базе блочных структур // Автоматика и телемеханика. 2004. № 2. С. 101—111.
11. **Залюбовский В. В.** О представлении перестановками допустимых решений одномерной задачи упаковки в контейнеры // Труды XIII Байкальской школы-семинара. Иркутск. 2005. Том 1. С. 461—468.
12. **Пападимитриу Х., Стайглиц К.** Комбинаторная оптимизация. Алгоритмы и сложность. М.: Мир, 1998. 512 с.
13. **Berkey J. O., Wang P. Y.** Two-dimensional finite bin-packing algorithms // J. Oper. Res. Soc. 1987. N 38(5). P. 423—429.
14. **Martello S., Vigo D.** Exact solution of the finite two-dimensional bin packing problem // Management Science. 1998. N 44. P. 388—399.
15. **Картак В. М., Месягутов М. А., Мухачева Э. А., Филиппова А. С.** Локальный поиск ортогональных упаковок с использованием нижних границ // Автоматика и телемеханика. 2009. № 6. С. 167—180.
16. **Bortfeld A.** A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces // European journal of Operational Research. 2006. N 172(8). P. 814—837.
17. **Степанов В. И.** Логистика: учеб. М.: Проспект, 2007. 485 с.

УДК 556.536.013:004.032.24

**В. П. Май,** канд. техн. наук, вед. научн. сотр.,
Институт автоматики и процессов управления
ДВО РАН, г. Владивосток,
e-mail: may@iacp.dvo.ru,
**Д. В. Мукомел,** вед. инженер-программист,
Тихоокеанский океанологический институт
ДВО РАН, г. Владивосток

# Моделирование поверхностного водостока с реализацией параллельных вычислений на многопроцессорной системе[1]

*Выбрана модель и разработана вычислительная схема для расчета динамики поверхностного водостока на рельефе местности. Разработан и реализован алгоритм параллельных вычислений на компьютерном кластере с учетом динамической нагрузки вычислительной среды. Приведен пример применения результатов работы для небольшого водосбора.*

***Ключевые слова:*** *моделирование, поверхностный водосток, вычислительная схема, алгоритм параллельных вычислений*

## Введение

Система предварительного прогноза последствий интенсивных осадков в виде тайфунов и ливневых дождей позволит успешнее бороться со стихией и своевременно проводить необходимые мероприятия в районах предполагаемого затопления.

Из работ, связанных с этой проблемой, следует выделить систему моделей, разработанную в Институте водных проблем РАН [1, 2]. Среди зарубежных можно отметить Европейскую гидрологическую систему SHE [3, 4], а также разработки американских специалистов — HSPF, HEC-HMS [5, 6, 7]. Однако в этих разработках тогда не было возможности использовать достижения современных вычислительных средств, которые позволяют значительно сократить время работы систем и повысить эффективность используемых моделей.

Данная работа посвящена разработке вычислительной схемы для модели поверхностного водостока на рельефе местности с реализацией параллельных вычислений на основе многопроцессорной системы.

## 1. Модель поверхностного водостока

Если совокупно рассмотреть процессы, происходящие с влагой на территории бассейна реки и оказывающие непосредственное влияние на водосток, то схематично их можно отобразить в виде рис. 1 [5].

В данной достаточно простой интерпретации осадки падают на растительность, земную поверхность и открытую поверхность водоемов. В реальной гидрологической системе значительное количество воды, выпавшее в качестве осадков, возвращается в атмосферу через испарение. Во время же гидрологического события эти процессы ограничены. Осадки могут стечь с растений и присоединиться к выпавшим непосредственно на землю. Здесь вода может скапливаться и в зависимости от типа почвы, рельефа, предшествующего увлажнения и других свойств водораздела, часть воды может просочиться в землю. Эта вода временно задерживается в верхних, частично насыщенных, слоях почвы. Отсюда она может снова подняться на поверхность вследствие капиллярного эффекта, двигаться горизонтально под поверхностью или просочиться в водоносный слой под речным бассейном. Подземные воды передвигаются медленно и, в конечном счете, попадают в речное русло. Вода, которая не испарилась и не просочилась под землю, двигается к каналам по поверхности. Таким образом, результирующий поток в каналах обычно рассматривается как сток с водораздела [5—11].

Однако для ряда задач, как и в нашем случае, нет необходимости моделировать все перечисленные выше факторы. В общем случае для моделирования воды в потоках или на поверхности модель состоит из уравнений сохранения массы и сохранения энергии (импульса). Обычно в качестве основы принимают уравнения Сен-Венана и различные его приближения и упрощения [8, 12, 13, 14].

Поскольку в нашей задаче целью моделирования поверхностного водостока является предсказание паводков и районов затопления, то информация о глубине и скорости водного потока более важна, нежели сведения об объемах стока. Предусматривается, что расчет будет выполняться с данными, включающими полную территорию водосбора района ожидаемого затопления, без входящих речных потоков. При таком предположении возможен свободный сток на всех границах области, что значительно упрощает формулировку граничных условий.

## 2. Вычислительная схема для поверхностного водостока

### *2.1. Ведущие уравнения*

Выбранная общая форма ведущих уравнений [15] задает законы сохранения массы и импульса.

[1] Работа выполнена при финансовой поддержке ДВО РАН в рамках Программы П2 фундаментальных исследований Президиума РАН, проект 09-I-П2-05.

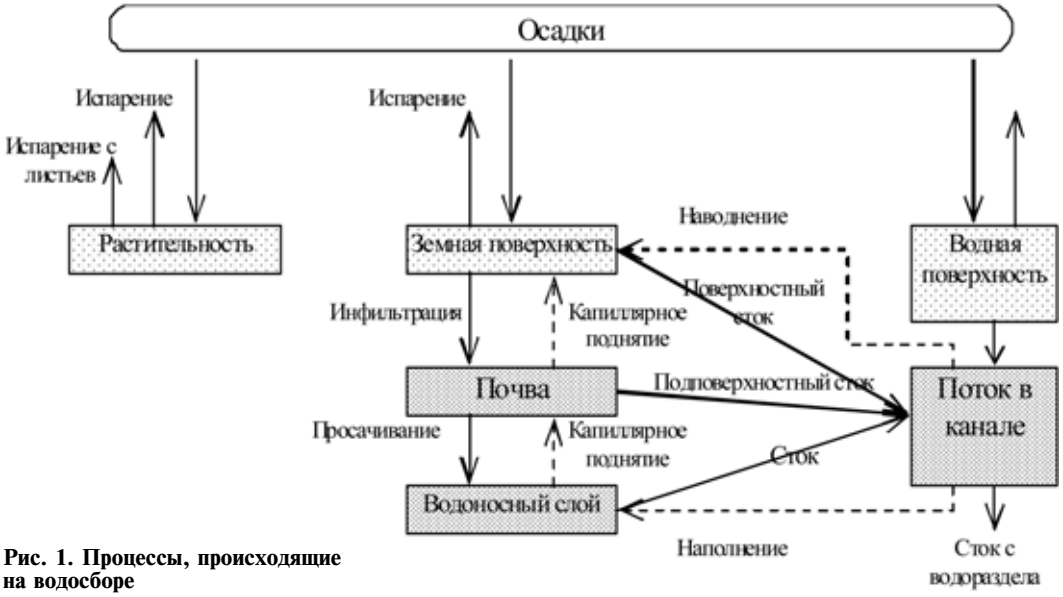



**Рис. 1. Процессы, происходящие на водосборе**

Закон сохранения массы при заданных условиях приводит к уравнению вида

$$\frac{\partial h}{\partial t} = q_{\text{о}} - q_{\text{ф}} - \operatorname{div}(h\bar{V}), \qquad (1)$$

где $h$ — уровень воды; $q_{\text{о}}$ — скорость выпадения осадков; $q_{\text{ф}}$ — скорость фильтрации; $\bar{V}$ — средняя скорость потока по высоте.

Закон сохранения импульса выражается в виде:

$$\left.\begin{aligned} \bar{V}_x &= f\Big(\frac{\partial}{\partial x}(H+h),\ h,\ k_1,\ k_2,\ ...,\ k_n\Big) \\ \bar{V}_y &= f\Big(\frac{\partial}{\partial y}(H+h),\ h,\ k_1,\ k_2,\ ...,\ k_n\Big) \end{aligned}\right\}, \qquad (1)$$

где $H$ — высота рельефа; $k_1$, $k_2$, ..., $k_n$ — система коэффициентов, определяющих вид функции $f(...)$.

Подобное представление позволяет использовать различные формы закона сохранения импульса, не изменяя структуры закона сохранения массы. При такой постановке задачи основной решаемый вопрос — дискретизация по времени. Для получения значений скорости могут быть использованы как физические, так и эмпирические законы или, например, таблицы скоростей. Поэтому в дальнейшем сначала рассматривается дискретизация ведущих уравнений по времени, а затем один из вариантов дискретизации по пространству.

### *2.2. Разбиение моделируемой области*

Для разбиения моделируемой области используется представление земной поверхности в виде квадратных ячеек постоянного размера с известной высотой над уровнем моря. Площадь, покрываемая одной квадратной ячейкой, будет в рассматриваемом случае порядка нескольких метров, что является очень точным представлением, так как неизвестные аналогичные модели обычно оперируют квадратными километрами. На площади ячейки проводится осреднение всех гидрометеорологических параметров и их сохранение в центре ячейки.

Поскольку в данной работе мы рассматриваем только поверхностный водосток, то непосредственно для каждой ячейки необходимо сохранять лишь высоту ячейки над уровнем моря и высоту водяного столба. Для описания динамики системы на границе ячеек рассчитывают средние потоки за временной шаг, и система переходит к следующему состоянию путем реализации во времени (перетекания) найденных потоков.

На рис. 2 представлены ячейки вычислительной сетки и величины, имеющие отношение к поверхностному водостоку. Здесь $H_{i,j}$ — высота ячейки $(i, j)$ над уровнем моря; $h_{i,j}$ — глубина поверхностной воды в центре ячейки $(i, j)$; $F_{i,j}$ — поток в точке $(i, j)$. Тогда $F_{i+1/2,j}$ — поток в точке $i + 1/2, j$, т. е. на интерфейсе ячеек $(i, j)$ и $(i + 1, j)$. Для решения задачи необходимо найти $h^{n+k}$ (где $k$ — целое положительное число) по известному $h^n$ для каждой клетки вычислительной сети на каж-

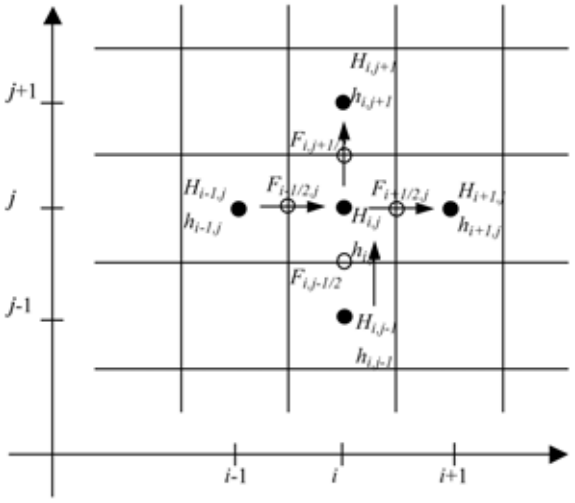



**Рис. 2. Фрагмент вычислительной сетки**

дой итерации. Расположенные в центре ячейки переменные величины, характеризующие состояние среды, будем называть [16] консервативными или сохраняемыми переменными (на рисунке это черные точки), а величины на смежной границе ячеек, характеризующие обмен между соседними ячейками, — потоковыми (пустые точки). Таким образом, $h_{i,j}$ — это сохраняемая переменная, а $F_{i+1/2,j}$ — потоковая. Потоковые переменные вычисляют на каждом временном шаге на основании имеющихся консервативных переменных.

### *2.3. Шаг по времени*

Для дальнейшего рассмотрения и построения вычислительной схемы перепишем уравнение (1) в виде

$$\frac{\partial h}{\partial t} = -\mathrm{div}(h\bar{V}) + K, \qquad (2)$$

где $K = q_0 - q_\Phi$ описывает приток и отток сохраняемой переменной за счет других факторов, не учитываемых уравнением непосредственно.

Учитывая, что на исследуемой сетке местом сохранения консервативной переменной для произвольной ячейки $(i, j)$ является ее центр, получим $\mathrm{div}(h\bar{V}) \approx \frac{\int_{\Gamma(i,j)} \overline{(hV)}_n dS}{V_{(i,j)}}$, где $\Gamma(i, j)$ — контур ячейки; $V_{(i,j)}$ — ее объем.

Выразив интеграл суммой по четырем границам ячейки и перейдя к сеточным обозначениям, получим

$$\mathrm{div}(h\bar{V}) \approx \frac{((hV)_{nx(i+1/2,j)} + (hV)_{nx(i-1/2,j)})\Delta x + \to}{\Delta x \Delta y}$$
$$\to \frac{+ ((hV)_{ny(i,j+1/2)} + (hV)_{ny(i,j-1/2)})\Delta y}{}.$$

Поскольку рассматривается фиксированная квадратная сетка, то обозначив $\Delta x = \Delta y = \Delta l$ и перенаправив скорости в соответствии с осями координат, с учетом введенных выше обозначений получим уравнение (3) в виде

$$\frac{\partial h}{\partial t} = -\Big( \frac{h_{i-1/2,j} V_{i-1/2,j} - h_{i+1/2,j} V_{i+1/2,j} + }{\Delta l} \to$$
$$\to \frac{+ h_{i,j-1/2} V_{i,j-1/2} - h_{i,j+1/2} V_{i,j+1/2}}{} \Big) + K.$$

Представив производную по времени ее конечно-разностным приближением и введя зависимость от шага вычисления, получим

$$\frac{h^{n+1}_{i,j} - h^n_{i,j}}{\Delta t} = -\frac{1}{\Delta l}(h^n_{i-1/2,j} V^n_{i-1/2,j} -$$
$$- h^n_{i+1/2,j} V^n_{i+1/2,j} + h^n_{i,j-1/2} V^n_{i,j-1/2} -$$
$$- h^n_{i,j+1/2} V^n_{i,j-1/2}) + K^n,$$

где верхний индекс обозначает номер шага по времени.

Обозначим

$$F^n_{i,j} = h^n_{i,j} V^n_{i,j} \qquad (3)$$

и выразим значение консервативной переменной $h$ на временном шаге $n + 1$:

$$h^{n+1}_{i,j} = h^n_{i,j} - \frac{\Delta t}{\Delta l}(F^n_{i-1/2,j} - F^n_{i+1/2,j} +$$
$$+ F^n_{i,j-1/2} - F^n_{i,j-1/2}) + \Delta t K^n.$$

Опыт показывает [8, 15], что вычисления по такой временной схеме являются неустойчивыми, поэтому для шага по времени используется TVD (Total Variation Diminishing) метод Рунге—Кутты [13, 17] второго порядка, учитывающий транзитивные и обратные потоки. Неустойчивость возникает [8], когда скорость распространения превышает линейные размеры ячейки, отсюда вытекает, что метод должен быть устойчивым, при следующем ограничении на временной шаг:

$$\Delta t \leqslant \min_{\text{по всем ячейкам } i,j} \times$$
$$\times \Big( \frac{\Delta l}{4\max(|V_{i-1/2,j}|; |V_{i+1/2,j}|; |V_{i,j-1/2}|; |V_{i,j-1/2}|)} \Big).$$

Следует заметить, что с таким шагом по времени решение одномерной задачи является устойчивым [18] и точным [16]. Для плоской задачи принимается ограничение [18], что $C_x + C_y <= 1$, из которого и вычисляется шаг по времени.

### *2.4. Пространственное вычисление потоковых переменных*

На каждом временном шаге по значениям в центре ячейки для каждой клетки вычисляется значение потоковых переменных. При этом не имеет смысла использовать методы более точные, чем вычисления на временном шаге. В данной работе выбран метод вычисления, основанный на формуле для ламинарного потока.

В качестве основы моделей склонового стока обычно используют уравнения со следующими предположениями [8, 14]:

- уклон подстилающей поверхности $i$ достаточно мал, и можно считать, что $\sin i = 1$, $\cos i = 1$;
- скорость течения не изменяется по глубине.

На их основе можно использовать эмпирическую формулу Шези—Маннинга для турбулентного потока [8, 14] $v = \frac{i^{1/2} h^{2/3}}{n}$, где $v$ — скорость водяного потока; $i$ — уклон поверхности; $n$ — коэффициент шероховатости. Тогда $F$ из выражения (3) может быть вычислено как

$$F^n_{i+1/2,j} = \frac{(i^n_{i+1/2,j})^{1/2}(h^n_{i+1/2,j})^{5/3}}{n}. \qquad (4)$$

Значение $F_{i+1/2,j}$ будем вычислять по формуле

$$F_{i+1/2,j} = F^*((F_L)_{i+1/2,j}, (F_R)_{i+1/2,j}),$$

где $(F_L)_{i+1/2,j}$, $(F_R)_{i+1/2,j}$ — значения справа и слева от точки $i+1/2, j$; $F^*$ — решатель Реймана, в качестве которого будем использовать

$$F^*((F_L)_{i+1/2,j}, (F_R)_{i+1/2,j}) = \begin{cases} (F_R)_{i+1/2,j}, \\ \text{если } (F_L)_{i+1/2,j} < 0 \\ (F_L)_{i+1/2,j}, \\ \text{если } (F_L)_{i+1/2,j} > 0. \end{cases}$$

$F_R$ и $F_L$ найдем по формулам

$$(F_R)^n_{i+1/2,j} = \frac{((i_R)^n_{i+1/2,j})^{1/2}((h_R)^n_{i+1/2,j})^{5/3}}{n},$$
$$(F_L)^n_{i+1/2,j} = \frac{((i_L)^n_{i+1/2,j})^{1/2}((h_L)^n_{i+1/2,j})^{5/3}}{n}, \qquad (5)$$

где $n = (n_{i,j} + n_{i+1,j})/2$; $h_R = h_{i+1,j}$, $h_L = h_{i,j}$.

Используя данные работы [8], $i$ можно вычислить как

$$i = i_R = i_L = \frac{(H_{i,j} + h_{i,j}) - (H_{i+1,j} + h_{i+1,j})}{\Delta l},$$

где $H_{i,j}$ высота центра ячейки $(i, j)$ над уровнем моря.

Тогда (5) принимает вид

$$(F_R)^n_{i+1/2,j} = \operatorname{sign}(i)\frac{|i|^{1/2}((h_R)^n_{i+1/2,j})^{5/3}}{n};$$
$$(F_L)^n_{i+1/2,j} = \operatorname{sign}(i)\frac{|i|^{1/2}((h_L)^n_{i+1/2,j})^{5/3}}{n}, \qquad (6)$$

т. е. направление потоковой переменной определяется разницей высот водяного столба. Поскольку в заданную область входит вся площадь водосбора, то на границах предполагается свободный сток воды, т. е. $H = 0$ и $h = 0$ за пределами выделенной площади.

## 3. Алгоритм параллельных вычислений для модели водостока с динамическим балансом нагрузки

### *3.1. Структура параллельных вычислений*

Приведенная выше схема водостока является очень емкой по количеству необходимых вычислений, а с учетом стремления к максимальной детализации рельефа время выполнения расчетов становится слишком большим, и запоздалый прогноз теряет смысл. Если учесть, что при гораздо более простой схеме [8] вычисления занимали несколько дней, то рассчитывать на приемлемое время вычислений на персональном компьютере не приходится. Поэтому для вычислений целесообразно использовать кластер микрокомпьютеров, что является в данном случае наиболее подходящим вариантом.

Для решения задачи используется архитектура *Master—Slave*, когда один компьютер (*Master*) занимается распределением вычислительных задач, вводом и выводом, а остальные компьютеры (*Slave*) занимаются исключительно вычислениями и коммуникациями друг с другом.

### *3.2. Разбиение области вычислений*

Для дальнейшего описания введем обозначения: $P_i$ — обозначение процесса с номером $i$; $N$ — число процессов в системе (от $P_0$ до $P_{N-1}$). Процесс $P_0$ будем считать мастером, а все остальные — рабочими.

Площадь водосбора собирается из квадратов по 1 км$^2$, которые, в свою очередь, разделены на более мелкие (рис. 3), являющиеся ячейками вычислительной сетки.

Разобьем полную площадь водосбора на полоски $L_i$ вдоль оси $Y$ высотой в одну элементарную ячейку. Поскольку время вычислений для каждой полоски различно, то поставим в соответствие каждой полоске ее вес $p_i$ и примем его равным числу клеток в полоске, тогда $\sum_{i=0}^{MY-1} p_i = MT$, где $MT$ — общее число ячеек на площади водосбора; $MY$ — число полос на всем водосборе (рис. 3).

При дальнейшем рассмотрении будем считать, что $MY \gg N$ (необходимо, чтобы было как минимум $MY > 10N$).

Первоначально каждому процессу распределяется по возможности одинаковое число ячеек так, чтобы процесс $P_i$ взаимодействовал только с $P_{i+1}$ и $P_{i-1}$ и мастером. Тогда процесс $P_1$ получит полоски от $L_0$ до $L_{Q_1}$ (далее номера первой и последней полосок, отведенных процессу, будем называть верхней и нижней границей разбиения и обозначать, соответственно, $Q_{i-1}+1$ — верхняя, а $Q_i$ — нижняя граница $P_i$). Процессу $P_i$ изначально передаются полоски от $L_{Q_{i-1}+1}$ до $L_Q$, где $Q_i$ выбирается итерационно ($Q_{i-1}$ было выбрано на предыдущем шаге) так, чтобы $\left(\sum_{i=Q_{i-1}+1}^{Q_1} p_i - \left[\frac{MT}{(N-1)}\right]\right) \to$

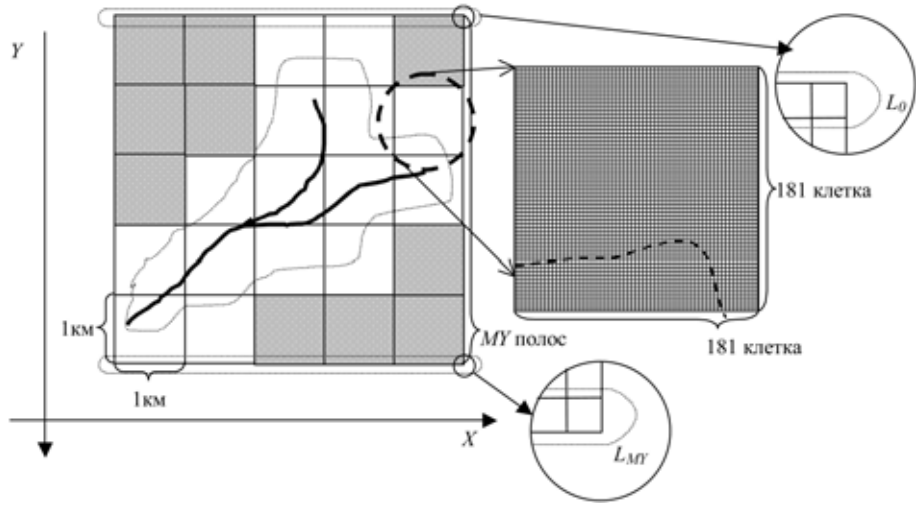



**Рис. 3. Структура представления речного водосбора.**
На рисунке серым цветом отображены клетки, которые не входят в территорию водосбора и при вычислениях не учитываются. Пунктирными линиями ограничена территория водосбора и обозначены первая и последняя полоски вычислительных ячеек

$\to \min$. И наконец, $P_{N-1}$ получает полоски от $L_{Q_{N-2}+1}$ до $L_{MY}$.

Информация о разделении водостока между процессами дублируется в каждом процессе и используется для перераспределения работы.

### *3.3. Нормальный ход вычислений*

При нормальном ходе вычислений (без балансировки) каждый подчиненный процесс проводит вычисления на выделенном ему участке, собирая и передавая всю статистическую информацию мастеру.

Задачами процесса-мастера являются обработка статистики, прием и сохранение результатов. Задачами процесса-рабочего — производство вычислений в соответствии с моделью, выделение из получаемого общего объема тех данных, которые необходимы для вывода, и отправка их мастеру, обмен промежуточной информацией с другими рабочими.

Для расчета потоковых переменных на границе каждому процессу необходима информация не только со своего участка, но и с участка соседа. В приведенной выше схеме это данные на две полосы шире, чем выделенный процессу участок. Поэтому реальных данных, переданных каждому процессу, больше, чем было описано выше. В таком разбиении две нижние и две верхние полоски не просчитываются самим вычислительным процессом, а используются для вычисления потоковых переменных. Данные же о значении консервативных переменных на них вычисляют соседние процессы. Таким образом, если занумеровать локально полоски в процессе $P_i$ сверху вниз от единицы до $n$, то процессу $P_{i-1}$ нужно отослать полоски 3 и 4, а процессу $P_{i+1}$ — полоски $n-2$ и $n-3$. От соседей принимаются, соответственно полоски 0 и 1 от верхнего соседа $P_{i-1}$ и полоски $n-1$ и $n$ от нижнего соседа $P_{i+1}$.

После проведения шага вычислений подчиненный процесс отправляет мастеру полное время вычислительного шага с учетом и без учета времени ожидания ответов от соседей и проверяет очередь входящих сообщений. При получении сигнала от мастера о перераспределении работы на одном из следующих шагов устанавливается соответствующий флаг. При достижении требуемого шага все вычислительные процессы синхронизируются и переходят к шагу балансировки.

### *3.4. Балансировка загрузки системы*

Балансировка нагрузки между процессами выполняется на основе данных о времени, затраченном каждым процессом на проведение вычислений на своем участке. Это время $T_{c_i}$, $i = 1...N-1$, получается мастером после каждого шага от каждого подчиненного процесса.

Новые границы области для процессов назначаются итерационно, начиная с $P_1$, при этом для процесса $P_i$ минимизируется разность $\left( \left( \sum_{i=Q'_{i-1}}^{Q'_1} p_j \right) - \frac{W_i MT}{\sum_{j=1}^{n-1} W_j} \right)$, где $W_i = \frac{\sum_{j=Q_{i-1}}^{Q_i} p_j}{\sum_{k=0}^{\text{размер истории}-1} T_{c_i}^k}$ характеризует вычислительные способности процессора (фактически это число ячеек, обрабатываемых за единицу времени). Верхний индекс означает, сколько шагов назад получено это значение; $Q'_{i-1}$ и $Q'_i$ — границы разбиения на следующем шаге, т. е. процессу выделяется область, соответствующая его вычислительным способностям.

После вычисления мастером новых значений границ они направляются вычислительным процессам, которые самостоятельно обмениваются данными. Обмен между рабочими участками водосбора осуществляется в два этапа: отсылка и прием. Из соображений эффективности они вы-

Таблица 1

**Значения критерия устойчивости на срезах расчетного водосбора**

| Номер среза | Значение критерия через | | | | |
|---|---|---|---|---|---|
| | 10 мин | 30 мин | 1 ч | 2 ч | 3 ч |
| 1 | 0,001558 | 0,002091 | 0,002753 | 0,005279 | 0,005443 |
| 2 | 0,000258 | 0,000396 | 0,000754 | 0,000456 | 0,000195 |
| 3 | 0,004918 | 0,012188 | 0,012783 | 0,005661 | 0,002534 |
| 4 | 0,009150 | 0,023407 | 0,024863 | 0,012549 | 0,007450 |
| 5 | 0,001983 | 0,005774 | 0,009271 | 0,004410 | 0,001932 |
| 6 | 0,000585 | 0,000367 | 0,000206 | 0,000107 | 0,000059 |

полняются в разном порядке на процессах с четными и нечетными номерами. На этапе отсылки процесс просматривает имеющиеся данные и, если находит данные, принадлежащие на следующем шаге другому процессу, то отправляет их непосредственно ему. На этапе приема сначала вычисляется, какие и сколько данных должны быть получены, а затем принимаются данные от любого процесса, пока все, что нужно принять, не будет принято.

### 4. Пример реализации результатов работы

Предложенная выше вычислительная схема поверхностного водостока, а также алгоритм ее распараллеливания на кластере микрокомпьютеров были проверены на небольшом водосборе на территории полуострова Муравьева-Амурского. Вычисления проводились на кластере микрокомпьютеров ИАПУ ДВО РАН.

При моделировании принято упомянутое выше представление о формировании речных потоков (рис. 4, см. четвертую сторону обложки) на подробном рельефе местности (около 5 м$^2$ на ячейку). Для точек, выделенных кружками на рис. 4, при стоке быстро выпавших осадков в объеме 100 мм приводятся расчетные гидрографы.

Результаты исследования устойчивости вычислительной схемы по предложенному выше критерию приведены в табл. 1 для срезов водосбора, изображенных на рис. 5 (см. четвертую сторону обложки). Срезы *1*, *3*, *4*, *5* определены вдоль потоков, срез *2* — вдоль склона, а срез *6* — поперек склона, что охватывает наиболее интересные участки водосбора. Эти значения значительно меньше среднего значения при неустойчивом решении, минимум которого был в районе 0,08, что говорит о том, что решение, полученное по предлагаемой расчетной схеме, устойчиво.

На рис. 6 (см. четвертую сторону обложки) изображены направления векторов скорости, рассчитываемых моделью. Они определяют пре-

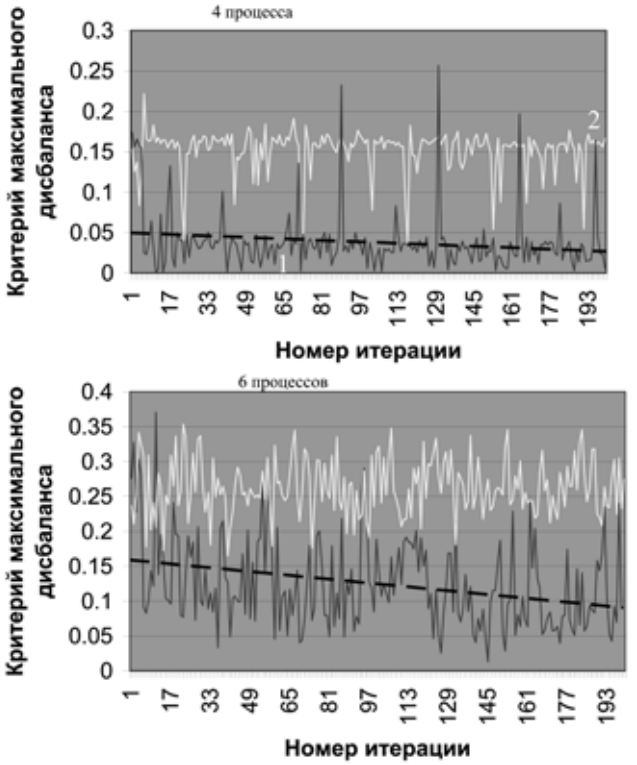



**Рис. 7. Зависимость критерия максимального дисбаланса системы от времени. Пунктиром обозначена линия линейного тренда для вычислений с использованием алгоритма динамической балансировки**

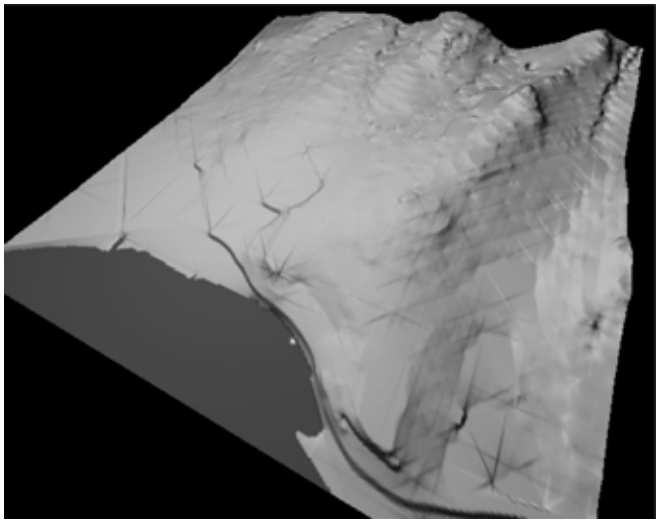


**Рис. 8. Общий вид территории, для которой проводилось тестовое моделирование**

имущественное направление стока воды. Хорошо заметны основные направления водных потоков, формируемые последовательностями направленных друг за другом стрелок. Оценку эффективности распараллеливания расчетного алгоритма проводили с использованием критерия MAX_IMB.

Таблица 2

**Сравнение эффективности вычислений с проведением динамического баланса и без него**

| Число процессов | Использование динамического баланса | Максимальный дисбаланс по общим результатам вычислений | Число вызовов алгоритма балансировки | Время расчета модели, мин | Время синхронизации, мин | Общее время работы, мин |
|---|---|---|---|---|---|---|
| 2 | – | – | – | 736,935 | – | 737,6102 |
| 3 | +<br>– | 0,026178<br>0,035503 | 0<br>– | 362,168<br>343,86 | 0,017956<br>– | 373,6464<br>359,4595 |
| 4 | +<br>– | 0,01737<br>0,17643 | 1<br>– | 263,53<br>252,14 | 0,094887<br>– | 346,5441<br>356,4022 |
| 5 | +<br>– | 0,014434<br>0,284335 | 25<br>– | 215,321<br>229,919 | 1,12396<br>– | 314,5642<br>353,7255 |
| 6 | +<br>– | 0,124836<br>0,333708 | 45<br>– | 165,489<br>167,416 | 2,23<br>– | 263,4115<br>304,7808 |
| 7 | +<br>– | 0,115476<br>0,258385 | 52<br>– | 123,386<br>135,182 | 2,6438<br>– | 228,547<br>259,925 |

Динамика значений критерия (рис. 7) вычислялась после каждой итерации для различного числа процессов.

Для сравнения результаты работы алгоритма с динамическим балансом и без него для различного числа процессов приведены в табл. 2.

Из табл. 2 видно, что использование предложенного алгоритма динамического перераспределения нагрузки приводит к сокращению времени вычислений по сравнению со статичным распределением и к более эффективному разделению общей работы, что позволяет сократить время простоя процессоров в ожидании окончания вычислений соседями.

Все приведенные расчеты выполнялись для территории, приведенной на рис. 8, вид которой сгенерирован на основании данных о возвышенности земной поверхности над уровнем моря.

## Заключение

Разработана расчетная схема модели поверхностного водостока и алгоритм ее параллельного вычисления с динамическим балансом нагрузки.

Реализовано и опробовано на модельном водосборе программное средство.

### Список литературы


1. **Кучмент Л. С., Демидов В. Н., Мотовилов Ю. Г., Смахтин В. Ю.** Система физико-математических моделей гидрологических процессов и опыт ее применения к задачам формирования речного стока // Водные ресурсы. 1986. № 5. С. 24—36.

2. **Кучмент Л. С., Гельфанд А. Н.** Динамико-стохастические модели формирования речного стока. М.: Наука, 1993. 104 с.

3. **Mike SHE WM** — User Manual. Overland Flow Module. DHI Water & Environment 2000. http://s1004.okstate.edu/S1004/Regional-Bulletins/Modeling-Bulletin/MIKESHEfinal.html

4. **Mike SHE WM** — User Manual Overview. DHI Water & Environment 2000. http://iahs.info/redbooks/a231/iahs_231_0427.pdf

5. **Hydrologic** modeling system HEC-HMS. Technical Reference Manual. US Army Corps of Engineers. 2000. http://www.waterengr.com/HECHMS.html

6. **HEC-1** Flood Hydrograph Package Users Manual. 1998. http://www.hec.usace.army.mil/software/legacysoftware/hec1/documentation/hec1user.pdf

7. **Hydrocomp** Forecast and Analysis Modeling (HFAM) // http://www.hydrocomp.com/HFAMinfo.htm

8. **Voinov A., Fitz C., Costanza R.** Surface water flow in landscape models: Everglades case study // Ecological Modelling. 1998. V. 108. N 1—3. P. 131—144.

9. **Summary of DR3M.** URL: http://water.usgs.gov/software/dr3m.html

10. **Model** abstract for Topmodel. URL: http://smig.usgs.gov/SMIC/topmodel.html

11. **Hydrologic** Simulation Models: An Overview. URL: http://www.hydrocomp.com\simoverview.html

12. **Heniche M., Secretan Y., Boudreau P., Leclerc M.** A two-dimensional finite element drying-wetting shallow water model for rivers and estuaries // Advances in water resources. 2000. V. 23. Is. 4. P. 359—372.

13. **Ji-Wen Wang, Ru-Xun Liu.** A comarative study of finite volume methods on unstructured meshes for simulations of 2D shallow water wave problems // Mathematics and Computers in Simulation. 2000. V. 53. P. 171—184.

14. **Румянцев В. А., Кондратьев С. А., Капотова Н. И., Ливанова Н. А.** Опыт разработки и применения математических моделей речных бассейнов. Л.: Гидрометеоиздат, 1985. 93 с.

15. **Бобков В. А., Борисов Ю. С., Май В. П.** Моделирование динамики водостока на рельефе местности // Информационные технологии и вычислительные системы. 2005. № 2. С. 43—50.

16. **Головизин В. М., Карабасов С. А., Кабринский И. М., Суходулов В. А.** Усовершенствованный алгоритм прыжкового переноса для одномерного линейного гиперболического уравнения первого порядка с переменными коэффициентами // Труды Всероссийской конференции "Математическое моделирование и проблемы экологической безопасности". Ростов-на-Дону: ЮГИНФО, 2000. 259 с.

17. **Gottlieb S., Shu C. W.** Total variation diminishing Runge—Kutta schemes // Mathematics of Computation. 2000. V. 67. N 221. P. 73—85.

18. **Роуч П.** Вычислительная гидродинамика. М.: Мир, 1980. 616 с.

УДК 004.38

**Н. В. Белянина,** канд. техн. наук, зав. каф.,
**М. Н. Прокопенко,** канд. техн. наук, инженер,
e-mail: tk-apit@yandex.ru,
**С. А. Серовиков,** аспирант,
Современная гуманитарная академия

# Выбор программно-аппаратной концепции организации распределенной системы экологического мониторинга

*Проведен анализ существующих аппаратно-программных решений для организации систем распределенных вычислений. Дано общее описание задачи экологического мониторинга. Обоснован выбор концепции реализации системы экологического мониторинга на основе распределенных вычислений.*

***Ключевые слова:*** *программно-аппаратные средства, распределенные вычислительные системы, экологический мониторинг*

## Введение

Персональные компьютеры в своем развитии достигли высокого уровня совершенства. Они компактны, имеют высокую производительность и достаточно просты в обращении. Все эти качества привели к их массовому использованию. Но ПК остается удобным инструментом до тех пор, пока не приходится решать задачи, требующие значительно бóльших вычислительных мощностей, например задачи экологического мониторинга загрязнения воздушного бассейна ограниченной территории выбросами промышленных предприятий и автотранспорта в режиме реального времени.

Сущность задачи заключается в расчете значений концентрации загрязняющих веществ в каждой точке ограниченного заданного пространства с учетом мощности выбросов, метеорологических параметров среды, направления и скорости ветра, застройки территории. Сложность заключается в том, что методика расчета предполагает решение лишь для ограниченных линейных или точечных источников загрязнения, а рассматриваемая территория, например город, состоит из множества таких источников, оказывающих не только аддитивное, но и мультипликативное влияние друг на друга.

Для решения подобных задач необходимо применение суперкомпьютеров. Переходя от ПК к суперкомпьютеру, разработчику (пользователю) приходится сталкиваться с большими трудностями в их освоении и дальнейшем использовании. Одной из основных трудностей является отсутствие характерной для ПК пользовательской среды. Кроме того, существует множество разных типов суперкомпьютеров, которые не совместимы друг с другом.

Разрабатывая программу, предназначенную для параллельных вычислений, важно выбрать наиболее подходящий суперкомпьютер, поскольку параллельные компьютеры не могут работать с одинаковой производительностью на любых программах. Если структура программы не соответствует особенностям их архитектуры, то производительность неизбежно падает. Например, ориентация на векторно-конвейерный компьютер или вычислительный кластер с распределенной памятью во многом определит метод решения задачи. В одном случае в программе необходимо векторизовать внутренние циклы, а в другом надо думать о распараллеливании значительных фрагментов кода. Чем больше структура программы соответствует особенностям архитектуры компьютера, тем выше эффективность вычислений, а производительность ближе к пиковым показателям.

## Аппаратные решения в классе распределенных систем

Рассмотрим концепцию аппаратных решений, предназначенных для реализации распределенных систем (РС). Исследователи выделяют множество различных схем классификаций РС: схемы М Флинна, Р. Хокни, Т. Фенга, В. Хендлера, Л. Шнайдера и др. Но ни одна из них не стала действительно популярной и широко распространенной [1]. Наиболее простыми являются системы, построенные из набора независимых компьютеров. На рис. 1 представлены различные базовые архитектуры взаимодействия процессоров и памяти РС: *мультипроцесссорная* (МП) и *мультикомпьютерная* (МК) [2]. Принципиальное различие заключается в том, что МП-системы имеют единое адресное пространство, совместно используемое всеми процессорами, а в МК-систе-

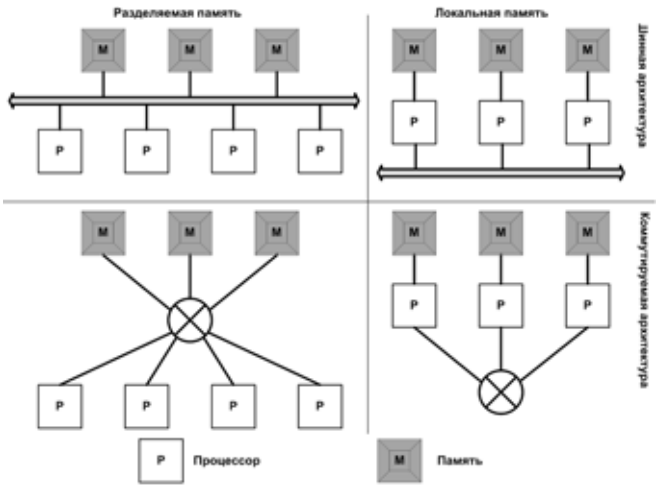



**Рис. 1. Различные базовые архитектуры взаимодействия процессоров и памяти распределенных компьютерных систем**

мах каждый компьютер использует свою собственную память.

Каждая из этих категорий может быть подразделена на дополнительные категории на основе архитектуры соединяющей их сети. На рис. 1 показаны *шинная* и *коммутируемая* архитектуры. Под шиной понимается одиночная сеть, плата, шина, кабель или другая среда, соединяющая все компьютеры между собой.

Коммутируемые системы в отличие от шинных, не имеют единой магистрали. Между компьютерами проведены отдельные каналы, выполненные с применением различных технологий связи.

Существует также разделение РС на *гомогенные* и *гетерогенные*. Это разделение применяется исключительно к системам МК. Для гомогенных систем МК характерна одна соединяющая компьютеры сеть, использующая единую технологию. Одинаковы все процессоры, которые в основном имеют доступ к одинаковым объемам собственной памяти. Гомогенные системы МК нередко используют в качестве параллельных систем (работающих с одной задачей), в точности как МП. Гетерогенные системы МК содержат различные компьютеры, соединенные разнообразными сетями.

Для осуществления выбора аппаратной концепции реализации РС рассмотрим системы МП и МК более подробно. Системы МК имеют одну характерную особенность: все процессоры имеют прямой доступ к общей памяти. Системы МП шинной архитектуры состоят из некоторого числа процессоров, подсоединенных к общей шине, а через нее — к модулям памяти.

Проблема шиной архитектуры состоит в том, что в случае уже четырех или пяти процессоров шина оказывается стабильно перегруженной и производительность резко падает [2]. Решение состоит в размещении между процессором и шиной высокоскоростной кэш-памяти. Однако введение кэша создает проблемы в согласованности памяти, что усложняет программирование системы. При этом остается проблема масштабируемости системы.

Один из альтернативных вариантов — разделить общую память на модули и связать их с процессорами через *коммутирующую решетку*, как показано на рис. 2, *а*. Каждое пересечение представляет собой маленький электронный *узловой коммутатор*, который может открываться и закрываться аппаратно. При этом к памяти могут одновременно обращаться несколько процессоров, кроме случаев, когда два процессора одновременно хотят получить доступ к одному и тому же участку памяти.

Недостатком коммутирующей решетки является то, что при наличии $n$ процессоров и $n$ модулей памяти требуется $n^2$ узловых коммутаторов. Для

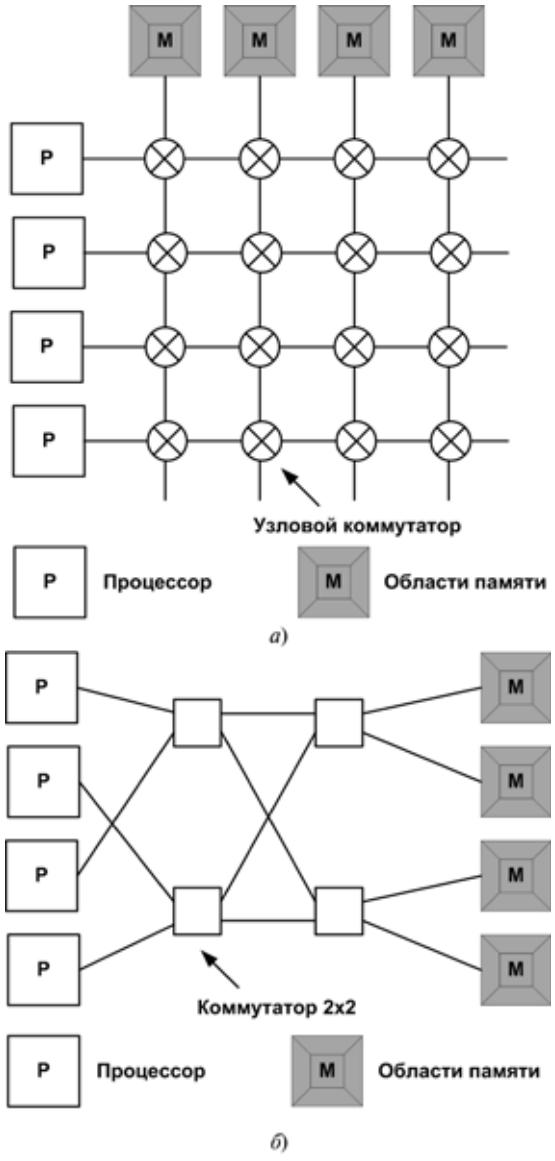



**Рис. 2. Варианты разделения общей памяти на модули:**
*а* — коммутирующая решетка; *б* — коммутирующая омега-сеть

больших значений $n$ это число может превысить технологические возможности.

Альтернативная коммутирующая сеть, требующая меньшего числа коммутаторов — *омега-сеть*, представлена на рис. 2, *б*. Эта сеть содержит четыре коммутатора 2×2, т. е. каждый из них имеет по два входа и два выхода. Каждый коммутатор может соединять любой вход с любым выходом. Недостаток коммутирующих сетей состоит в том, что сигнал, идущий от процессора к памяти или обратно, вынужден проходить через несколько коммутаторов. Поэтому, чтобы снизить задержки между процессором и памятью, коммутаторы должны иметь очень высокое быстродействие.

Существуют также системы, в которых с каждым процессором ассоциируется некоторая область памяти, к которой он может быстро получить доступ. Доступ к другой области памяти происходит значительно медленнее. Эта идея реализована в системах с *неунифицированным доступом к памяти* (Non Uniform Memory Access, NUMA) [3]. Хотя системы NUMA имеют лучшее среднее время доступа к памяти, чем системы на базе омега-сетей, у них есть свои проблемы, связанные с тем, что размещение программ и данных необходимо выполнить так, чтобы большая часть обращений шла к локальной памяти.

В отличие от МП построить МК-систему относительно несложно. Каждый процессор напрямую связан со своей локальной памятью. Единственная проблема — это общение процессоров между собой.

В гомогенных МК-системах, известных под названием *системных сетей*, узлы монтируют в большой стойке и соединяют единой, обычно высокоскоростной, сетью. Существуют системы на основе шинной архитектуры и на основе коммутации.

В МК-системах с шинной архитектурой процессоры соединяют с помощью разделяемой сети множественного доступа, например Fast Ethernet. Как и в случае МП-систем с шинной архитектурой, МК-системы с шинной архитектурой имеют ограниченную масштабируемость.

В коммутируемых МК-системах существует множество различных топологий [4]. Две популярные топологии — квадратные решетки и гиперкубы — представлены на рис. 3. Решетки просты для понимания и удобны для разработки на их основе печатных плат. Они прекрасно подходят для решения двухмерных задач.

*Гиперкуб* представляет собой куб размерности $n$. Гиперкуб, показанный на рис. 3, *б*, четырехмерен. Каждая вершина — это процессор. Каждое ребро — это связь между двумя процессорами. Соответствующие вершины обоих кубов соединены между собой.

Коммутируемые МК-системы могут быть очень разнообразны. Так, *процессоры с массовым параллелизмом* (massively parallel processors, MPP)

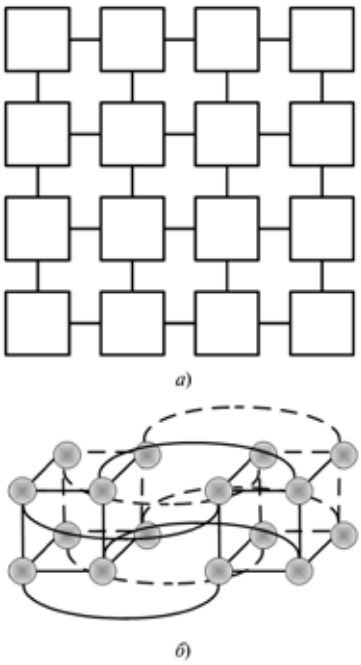



**Рис. 3. Топологии коммутируемых МК систем:**
*а* — решетка; *б* — гиперкуб

представляют собой гигантские суперкомпьютеры высокой стоимости, содержащие тысячи процессоров, объединенных в общую систему патентованными высокоскоростными соединительными сетями. В них предпринимаются специальные меры для защиты системы от сбоев. *Кластеры рабочих станций* (clusters of workstations, COW) в основе своей содержат стандартные персональные компьютеры или рабочие станции, соединенные посредством коммерческих коммуникационных компонентов. В них обычно не предпринимается никаких мер для повышения скорости ввода—вывода или защиты от сбоев в системе. Подобный подход делает COW проще и дешевле.

Гетерогенные МК-системы составляют наибольшее число существующих в настоящее время РС. Компьютеры, являющиеся частями этой системы, могут быть крайне разнообразны, например, роль некоторых из этих компьютеров могут исполнять высокопроизводительные параллельные системы, МП или гомогенные МК. Соединяющая их сеть также может быть сильно неоднородной. Многие крупномасштабные гетерогенные МК-системы нуждаются в глобальном подходе. Это означает, что приложение не может предполагать, что ему постоянно будут доступны определенная производительность или определенные службы.

### Программное обеспечение распределенных вычислений

Ни одна аппаратная платформа не может рассматриваться в отрыве от системного ПО, поэто-

му обратимся к концепции программных решений для распределенных систем.

Операционные системы для распределенных компьютеров можно разделить на две категории — сильно связанные и слабо связанные системы [5]. В сильно связанных системах операционная система в основном старается работать с одним глобальным представлением ресурсов, которыми она управляет. Слабо связанные системы могут представляться набором операционных систем, каждая из которых работает на собственном компьютере. Однако эти операционные системы функционируют совместно, делая собственные службы доступными другим.

Сильно связанные операционные системы обычно называют *распределенными операционными системами* (РОС) и используют для управления МП и гомогенными МК-системами. Основная цель РОС состоит в сокрытии тонкостей управления аппаратным обеспечением, которое одновременно используется множеством процессов.

Слабо связанные *сетевые операционные системы* (СОС) используют для управления гетерогенными МК-системами. Для поддержки прозрачности служб наличия только СОС недостаточно. Необходимо добавить к ним дополнительные компоненты, известные как *системы промежуточного уровня* (СПУ), которые и лежат в основе современных РС.

Функциональность РОС в основном не отличается от функциональности традиционных операционных систем, предназначенных для компьютеров с одним процессором, за исключением того, что она поддерживает функционирование нескольких процессоров. Выделяют два типа РОС:

- *мультипроцессорная операционная система* (МПОС) управляет ресурсами МП. Основная ее задача — обеспечить прозрачность числа процессоров для приложения. Поскольку процессоры используют общую память, необходимо защитить данные от одновременного доступа к ним. Защита осуществляется посредством примитивов синхронизации (два наиболее важных примитива — это семафоры и мониторы);
- *мультикомпьютерная операционная система* (МКОС) предназначена для гомогенных МК. МКОС обладают гораздо более разнообразной структурой и значительно сложнее, чем МПОС. Взаимодействие процессоров в такой системе осуществляется с помощью *передачи сообщений*. МКОС в основном организованы так, как показано на рис. 4.

Каждый узел имеет свое ядро, которое содержит модули для управления локальными ресурсами, и отдельный модуль для межпроцессорного взаимодействия.

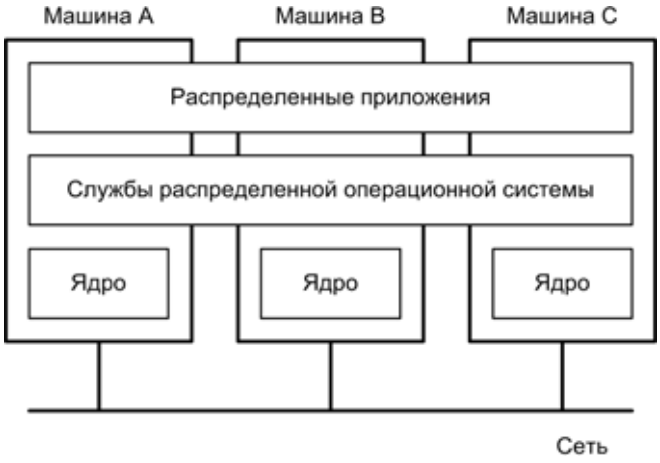



**Рис. 4. Общая структура мультикомпьютерных операционных систем**

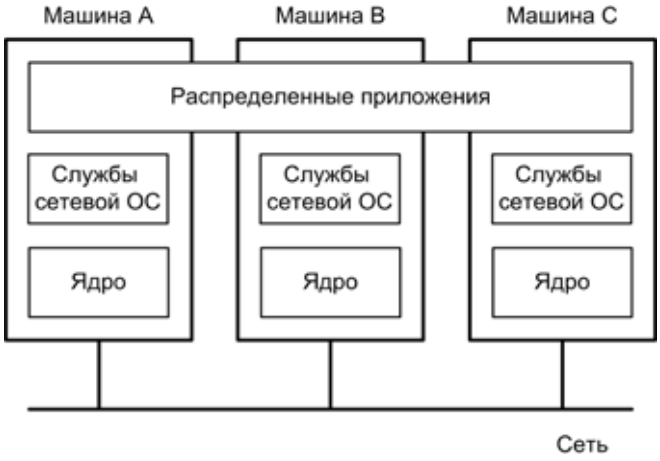



**Рис. 5. Общая структура сетевой операционной системы**

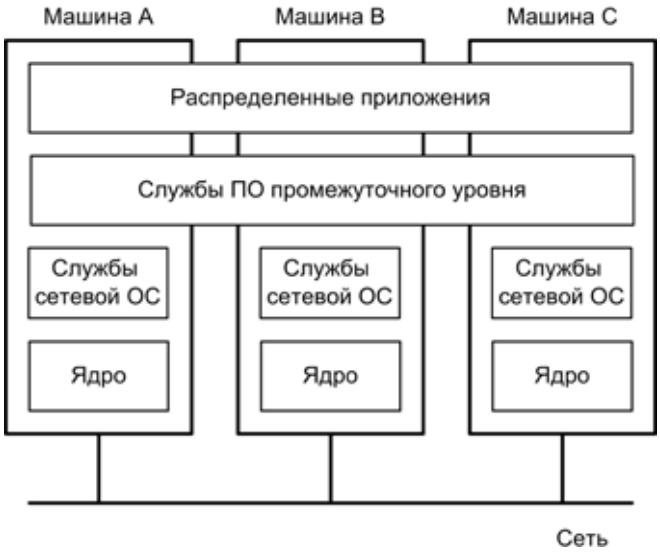



**Рис. 6. Общая структура распределенных систем с промежуточным уровнем**

**Сравнение распределенных операционных систем, сетевых операционных систем и распределенных систем промежуточного уровня**

| Характеристика | Распределенная операционная система | | Сетевая операционная система | Распределенная система промежуточного уровня |
|---|---|---|---|---|
| | Мультипроцессорная | Мультикомпьютерная | | |
| Степень прозрачности | Очень высокая | Высокая | Низкая | Высокая |
| Идентичность операционных систем на всех узлах | Поддерживается | Поддерживается | Не поддерживается | Не поддерживается |
| Число копий ОС | 1 | *N* | *N* | *N* |
| Коммуникации на основе | Совместно используемой памяти | Сообщений | Файлов | В зависимости от модели |
| Управление ресурсами | Глобальное, централизованное | Глобальное, распределенное | Отдельно на узле | Отдельно на узле |
| Масштабируемость | Отсутствует | Умеренная | Да | Различная |
| Открытость | Закрытая | Закрытая | Открытая | Открытая |

Поверх каждого локального ядра лежит уровень программного обеспечения общего назначения, реализующий операционную систему в виде виртуальной машины, поддерживающей параллельную работу над различными задачами.

В противоположность РОС, для СОС не требуется, чтобы аппаратное обеспечение, на котором они функционируют, было гомогенно и управлялось как единая система. Напротив, обычно их строят для набора однопроцессорных систем, каждая из которых имеет собственную операционную систему, как показано на рис. 5.

СОС выглядят значительно примитивнее РОС. Основной минус СОС в недостаточной прозрачности и сложности использования. Преимуществом данных систем является легкое масштабирование.

Поскольку СОС не дают представления одной согласованной системы, дополнительно используется СПУ, что позволяет скрыть от пользователя разнородность набора аппаратных платформ и повысить прозрачность распределения. Она находится посредине между приложением и СОС, как показано на рис. 6.

Существует несколько моделей промежуточного уровня:

- распределенная файловая система (*distributed file system*);
- удаленный вызов процедур (*Remote Procedure Calls*, RPC);
- распределенные объекты (*distributed objects*);
- распределенные документы (*distributed documents*).

Краткое сравнение РОС, СОС и распределенных систем промежуточного уровня приводится в таблице.

## Выводы

исходя из проведенного анализа программно-аппаратных средств организации распределенных вычислений можно сделать вывод: наиболее подходящим средством для решения задач экологического мониторинга является система вычислительных кластеров параллельных вычислений.

Во-первых, кластер может собрать и поддерживать небольшая лаборатория. Во-вторых, стоимость кластерных решений значительно ниже стоимости традиционных суперкомпьютеров. В-третьих, для их построения, как правило, используются массовые процессоры, стандартные сетевые технологии и свободно распространяемое программное обеспечение.

### Список литературы


1. **Воеводин В. В., Воеводин Вл. В.** Параллельные вычисления. — СПб.: БХВ-Петербург, 2004. 608 с.
2. **Таненбаум Э., ванн Стеен. М.** Распределенные системы. Принципы и парадигмы. СПб.: Питер, 2003. 877 с.
3. **Букатов А. А., Дацюк В. Н., Жегуло А. И.** Программирование многопроцессорных вычислительных систем. Ростов-на-Дону: ЦВВР, 2003. 208 с.
4. **Гергель В. П., Стронгин Р. Г.** Основы параллельных вычислений для многопроцессорных вычислительных систем. Нижний Новгород: Изд-во ННГУ им. Н. И. Лобачевского, 2003. 184 с.
5. **Таненбаум Э.** Современные операционные системы. 2-е изд. — СПб.: Питер, 2004. 1040 с.
6. **Макс К. Гофф.** Сетевые распределенные вычисления: достижения и проблемы. М.: КУДИЦ-ОБРАЗ, 2005. 320 с.

УДК 004.7

**Б. Я. Штейнберг,** д-р техн. наук,
ст. науч. сотр., зав. каф.,
Южный федеральный университет,
г. Ростов-на-Дону,
e-mail: bosteinb@mail.ru

# Блочно-аффинные размещения данных в параллельной памяти

*Описаны блочно-аффинные размещения массивов для параллельных компьютеров с распределенной памятью и общей распределенной памятью. Исследованы некоторые свойства таких размещений. Такие размещения массивов могут быть использованы при автоматическом распараллеливании программ.*

***Ключевые слова:*** *параллельные вычисления, размещение данных, распределенная память, общая распределенная память*

## Введение

В данной статье описываются блочно-аффинные размещения многомерных массивов в параллельной памяти. Формальное описание класса размещений массивов может быть удобно для решения следующих проблем:

- автоматического или полуавтоматического размещения данных с оптимизацией пересылок данных при распараллеливании программ для компьютеров с параллельной памятью;
- введения оператора описания размещения массива в параллельной памяти для параллельных языков программирования (концепция MPI, согласно которой в каждом модуле памяти массив находится полностью, не всегда эффективна или удобна);
- определения границ возможностей распараллеливания программ, в частности, для доказательства невозможности параллельного выполнения некоторых алгоритмов без пересылок данных.

Блочно-аффинные размещения являются обобщением часто рассматриваемых при распараллеливании способов размещения матриц (двумерных массивов) по строкам, по столбцам или в скошенном виде [3, 4, 6, 9]. Размещение блочных матриц также может быть описано как блочно-аффинное. Это же формальное описание включает в себя и размещения одномерных массивов [10, стр. 91]. Иногда встречаются размещения данных, которые не относятся к блочно-аффинным, и основанные на них алгоритмы являются асимптотически более эффективными, но такие алгоритмы предполагают сложные коммуникационные системы для пересылок данных [12].

В работе [13] отмечен опережающий рост быстродействия вычислений современных процессоров по отношению к росту быстродействия доступа к данным. Эта тенденция повышает потребность в параллельном доступе к данным, т. е. в использовании параллельной памяти, поскольку общая память способна снабжать данными лишь ограниченное число параллельно выполняемых процессов.

Поиск оптимального с точки зрения пересылок размещения данных в распределенной памяти по наиболее долго вычисляемому участку программы можно выполнять, например, перебором вариантов [4] и с использованием операций типа выравнивания [6], транспонирования или скашивания [3]. Условия бесконфликтного размещения одномерных массивов для компьютеров с архитектурой перестраиваемого конвейера и общей распределенной памятью (*shared distributed memory*) рассматривались в [5]. Там же на основе таких условий предложены алгоритмы размещения данных с минимизацией переразмещений.

Оптимальные по числу межпроцессорных пересылок алгоритмы переразмещения массивов с доказательством неулучшаемости для полнодоступной коммутации и коммуникационной сети гиперкуб рассмотрены в работах [10, 11].

Результаты данной работы ориентированы на многопроцессорные вычислительные системы с распределенной памятью и вычислительные системы с архитектурой перестраиваемого конвейера и общей распределенной памятью (со структурно процедурной организацией вычислений [2]).

На сегодняшний день эффективное автоматическое распараллеливание реализовано только для параллельных компьютеров с общей памятью. Данная работа выполнена в рамках проекта [8] и может рассматриваться как шаг в направлении решения проблемы создания эффективного автоматического (полуавтоматического) распараллеливания программ для вычислительных систем с параллельной памятью.

## Два типа параллельной памяти в вычислительных архитектурах

Будем предполагать, что рассматриваемая параллельная вычислительная система содержит $p$ вычислительных устройств и $p$ модулей памяти, между которыми возможен обмен данными через коммуникационное устройство.

Важной характеристикой вычислительной архитектуры является вид используемой памяти.


**Рис. 1. Многопроцессорная система с распределенной памятью**


**Рис. 2. Многопроцессорная система с общей распределенной памятью**

В данной работе будут рассматриваться два типа параллельной памяти: распределенная (рис. 1) и общая распределенная (рис. 2).

Распределенная память (*distributed memory*) очень распространена и используется как в многопроцессорных системах типа SIMD (ILLIAK-IV, ПС-2000), так и в системах типа MIMD (в кластерных вычислительных системах, в мультитранспьютерных).

Реже используется общая распределенная память (*shared distributed memory*) [2, 5, 7]. Подобная память рассматривается и в работе [13].

Будем использовать термин "параллельная память", понимая под этим одно из двух: распределенную память либо общую распределенную. Неформально говоря, параллельная память означает возможность одновременного доступа к нескольким данным. Размещение данных в параллельной памяти — нетривиальная задача, существенно влияющая на быстродействие программы.

Для размещения данных в параллельной памяти существует принципиальное различие между распределенной памятью и общей распределенной (разделяемой) памятью. В случае распределенной памяти аргументы одной бинарной операции должны быть в одном модуле памяти, а в случае общей распределенной памяти аргументы одной операции могут (должны) быть в разных модулях памяти. Но типы размещения массивов одинаковы.

### Блочно-аффинное размещение и его частные случаи

Размещением $m$-мерного массива $X[1..N_1; 1..N_2; ...; 1..N_m]$ будем называть функцию, где каждому набору индексов $I = (I_1, I_2, ..., I_m)$, $(1 \leqslant I_1 \leqslant N_1, ..., 1 \leqslant I_m \leqslant N_m)$ ставится в соответствие номер модуля памяти, в котором должен находиться элемент массива $X(I_1, ..., I_m)$. Размещение данных в отличие от распределения данных не указывает, в какую именно ячейку модуля памяти попадает данный элемент массива. Размещение данных должно быть таким, чтобы, с одной стороны, к данным удобно было обращаться, а с другой стороны — чтобы число пересылок данных было минимально. Последнее требование приводит к зависимости размещения данных от исполняемой программы. Широкое множество способов размещения данных описано в [1], но без связи с исполняемым кодом.

**Блочно-аффинные по модулю *p* размещения данных.** Пусть натуральные числа $p$, $d_1$, $d_2$, ..., $d_m$ и целые константы $s_0$, $s_1$, $s_2$, ..., $s_m$ зависят только от $m$-мерного массива $X$. Блочно-аффинное по модулю $p$ размещение $m$-мерного массива $X$ — это такое размещение, при котором элемент $X(i_1, i_2, ..., i_m)$ находится в модуле памяти с номером

$$u = (]i_1/d_1[*s_1 + ]i_2/d_2[*s_2 + ... + ]i_m/d_m[*s_m + s_0) \bmod p.$$

Здесь и далее $]a[$ — наименьшее целое, которое не меньше вещественного числа $a$, а * — символ операции умножения чисел.

Число $s_0$ будем называть сдвигом массива $X$. Это число показывает номер модуля памяти, в котором размещается нулевой элемент $X(0, 0, ..., 0)$.

При описанном блочно-аффинном способе размещения $m$-мерный массив представляется как массив блоков размерности $d_1 \times d_2 \times ... \times d_m$, который размещается так, что у каждого блока все элементы оказываются в одном модуле памяти.

**Блочное размещение двумерного массива в четырех модулях памяти**

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| $X(1,1)$<br>$X(2,1)$<br>$X(9,1)$<br>$X(10,1)$ | $X(3,1)$<br>$X(4,1)$ | $X(4,1)$<br>$X(6,1)$ | $X(7,1)$<br>$X(8,1)$ |
| $X(1,2)$<br>$X(2,2)$<br>$X(9,2)$<br>$X(10,2)$ | $X(3,2)$<br>$X(4,2)$ | $X(5,2)$<br>$X(6,2)$ | $X(7,2)$<br>$X(8,2)$ |

**Пример 1.** Возьмем двумерный массив $X$ со значениями параметров: $p = 4$, $m = 2$, $N_1 = 10$, $N_2 = 2$ и добавим параметр $d = 2$. Тогда блочное размещение элементов этого массива будет иметь вид, представленный в таблице.

Следующие способы размещения данных являются частными случаями аффинного по модулю $p$ размещения.

**Аффинные по модулю $p$ размещения данных** — это такие размещения, при которых элемент массива $X(i_1, i_2, ..., i_m)$ находится в модуле памяти с номером

$$u = (s_1 * i_1 + s_2 * i_2 + ... + s_m * i_m + s_0) \bmod p,$$

где $s_0, s_1, s_2, ..., s_m$ — константы, зависящие только от массива $X$.

**{−1, 0, +1}-размещения** — это аффинные по модулю $p$ размещения, в которых константы $s_1, s_2, ..., s_m$ принадлежат множеству {−1, 0, +1}.

К таким размещениям, в частности, относятся размещение матрицы по строкам или по столбцам или по диагоналям или скошенная форма хранения матрицы.

**{0, +1}-размещения** — это аффинные по модулю $p$ размещения, в которых константы $s_1, s_2, ..., s_m$ принадлежат множеству {0, +1}.

К такому типу размещений относятся размещения матриц по строкам, по столбцам и в скошенном виде. Размещение по диагоналям к такому типу не относится.

**Покоординатные размещения со сдвигами** — это такие {0, +1}-размещения, в которых лишь одна из констант $s_1, s_2, ..., s_m$ может быть равна 1.

**Покоординатные размещения** — это такие {0, +1}-размещения, в которых лишь одна константа $s_1, s_2, ..., s_m$ может быть равна 1 и сдвиг равен нулю $s_0 = 0$.

Размещения матрицы по строкам или по столбцам являются покоординатными.

Размещение массива $A[1..N_1; 1..N_2; ...; 1..N_m]$ будем называть блочным с блоком $d$, если для любого $I = (I_1, I_2, ..., I_m)$, $(1 \leqslant I_1 \leqslant N_1, ..., 1 \leqslant I_m \leqslant N_m)$ элемент массива $A[I_1, I_2, ..., I_m]$ находится в модуле памяти с номером $]I_1/d[ \bmod p$.

Естественным образом можно определить подтипы блочно-аффинного размещения: блочное по модулю $p$ {−1, 0, +1}-размещение, блочное по модулю $p$ покоординатное размещение и др.

Горизонтальные и вертикальные размещения одномерных массивов [10, стр. 91], как частный случай, также входят в класс блочно-аффинных размещений.

Описания блочно-аффинных размещений данных могут войти в параллельные языки программирования. Библиотека MPI предполагает, что в каждом модуле памяти хранится весь массив полностью, хотя это не всегда удобно. Язык параллельного программирования COLAMO [14] допускает использование двумерных массивов, причем одна из координат — это номер модуля памяти, в котором хранятся соответствующие элементы массива. Пользователю может быть проще иметь привычный многомерный массив с добавлением указания о размещении его элементов. Такое указание должно содержать целые параметры $p, d_1, d_2, ..., d_m, s_0, s_1, s_2, ..., s_m$. Таким образом, разработчик параллельной программы получает возможность использования широкого класса блочно-аффинных размещений. Реализация таких размещений в компиляторах представляется вполне реализуемой.

## Условия размещения массивов в распределенной памяти без пересылок

Пусть имеется гнездо циклов, содержащее два вхождения аффинно размещенных массивов:

$$\begin{gathered} DO\ 1\ I_1 = 0,\ N_1 \\ \ldots\ldots\ldots\ldots\ldots \\ DO\ 99\ I_n = 0,\ N_n \end{gathered} \qquad (1)$$

$$99... = X\Big(a_{10} + \sum_{j=1}^{n} a_{1j} * I_j, ..., a_{m0} + \sum_{j=1}^{n} a_{mj} * I_j) + Y(b_{10} + \sum_{j=1}^{n} b_{1j} * I_j, ..., b_{k0} + \sum_{j=1}^{n} b_{kj} * I_j\Big).$$

Будем полагать, что элемент $m$-мерного массива $X(i_1, i_2, ..., i_m)$ находится в модуле памяти с номером

$$u = (s_1 * i_1 + s_2 * i_2 + ... + s_m * i_m + s_0) \bmod p,$$

где $s_0, s_1, s_2, ..., s_m$ — константы, зависящие только от массива $X$, а элементы $k$-мерного массива $Y$ размещены аналогично, но с параметрами $t_1, t_2, ..., t_k, t_0$.

Для всех точек пространства итераций (1) сложение двух элементов массивов $X$ и $Y$ можно выполнить без пересылок данных тогда и только то-

гда, когда эти элементы будут в одном и том же модуле памяти при всех значениях счетчиков циклов.

**Теорема 1.** Пусть в гнезде циклов (1) есть два вхождения

$$X(a_{11} * i_1 + ... + a_{1n} * i_n + a_{10}, a_{21} * i_1 + ... + a_{2n} * i_n + a_{20}, ..., a_{m1} * i_1 + ... + a_{mn} * i_n + a_{m0});$$

$$Y(b_{11} * i_1 + ... + b_{1n} * i_n + b_{10}, b_{21} * i_1 + ... + b_{2n} * i_n + b_{20}, ..., b_{kl} * i_1 + ... + b_{kn} * i_n + b_{k0})$$

$m$-мерного массива $X$ и $k$-мерного массива $Y$ соответственно.

Для того чтобы эти два вхождения были в одном и том же модуле памяти для всех значений счетчиков циклов $i_1, i_2, ..., i_n$, необходимо и достаточно, чтобы выполнялись следующие условия:

$$s_1 * a_{11} + s_2 * a_{21} + ... + s_m * a_{m1} = t_1 * b_{11} + t_2 * b_{21} + ... + t_k * b_{k1} \bmod p;$$

.....

$$s_1 * a_{1n} + s_2 * a_{2n} + ... + s_m * a_{mn} = t_1 * b_{1n} + t_2 * b_{2n} + ... + t_k * b_{kn} \bmod p;$$

$$s_1 * a_{10} + s_2 * a_{20} + ... + s_m * a_{m0} + s_0 = t_1 * b_{10} + t_2 * b_{20} + ... + t_k * b_{k0} + t_0 \bmod p.$$

*Доказательство.* Поскольку оба вхождения из условия теоремы находятся в одном и том же модуле памяти, должно выполняться равенство

$$s_1 * (a_{11} * i_1 + ... + a_{1n} * i_n + a_{10}) + s_2 * (a_{21} * i_1 + ... + a_{2n} * i_n + a_{20}) + ... + s_m * (a_{m1} * i_1 + ... + a_{mn} * i_n + a_{m0}) + s_0 = t_1 * (b_{11} * i_1 + ... + b_{1n} * i_n + b_{10}) + t_2 * (b_{21} * i_1 + ... + b_{2n} * i_n + b_{20}) + ... + t_k * (b_{k1} * i_1 + ... + b_{kn} * i_n * i_n + b_{k0}) + t_0 \bmod p.$$

Так как это равенство должно тождественно выполняться для всех значений счетчиков циклов $i_1, i_2, ..., i_n$, коэффициенты при этих счетчиках должны быть равны. Отсюда и вытекает заключение теоремы.

Конец доказательства.

Аналогично можно доказать, что в гнезде циклов (1) два вхождения $m$-мерного массива $X$:

$$X(a_{11} * i_1 + ... + a_{1n} * i_n + a_{10}, a_{21} * i_1 + ... + a_{2n} * i_n + a_{20}, ..., a_{m1} * i_1 + ... + a_{mn} * i_n + a_{m0}) \quad (2)$$

и

$$X(b_{11} * i_1 + ... + b_{1n} * i_n + b_{10}, b_{21} * i_1 + ... + b_{2n} * i_n + b_{20}, ..., b_{m1} * i_1 + ... + b_{mn} * i_n + b_{m0}) \quad (3)$$

будут в одном и том же модуле памяти для всех значений счетчиков циклов $i_1, i_2, ..., i_n$, тогда и только тогда, когда выполняются следующие условия:

$$s_1 * (a_{11} - b_{11}) + s_2 * (a_{21} - b_{21}) + ... + s_m * (a_{m1} - b_{m1}) = 0 \bmod p;$$

.....

$$s_1 * (a_{1n} - b_{1n}) + s_2 * (a_{2n} - b_{2n}) + ... + s_m * (a_{mn} - b_{mn}) = 0 \bmod p; \quad (4)$$

$$s_1 * (a_{10} - b_{10}) + s_2 * (a_{20} - b_{10}) + ... + s_m * (a_{m0} - b_{m0}) = 0 \bmod p.$$

Опираясь на полученные условия, для некоторых алгоритмов можно вычислять размещения данных, при которых параллельные вычисления можно проводить без пересылок данных или доказывать отсутствие таковых в классе блочно аффинных размещений.

**Пример 2.** Рассмотрим фрагмент программы, вычисляющий симметрическую составляющую матрицы:

$$do\ i = 1, n$$
$$do\ j = 1, n$$
$$B(i, j) = [A(i, j) + A(j, i)]/2.$$

Для каждого значения $i$ и каждого значения $j$ элементы $B(i, j)$, $A(i, j)$ и $A(j, i)$ должны оказаться в одном и том же модуле памяти. Это возможно тогда и только тогда, когда существуют такие целые числа $s_1, s_2, t_1, t_2$, что выполняются равенства

$$s_1 * i + s_2 * j = s_1 * j + s_2 * i = t_1 * i + t_2 * j \bmod p.$$

Это возможно лишь при $s_1 = s_2 = t_1 = t_2 \bmod p$. Например, все эти коэффициенты равны 1. Тогда в $k$-м модуле памяти будет находиться $k$-я косая диагональ (перпендикуляр к диагоналям) матрицы $A$ и $k$-я косая диагональ $B$, т. е. множества элементов $A(i, j)$ и $B(i, j)$, для которых $i + j = k (\bmod p)$.

**Пример 3.** Рассмотрим фрагмент программы, вычисляющий произведение двух матриц:

$$do\ i = 1, n$$
$$do\ j = 1, n$$
$$do\ k = 1, n$$
$$X(i, j) = X(i, j) + A(i, k) * B(k, j).$$

Для каждого значения $i$, каждого значения $j$ и каждого значения $k$ элементы $X(i, j)$, $A(i, k)$ и $B(k, j)$ должны оказаться в одном и том же модуле памяти. Это возможно тогда и только тогда, когда существуют такие целые числа $s_1, s_2, t_1, t_2, u_1, u_2$, что выполняются равенства

$$s_1 * i + s_2 * j = t_1 * i + t_2 * k = u_1 * k + u_2 * j \bmod p.$$

Это возможно для всех значений $i, j, k$ лишь при выполнении равенств $s_1 = s_2 = t_1 = t_2 = u_1 =$

$= u_2 = 0 \bmod p$, т. е. когда все данные находятся в одном и том же модуле памяти. Итак, задача вычисления произведения матриц не может быть распараллелена на суперкомпьютере с MIMD-архитектурой без межпроцессорных пересылок данных.

**Пример 4.** Распараллеливание алгоритма Флойда.

Алгоритм Флойда имеет на входе матрицу $A$ весов дуг графа и преобразует ее в матрицу кратчайших расстояний:

$$do\ k = 1,\ n$$
$$do\ i = 1,\ n$$
$$do\ j = 1,\ n$$
$$if\ A(i, j) > A(i, k) + A(k, j)\ \text{then}$$
$$A(i, j) = A(i, k) + A(k, j).$$

Условие выполнения этого гнезда циклов без межпроцессорных пересылок на параллельном суперкомпьютере с распределенной памятью запишется в виде следующих равенств:

$$s_1 * i + s_2 * j = s_1 * i + s_2 * k = s_1 * k + s_2 * j.$$

Эти равенства могут выполняться тождественно для всех $i, j, k$ только при $s_1 = s_2 = 0$, т. е. при последовательном вычислении.

Приведенные выше примеры 2—4 иллюстрируют метод поиска таких блочно-аффинных размещений массивов, при которых параллельное вычисление алгоритма возможно без пересылок данных, либо иллюстрируют доказательства отсутствия таких размещений. Этот метод основан на полученной теореме 1 и условиях (4). Суть метода в том, чтобы в последовательной программе алгоритма рассмотреть все пары вхождений переменных, которые являются аргументами одних и тех же вычислений, составить для них уравнения с неопределенными параметрами размещений и найти решение полученной системы линейных алгебраических уравнений.

## Условия бесконфликтного размещения массивов в общей распределенной памяти

В работе [2] описана вычислительная система с архитектурой перестраиваемого конвейера и общей распределенной памятью. Эта вычислительная система требует в одно элементарное вычислительное устройство (сумматор, умножитель, ...) подавать одновременно два аргумента одной бинарной операции. Для реализации такой возможности оба аргумента одной операции должны быть в разных модулях памяти.

Возьмем гнездо циклов (1) и будем рассматривать вопрос о том, когда оба его вхождения оказываются в разных модулях памяти.

**Теорема 2.** Пусть в гнезде циклов (1) есть два вхождения

$$X(a_{11} * i_1 + ... + a_{1n} * i_n + a_{10},\ a_{21} * i_1 + ... + a_{2n} * i_n + a_{20},\ ...,\ a_{m1} * i_1 + ... + a_{mn} * i_n + a_{m0})$$

и

$$Y(b_{11} * i_1 + ... + b_{1n} * i_n + b_{10},\ b_{21} * i_1 + ... + b_{2n} * i_n + b_{20},\ ...,\ b_{k1} * i_1 + ... + b_{kn} * i_n + b_{k0})$$

$m$-мерного массива $X$ и $k$-мерного массива $Y$ соответственно.

Для того чтобы эти два вхождения были в разных модулях памяти для всех значений счетчиков циклов $i_1, i_2, ..., i_n$, необходимо и достаточно, чтобы число

$$\text{НОД}\{[s_1 * a_{11} + s_2 * a_{21} + ... + s_m * a_{m1}) - (t_1 * b_{11} + t_2 * b_{21} + ... + t_k * b_{k1})],\ ...,\ [(s_1 * a_{1n} + s_2 * a_{2n} + ... + s_m * a_{mn}) - (t_1 * b_{1n} + t_2 * b_{2n} + ... + t_k * b_{kn})],\ p\}$$

не являлось делителем числа

$$(s_1 * a_{10} + s_2 * a_{20} + ... + s_m * a_{m0} + s_0) - (t_1 * b_{10} + t_2 * b_{20} + ... + t_k * b_{k0} + t_0).$$

*Доказательство.* Если для некоторого вектора счетчиков циклов $(i_1, i_2, ..., i_n)$ оба вхождения $X$ и $Y$ находятся в одном и том же модуле памяти, то должно выполняться равенство

$$s_1 * (a_{11} * i_1 + ... + a_{1n} * i_n + a_{10}) + s_2 * (a_{21} * i_1 + ... + a_{2n} * i_n + a_{20}) + ... + s_m * (a_{m1} * i_1 + ... + a_{mn} * i_n + a_{m0}) + s_0 = t_1 * (b_{11} * i_1 + ... + b_{1n} * i_n + b_{10}) + t_2 * (b_{21} * i_1 + ... + b_{2n} * i_n + b_{20}) + ... + t_k * (b_{k1} * i_1 + ... + b_{kn} * i_n + b_{k0}) + t_0 \bmod p.$$

Это равносильно существованию решения диофантова уравнения относительно переменных $(i_0, i_1, i_2, ..., i_n)$:

$$[(s_1 * a_{11} + s_2 * a_{21} + ... + s_m * a_{m1}) - (t_1 * b_{11} + t_2 * b_{21} + ... + t_k * b_{k1})] * i_1 + ... + [(s_1 * a_{1n} + s_2 * a_{2n} + ... + s_m * a_{mn}) - (t_1 * b_{1n} + t_2 * b_{2n} + ... + t_k * b_{kn})] * i_n + (s_1 * a_{10} + s_2 * a_{20} + ... + s_m * a_{m0} + s_0) - (t_1 * b_{10} + t_2 * b_{20} + ... + t_k * b_{k0} + t_0) + p * i_0 = 0.$$

Условие существования такого решения имеет следующий вид: число

$$\text{НОД}\{[s_1 * a_{11} + s_2 * a_{21} + ... + s_m * a_{m1}) - (t_1 * b_{11} + t_2 * b_{21} + ... + t_k * b_{k1})],\ ...,\ [(s_1 * a_{1n} + s_2 * a_{2n} + ... + s_m * a_{mn}) - t_1 * b_{1n} + t_2 * b_{2n} + ... + t_k * b_{kn})],\ p\}$$

является делителем числа

$$(s_1 * a_{10} + s_2 * a_{20} + ... + s_m * a_{m0} + s_0) - (t_1 * b_{10} + t_2 * b_{20} + ... + t_k * b_{k0} + t_0).$$

Взяв отрицание этого условия, получим утверждение теоремы.

Конец доказательства.

Аналогично могут быть описаны условия, при которых два вхождения одного массива всегда оказываются в разных модулях памяти.

В работе [5] описаны алгоритмы бесконфликтного размещения одномерных массивов для выполнения на архитектуре [2]. Результаты данного параграфа позволяют обобщить эти алгоритмы на случай многомерных массивов.

**Список литературы**


1. **Метлицкий Е. А., Каверзнев В. В.** Системы параллельной памяти. Теория, проектирование, применение. Л.: ЛГУ, 1989. 240 с.
2. **Каляев А. В., Левин И. И.** Модульно-наращиваемые многопроцессорные системы со структурно-процедурной организацией вычислений. М.: Янус-К, 2003. 380 с.
3. **Прангишвили И. В., Виленкин С. Я., Медведев И. Л.** Параллельные вычислительные системы с общим управлением. М.: Энергоатомиздат, 1983. 312 с.
4. **Фролов А. В.** Оптимизация размещения массивов в ФОРТРАН-программах на многопроцессорных вычислительных системах // Программирование. 1998. № 3. С. 70—80.
5. **Штейнберг Б. Я.** Бесконфликтные размещения массивов при параллельных вычислениях // Кибернетика и системный анализ. 1999. № 1. С. 166—178.
6. **Kulkurni D., Stumm M.** Loop and Data Transformations: A Tutorial. Technical Report CSRI 337 // Computer Systems Research Institute, University of Toronto. June 1993. 53 p.
7. **Almasi G. S., Gottlib A.** Highly Parallel Computing. The Benjamin/Cummings Publishing Company, inc. 1989.
8. **www.ops.rsu.ru** Открытая распараллеливающая система.
9. **Гергель В. П., Стронгин Р. Г.** Основы параллельных вычислений для многопроцессорных вычислительных систем. 2-е изд. Н.-Новгород: ННГУ. 2003.
10. **Штейнберг Б. Я.** Математические методы распараллеливания рекуррентных программных циклов на суперкомпьютеры с параллельной памятью. Ростов-на-Дону: Изд-во Ростовского ун-та, 2004. 192 с.
11. **Штейнберг Б. Я.** Оптимальные параллельные переразмещения двумерных массивов // Программирование. 1993. № 6. С. 81—88.
12. **Gupta H., Sadayappan P.** Communication Efficient Matrix-Multiplication on Hypercubes // Technical Report. Stanford InforLab. (Publication Note: Sixth Annual ACM Symposium on Parallel Algorithms and Architectures, 1994 (SPAA 1994). Extended version in Parallel Computing, 22 (1996) 75—99.) http://ilpubs.stanford.edu:8090/59/
13. **Корнеев В. В.** Проблемы программирования суперкомпьютеров на базе многоядерных мультитредовых кристаллов. Научный сервис в сети Интернет: масштабируемость, параллельность, эффективность// Тр. Всероссийской суперкомпьютерной конференции (21—26 сентября 2009 г., г. Новороссийск). М.: Изд-во МГУ, 2009. 524 с.
14. www.parallel.ru



УДК 519.6 + 004.3.012



**Д. В. Макошенко,** аспирант,
Южный Федеральный университет,
Intel Corporation, руководитель группы,
e-mail: denis.makoshenko@intel.com


# Назначение переменных на регистры с помощью древовидного параметрического алгоритма раскраски графа


*Рассматривается задача назначения переменных на физические регистры, важная при трансляции программы с языка высокого уровня в машинные коды. Для решения задачи предлагается новый древовидный параметрический алгоритм. Приводятся предложения по эффективной реализации алгоритма, позволяющие использовать новый метод в коммерческих инструментах для оптимизации кода.*



***Ключевые слова:*** *компилятор, оптимизация кода, распределение регистров, раскраска графов, древовидный поиск*


## Введение

В данной работе рассматривается задача назначения переменных на физические регистры, важная при трансляции программы с языка высокого уровня в машинные коды.

Если физических регистров для хранения переменных не хватает, то требуется добавление дорогостоящих инструкций обращения в память. Поэтому актуальна задача оптимального распределения переменных на физические регистры.

Каждой программе может быть поставлен в соответствие некоторый граф, позволяющий моделировать назначение переменных на регистры [1]. Если хроматическое число этого графа не превосходит число доступных в архитектуре физических регистров, то переменные могут быть размещены на физических регистрах без добавления инструкций обращения в память. Хроматическое число графа находится с помощью его раскраски. Задача раскраски произвольного графа является NP-полной [2].

Чейтин [1] предложил, а Бриггс [3] и Вигдал [4] развили несколько эвристических методов распределения переменных на регистры на основе

задачи раскраски графа. Данная статья развивает предложенные методы раскраски и предлагает новый алгоритм. Приводятся предложения по эффективной реализации алгоритма.

### Модель внутреннего представления

Будем рассматривать подход к трансляции программы, при котором трансляция программы выполняется для каждой процедуры отдельно.

Предполагается, что исходная процедура дается во внутреннем представлении, основой которого является граф потока управления [5].

Рассмотрим понятие *виртуального регистра*. Выберем некоторую вычислительную архитектуру и рассмотрим ее регистровые файлы (наборы однотипных регистров). Каждому такому файлу поставим в соответствие бесконечное множество виртуальных регистров. Виртуальный регистр наследует все свойства физических регистров из того файла, которому он соответствует.

Будем называть генератором виртуального регистра $v$ такую инструкцию, которая присваивает значение $v$; использованием виртуального регистра $v$ — такую инструкцию, которая использует значение $v$. Обращением к виртуальному регистру называется либо его генератор, либо его использование.

**Пример 1.** Ниже приведен фрагмент графа потока управления. Инструкция (1) — это генератор $v3$, а инструкция (2) — использование $v3$. Инструкция (3) является как использованием, так и генератором $v4$.

(1) $v3 = v2 + v1$;
(2) $v4 = v3 + 5$;
(3) $v4 = v2 \times v4$.
■

Далее в данной работе будет предполагаться, что проведена замена исходных переменных процедуры на виртуальные регистры. Таким образом, в процедуре больше нет исходных переменных, инструкции оперируют виртуальными регистрами. Из этого следует, что виртуальные регистры и только они будут являться кандидатами на назначение на физические регистры.

Время жизни LifeTime($v$) виртуального регистра $v$ — это множество вершин графа потока управления, таких что в них виртуальный регистр $v$ жив.

**Пример 2.** На рис. 1 приведены исходная процедура и ее граф потока управления. Исходные переменные заменены виртуальными регистрами. Параметры $b$ и $c$ назначены на виртуальные регистры $v0$ и $v1$ соответственно. Чтобы внутреннее представление всегда имело вид, в котором для каждого использования есть свой генератор, для параметров добавлены псевдо-определения в месте входа в процедуру. Для виртуального регистра $v$ время жизни LifeTime($v3$) = {4, 5, 6, 8, 9, 10}.

```
int foo(int b, int *c)
a=b+*c;
if (a>0){
 e=b+a;
 return e;
} else {
  *c=0;
  return b;
}
```

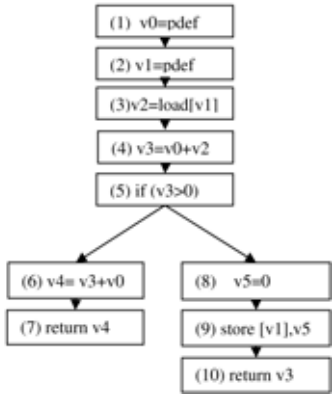



**Рис. 1. Исходная процедура и ее граф потока управления**

### Модель памяти вычислительной системы

Рассмотрим модель памяти вычислительной системы, которая состоит из двух частей: физических регистров и основной памяти. Инструкции, реализующие различные операции, ожидают их аргументы на физических регистрах, результат также помещается в физический регистр. Физические регистры — это "быстрая" память, т. е. время записи и чтения из них находится в пределах такта машинного времени. Недостатком физических регистров является их малочисленность. Например, в архитектуре Intel 64 [6] число регистров общего назначения равно 16. Размер основной памяти может достигать нескольких гигабайтов, в то время как время доступа к ней может достигать десятков и даже сотен тактов машинного времени.

Данная модель отражает несколько упрощенную организацию подсистемы памяти в современных вычислительных архитектурах, например, x86 [6] и Итаниум [7].

### Задача оптимального распределения физических регистров

Для поиска распределения виртуальных регистров на физические введем понятие *графа несовместимости* (ГН). ГН — это неориентированный граф, описывающий зависимости между временами жизни виртуальных регистров. Вершины графа — это виртуальные регистры. Ребро соединяет две вершины, если времена жизни соответствующих вершинам виртуальных регистров пересекаются. Очевидно, что любое назначение виртуальных регистров на физические будет корректным в случае, если никакие две смежные вершины в ГН не были назначены на один физический регистр.

Пусть число регистров, доступных в данной архитектуре, равно $N$. Будем отождествлять каждую краску с одним физическим регистром.

Если полученный ГН удалось раскрасить $N$ красками, то распределение физических регистров завершено. В самом деле, никакие два виртуальных регистра в этом случае не будут назначены на один физический — условие корректности соблюдено.

В противном случае нужно трансформировать процедуру таким образом, чтобы хроматическое число соответствующего ей ГН было меньше, чем хроматическое число для исходного ГН. Для этого необходимо уменьшать времена жизни виртуальных регистров процедуры (данная тема выходит за рамки этой статьи).

На время исполнения процедуры влияют дополнительные операции с основной памятью, которые появляются при уменьшении времен жизни виртуальных регистров. Как было обсуждено в предыдущем параграфе, время операций с основной памятью велико, поэтому очень важно нахождение оптимального распределения виртуальных регистров на физические.

Способность алгоритма находить оптимальное распределение регистров зависит от двух факторов:

- способности алгоритма находить наилучшую раскраску графа для размещения как можно больше виртуальных регистров целиком на физических;
- способности алгоритма находить оптимальную комбинацию виртуальных регистров, время жизни которых будет уменьшено, и они не будут целиком лежать на физических.

Ниже обсуждаются различные методы раскраски ГН.

### Переборный алгоритм раскраски графа несовместимости

Заметим, что если граф $G$ полный, то его единственная возможная раскраска — каждая вершина — получает свой собственный цвет. Если граф $G$ не полный, воспользуемся методом из [8, стр. 47]. Выберем две несмежные вершины $u$ и $v$. Обозначим: $(G; <u - v>)$ — исходный граф, в котором вершины $u$ и $v$ соединены ребром; $(G: <u = v>$ — исходный граф, в котором вершины $u$ и $v$ стянуты в одну.

Рассмотрим множество $M(G)$ всех раскрасок графа несовместимости $G$. Тогда $M(G)$ представимо в виде объединения двух непересекающихся множеств:

$$M(G) = M((G; <u - v>)) \cup M((G: <u = v>)). \quad (1)$$

Пользуясь соотношением (1), построим дерево $C$ перебора всех раскрасок графа несовместимости $G$. Корнем дерева $C$ является вершина, которой соответствует множество $M(G)$. Каждой вершине этого дерева соответствует некоторое подмножество множества $M(G)$. Определим правила для построения сыновей.

Рассмотрим произвольную вершину $n$ дерева $C$ и соответствующее ей подмножество $M(n) \subset M(G)$. Рассмотрим два условия:

- существуют две несмежные вершины $u$ и $v$ графа $G$;
- множество $M(n)$ содержит раскраски, такие что несмежные вершины $u$ и $v$ красятся как одним и тем же, так и разными цветами.

У вершины $n$ существуют два сына тогда и только тогда, когда оба условия выполнены. В противном случае вершина $n$ является листом. Пусть оба условия выполнены. Обозначим $L$ множество всех тех раскрасок, принадлежащих $M(n)$, в которых вершины $u$ и $v$ красятся одним цветом. Обозначим $R$ множество всех тех раскрасок, принадлежащих $M(n)$, где вершины $u$ и $v$ красятся разными цветами. Тогда множество $M(n)$ разбивается на подмножества $L$ и $R$: левому сыну вершины $n$ соответствует множество $L$, правому сыну вершины $n$ — множество $R$.

Для удобства построения дерева $C$ каждой его вершине $n$ можно поставить в соответствие некоторый граф $G(n)$, который строится из графа $G$ следующим образом:

- если для любой раскраски из $M(n)$ вершины $u$ и $v$ покрашены в разные цвета, тогда в графе $G$ эти вершины соединяются ребром;
- если для любой раскраски из $M(n)$ вершины $u$ и $v$ покрашены в один и тот же цвет, тогда в графе $G$ эти вершины стягиваются в одну.

Таким образом, для каждой вершины $n$ между множеством раскрасок $M(n)$ и множеством всех раскрасок графа $G(n)$ существует взаимно однозначное соответствие. Тогда перебор всех раскрасок графа $G$ сводится к перебору всех графов, получаемых из графа $G$ с помощью операций добавления ребра и стягивания вершин.

Дерево $C$ строится итеративно. Для этого множество вершин дерева разбивается на подмножества, просмотренных и помеченных. В начале работы алгоритма множество просмотренных вершин пусто, множество помеченных содержит корень дерева $C$. На каждом шаге алгоритм выбирает произвольную помеченную вершину $n$ и перемещает ее во множество просмотренных. Для вершины $n$ проверяются два условия:

- вершина $n$ является листом в текущем дереве $C$;
- граф $G(n)$, соответствующий вершине $n$, не является полным.

Если оба условия выполняются, то выбирается пара вершин $u$ и $v$, принадлежащих множеству вершин $G(n)$ и не соединенных ребром. Правый сын вершины $n$ — это вершина, которой соответствует граф $(G(n): <u = v>)$. Левый сын вершины $n$ — это вершина, которой соответствует граф

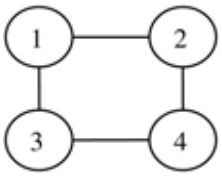



**Рис. 2. Исходный граф *G***

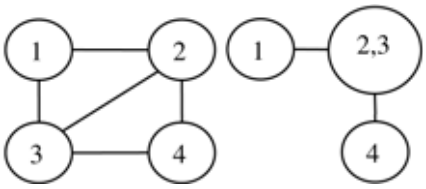



**Рис. 3. Графы, соответствующие потомкам *K* и *L* корня *C***

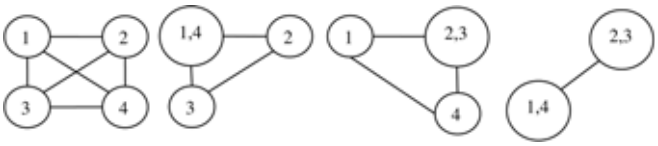



**Рис. 4. Графы, соответствующие слева направо: левому сыну *K*, правому сыну *K*, левому сыну *L*, правому сыну *L***

($G(n)$; $<u - v>$). Оба сына вершины $n$ помещаются во множество помеченных.

Алгоритм останавливает свою работу, когда множество помеченных вершин пусто.

Рассмотрим множество графов $L(C)$, соответствующих листьям дерева $C$. Каждый граф, принадлежащий множеству $L(C)$, может быть раскрашен единственным образом. Объединение раскрасок всех графов, принадлежащих множеству $L(C)$, образует множество $M(G)$.

Пусть $G'$ — это граф из $L(C)$. По раскраске $G'$ можно найти раскраску $G$. Различные вершины $G$, которые были стянуты в одну вершину $v$ графа $G'$, раскрашиваются той же краской, что и вершина $v$. Остальные вершины $G$ раскрашиваются теми же красками, что и вершины $G'$. Чтобы найти раскраску графа $G$ с наименьшим числом цветов, нужно найти граф $G' \in L(C)$ с наименьшим числом вершин. Минимальная раскраска графа $G$ имеет число цветов, равное числу вершин в $G'$.

**Пример 3.** Рассмотрим граф $G$, состоящий из четырех вершин (рис. 2).

Построим дерево всех раскрасок $G$. Граф $G$ соответствует корню этого дерева. Рассмотрим несмежные вершины 2 и 3 графа $G$. Обозначим сыновей корня дерева $C$ как $K$ и $L$ (рис. 3). Графы, соответствующие левому сыну $K$, правому сыну $K$, левому сыну $L$ и правому сыну $L$, приведены на рис. 4.

Сложность приведенного алгоритма экспоненциальная, что делает его неприемлемым в случае графов больших размерностей.

### Параметрический древовидный алгоритм раскраски ГН

Пусть $C'$ — некоторое поддерево дерева $C$ всех вариантов раскраски, причем $C'$ содержит корень $C$. Пусть параметр LeftChilds — это максимальное число левых сыновей на любом пути от корня к листу в дереве $C'$. Рассмотрим два значения $i$ и $j$ параметра LeftChilds. Если выполняется $0 \leqslant i \leqslant j$, то дерево $C'(i)$ вложено в дерево $C'(j)$.

Параметрический алгоритм строит дерево $C'$. Чем выше значение LeftChilds, тем большее число шагов будет выполнять параметрический алгоритм. Таким образом, в зависимости от требований к качеству раскраски алгоритм позволяет тратить больше времени для получения лучшего результата.

Пусть величина Built($n$) — это число левых сыновей на пути от корня дерева $C'$ к вершине $n$. Обозначим множество $\Gamma(v)$ как множество всех вершин, смежных вершине $v$. Для двух вершин ГН $u$ и $v$ введем в рассмотрение предикат:

$$I(u, v) = (\Gamma(v) \subset \Gamma(u) == \text{TRUE}) \vee \\ \vee (\Gamma(u) \subset \Gamma(v) == \text{TRUE}). \qquad (2)$$

Параметрический алгоритм является модификацией алгоритма, рассмотренного в предыдущем разделе, заключающейся в методе построения сыновей для помеченной вершины.

Рассмотрим метод построения сыновей для такой помеченной вершины $n$ дерева $C'$, что соответствующий ей граф $G(n)$ не является полным. Выбирается пара вершин $u$ и $v$, принадлежащих множеству вершин $G(n)$, не соединенных ребром. Правый сын вершины $n$ — это вершина, которой соответствует граф ($G(n)$: $<u = v>$). Левый сын вершины $n$, которому соответствует граф ($G(n)$; $<u - v>$), может существовать в дереве $C'$, а может и не существовать. Он существует, если выполняются оба условия:

- предикат (Built($n$) < LeftChilds) истинен;
- для пары вершин $u$ и $v$ предикат (2) ложен.

**Лемма.** Рассмотрим граф несовместимости $G$. Возьмем две его произвольные вершины $u$ и $v$. Пусть выполняется $\Gamma(v) \subset \Gamma(u)$ или $\Gamma(u) \subset \Gamma(v)$. Тогда хроматическое число графа ($G$: $<u = v>$) не больше, чем $G$.

Доказательство. Рассмотрим случай, когда выполняется $\Gamma(v) \subset \Gamma(u)$. В данном случае стягивание вершин $u$ и $v$ эквивалентно операции выкидывания вершины $v$ вместе со всеми инцидентными ей ребрами из графа $G$. Очевидно, что такая операция не может ухудшить хроматическое число $G$. Случай $\Gamma(u) \subset \Gamma(v)$ доказывается аналогично.

Конец доказательства.

Согласно лемме, для пары вершин $u$ и $v$ истинность предиката (2) означает, что ($G(n)$: $<u = v>$) имеет не худшее хроматическое число, чем $G(n)$. В этом случае построение левого сына излишне, поскольку его хроматическое число не лучше хроматического числа правого сына. Поэтому левый сын $G(n)$ строится только в случае, когда для пары вершин $u$ и $v$ предикат — ложь.

У каждой вершины дерева $C'$, не являющейся листом, есть правое и левое поддерево. Рассмотрим множество левых поддеревьев LeftTrees дерева $C'$. Множество LeftTreeRoots — это множество

вершин дерева $C'$, являющихся корнями поддеревьев, принадлежащих LeftTrees.

Рассмотрим подмножество вершин NotVisited, принадлежащих LeftTreeRoots, таких что параметрический алгоритм не перебирал раскрасок соответствующих их поддеревьям из LeftTrees. Параметрический алгоритм всегда находит наилучшую раскраску, если для всех вершин из NotVisited предикат (2) истинен. Таким образом, при некотором, быть может даже нулевом, значении параметра LeftChilds параметрический алгоритм может находить наилучшую раскраску. Поскольку алгоритм Чейтина—Бриггса—Вигдала является жадным, для некоторых графов он не способен найти наилучшую раскраску.

Рациональным применением рассмотренного алгоритма может быть его применение после того, как алгоритм Чейтина—Бриггса не нашел в графе несовместимости вершин степени меньше $N$. В таком случае комбинация параметрического алгоритм и алгоритма Чейтина—Бриггса всегда будет находить по крайней мере не худшую раскраску.

### Замечания по реализации и оценка сложности параметрического алгоритма раскраски

Рассмотрим реализацию параметрического алгоритма. Пусть все вершины ГН $G(V, E)$ пронумерованы, $K$ — это число вершин во множестве $V$.

На каждом шаге построения дерева $C'$ нужно выбирать неупорядоченную пару несмежных вершин, которые еще не были просмотрены. Будем просматривать пары вершин в лексикографическом порядке. Пусть уже просмотрены пары вершин, предшествующие паре $(s, t)$. Тогда, чтобы найти пару несмежных вершин, будем просматривать пары в следующем порядке: $(s, t)$, $(s, t + 1)$, $(s, t + 2)$, ..., $(s, K)$, $(s + 1, s + 2)$, $(s + 1, s + 3)$, ..., $(K - 1, K)$.

Пусть найдена пара несмежных вершин $(u, v)$. После операции стягивания вершин $u$ и $v$ номер результирующей вершины равен $u$. Если вершина с номером $u$ имеет ребра ко всем вершинам графа, то она выбрасывается вместе с этими ребрами, и размерность задачи уменьшается.

На каждом шаге алгоритма необходимо знать значение предиката (2) для выбранной пары вершин $u$ и $v$. Очевидно, что предикат (2) можно переформулировать в следующих терминах:

$$I(u, v) = (\deg(uv) == \deg(v)) \vee \\ \vee (\deg(uv) == \deg(u)), \qquad (3)$$

где вершина $uv$ является результатом стягивания вершин $u$ и $v$.

Граф $G$ будем представлять с помощью битовой матрицы $Q$ размерности $K \times K$. Если $i$-я и $j$-я вершины смежны, то $j$-й бит в $i$-й строке равен единице; если не смежны, то $j$-й бит в $i$-й строке равен нулю.

Введем вспомогательный битовый вектор $S$ размерности $K$. Вектор $S$ помогает эмулировать операции стягивания вершин и выкидывания вершины из графа. В векторе $S$ бит с номером $i$ соответствует $i$-й вершине графа $G$. В начале работы алгоритма положим все биты вектора $S$ равными единице. Если $i$-й бит вектора $S$ равен нулю, это означает, что в текущем графе вершина с номером $i$ выкинута.

Пусть необходимо стянуть $i$-ю и $j$-ю вершины. Обозначим строку матрицы $Q$ с номером $i$ как $Q_i$. Пусть множество $E$ — это множество номеров вершин, смежных $j$-й вершине. Перечислим шаги, необходимые для того, чтобы полностью описать граф после стягивания пары вершин:

1. Строятся векторы $T_1 = Q_i \wedge S$ и $T_2 = Q_j \wedge S$, где операция $\wedge$ означает побитовое "и".

2. Строится вектор $R = T_1 \vee T_2$, где операция $\vee$ означает побитовое "или".

3. Для вычисления предиката (3) необходимо посчитать число битов, равных единице для векторов $T_1$, $T_2$ и $R$.

4. Вершина, являющаяся результатом стягивания, получает номер $i$, и поэтому $Q_i = R$.

5. Бит с номером $j$ в векторе $S$ необходимо приравнять нулю.

6. Для каждой строки с номером 1, принадлежащим множеству $E$, ее $i$-й бит необходимо приравнять единице.

7. После просмотра элемента $(i, K)$ $i$-й бит вектора $S$ необходимо приравнять нулю. Эта операция эмулирует выкидывание вершины, смежной всем вершинам в графе.

Покажем, что 6-й шаг излишен. Заметим, что приравнивание $i$-го бита в строке с номером 1 единице выполняется, если после стягивания $i$-я и 1-я вершины соединены ребром. Но в этом случае операция стягивания $i$-й и 1-й вершины выполняться не будет. Таким образом, пока просматривается $i$-я строка, 6-й шаг проводить нет необходимости. В момент, когда просмотр $i$-й строки закончен, $i$-я вершина имеет ребра во все вершины текущего графа, и 7-й шаг выкидывает ее из графа. Поскольку $i$-я вершина не участвует больше в операциях с графом, показано, что выполнять 6-й шаг необходимости нет.

Современные архитектуры, например [6, 7], зачастую обладают регистрами размером не менее 64 бит. Такие регистры и битовые операции над ними типа "побитовое и", "побитовое или", "подсчет числа единичных битов" позволяют эффективно реализовать шаги с номерами 1, 2 и 3. Пусть размер регистров, над которыми могут эффективно проводиться вышеперечисленные операции,

равен register_size. Тогда сложность первых трех шагов операции стягивания равна:

$$\text{Complexity} = (K + K \bmod \text{register\_size})/\text{register\_size} \times 6.$$

Заметим также, что число операций стягивания при прохождении пути от корня дерева $C'$ до его листа невелико и равно $K - N$. Таким образом, параметрический алгоритм пригоден для реализации в индустриальных компиляторах.

### Заключение

Построенный алгоритм раскраски ГН является развитием методов раскраски графов, предложенных в литературе и допускает эффективную реализацию. В дальнейших работах автор предполагает обобщить методы трансформации программы в случае, если ГН не раскрашиваем доступным числом цветов. Предполагается реализация параметрического алгоритма в одном из компиляторов: либо GCC [9], либо OPS [10].

#### Список литературы


1. **Chaitin G., Auslander M., Chandra A., Cocke J., Hopkins M., and Markstein P.** Register allocation via coloring // Computer Languages. 1981. Vol. 6. P. 47—57.
2. **Aho A. V., Hopcroft J. E. and Ulman J. D.** The Design and Analysis of Computer Algorithms. Addison Wesley Reading, MA, USA, 1974.
3. **Briggs P.** Register allocation via graph coloring. A thesis submitted in partial fulfillment of the requirement for the Degree Doctor of Philosophy. Houston, Texas, USA, 1992.
4. **Vegdahl S.** Using Node Merging to Enhance Graph Coloring // SIGPLAN'99 {PLDI}, Atlanta, GA, USA. 1999.
5. **Muchnik S.** Advanced compiler design and implementation. Morgan-Kaufmann, San-Francisco, CA, USA, 1997.
6. **Intel®** 64 and IA-32 Architectures Optimization Reference Manual, http://www.intel.com/products/processor/manuals/
7. **Intel®** Itanium® Architecture Software Developer's Manual, http://www.intel.com/design/itanium/manuals/iiasdmanual.htm
8. **Зыков А. А.** Основы теории графов. Ростов-на-Дону: Вузовская книга, 2004.
9. **GCC** home page, http://gcc.gnu.org/
10. **OPS** home page, http://www.ops.rsu.ru


# ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ


УДК 004.8



**Т. М. Леденева,** д-р техн. наук, проф.,
**М. А. Сергиенко,** аспирант,
Воронежский государственный университет,
e-mail: fers2003@list.ru


## Организация структуры нечеткой базы правил


*Представлен способ организации структуры нечеткой базы правил в виде иерархии на основе теории графов.*



***Ключевые слова:*** *нечеткая база правил, граф, порядковая функция*


### Введение

*Нечеткая система управления* (*НСУ*) — это интеллектуальная система, использующая нечеткое описание управляемого процесса и системы его управления в виде базы нечетких правил для генерации последовательности управляющих решений, обеспечивающих достижение целей управления. Основой для построения НСУ служит схема управления с участием эксперта, который на основе опыта и знаний об управлении объектом формирует качественное описание процесса управления. Это описание преобразуется в базу продукционных правил и в дальнейшем используется в системе управления уже без участия эксперта. Идея нечеткого управления заключается именно в подражании действиям опытного человека-оператора.

Актуальная проблема разработки НСУ — формирование базы знаний, которая, как правило, представляется совокупностью базы данных и базы правил. В данной статье представлен подход к представлению базы правил в виде иерархии.

Сформированная база правил должна удовлетворять следующим критериям.

- *Критерий полноты*: база правил полна тогда и только тогда, когда для любых значений входных переменных существует, хотя бы одно правило, которое активизируется в процессе нечеткого вывода.
- *Критерий противоречивости*: в базе правил не должно быть правил с одинаковыми посылками и разными заключениями.

База правил, удовлетворяющая этим критериям, называется соответственно полной и непротиворечивой.

### Иерархическое представление базы правил

Под *нечеткой базой правил* будем понимать совокупность нечетких "если-то" правил. Обобщенный формат таких правил имеет вид

**Если** *посылка*, **то** *заключение*.

Рассмотрим базу нечетких "если-то" правил $\Re = \{R_i\}$ вида

$$R_i: \text{Если } x_1 \text{ есть } A_{i1} \text{ И } ... \text{ И } x_m \text{ есть } A_{im},$$

$$\text{ТО } y \text{ есть } B_i \ (i = \overline{1, n}),$$

где $x_j$ $(j = \overline{1, m})$ — входные переменные, которые могут быть как четкими, так и нечеткими; $X_1 \times ... \times X_m$ — область определения входной переменной (посылки), $x_j \in X_j$, $X_j$ — область определения соответствующей переменной, $y \in Y$, $Y$ — область определения выходной переменной (заключения); $A_{ij}$, $B_i$ — нечеткие множества, определенные на $X_j$, $Y$ с функциями принадлежности $\mu_{A_{ij}}(x_j)$, $\mu_{B_i}(y)$ соответственно.

*Степенью нечеткого включения* посылки правила $R_i$ в посылку правила $R_j$ назовем величину

$$I(R_i < R_j) = \text{imp}(R_i, R_j), \ (i, j = \overline{1, n}), \qquad (1)$$

imp — оператор импликации [2].

С помощью (1) вычислим матрицу включений правил

$$I = \{\text{imp}(R_i, R_j)\} = \{\text{imp}_{ij}\}_{n \times m}, \qquad (2)$$

где $\text{imp}_{ij} \in [0, 1]$. На основе полученной матрицы (2) построим ориентированный граф взаимодействия правил $G(V, E)$, обладающий следующими свойствами:

- $V$ — множество вершин графа, каждая вершина — это нечеткое правило $R_i$, $(i = \overline{1, n})$; (3)
- $E = \{\text{edge}_{ij} : \text{imp}_{ij} > \text{imp}_{ji}, \ (i, j = \overline{1, n})\}$ — множество дуг графа. (4)

Рассмотрим случаи, когда граф $G(V, E)$ является бесконтурным и содержит контуры.

1. $G(V, E)$ ***не содержит контуры.***

Определим обычные подмножества $N_0$, $N_1$, ..., $N_l$ такие, что

$$N_0 = \{R_i : E^{-1}\{R_i\} = \varnothing\},$$

$$N_1 = \{R_i : E^{-1}\{R_i\} \subset N_0\} - N_0,$$

$$\vdots$$

$$N_l = \left\{R_i : E^{-1}\{R_i\} \subset \bigcup_{k=0}^{l-1} N_k\right\} - \bigcup_{k=0}^{l-1} N_k,$$

где $l$ — наименьшее целое число, такое, что $EN_k = \varnothing$.

Полученные подмножества $N_k$, $(k = \overline{0, l})$ образуют разбиение $V$ и полностью и строго упорядочены отношением $N_k < N_{k'} \Leftrightarrow k < k'$.

Функция $O(R_i)$, определенная условием

$$R_i \in N_k \Rightarrow O(R_i) = k,$$

называется порядковой функцией обычного графа без контуров.

Другими словами, можно представить разложение множества вершин обычного графа $G$ без контуров на обычные подмножества, непересекающиеся и упорядоченные, так, что если одна из вершин принадлежит подмножеству, несущему номер $k$, то все вершины, следующие за данной вершиной, должны располагаться в подмножестве с номером, большим, чем $k$. Обычные подмножества такого разложения называются *уровнями*.

Рассмотрим пример построения иерархии правил на основе обычного графа без контуров (рис. 1).

На основе *метода определения уровней графа без контуров* [3] построим порядковую функцию и представим базу правил в виде иерархии (рис. 2).

Когда граф содержит, по крайней мере, один контур, найдется строка $\Lambda_i$, в которой невозможно добиться появления нового нуля. Этот факт дает автоматическое средство для выявления контуров в графе.

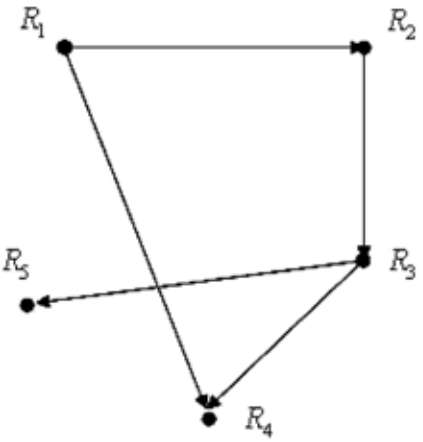



**Рис. 1. Граф взаимодействия правил**

Порядковая функция

| | |
|---|---|
| $R_1$ | 0 |
| $R_2$ | 1 |
| $R_3$ | 2 |
| $R_4$ | 3 |
| $R_5$ | 3 |

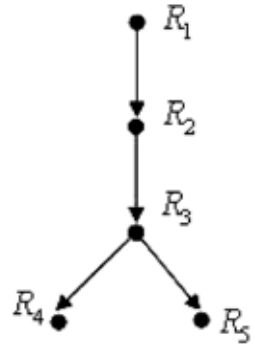



**Рис. 2. Иерархия правил**

Чтобы получить порядковую функцию при обратном упорядочивании уровней, необходимо применить ту же самую процедуру к транспонированной булевой матрице. Построение порядковой функции позволяет автоматически получать диаграмму Хассе [3], представляющую собой график частично упорядоченного множества, и определять уровни этой диаграммы.

2. $G(V, E)$ ***содержит контуры.***

Для соответствующего графа определим матрицу достижимостей:

$$\mathbf{D} = \{d_{ij}\}, \text{ где } d_{ij} = \begin{cases} 1, \textit{ если вершина } R_j \\ \textit{достижима из } R_i, \\ 0 \textit{ в противном случае.} \end{cases}$$

Множество вершин $D(R_i)$ графа $G$, достижимых из заданной вершины $R_i$, состоит из таких элементов $R_j$, для которых $(i, j)$-й элемент в матрице достижимостей равен 1. Очевидно, что все диагональные элементы в матрице **D** равны 1, поскольку каждая вершина достижима из себя самой с помощью пути длины 0.

Матрица контрадостижимостей $\mathbf{Q} = \{q_{ij}\}$ определяется следующим образом:

$$q_{ij} = \begin{cases} 1, \textit{ если из вершины } R_j \textit{ можно} \\ \textit{достигнуть } R_i, \\ 0 \textit{ в противном случае.} \end{cases}$$

Из определений очевидно, что столбец $R_i$ матрицы **Q** совпадает со строкой $R_i$ матрицы **D**, т. е. $\mathbf{Q} = \mathbf{D}^t$, где $\mathbf{D}^t$ — матрица, транспонированная к матрице **D**.

Так как $D(R_i)$ является множеством вершин, достижимых из $R_i$, а $Q(R_j)$ — множеством вершин, из которых можно достигнуть $R_j$, то $D(R_i) \cap Q(R_j)$ — множество таких вершин, каждая из которых принадлежит, по крайней мере, одному пути, идущему от $R_i$ к $R_j$. Эти вершины называются существенными или неотъемлемыми относительно двух концевых вершин $R_i$ и $R_j$, а все оставшиеся вершины — несущественными или избыточными, поскольку их удаление не влияет на пути от $R_i$ к $R_j$.

Ориентированный граф называется сильно связным, если для любых двух различных вершин $R_i$ и $R_j$ существует, по крайней мере, один путь, соединяющий $R_i$ с $R_j$. Это определение означает также, что любые две вершины такого графа взаимно достижимы.

Сильной компонентой (СК) графа $G$ будем называть максимально сильный связный подграф графа $G$. Поскольку в сильно связном графе произвольная вершина $R_j$ достижима из любой другой вершины $R_i$, то в ориентированном графе существует одна и только одна СК, содержащая данную вершину $R_i$. В самом деле, если бы вершина $R_i$ принадлежала двум или большему числу сильных компонент, то существовал бы путь из любой вершины одной СК в произвольную вершину другой СК и, следовательно, объединение этих сильных компонент было бы сильно связным графом, что противоречит определению СК.

Если вершина $R_i$ одновременно является начальной и конечной вершиной пути, то множество вершин, существенных относительно этих двух идентичных концов, совпадает с пересечением $D(R_i) \cap Q(R_i)$. Поскольку все эти существенные вершины достижимы из $R_i$ и, кроме того, из каждой такой вершины достижима вершина $R_i$, то все они взаимно достижимы. Более того, если нет другой вершины, существенной относительно концов $R_i$ и $R_i$, то множество $D(R_i) \cap Q(R_i)$ однозначно определяет СК графа $G$, содержащую вершину $R_i$.

Если эти вершины удалить из графа $G$, то в оставшемся графе можно таким же способом выделить новую СК, содержащую $R_j \in V - D(R_i) \cap Q(R_i)$. Эту процедуру можно повторять до тех пор, пока все вершины графа $G$ не будут сгруппированы в соответствующие СК. После завершения этой процедуры граф $G$ будет разбит на свои СК.

Граф $G^*(V^*, E^*)$ называют конденсацией графа $G(V, E)$, если каждая вершина графа $G^*$ представляет множество вершин некоторой СК графа $G$ (разным компонентам из $G$ соответствуют разные вершины в $G^*$), дуга $(R_i^*, R_j^*)$ существует в $G^*$ тогда и только тогда, когда в $G$ существует дуга $(R_i, R_j)$ такая, что $R_i$ принадлежит компоненте, соответствующей вершине $R_i^*$, а $R_j$ — компоненте, соответствующей вершине $R_j^*$.

Совершенно очевидно, что конденсация $G^*$ не содержит циклов, поскольку наличие цикла означает, что любые вершины этого цикла взаимно достижимы, а поэтому совокупность всех вершин цикла принадлежит СК в $G^*$ и, следовательно, содержится в СК графа $G$, что противоречит определению конденсации, в силу которого вершины из $G^*$ соответствуют СК в $G^*$.

Рассмотрим пример построения конденсации.

Для графа $G$, приведенного на рис. 3, найдем СК. Найдем СК в графе $G$ для вершины $R_1$:

$$D(R_1) = \{R_1, R_2, R_4, R_5, R_6, R_7, R_8, R_9, R_{10}\},$$

$$Q(R_1) = \{R_1, R_2, R_3, R_5, R_6\},$$

следовательно, СК, содержащая вершину $R_1$, является порожденным подграфом $\langle D(R_1 \cap Q(R_1)\rangle = \langle\{R_1, R_2, R_5, R_6\}\rangle$.

Аналогично, СК, содержащая $R_8$, есть порожденный подграф $\langle\{R_8, R_{10}\}\rangle$; СК, содержащая $R_7$, — подграф $\langle\{R_4, R_7, R_9\}\rangle$; СК, содержащая $R_{11}$, — подграф $\langle\{R_{11}, R_{12}, R_{13}\}\rangle$; СК, содержащая $R_3$, — подграф $\langle\{R_3\}\rangle$. Конденсация $G^*$ приведена на рис. 4.

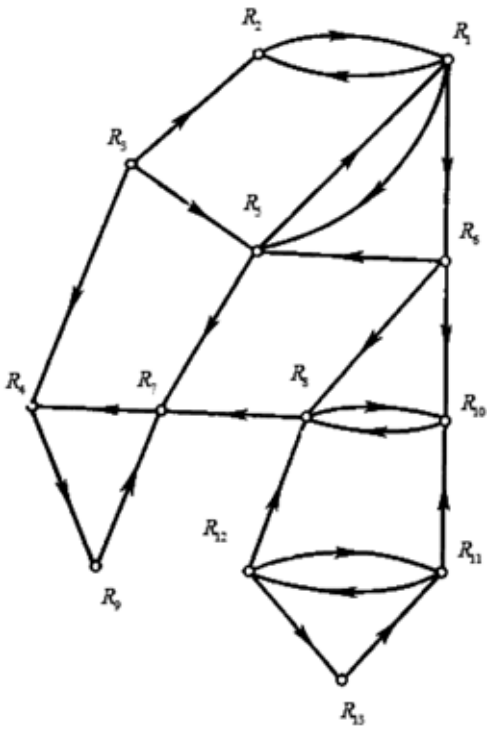


**Рис. 3. Граф взаимодействия правил**

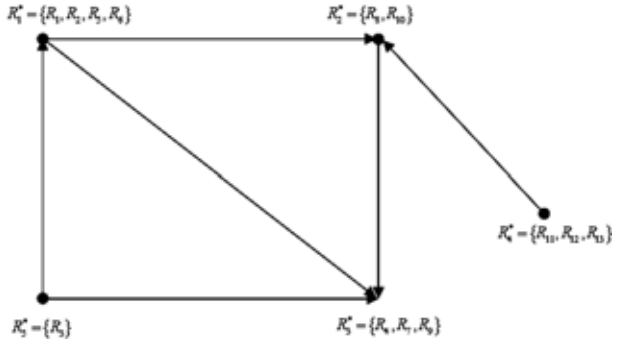


**Рис. 4. Конденсация *G** графа *G***

| | $R_1$ $R_2$ $R_3$ $R_4$ | $R_5$ $R_{10}$ | $R_6$ $R_7$ $R_8$ | $R_{11}$ $R_{12}$ $R_{13}$ | $R_9$ |
|---|---|---|---|---|---|
| $R_1$ | 1 1 1 1 | | | | |
| $R_2$ | 1 1 1 1 | | | | |
| $R_3$ | 1 1 1 1 | 0 | 0 | 0 | 0 |
| $R_4$ | 1 1 1 1 | | | | |
| $R_5$ | | 1 1 | | | |
| $R_{10}$ | 0 | 1 1 | 0 | 0 | 0 |
| $R_6$ | | | 1 1 1 | | |
| $R_7$ | 0 | 0 | 1 1 1 | 0 | 0 |
| $R_8$ | | | 1 1 1 | | |
| $R_{11}$ | | | | 1 1 1 | |
| $R_{12}$ | 0 | 0 | 0 | 1 1 1 | 0 |
| $R_{13}$ | | | | 1 1 1 | |
| $R_9$ | 0 | 0 | 0 | 0 | 1 |

**Рис. 5. Матрица D ⊗ Q**

Процедуру, описанную выше, можно сделать более удобной, если непосредственно использовать матрицы **D** и **Q**. Пусть запись **D** ⊗ **Q** означает поэлементное умножение этих матриц, тогда сразу видно, что строка $R_i$ матрицы **D** ⊗ **Q** содержит единицы только в тех столбцах $R_j$, для которых выполняется условие: вершины $R_i$ и $R_j$ взаимно достижимы. Таким образом, две вершины находятся в одной и той же СК тогда и только тогда, когда соответствующие им строки (или столбцы) в матрице **D** ⊗ **Q** идентичны. Вершины, которым соответствуют строки, содержащие 1 в столбце $R_j$, образуют множество вершин СК, содержащей $R_j$. Отсюда следует, что матрицу **D** ⊗ **Q** можно преобразовать путем транспонирования строк и столбцов в блочно-диагональную. Каждая из диагональных подматриц этой матрицы соответствует СК графа *G* и содержит только единичные элементы, а все остальные элементы этой матрицы равны 0.

Для приведенного выше примера матрица **D** ⊗ **Q**, преобразованная соответствующим образом, имеет вид, изображенный на рис. 5.

Таким образом, алгоритм построения иерархии нечетких правил примет вид.

- *Шаг 1.* Вычисление элементов матрицы включений правил (2).
- *Шаг 2.* Построение ориентированного графа взаимодействия правил *G*, обладающего свойствами (3), (4).
- *Шаг 3.* Проверка полученного графа на наличие контуров. Если полученный граф содержит контуры, то строится конденсация *G** графа *G*, которая не содержит контуров.
- *Шаг 4.* Построение для ориентированного графа без контуров порядковой функции, позволяющей разложить базу правил на уровни.

## Заключение

Предложенный алгоритм позволяет представлять нечеткую базу правил в виде иерархии. Правила в такой структуре будут располагаться по уровням согласно их посылкам. Использование порядковой функции позволяет ускорить поиск правил, сформулировать систему приоритетов правил. Избавление от контуров приводит к сокращению базы правил.

### Список литературы


1. **Kandel A., Langholz G.** Fuzzy Control Systems // CRC Press LLC, 1993. P. 187.
2. **Леденева Т. М.** Обработка нечеткой информации. Воронеж: Изд-во Воронежского государственного университета, 2006. 233 с.
3. **Кофман А.** Введение в теорию нечетких множеств. М.: Радио и связь, 1982. 428 с.
4. **Кристофидес Н.** Теория графов. М.: Мир, 1978. 427 с.

УДК 004.89:004.4

**А. В. Черний,** аспирант,
e-mail: cherny@tpu.ru,
**А. Ф. Тузовский,** д-р техн. наук, проф.,
e-mail: tomo@osu.cctpu.edu.ru,
Томский политехнический университет

# Semantic Web масштаба организации

*Описано применение методологии Semantic Web в организациях для решения задачи интеграции разнородных ресурсов информации и данных. Описан подход к разработке такой системы, предложена архитектура, используемые технологии и программные продукты. Система основана на наборе онтологий и нацелена на извлечение, интеграцию, категоризацию, поиск разнородных объектов знаний организации на основе имеющихся в организации информационных систем и документов. Система позволяет интегрировать знания нескольких организаций-партнеров. Описанные методология и система представляют собой универсальную платформу для создания систем управления знаниями и могут быть применены в любой предметной области.*

***Ключевые слова:*** *онтология, Semantic Web, OWL, RDF, триплеты, системы управления знаниями, семантическое аннотирование, семантический поиск, интеграция*

## Введение

В настоящее время ведутся активные исследования и разработки в области реализации концепции Semantic Web. Данная концепция предложена консорциумом W3C в качестве новой модели развития Web сети — Semantic Web. Основным содержанием данного подхода является использование семантических моделей (онтологий) для описания метаданных ресурсов всемирной сети. Появились стандарты языков описания онтологий и метаданных (например, RDF[1], OWL[2], SPARQL[3]), а также множество инструментов для работы с онтологиями, таких как редакторы онтологий (например, Protégé) и системы логического вывода (например, Pellet, Racer, Joseki) [4]. Однако следует отметить трудность реализации концепции Semantic Web в рамках глобальной сети (Интернет) в ближайшее время, ввиду проблем с созданием и поддержкой описаний, а также в связи со сложностью интеграции ресурсов. Но идеи и существующие технологии Semantic Web могут быть использованы для развития информационных систем в рамках локальных сетей (интранет). В данной статье описан подход к развитию существующих информационных систем организации с помощью семантических технологий, использующихся в глобальной сети, таких как смысловой поиск, категоризация, навигация по ресурсам, интеграция и т. д.

## Описание основных понятий

Вся информация и данные организации содержатся во множестве разнообразных источников $R$ (документы, файлы, базы данных, внешние ресурсы, на которые есть ссылки) $R = \{R_1, ..., R_n\}$. Кроме этого, в организации имеется множество потребителей информации (специалисты, каталоги, бизнес-процессы и входящие в них задачи), которые заинтересованы в своевременном получении некоторого вида информации.

Для организации работы с разнотипной и распределенной информацией и данными предлагается использовать онтологическое моделирование [5]. Под онтологической моделью (онтологией) $O$ в данной статье понимается знаковая система $\langle C, T, P, F, L, A\rangle$, где $C$ — множество элементов, которые называются понятиями; $T$ — частичный порядок на множестве $C$, задающий отношения "подкласс" и "суперкласс"; $P$ — множество элементов, которые называются свойствами (двуместными предикатами); $F$ — функция, которая назначает каждому элементу множества $P$ множество элементов из множества $C$ (с учетом их иерархии в $T$), к которым оно применимо (область действия, область, *domain*), и множество элементов из множества $C$ или *литералов* (экземпляров примитивных типов, таких как строки и числа), которые могут быть их значениями (область возможных значений, интервал, *range*); $L = \{L^C, L^P, \alpha^C, \alpha^P\}$ — множество текстовых меток, которые определяют профессиональные термины организации и их соответствие, соответствие $\alpha^C$ — элементам множества $C$, $\alpha^P$ — элементам множества $P$; $A$ — набор аксиом онтологии — утверждения об элементах предметной области, которые считаются верными, выраженных с использованием соответствующего логического языка.

Создание единой онтологии для детального описания модели знаний организации является весьма трудоемкой задачей. Для решения задачи построения онтологии организации предлагается использовать онтологическую модель следующей структуры [6]: $O = \{O_о, O_р, O_з\}$, где $O_о$ — онтология организации; $O_р$ — онтология информационных ресурсов; $O_з = \{O_1, ..., O_m\}$ — иерархически организованная, последовательно расширяемая система онтологии основных областей знаний $O_i$, значимых для работы организации. Выделение иерархии областей знаний организации дает возможность создавать отдельно онтологии разных подобластей

знаний, которые могут иметь разную детальность, в зависимости от потребностей их моделирования. Онтология организации $O_o$ включает основные понятия, которые описывают структуру, состав элементов и работу организации (подразделения, специалисты, заказчики, проекты, бизнес-процессы и пр.). Онтология информационных ресурсов $O_p$ включает описание всех видов ресурсов данных и информации организации (документы, файлы, базы данных, программы и пр.).

На основе такой онтологической модели знаний $O$ описание ресурса (объекта) $R_i$ может быть представлено в виде набора семантических метаданных [6] следующей структуры: $M_i = (M_{ki}(O), M_{ci}(O))$, где $M_{ki}(O)$ — это *контекстные метаданные* объекта знаний, описывающие взаимосвязи объекта с другими объектами и понятиями организации или литералами, а $M_{ci}(O)$ — *контентные метаданные* ресурса, описывающие информацию, которая содержится в объекте. *Контекстные метаданные* соответствуют набору значений свойств понятия $c_j \in C$, экземпляром которого является описываемый объект в онтологии организации, т. е. $M_{ki}(O) = (p_1(O_i, v_1) \vee p_2(O_i, v_2) \vee ... \vee p_r(O_i, v_r))$, где утверждение $p_i(O_i, v_i)$ состоит из предиката (отношения) $p_i \in P$, описанного в онтологии организации, *URI* (универсальный идентификатор ресурса) описываемого объекта $O_i$, для которого определяются метаданные, и значения $v_i$, которое может быть либо *URI* некоторого экземпляра понятий онтологии организации, либо некоторый литерал (текст, число, дата) в соответствии с областью действий и областями возможных значений свойства $p_i$. *Контентные метаданные* $M_{ci}(O)$ описываются наборами утверждений из $O_3$, т. е. семантические метаданные объектов знаний описываются отношениями и понятиями из онтологии основных предметных областей знаний организации в виде набора кортежей: $M_{ci}(O) = (\{p_1(s_1, v_1), k_1\} \vee \{p_2(s_2, v_2), k_2\} \vee ... \vee \{p_k(s_k, v_k), k_k\})$, где $\{p_i(s_i, v_i), k_i\}$ — кортеж, включающий утверждение $p_i(s_i, v_i)$ (соответствующее *RDF* триплету $(s_i, p_i, v_i)$) и $k_i$ — важность данного утверждения для описания контента объекта знаний $i$. Утверждение $p_i(s_i, v_i)$ состоит: из предиката (отношения) $p_i \in P$, описанного в онтологии областей знаний; объекта $s_i$, которым может быть понятие онтологии $c_j \in C$ или экземпляр онтологии $i$ (ссылка на контекстные метаданные некоторого экземпляра онтологии), значения $v_i$, которое может быть либо *URI* некоторого экземпляра, либо некоторым литералом (текстом, числом, датой).

При этом следует отметить, что не все экземпляры понятий онтологии будут соответствовать ресурсам информации или данных организации, например, в онтологию также могут быть включены такие понятия, как Специалист, Заказчик, Проект и т. п.

Совокупность описаний онтологической модели и метаданных всех ресурсов организации составляют ***онтологическую базу знаний*** системы.

### Измерение близости метаописаний ресурсов

Учитывая, что модели описаний объектов знаний связаны за счет использования единой модели знаний организации, имеется возможность оценки их подобия (сходства) между собой на основе некоторой метрики подобия $\Phi(M_i, M_j)$. В статье предложены методы оценки подобия контекстных и контентных метаданных объектов знаний. Данный функционал описывает близость между отдельными элементами знаний, содержащимися в разных объектах знаний.

Формализованное представление онтологической модели знаний, а также метаописаний ресурсов создает возможность для измерения близости (подобия) ресурсов в интеллектуальном пространстве организации. Подобие между метаданными $\Phi(M_{3i}, M_{3j})$ может быть определено через подобие входящих в них утверждений:

$$\Phi(M_{3i}, M_{3j}) = \sum_{I_i \in MD_i} \sum_{I_j \in MD_j} \mathrm{sim}(T_i, T_j),$$

где $\Phi(M_{3i}, M_{3j})$ — близость метаописаний объекта $i$ и объекта $j$; $\mathrm{sim}(T_i, T_j)$ — близость утверждений (триплетов) $T_i$ и $T_j$, входящих в сравниваемые метаописания. Величины $\mathrm{sim}(T_i, T_j)$ могут быть определены с использованием следующего выражения:

$$\mathrm{sim}(T_i, T_j) = \mathrm{sim}((c_i, r_j, i_k k_1), (c_x, r_y, i_z k_w)) =$$
$$= \frac{\mathrm{sim}c(c_i, c_x) + \mathrm{sim}r(r_i, r_y) + \mathrm{sim}i(i_k, i_z)}{3} f(k_l, k_w),$$

где $\mathrm{sim}c(c_i, c_k)$ — семантическая близость понятий, используемых в утверждениях; $\mathrm{sim}r(r_i, r_y)$ — семантическая близость отношений онтологий; $\mathrm{sim}i(i_k, i_z)$ — семантическая близость контекстных метаданных экземпляров понятий онтологий; $f(k_l, k_w)$ — функция учета коэффициентов важности утверждений (используются разные варианты).

**Оценка семантической близости двух понятий.** Поясним вычисление семантической близости более подробно на примере ее оценки для двух понятий.

В онтологии $O$ на множестве понятий $C$ задано отношение нестрогого — частичного порядка $T_C$. Утверждение $T_C(c_k, c_l)$ означает, что $c_k$ предшествует $c_l$ или что $c_l$ следует за $c_k$. Причем $T_C$ задано так, что среди элементов множества $C$ существует *единственный минимальный* (базовый) элемент $c_{top} \in C$. Иерархия понятий с единственной вер-

шиной (таксономия понятий), заданная отношением $T_C$, используется для определения семантической близости понятий.

Для каждого понятия $c_i \in C$ существует множество $C_{ANC}(c_i)$, являющееся подмножеством $C$ и содержащее понятия, предшествующие понятию $c_i$, а также само понятие $c_i$:

$$C_{ANC}(c_i) = \{c_j \in C \mid T_C(c_j, c_i) \vee c_j = c_i\}.$$

Для оценки семантической близости двух понятий $\mathrm{sim}_C(c_k, c_l)$ вводятся два показателя, основанные на сравнении множеств $C_{ANC}(c_i)$:

$$\mathrm{sim}_C(c_k, c_l) = k_{st}\frac{\left|C_{ANC}(c_k) \cap C_{ANC}(c_l)\right|}{\left|C_{ANC}(c_k) \cup C_{ANC}(c_l)\right|};$$

$c_{\mathrm{top}} \in C_{ANC}(c_k) \cap C_{ANC}(c_l)$ и $c_{\mathrm{top}} \in C_{ANC}(c_k) \cup C_{ANC}(c_l)$ при любых $c_k \in C$ и $c_1 \in C$.

$$k_{st} = \begin{cases} 0, \text{ если } C_{ANC}(c_k) \cap C_{ANC}(c_l) = c_{\mathrm{top}} \\ 1, \text{ иначе} \end{cases}$$

$\mathrm{sim}_C(c_k, c_l) \in [0; 1]$.

На рис. 1 представлена простая семантическая модель, на которой поясняется оценка семантической близости между понятиями.

На приведенном рисунке $\mathrm{sim}(c_a, c_b) > \mathrm{sim}(c_b, c_c)$, $\mathrm{sim}(c_d, c_b)$, $\mathrm{sim}(c_d, c_a)$, $\mathrm{sim}(c_d, c_c) = 0$, так как для этих оцениваемых понятий коэффициент $k_{st} = 0$.

На основе использования данного функционала могут быть решены разные задачи интеграции разнородной информации и данных: смысловой поиск документов (любой информационный ресурс, описанный с помощью метаданных); классификация объектов знаний по системе рубрик; рекомендация (предоставление) новых объектов знаний специалистам.

**Семантический поиск.** Семантический (смысловой) поиск состоит в поиске объектов базы знаний, у которых семантические метаданные близки (значение семантической близости превышает некоторое пороговое значение) семантическому описанию поискового запроса. Объектом-эталоном при семантическом поиске является поисковый запрос, представленный в виде контентных метаданных. Процедура формирования множества объектов-кандидатов для выполнения среди них семантического поиска заключается в выборе пользователем тех понятий из онтологии информационных ресурсов, которым соответствуют требуемые типы объектов. В результате семантического поиска пользователю предлагается список найденных по запросу объектов, упорядоченных по значению семантической близости и дополнительно сгруппированных по типам объектов.

**Категоризация.** Под задачей категоризации понимается распределение ресурсов информации между иерархически организованным набором

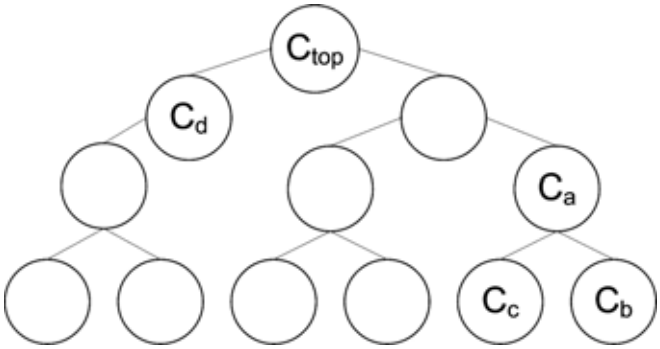



**Рис. 1. Схема онтологии для иллюстрации оценки семантической близости**

рубрик (категорий). Каждая рубрика характеризуется семантическими метаданными, описывающими информацию, которая должна в них содержаться. Родительские и дочерние рубрики в иерархии классификации связаны между собой отношением включения (*subsumption*) между их семантическими метаданными. При решении задачи классификации в качестве объекта-эталона выступают метаданные рубрики классификатора. Из базы знаний извлекаются семантические метаданные всех объектов (за исключением рубрик), для которых нужно проверить соответствие рубрике. В процессе обработки результатов из множества кандидатов удаляются те объекты, контентные метаданные которых не содержат хотя бы одного понятия или экземпляра, присутствующего в семантических метаданных рубрики. Во множестве объектов-кандидатов остаются лишь те объекты, которые имеют показатель релевантности больше нуля, т. е. относятся к рубрике.

## Архитектура системы

Информационная система организации состоит из множества компьютеров, объединенных в локальную сеть, которые содержат разнообразные информационные ресурсы разных форматов. Рассматриваемая в статье система позволяет интегрировать и структурировать все общие ресурсы информации и данных. Она представляет собой набор серверов и клиентских приложений, которые должны быть установлены в локальной сети организации. Структура системы показана на рис. 2.

Серверы отвечают за совместное использование онтологической базы знаний и решения базовых задач по работе с онтологической моделью (редактирование и пополнение онтологии), метаданными (формирование семантических метаданных — аннотирование, хранение базы знаний) и информационными ресурсами (поиск, категоризация, навигация по метаданным).

Архитектура системы приведена на рис. 3.


**Рис. 2. Структура системы**

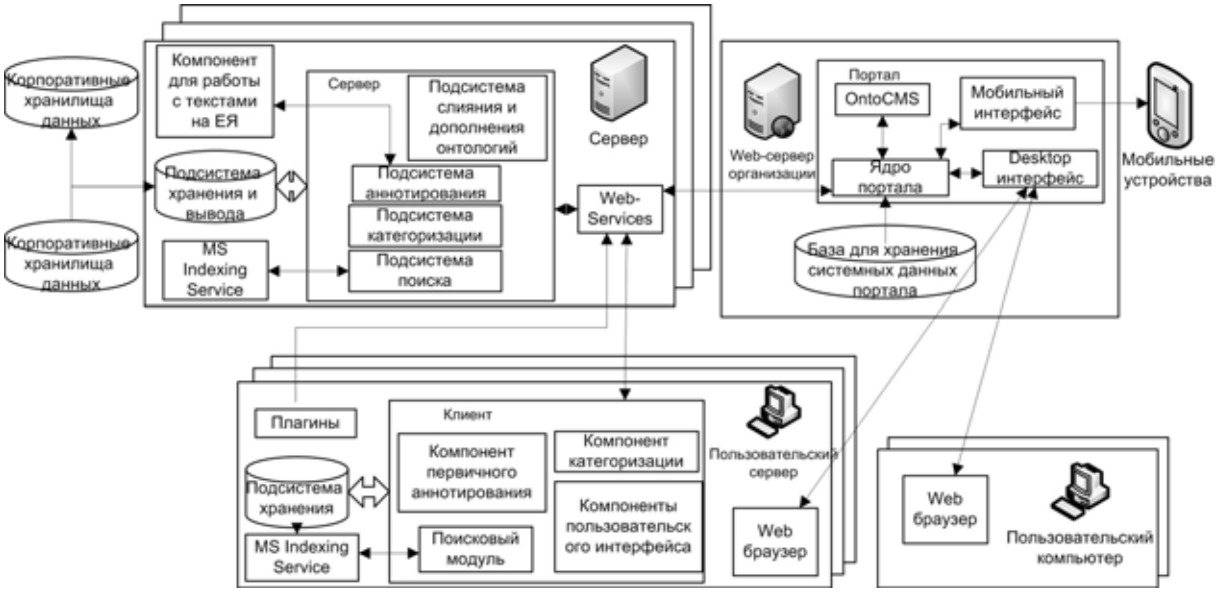



**Рис. 3. Архитектура системы**

**Описание компонентов системы.** В состав сервера входят такие подсистемы: хранения и логического вывода; аннотирования; поиска; пополнения онтологии; категоризации.

**Подсистема хранения и логического вывода.** Данная подсистема состоит из трех частей: модуля логического вывода; модуля хранения; модуля интеграции.

*Модуль логического вывода* отвечает за построение иерархий и создание новых триплетов на основе базовой онтологии за счет механизмов дескриптивной логики. Таким образом, в системе появляются связи, явно не заданные, но актуальные.

*Модуль хранения* обеспечивает хранение описания онтологической модели в реляционных базах данных.

**Модуль интеграции.** Данный модуль предоставляет функционал по интеграции онтологии системы и баз данных. Это дает возможность, не изменяя существующие в организации информационные системы, использовать данные этих ИС и иметь в онтологии всегда актуальные данные без приложения дополнительных усилий. Каждая таблица в БД ставится в соответствие с понятием онтологии. Каждый столбец этой таблицы является свойством понятия, а каждая строка — экземпляром (рис. 4).

При интеграции в свойствах каждого понятия указывается, с каким полем таблицы связано каждое свойство этого понятия. В БД добавляются триггеры, которые позволяют отслеживать обновления экземпляров в БД и актуализировать эти значения в онтологии.

Выработанная организацией базовая онтология является доступной для партнеров организации информацией (только онтология, без метаданных).

Таким образом, становится возможным слияние онтологий из разных источников, что позволяет партнерам использовать все преимущества единого информационного пространства нескольких взаимодействующих организаций. И, как следствие, более полно описанные документы, семантически описанные совместные бизнес-процессы и т. п.

Слияние онтологий выполняется совместно менеджерами по работе со знаниями организаций

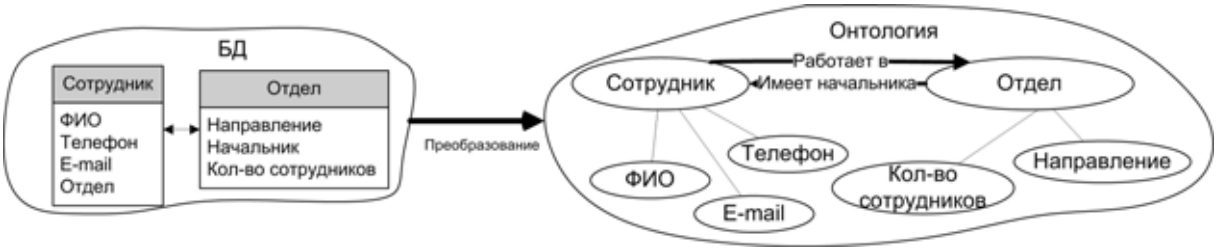



**Рис. 4. Формирование онтологической модели на основе БД**

и основано на свойствах языка *OWL*, который позволяет упростить этот процесс, находя, например, идентичные понятия и ставя их в однозначное соответствие и так далее.

**Подсистема аннотирования.** Эта подсистема выделяет из документов понятия, экземпляры понятий и связи между ними. Она основывается на использовании программного обеспечения, выполняющего грамматический разбор текстов на естественном языке. На основе этого строится описание (граф) документа, который сохраняется в виде триплетов на сервере и (если позволяет формат документа) в самом документе. С каждым триплетом, полученным из документа, связывается значение, определяющее соответствие триплета смыслу документа — значимость этого утверждения. По полученным аннотациям впоследствии проводится семантический поиск.

Аннотации, полученные таким путем, считаются первичными, и пользователю предоставляется возможность после ознакомления с документом в дружелюбном пользовательском интерфейсе откорректировать и дополнить автоматическую аннотацию.

**Подсистема поиска.** Как известно, семантический поиск дает намного более релевантные результаты, чем полнотекстовый. Но при этом считается [7], что лучшим вариантом является их совмещение.

Ввиду этого приемлемой видится система, в которой пользователь может самостоятельно задать, какой из следующих видов поиска его интересует: только семантический, только полнотекстовый или совмещенный, когда пользователь задает набор триплетов и уточняющий полнотекстовый запрос, либо помечает при наборе в строку полнотекстового запроса, что его необходимо интерпретировать и как семантический. Во втором случае система разбирает его по тем же правилам, что и тексты при аннотировании и на этой основе строит триплеты для запроса. Таким образом, пользователь получает первыми результаты, которые удовлетворяют семантическому запросу и содержат искомый текст, затем прочие в порядке убывания релевантности.

**Подсистема категоризации.** Эта подсистема служит для распределения по категориям ресурсов компании (документов, сотрудников, ссылок, программ и т. п.) на основе их семантического описания.

**Пользовательский клиент.** Данный модуль включает возможности категоризации, поиска (как было описано ранее), а также подсистемы взаимодействия с пользователем. Главной частью этой подсистемы являются модули интеграции приложений.

*Модули интеграции приложений* с системой обеспечивают средство интеграции различных приложений по работе с документами с серверами системы для аннотирования и поддержки связи информации документов с онтологической базой знаний. Например, в приложении Microsoft Word, модуль интеграции позволяет выделять понятия предметной области и выполнять с ними некоторые манипуляции. Например, читая некоторый документ, в котором есть ссылка на другой документ, имеется возможность, щелкнув кнопкой "мыши" на упоминание о нем, перейти к карточке документа и узнать, кем и когда он был создан, кем изменялся и пр., либо сразу открыть документ. Такого рода модули могут быть разработаны для браузеров, офисных приложений и т. п.

**Подсистема Web-сервера.** Web-сервер может присутствовать как в каждом подразделении, так и быть вынесенным на корпоративный уровень и главной составляющей в нем является семантический портал.

*Семантический портал* отвечает за создание и доступ к функционалу системы через web-интерфейс и состоит из базовой части и системы управления контентом на основе семантических метаданных.

*Базовая часть семантического портала* предоставляет набор классов и интерфейсов (API) для реализации других модулей и компонентов портала. Так как система является лишь платформой для создания систем интеграции информации и данных и обеспечивает лишь базовый функционал, который есть в любой организации, то для него, конечно, необходимо разрабатывать дополнительные специализированные модули, предос-

тавляющие дополнительную (специфическую для конкретной организации) функциональность.

*Система управления контентом на основе семантических метаданных* (ресурсов информации и данных) представляет собой Content Management System (CMS) с расширенным функционалом для работы с онтологиями и семантическими метаданными. Например, любой элемент управления на web-странице может формировать SPARQL запрос к онтологической базе знаний и представлять полученные результаты пользователю в удобном виде. На основе онтологической CMS реализуется подсистема поддержки задач бизнес-процессов требуемыми ресурсами информации и данных. Для поддержки бизнес-процессов необходимо выполнить их семантическое описание, оно состоит в том, что бизнес-процесс в целом, и каждая его задача в частности, описываются в понятиях онтологии. Это дает множество преимуществ. При выполнении некоторой задачи бизнес-процесса сотрудник может узнать всю информацию, полезную для выполнения этой задачи, получить описание сотрудников организации, которые выполняли данную задачу ранее (и при необходимости связаться с ними) и многое другое.

**Реализация системы.** В реализации предлагаемого подхода используют разные сторонние программные продукты. В качестве базовой подсистемы используется OWLAPI [8]. Для аннотирования могут быть использованы компоненты AOT [9] или более дорогие и более функциональные компоненты RCO [10]. В качестве движка полнотекстового поиска используется Microsoft Indexing Service, входящий в состав ОС Windows.

## Аналоги

В настоящее время за рубежом уже разработано достаточно много программных продуктов, основанных на семантических технологиях, например такие, как системы компаний Virtuoso, Intellidimension. Однако эти системы, в основном, предоставляют инструментальные средства разработки или предназначены для решения некоторых специфических задач, в то время как предлагаемое решение направлено на семантическую интеграцию знаний и информации различного типа организаций, использующих информационные системы и накопивших большое количество информационных ресурсов.

## Заключение

В целом, можно сказать, что разработанная система позволяет не только интегрировать всю информацию, которая имеется в организации, начиная от знаний о служащих, заканчивая информацией о производимых деталях или услугах, но и создавать коллективное информационное пространство, которое, впоследствии, приблизит эпоху Semantic Web.

### Список литературы


1. **Resource** Description Framework Specification. 1999. URL: http://www.w3.org/RDF/
2. **OWL** Web Ontology Language Overview. 2004. URL: http://www.w3.org/TR/owl-features/
3. **SPARQL** Query Language for RDF (W3C Recommendation). — 2008. URL: http://www.w3.org/TR/rdf-sparql-query/
4. **Allemang D., Hendler J.** Semantic Web for the Working Ontologist Modeling in RDF, RDFS and OWL. Morgan Kaufmann Publishers, 2008. 350 p.
5. **Staab S., Studer R.** (eds.) Handbook on Ontologies (International Handbooks on Information Systems). Springer Verlag, 2004. 660 p.
6. **Тузовский А. Ф.** Формирование семантических метаданных для объектов системы управления знаниями // Известия Томского политехнического университета. 2007. Т. 310, № 3. С. 184—188.
7. **Добров Б. В., Иванов В. В., Лукашевич Н. В., Соловьев В. Д.** Онтологии и тезаурусы: модели, инструменты, приложения: Лекция № 3. 2008. URL: http://www.intuit.ru/department/expert/ontoth/3/
8. **Documentation** about OWL API. 2008. URL: http://owlapi.sourceforge.net/documentation.html
9. **Пакет** документации к программному продукту RML — 2006. URL: http://www.aot.ru/technology.html.
10. **Описание** компонента RCO Fact Extractor. — 2007. URL: http://rco.ru/product.asp?ob_no = 1131


## *Поздравляем юбиляра!*

Заслуженному деятелю науки Республики Башкортостан и Российской Федерации, профессору Уфимского государственного авиационно-технического университета, постоянному и любимому автору журнала "Информационные технологии", доктору технических наук

*Элите Александровне*
***МУХАЧЕВОЙ,***
исполнилось **80 лет.**

Элита Александровна является известным ученым и талантливым преподавателем, специалистом по методам оптимизации и прикладным задачам исследования операций, руководителем одной из ведущих отечественных научных школ в области математического программирования в задачах раскроя-упаковки.

Высокий профессионализм, широкая эрудиция, большая трудоспособность, чуткость и отзывчивость снискали ей большое уважение учеников и коллег.

***Желаем Вам, Элита Александровна, крепкого здоровья, большого счастья и новых творческих успехов.***

УДК 519.7:007.52; 519.81; 519.2

**С. И. Колесникова,** канд. физ.-мат. наук, доц.,
Томский госуниверситет систем управления
и радиоэлектроники,
e-mail: skolesnikova@yandex.ru,
**В. С. Лаходынов,** аспирант,
e-mail: lahodynov@yahoo.com,
**Ю. Р. Цой,** канд. техн. наук, доц.,
Томский политехнический университет,
e-mail: yurytsoy@gmail.com

# Исследование качества распознавания состояний стохастической системы*

*Рассматривается задача распознавания (диагностирования) состояний зашумленной динамической системы, наблюдаемые значения которой представлены в виде временного ряда. Используемый математический аппарат приведенных двух подходов к решению задачи распознавания состояний в режиме реального времени включает методы распознавания образов, теории вероятностей и теории информации. Обсуждается новый адаптивный метод учета особенностей квантования сигнала, приводящий к существенному выигрышу во времени распознавания состояний (без потери качества). Работоспособность алгоритмов распознавания состояний и сравнительный анализ числовых результатов их работы показаны на данных модели системы, разработанной в среде MatLab.*

***Ключевые слова:*** *состояние динамической системы, фильтрация, методы распознавания образов, эффективность распознавания, оценка качества алгоритмов, статистические оценки*

## Введение

В современных цифровых системах управления при определении состояния объекта часто присутствует шумовая составляющая, связанная с помехами в канале связи либо с ошибками измерения. Одной из целей разработки модели для распознавания состояний динамической системы (ДС) является осуществление интеллектуального мониторинга и адаптивного управления сложными системами с параметрическими неопределенностями и случайными возмущениями в каналах передачи информации [1—3] (рис. 1).

Модель управляемой динамической системы включает в себя:

- собственно динамический объект;
- подсистему, реализующую управляющее воздействие;
- модель внешнего воздействия ("шумящей" среды).

В ряде прикладных технических задач (в теории управления, в частности) основной подход к идентификации ДС основан на построении модели в виде дифференциальных (разностных) уравнений, связывающих входные и выходные переменные. Однако наличие нелинейных зависимостей (между "входом" и "выходом"), наличие разного вида неопределенностей, нередко характеризующих экспертное мнение о поведении ДС, является серьезным препятствием для применения традиционных математических моделей к процессу идентификации состояний ДС [1—3].

В статье рассматриваются и сопоставляются два подхода к распознаванию состояний ДС (и реализующие их соответствующие алгоритмы) на основе методов теории распознавания образов [4, 5] и теоретико-информационных понятий [6, 7]. Приводятся сравнительные результаты обширного экспериментального исследования на шести

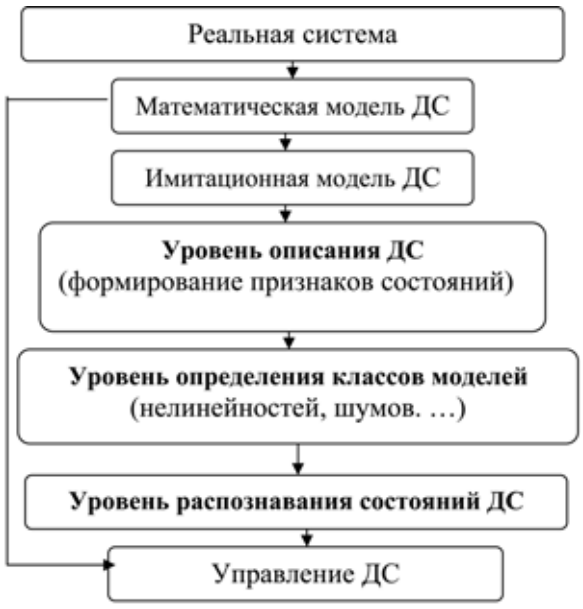



**Рис. 1. Технологическая схема распознавания состояний ДС для управления**

* Работа частично поддержана РФФИ (проект № 09-01-99014-р_офи).

различных моделях электромеханических систем (ЭМС). Показано, что процедура фильтрации (традиционно применяемая для зашумленных сигналов в целях получения достоверной информации о "тонких" процессах, протекающих в объекте управления) в тех задачах, где распознавание состояния ДС (в реальном режиме времени) является самостоятельно-достаточной задачей, не оказывает существенного влияния (в смысле качества и времени распознавания) на итоговый результат.

### Основные определения и понятия

Под *состоянием* будем понимать совокупность элементов (подмножество) фазового пространства ДС (скаляр, точка многомерного пространства, вектор); под *системой* (программой) реального времени будем понимать систему, гарантирующую время реакции (отклика) на измерение контролируемого сигнала в темпе протекания процессов в системе (например, системы обнаружения разладки ДС).

Под *дискретизацией* сигнала $Y(t)$ понимается разбиение сигнала по временной составляющей (на графике — по горизонтали): $t \in \{t_0, t_1, ..., t_n\}$, $t_j = j\Delta$, $j = \overline{0, n}$), с шагом дискретизации $\Delta > 0$, $Y_j = Y(t_j)$.

Под *квантованием* сигнала понимается разбиение диапазона значений (амплитуды $Y_j$) сигнала на отрезки равной длины (деление исходного измеряемого значения $Y_j$ на постоянную величину (шаг квантования $\delta$) и взятие целой части от частного). Глубина квантования (динамический диапазон, или уровень квантования) — целое число шагов квантования $v_j$, "уложившееся" в значение амплитуды сигнала $Y_j$. Сигнал, подверженный операциям дискретизации и квантования, называется *цифровым*.

Под *градациями* квантования будем понимать уникальные значения уровней квантования (по сути, значения случайной величины — уровня квантования). Градации состоят из одного значения при стандартном квантовании (домен $U$ — это множество градаций при стандартном квантовании).

В адаптивном методе учета особенностей квантования сигнала градации — это уникальные множества (непересекающиеся), элементами которых являются уровни квантования, близкие в определенном смысле. Как правило (в контексте данной работы), в одну градацию входят "малочисленные" (с малой частотой, редко встречающиеся) "соседние" уровни квантования. При адаптивном методе учета особенностей квантования мощности градаций (как множеств значений уровней квантования) могут быть разными.

Параметр квантования $h$ — это максимальный "зазор" между значением $Y_j$ и уровнем его квантования $v_j$, т. е. $h \geqslant |Y_j - \delta v_j|$ или $Y_j = \delta v_j + h_j$, $h_j \leqslant h$.

### Общая постановка задачи

Пусть задан случайный процесс (СП)

$$\{X,\ Y\} = \{X(t),\ Y(t),\ t_0 \leqslant t \leqslant T\};$$
$$Y(t) = f(X(t),\ \xi(t)), \qquad (1)$$

характеризующий состояние ДС, функционирующей на интервале времени от $t_0$ до $T$, где $X(t)$ — вектор ненаблюдаемых переменных состояния системы; $Y(t)$ — случайная наблюдаемая $l$-мерная векторная функция; $\xi(t)$ — шум достаточно общей природы с ограниченной дисперсией.

Относительно СП $\{X,\ Y\}$ выдвинуто $I > 1$ альтернативных гипотез $\Omega = \{\Omega_1, \Omega_2, ..., \Omega_I\}$, составляющих полную группу событий и физически интерпретируемых как классы состояний динамической системы (СДС).

Задача состоит в отнесении наблюдаемых (в момент $t$ или на некотором фиксированном интервале $[t',\ t'']$) реализаций $Y(t)$ к классу $\Omega_i$, $i = \overline{1, I}$. Наблюдение величин $Y(t)$ осуществляется в соответствии с дискретным планом $t \in \{t_0, t_1, ..., t_n\}$, $t_j = t_0 + j\Delta$, $j = \overline{0, n}$, с шагом дискретизации $\Delta > 0$, по адекватной модели ДС (в общем случае нелинейной и нестационарной).

Практическое решение поставленной задачи затруднено не только наличием возможной нелинейной связи между откликом (выходом) и входным воздействием, но и отклонениями параметров модели от реальных значений (присутствием систематической и несистематической ошибок, неизбежными при моделировании), реальностью времени решения задачи распознавания (диагностики) СДС (для последующего мониторинга и возможного управления в реальном времени).

Несмотря на большое число методов интеллектуальной обработки диагностической информации (см., например, обзоры в [1, 2]) вопрос об идентификации СДС в такой постановке остается открытым.

### Подход к решению задачи на основе непараметрической идентификации нелинейной регрессионной модели сигнала

Систему (1) будем рассматривать как стохастический объект, на $l$ входах и скалярном выходе $Z$ которого наблюдаются соответственно случайные величины $Y = (Y_1, ..., Y_l)$ и $Z$, $Y \in R^l$, $Z \in R^l$, где $R^l$ — $l$-мерное евклидово пространство.

Положим, что значение выходной переменной $Z$ определяется неизвестным функциональным

преобразованием $F(\cdot)$ входных переменных $Y$, т. е. $Z = F(Y)$. В качестве приближения неизвестной функции $F(\cdot)$ будем использовать функцию регрессии, следуя работам [8—10]. Функция регрессии или условное математическое ожидание выхода стохастического объекта относительно входов является моделью, которая минимизирует среднеквадратическое отклонение выхода реального объекта и его модели:

$$r(y) = \int\limits_{R^1} zf(z \mid y)dz = \int\limits_{R^1} zf(y, z)dz/p(y) = a(y)/p(y), \quad (2)$$

где $f(y, z)$ — неизвестная плотность распределения наблюдаемой $(l+1)$-мерной случайной величины $(Y, Z) \in R^{l+1}$ в точке $(y, z)$; $f(z \mid y) = \dfrac{f(y, z)}{p(y)}$ — условная плотность распределения случайной величины $Z \in R^1$ при условии, что $Y = y$; $p(y)$ — маргинальная плотность распределения величины $Y$; $a(y) = \int\limits_{R^1} zf(y, z)dz$.

В качестве непараметрической ядерной оценки условного функционала (1) в точке $y$ в [9] предлагаются полурекуррентные статистики вида:

$$\begin{cases} a_{[n]r}(y) = \dfrac{1}{n} \sum\limits_{i=1}^{n} \dfrac{1}{\prod\limits_{k=1}^{l} h_{[i]k}} Z_i K\left(\dfrac{y - Y_i}{h_{[i]}}\right); \\ p_{[n]r}(y) = \dfrac{1}{n} \sum\limits_{i=1}^{n} \dfrac{1}{\prod\limits_{k=1}^{l} h_{[i]k}} K\left(\dfrac{y - Y_i}{h_{[i]}}\right); \\ r_{[n]}(x) = \dfrac{a_{[n]r}(y)}{p_{[n]r}(y)}. \end{cases} \quad (3)$$

Здесь $a_{[n]r}(y)$, $p_{[n]r}(y)$ — рекуррентные оценки функционала $a(y)$ и плотности $p(y)$ соответственно; $(Y_i, Z_i)$, $Y_i = (Y_{i,1}, ..., Y_{i,l})$, $i = \overline{1, n}$ — $(l+1)$-мерная выборка, характеризуемая плотностью $f(x, y)$; $K(u) = K(u_1, ..., u_l)$ — $l$-мерная ядерная функция; $h_{[i]k} > 0$ — последовательность чисел (параметров), сходящаяся к нулю для каждого $k = \overline{1, l}$, способы которых подробно рассмотрены в [8—10].

*Пример 1. Оценка функции авторегрессии.* Пусть на выходе объекта наблюдается последовательность одномерных величин $Y_{i+1}$, $Y_{i+2}$, ..., $Y_{i+p+1}$, $i = 1, 2, ...$, которые образуют процесс авторегрессии (AR) порядка $p$, т. е. связаны соотношением

$$Y_{i+p+1} = \Psi(Y_{i+1}, ..., Y_{i+p}) + \varepsilon_{i+p+1}, \quad i = 1, 2, ..., \quad (4)$$

где $\{\varepsilon_j\}$ — последовательность независимых и одинаково распределенных случайных величин. Пусть $Y_1, ..., Y_{n+p+1}$ — выборка объема $n + p + 1$, генерируемая процессом (4). В [10] показано, что для оценки функции $\Psi$ целесообразна статистика:

$$\Psi_{[n]p}(y) = \sum_{i=1}^{n} \frac{1}{\prod\limits_{k=1}^{p} h_{[i]k}} Y_{i+p+1} K\left(\frac{y_1 - Y_{i+1}}{h_{[i]1}}, ..., \frac{y_p - Y_{i+p}}{h_{[i]p}}\right) \Bigg/ \sum_{i=1}^{n} \frac{1}{\prod\limits_{k=1}^{p} h_{[i]k}} K\left(\frac{y_1 - Y_{i+1}}{h_{[i]1}}, ..., \frac{y_p - Y_{i+p}}{h_{[i]p}}\right).$$

*Пример 2. Фильтрация сигнала с аддитивным шумом.* Пусть наблюдается дискретный сигнал $Y(t)$ с аддитивным шумом. Представим сигнал в виде

$$Y(t) = Y(t-1) + \Delta Y(t),$$

где $\Delta Y(t) = (Y(t) - Y(t-1)) + \xi(t)$ — зашумленное изменение сигнала на интервале $[t-1, t]$) и введем обозначения: $t_i = t(T - n - p + i)$ — моменты времени, в которые проводились измерения сигнала на интервале наблюдения длительностью $T$; $\Delta Y^i = \Delta Y(t(T - n - p + i))$, $i = \overline{1, n+p}$ — выборочные значения изменения сигнала.

Рассматривая изменения сигнала $\Delta Y(t(i))$ как функцию $F(\cdot, \cdot)$ от времени и от значений на предыдущих интервалах времени, т. е. $\Delta Y(t(i)) = F\{t(i), \Delta Y(t(i-1)), ..., \Delta Y(t(i-p))\}$, и применяя непараметрические оценки, полученные в [8—10], будем искать оценку $\hat{Y}(t(T))$ значений сигнала $Y(t(T))$ в момент времени $t(T)$ в виде $\hat{Y}(t(T)) = Y(t(T-1)) + \hat{F}(t(T), \Delta Y^n, ..., \Delta Y^{n+p-1})$ (вид оценки $\hat{F}$ ввиду громоздкости выражения не приводится).

## Подход к решению задачи на основе методов распознавания образов

Информационная (и программная) модель распознавания СДС должна обеспечивать решение следующих задач:

1) формирование обобщенных образов состояний ДС на основе обучающей выборки;

2) идентификация (распознавание) состояния ДС на основе его наблюдаемых (дискретных) значений сигнала;

3) прогнозирование поведения ДС в условиях полного отсутствия управляющих воздействий;

4) решение задач 1)—3) должно обеспечиваться в режиме реального времени на реальных размерностях данных при ограничениях на вычислительные ресурсы ЭВМ.

Алгоритмы выбора информативных признаков, опирающиеся на дискретные методы распознавания образов [4, 5], включают конструирование эталонов на обучающей выборке по априорно указанным (экспертом) характерным участкам временных рядов, именуемых состояниями ДС.

**Алгоритм 1** *распознавания СДС на основе метода эталонов и функции обобщенного сходства* (*ФОС*) [6, 11]. Основу решающего правила составляют эталонные элементы, выбираемые для каждого образа из объектов обучающей выборки по алгоритму выбора эталонов-подпоследовательностей из работы [12]. Для распознавания нового объекта (значения сигнала) определяются расстояния от него до всех эталонов, выбираются ближайшие два эталона из разных классов и по расстояниям до них вычисляются значения функции ФОС [6] (FRiS-функции, следуя работе [11]). Решение принимается в пользу того класса, значения функции ФОС для которого максимально.

**Алгоритм 2** *распознавания СДС на основе выбора эталонов по определенному в* [12] *алгоритму и методу ближайших соседей (БС)* [4, 5].

**Алгоритм 3** *распознавания СДС* является новым и предназначается для обработки возможной ситуации неопределенности, связанной с "пограничностью" состояний $\Omega_i$, $i = \overline{1, I}$. Трехэтапный алгоритм 3 опирается, во-первых, на результаты работы [12] и на метод построения метаобъектов, предложенный в работе [13], и состоит в замене исходного множества близкорасположенных в признаковом пространстве объектов одним метаобъектом; во-вторых, на формализм ФОС объектов с метаобъектами (метаэталонами).

Прежде чем изложить суть алгоритма 4 распознавания СДС на основе информационного подхода, являющегося математической основой многих методов [5—7], введем обозначения. Обозначим $p_v^i$ ($i = \overline{1, I}$, $v \in U$) — вероятность "встретить" сигнал $v$-й градации $v \in U = \{0, 1, ..., u\}$, $N_u = |U|$, на интервале, сопоставленного $i$-му СДС. Совокупность $p^i = \{p_v^i, v \in U\}$, $i = \overline{1, I}$, — условное вероятностное распределение при фиксированном $i$.

**Алгоритм 4** *распознавания СДС* состоит из двух этапов. Итогом первого этапа является совокупность информативных (характеристических) признаков в виде оценок условных распределений квантованных значений сигнала в разных СДС. Итогом второго этапа является принятие решения о принадлежности наблюдаемого сигнала одному из СДС. Кратко его изложим.

1. По результатам наблюдений $Y^* = [Y^*(t), Y^*(t + 1), ..., Y^*(t + L - 1)]$ (назовем эту совокупность значений квантованного сигнала $L$-окном [12, 14]) вычисляем соответствующие относительные частоты $p_v^* = N_v^*/L$, $v \in U$ ($N_v^*$ — число встреч сигнала $v$-й градации).

2. Вычисляем информационное расстояние (следуя [6, 14]) между "эталонными" оценками $p^i$ условных вероятностных распределений случайной величины — числа встреч определенной градации сигнала в каждом СДС и полученными в результате наблюдений $p^* = \{p_v^*, v \in U\}$ по формулам:

$$D(p^i, p^*) = d\left(p^i, \frac{p^i + p^*}{2}\right) + d\left(p^*, \frac{p^i + p^*}{2}\right); \quad (5)$$

$$d\left(p^i, \frac{p^i + p^*}{2}\right) = \frac{1}{\log_2(N_U)} \sum_{v \in U} p_v^i \log_2 \frac{2p_v^i}{p_v^i + p_v^*};$$

$$d\left(p^*, \frac{p^i + p^*}{2}\right) = \frac{1}{\log_2(N_U)} \sum_{v \in U} p_v^* \log_2 \frac{2p_v^i}{p_v^i + p_v^*},$$

где $d(p^i, p^{i*}) = \sum_{v \in U} p_v^i \log_2 \frac{p_v^i}{p_v^{i*}}$ — оценка расстояния Кульбака—Лейблера (меры, характеризующей взаимную информацию двух вероятностных распределений в битах), неотрицательная и равная нулю, если равны сравниваемые распределения; $p^{i*} = (p^i + p^*)/2$ — распределение вероятностей, которое получается усреднением для каждой пары соответствующих вероятностей из исходных $p^*$ и $p^i$; $\frac{1}{\log_2(N_U)}$ — нормирующий коэффициент, обеспечивающий выполнение неравенства $0 \leqslant d(p^i, p^{i*}) \leqslant 1$.

3. Принимаем решение в пользу того СДС, для которого величина $D(p^i, p^*)$ является наименьшей: $i^* = \arg \min_{i = \overline{1, I}} (D(p^i, p^*))$.

*Утверждение 1.* Функция $D(p^i, p^*)$, определяемая по формуле (5), удовлетворяет всем свойствам расстояния.

Доказательство проводится непосредственной проверкой свойств расстояния (метрики).

## Решение прикладной задачи

В последнее время наметилось устойчивое расширение области применения частотно-регулируемых асинхронных приводов в различных промышленных механизмах [15, 16]. Это обусловлено многими факторами, в том числе снижением потребления энергии при внедрении таких электроприводов. В теории обобщенного электромеханического преобразователя получены уравнения динамики асинхронных двигателей (АД) [16]. Однако их использование для построения динамических наблюдателей состояния и отработки новых алгоритмов управления (робастное осторожное управление [2, 3], синерге-

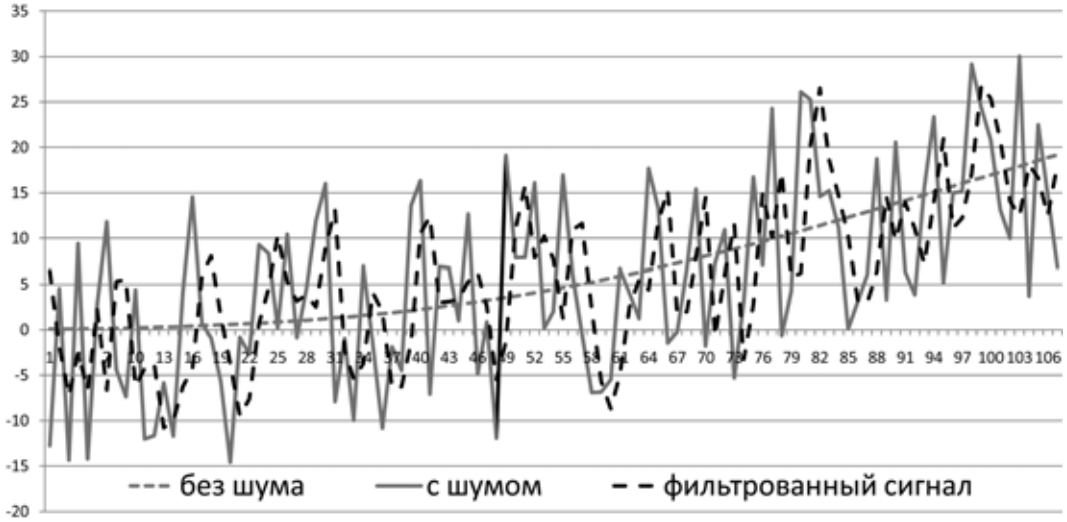



**Рис. 2. Результат фильтрации модельного сигнала; по оси *OX* — дискретные отсчеты времени, *OY* — уровень сигнала**

тическое управление [1] и др.) наталкивается на ряд трудностей, связанных с недостаточной разработанностью методов идентификации систем нелинейных дифференциальных уравнений.

Численная иллюстрация предложенных процедур и алгоритмов будет проводиться на примере модели АД, разработанной в среде MatLab, для которой по выходному сигналу модели было выделено (экспертом) четыре класса-состояния различной длительности.

### Результаты моделирования на ЭВМ

Экспериментальное исследование качества и времени распознавания СДС проводилось в целях оценивания степени влияния фильтрации и других различных факторов (вид и уровень шумов, число эталонов в состояниях, шаг квантования, тип используемой метрики).

- Рассмотрим сначала результаты численного моделирования фильтрации и прогнозирования измеряемых сигналов тока и напряжения АД по вышеизложенной методике. На сигналы модели (восемь моделей) были наложены различные шумы уровня, не превышающего 30 % от максимального значения полезного сигнала: равномерный шум на интервалах [−5; 5], [−10; 10], для сигнала тока и [−150; 150], [−300; 300] для сигнала напряжения соответственно; гауссовский шум с параметрами $N(0, 10)$, $N(0, 50)$ для сигнала тока и $N(0, 10\,000)$, $N(0, 50\,000)$ для сигнала напряжения соответственно. По полученным данным (временным рядам) были построены прогностические и отфильтрованные значения по 1000 точкам. При этом использовались 18 непараметрических моделей с различными порядками авторегрессии и размерами скользящего выборочного окна, а именно, исследовались модели AR со значениями параметра $p = 1, 2, 3$. Для каждой из них были рассмотрены варианты с размерами скользящего выборочного окна $N$, равными 5, 10, 15, 20, 25, 30. В качестве показателя уровня шума использована относительная погрешность $\gamma$, вычисленная по следующей формуле:

$$\gamma = \frac{1}{1000} \sum_{i=1}^{1000} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \cdot 100\ \%,$$

где $\hat{y}_i$ — значение сигнала, уровень шума в котором измеряется; $y_i$ — истинное значение сигнала.

В ходе анализа полученных реализаций фильтрации и прогноза было выявлено, что наилучшие результаты наблюдаются при величине скользящего выборочного окна, равной 20, и порядке авторегрессии $p = 1$; уровень шума в фильтрованном сигнале приращения измеряемых величин уменьшается на 40...60 %, и на 10...35 % в сигналах тока и напряжения соответственно; уровень шума в прогнозных значениях сигналов тока и напряжения на 10...20 % ниже, чем в значениях сигналов с шумами. Отметим, что с большим уровнем шумов сигнал фильтруется слабо (рис. 2).

- Покажем теперь, что процедура предварительной фильтрации для распознавания СДС не является критичным звеном в распознавании сигналов (в смысле времени и эффективности распознавания (доли правильно распознанных объектов)). Исследование проводилось в условиях разных метрик (Евклида (ЕМ), Хемминга (ХМ), информационной обобщенной метрики (ИнфОМ)) (формула (5)) и при различных типах и уровнях шумов.

В табл. 1, 2 даны показатели эффективности и времени распознавания при стандартном (СК) и адаптивном выборах шага квантования (АК) и использовании процедуры распознавания на основе

Таблица 1

**Эффективность распознавания при стандартном и адаптивном выборах шага квантования по трем метрикам**

| Тип сигнала для распознавания | Эффективность распознавания СК, % | | | Эффективность распознавания АК, % | | |
|---|---|---|---|---|---|---|
| | ИнфОМ | ЕМ | ХМ | ИнфОМ | ЕМ | ХМ |
| Без шума | 86,48<br>99,68 | 84,73<br>97,78 | 86,71<br>99,71 | 78,81<br>99,49 | 78,00<br>99,46 | 65,38<br>96,92 |
| С шумом 30 % | 76,19<br>95,49 | 76,05<br>96,52 | 76,4<br>95,27 | 73,62<br>81,91 | 80,08<br>81,91 | 77,05<br>80,86 |
| С фильтрацией | 82,67<br>94,49 | 71,81<br>95,13 | 81,47<br>94,24 | 84,98<br>81,6 | 82,93<br>82,93 | 79,83<br>86,49 |

Таблица 2

**Время распознавания при стандартном и адаптивном выборах шага квантования по трем метрикам**

| Диапазон числа шагов квантования | Диапазон времени распознавания СК | | | Диапазон времени распознавания АК | | |
|---|---|---|---|---|---|---|
| | ИнфОМ | ЕМ | ХМ | ИнфОМ | ЕМ | ХМ |
| 10—50 (Б/ф) (Ф) | 0,167<br>0,795 | 0,050<br>0,199 | 0,026<br>0,109 | 0,026<br>0,046 | 0,013<br>0,016 | 0,008<br>0,008 |
| | 1,267<br>6,793 | 0,510<br>1,999 | 0,321<br>1,121 | 0,225<br>0,541 | 0,123<br>0,114 | 0,090<br>0,091 |
| 60—100 (Б/ф) (Ф) | 0,951<br>1,583 | 0,951<br>1,583 | 0,124<br>0,209 | 0,026<br>0,047 | 0,013<br>0,017 | 0,008<br>0,009 |
| | 8,951<br>14,583 | 9,957<br>15,581 | 1,144<br>2,249 | 0,423<br>0,749 | 0,216<br>0,245 | 0,098<br>0,099 |
| Примечание: (Б/ф) — без фильтрации; (Ф) — с фильтрацией. | | | | | | |

трех указанных метрик. В 1-й строке (для каждого типа сигнала) указан худший результат (в процентах), во 2-й строке — лучший, параметр квантования $h = 0{,}05$, число уровней квантования указано для минимального значения амплитуды сигнала, время распознавания указано для 1-го объекта (измерения) в микросекундах без учета времени на обучение и фильтрацию.

### Выводы по результатам моделирования на ЭВМ

Сравнение точности и времени распознавания с предварительной фильтрацией исходных данных и без фильтрации (на данных модели ЭМС) показало:

- время распознавания с фильтрацией больше на один—два порядка;
- точность распознавания при использовании стандартной процедуры квантования для данных без фильтрации выше, чем точность распознавания для данных с фильтрацией, примерно на 1 %;
- точность распознавания для адаптивной процедуры квантования при использовании фильтрации в среднем выше на 3...5 %, чем без фильтрации. При параметре квантования, равном $h = 0{,}01$, точность распознавания отфильтрованного сигнала на 10 % выше, чем сигнала с шумом;
- для сигнала с шумом и отфильтрованного сигнала чуть более точно (примерно на 1 %) работает распознавание с использованием метрики Евклида по сравнению с использованием ИнфОМ и Хемминга метрик. При адаптивной процедуре квантования для отфильтрованного сигнала с метриками ИнфОМ и Хемминга точность распознавания выше на 2...4 %, чем для метрики Евклида.

Среди различных алгоритмов распознавания на основе эталонов, подробно изложенных и экспериментально исследованных в [12, 14] для решения обсуждаемой в этой работе задачи, лучше всего показывают себя алгоритм с использованием ФОС и алгоритм по методу БС. При этом алгоритм с использованием ФОС лучше работает для сигнала без шума, а алгоритм по методу БС — для сигнала с шумом и отфильтрованного сигнала.

Исследование по разным уровням и типам распределений шумов показало, что для равномерного (10 %-го шума) увеличение числа градаций способствует повышению точности распознавания до некоторого порога (небольшого, порядка 30 градаций), начиная с которого точность распознавания существенно падает (на ~20 %). Возможно, это связано с переобучением алгоритма распознавания (при увеличении числа градаций алгоритм "обучается на шуме"), так как замечено, что при малом уровне шума этот эффект отсутствует.

Исследование по разным шести видам моделей (нелинейных и нестационарных) показало, что адаптивное квантование, сохраняя качество распознавания, приводит к существенно меньшему времени распознавания по сравнению со стандартным квантованием (см. табл. 2). Процедура фильтрации в тех ситуациях, где распознавание состояния ДС в реальном режиме времени является самостоятельной задачей, не несет основополагающей нагрузки, хотя на ряде моделей на фоне уменьшения шума (см. рис. 2) алгоритмы распознавания показывают лучшие результаты (1...5 %).

### Заключение

В статье рассмотрены два подхода к распознаванию состояний ДС по наблюдаемым характеристикам в целях возможного использования этого знания для осуществления мониторинга ДС. Подходы основаны на методах распознавания образов и теории информации. Возможность распознавания состояний ДС за приемлемое время проде-

монстрирована на примере распознавания состояний различных ЭМС.

Процедура фильтрации нелинейных и нестационарных процессов [8...10] требует (при нахождении оптимальных параметров) проводить многомерную оптимизацию (в смысле критерия полного скользящего контроля), на которую (несмотря на введенные в [10] оптимизирующие преобразования, упрощающие вычисления) затрачиваются большие вычислительные и временные ресурсы. При этом необходимо знание ядерных функций (выбор которых связан с определенным субъективизмом). Показано, что в тех задачах, где распознавание состояния ДС в реальном режиме времени является самостоятельной целью, процедура фильтрации не несет основополагающей нагрузки. Предложенные алгоритмы могут быть использованы при организации робастного и адаптивного управления сложными электромеханическими системами [1...3].

*Авторы благодарят В. Г. Букреева, профессора Томского политехнического университета, за постановку задачи и полезные обсуждения.*

**Список литературы**

1. **Тюкин И. Ю., Терехов В. А.** Адаптация в нелинейных динамических системах. Санкт-Петербург: ЛКИ, 2008. 384 с.

2. **Глухов В. В.** Техническое диагностирование динамических систем. М.: Транспорт, 2000. 96 с.

3. **Поляк Б. Т., Щербаков П. С.** Робастная устойчивость и управление. М.: Наука, 2002. 303 с.

4. **Журавлев Ю. И., Гуревич И. Б**. Распознавание образов и анализ изображений // Искусственный интеллект. В 3-х кн. Кн 2. Модели и методы: Справочник / Под ред. Д. А. Поспелова. М.: Радио и связь, 1990. С. 149—190.

5. **Ту Дж., Гонсалес Р.** Принципы распознавания образов: Пер. с англ. / Под ред. Ю. И. Журавлева. М.: Мир, 1978. 411 с.

6. **Клир Дж.** Системология. Автоматизация решения системных задач. М.: Радио и связь, 1990. 538 с.

7. **Кульбак С.** Теория информации и статистика. М.: Наука, 1967. 408 с.

8. **Васильев В. А., Добровидов А. В., Кошкин Г. М.** Непараметрическое оценивание функционалов от распределений стационарных последовательностей. М.: Наука, 2004. 508 с.

9. **Györfi L., Kohler M. and Walk H.** Week and strong universal consistency of semirecursive kernel and partitioning regression estimates // Statist. Decisions. 1998. V. 16. P. 1—18.

10. **Кошкин Г. М., Лаходынов В. С.** Полурекуррентная непараметрическая идентификация условных функционалов слабозависимых последовательностей // Вестник ТГУ "Управление, вычислительная техника и информатика". 2008. № 1(2) С. 57—68.

11. **Zagoruiko N. G., Borisova I. A., Dyubanov V. V. and Kutnenko O. A.** Methods of Recognition Based on the Function of Rival Similarity // Pattern Recognition and Image Analysis. 2008. Vol. 18. N 1. P. 1—6.

12. **Колесникова С. И., Букреев В. Г.** Распознавание состояний динамической системы // Сб. докладов XI Междунар. научно-техн. конф. DSPA—2009. 2009. С. 619—622.

13. **Волченко Е. В.** Модифицированный метод потенциальных функций // Бионика интеллекта. 2006. № 1(64). С. 86—92.

14. **Колесникова С. И., Букреев В. Г.** Информационный подход к распознаванию состояний динамической системы // Сб. докладов X Междунар. научно-техн. конф. С&Т—2009, Воронеж: НПФ "Саквоее" ООО. 2009. Т. 1. С. 54—64.

15. **Изосимов Д. Б., Рыбкин С. Е.** Идентификация частоты вращения и составляющих вектора потокосцепления ротора асинхронного двигателя по измерениям токов и напряжений обмоток статора // Электричество. 2005. № 4. С. 32—40.

16. **Копылов И. П.** Математическое моделирование электрических машин. М.: Высш. шк. 2001. 327 с.

---

УДК 62.501.22:519.6

**Э. И. Владимирский,** канд. техн. наук, ст. науч. сотр., e-mail: eduard.vladimirsky@hotmail.com,
**Ф. К. Тагиев,** канд. техн. наук, доц.,
Азербайджанская государственная нефтяная академия, г. Баку

# Синергетический подход к формированию интегральных размерностей в интеллектуальных информационно-измерительных системах

*На основе принципов синергетики, нелинейной динамики и современных информационных технологий (IT-технологий) предлагается рассмотрение интегральных размерностей, характеризующих гетерогенные потоки информации в интеллектуальных измерительных системах.*

***Ключевые слова:*** *синергетика, нечетко-фрактальная размерность, вейвлет-анализ, размерность Реньи, размерность Ляпунова, фрактальность отношения сигнал/помеха*

## Введение

В теории гладких динамических систем существует направление, связанное с исследованием размерности инвариантных множеств. Речь идет не о классическом топологическом понятии размерности, а о различных модификациях так называемой размерности Каратеодори [1]. Она имеет не чисто топологический, а топологометрический характер.

Грубое описание топологического строения этих множеств состоит в следующем: локально они представляют собой произведение гладкого подмногообразия на множество канторовского типа. Таким образом, размерность (или, как обычно

в этих случаях говорят, фрактальная размерность) трактуется как количественная характеристика сложности топологического строения аттракторов в трансверсальных к нему направлениях [1].

Вместе с тем, были обнаружены некоторые соотношения между размерностью, метрической энтропией динамической системы и показателями Ляпунова [1]. Эти соотношения отражают глубокую внутреннюю связь между топологией аттрактора и свойствами действующей на нем динамики.

Общая трактовка размерностных характеристик состоит в следующем. Пусть $X$ — полное сепарабельное метрическое пространство и $m(\alpha, Z)$ — семейство $G$ — полуаддитивных внешних мер на $X$, зависящих от вещественного параметра $\alpha$. Допустим, что для всякого $Z \subset X$ существует такое критическое значение $\alpha_0$ параметра $\alpha$, что $m(\alpha, Z) = 0$ при $\alpha > \alpha_0$ и $m(\alpha, Z) = \infty$ при $\alpha < \alpha_0$ (при этом $m(\alpha, Z)$ может быть любым числом в интервале $[0, \infty]$.

Число $\alpha_0$ называется размерностной характеристикой множества $Z$ [1].

В [2] отмечено, что развитие синергетики и нелинейной динамики актуализировало новые возможности для метрологии и теории измерений. Отмечено также, что изменение количественных характеристик масштабно-инвариантных или близких к ним объектов следует проводить на новой методологической основе, связанной с идеями синергетики, в частности, теории фракталов.

В связи с этим необходимость получения качественно нового информационного продукта обусловлено такими характерными особенностями данных, как гетерогенность, нестационарность, темпоральность, хаотичность, неопределенность (нечеткость) и т. д. Кроме того, обработка взаимодействующих потоков информации может осуществляться в метрических и неметрических пространствах, что усложняет процесс анализа и принятия решений.

Отсюда, используя принципы синергетики, нелинейной динамики и современные информационные технологии (IT-технологии), предлагается рассмотрение интегральных размерностей, характеризующих потоки информации в интеллектуальных измерительных системах (ИИС).

## Нечетко-фрактальная размерность (Fuzzy fractal dimensions)

Пусть $X$ — произвольное множество, $d_H$ — фрактальная размерность, $\mu$ — функция принадлежности нечеткого подмножества $A$ множества $X$, $A \subset X$, $\mu_A(x) : X \to [0, 1]$.

Пусть $d_{H\mu}$ — нечетко-фрактальная мера. Пара $(X, d_{H\mu})$ — компактное нечетко-фрактальное метрическое пространство.

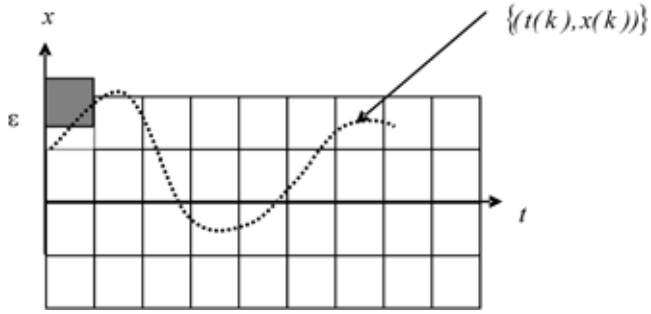



**Рис. 1. Пример прямого вычисления детерминированной фрактальной размерности**

Перейдем непосредственно к рассмотрению нечетко-фрактальной размерности.

***Реализация алгоритма 1.***

1. Представим временной ряд в виде $\{t(k), x(k)\}$, $k = \overline{1, N}$, где $x(k) \in R$, и $t(1), t(2), \ldots, t(N)$ — дискретные значения временных моментов.

Пусть исследуемый временной ряд, элементы которого представляют сумму каких-либо событий в единицу времени, дискретизирован на регулярной сетке, ячейки которой называют "боксами", размерностью $\varepsilon$ (пространство $R^1$) (рис. 1). Проблема возникает, когда размерность системы рассматривается в $R^n$.

2. Фрактальная размерность временного ряда $d_H$ определяется как

$$d_H = \lim_{\varepsilon \to 0} \log N(\varepsilon)/\log(\varepsilon^{-1}), \qquad (1)$$

где $N(\varepsilon)$ принимается как минимальное число шаров радиуса $\varepsilon$, необходимых для покрытия компактного множества [3].

3. Получаем гранулированное множество, состоящее из гиперсфер радиуса $\varepsilon$.

4. Определяем $N(\varepsilon)$ для всех сфер

$$N(\varepsilon) = \frac{1}{N(N-1)} \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} \theta_{ij}(\varepsilon), \qquad (2)$$

где $\theta_{ij}$ — сфера, определяемая как [4]

$$\theta_{ij} = \begin{cases} 1, \text{ если } \sqrt{(t(i)-t(j))^2 + (x(i)-x(j))^2} \leqslant \varepsilon; \\ 0 \text{ в противном случае.} \end{cases} \qquad (3)$$

5. Конструируем $\theta_{ij}$ вокруг точки $(t(i), x(i))$ (рис. 2).

В общем случае задача о покрытии множества семейством его подмножеств является нечеткой [5]. Тогда выражение для $N(\varepsilon)$ может быть записано в следующем виде [4]:

$$N(\varepsilon) = \frac{1}{N(N-1)} \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} \theta_{ij}\mu(\varepsilon). \qquad (4)$$

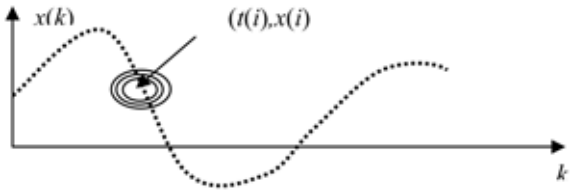



**Рис. 2. Конструкция сферы $\theta_{ij}$ [4]**

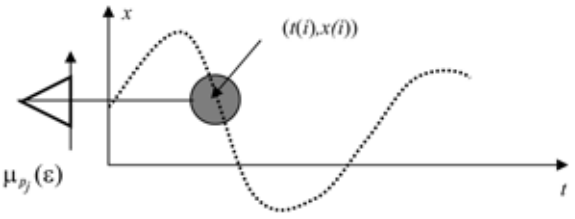



**Рис. 3. Коррелированное нечетко-фрактальное множество**

Таблица 1

**Наиболее употребляемые функции принадлежности**

| Классы *F*-множеств | Функции принадлежности |
|---|---|
| Триангулярное | $\theta(x) = \begin{cases} 1 - \dfrac{x-m}{\varepsilon^2}, \text{ если } \lvert x-m\rvert \leqslant \varepsilon; \\ 0 \text{ в противном случае} \end{cases}$ |
| Параболическое | $\theta(x) = \begin{cases} 1 - \dfrac{(x-m)^2}{\varepsilon^2}, \text{ если } \lvert x-m\rvert \leqslant \varepsilon; \\ 0 \text{ в противном случае} \end{cases}$ |
| Гауссово | $\theta(x) = \exp\left(-\dfrac{(x-m)^2}{\varepsilon^2}\right)$ |

Таблица 2

**Оценка нечетко-фрактальных размерностей в зависимости от нечетких функций [4]**

| Классы нечетких множеств | Фрактальная размерность $d_{H_\mu}$ |
|---|---|
| Множество | 1,3546 |
| Параболическое | 1,3897 |
| Гауссово | 1,3800 |

Графически это отражено на рис. 3, где $\mu_{p_j}(\varepsilon)$ — гауссова функция принадлежности, имеющая вид

$$\mu_{p_j}(\varepsilon) = \exp(-x(i) - x(j))^2/\varepsilon^2. \quad (5)$$

Подставим выражение (4) в (1), получим коррелированное соотношение, характеризующее нечетко-фрактальную размерность $d_{H_\mu}$.

В целях практического использования представлены табл. 1, 2.

### Фрактальная размерность минимального покрытия

В [6] на основе минимальных покрытий вводятся новые фрактальные характеристики: размерность минимального покрытия $D_\chi$ и индекс вариации $\chi$, тесно связанный с $D_\chi$. Использование этих показателей расширяет сферу применимости в различных информационных процессах.

Необходимо заметить, что если исходное множество погружено в евклидово пространство, то вместо покрытия этого множества шарами, можно использовать любые другие аппроксимации простыми фигурами (например, клетками) с геометрическим фактором (линейным размером) $\delta$. При этом наряду с исходной сферической размерностью $D$ имеют место новые фрактальные размерности (клеточная, внутренняя и т. д.), которые как предельные значения при $\delta \to 0$ обычно совпадают с $D$.

Пусть задан график вещественной непрерывной функции $y = f(t)$, определенной на некотором отрезке $[a, b]$. Равномерное разбиение отрезка имеет вид

$$\omega_m = [a = t_0 < t_1 < \ldots < t_m = b], \quad \delta = (b - a)/m. \quad (6)$$

Минимальное покрытие функции в классе покрытий, состоящих из прямоугольников, с основанием $\delta$ показано на рис. 4 [6].

Тогда высота прямоугольника на отрезке $[t_{i-1}, t_i]$ будет равна амплитуде $A_i(\delta)$, которая является разностью между максимальным и минимальным значением функции на этом отрезке.

Величина

$$V_f(\delta) = \sum_{i=1}^{m} A_i(\delta) \quad (7)$$

будет называться вариацией функции $f(t)$, соответствующей масштабу разбиения $\delta$ на отрезке $[a, b]$. Полная площадь минимального покрытия $S_\chi(\delta)$ записывается в виде

$$S_\chi(\delta) = V_f(\delta)\delta. \quad (8)$$

На основании размерности Хаусдорфа $D$ следует, что

$$V_f(\delta) \sim \delta^{-\chi} \text{ при } \delta \to 0, \quad (9)$$

где

$$\chi = D_\chi - 1. \quad (10)$$

Показатель $\chi$ называется индексом вариации, а размерность $D_\chi$ — размерностью минимального покрытия [6].

Размерность $D_\chi$ можно использовать при анализе вещественных функций с характерными особенностями (всплески, различные флуктуации, разрывы).

### Фрактальная размерность и вейвлет-анализ

Вейвлетный (*wavelet*) анализ является незаменимым средством для выявления "тонкостей"

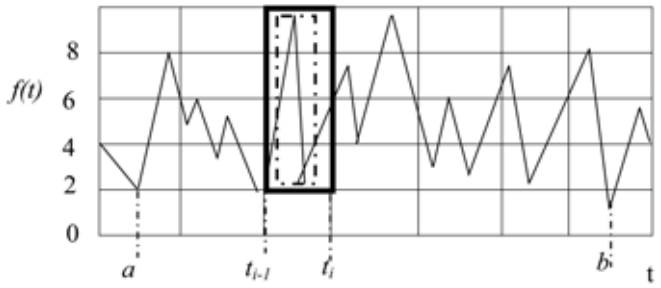



**Рис. 4. Минимальное (штрих-пунктирный прямоугольник) и клеточное (сплошной прямоугольник) покрытия функции $f(t)$ на интервале $[t_{i-1}, t_i]$ длиной $\delta$**

фрактальных (мультифрактальных) структур, а также нарушений регулярности временных рядов.

Однако эти структуры не могут "обитать" в гильбертовом пространстве, поскольку оно приспособлено для "жизни" хороших, непрерывных функций [7].

Но известно, что фрактальная размерность напрямую связана с показателями Липшица—Гельдера [8]. В свою очередь, точечные показатели Гельдера определяются с помощью вейвлетов. Здесь по масштабному поведению вейвлет-коэффициентов удается получить некоторые сведения о свойстве однородной регулярности функции. Так, линейная зависимость логарифма вейвлет-коэффициентов от масштаба $j$ указывала бы на проявление масштабно инвариантных свойств сигнала, т. е. на фрактальное поведение [8].

В связи с этим имеет место следующая постановка задачи.

**Постановка задачи.** Пусть $f(x) \in L^2(R)$ и вейвлет-преобразование имеет вид

$$f(x) = \sum_{j,k} \underline{\theta}_{j,k} \Psi_{j,k}(x), \qquad (11)$$

а коэффициенты вейвлет-разложения

$$\underline{\theta}_{j,k} = \int f(x) \Psi_{j,k}(x) dx, \qquad (12)$$

где $\Psi_{j,k}$ — базовый вейвлет; $j$, $k$ — целые числа $\{j = 0, 1, ..., k = 0, ..., 2^{j-1}\}$.

На основании свойств функциональных пространств Гальдера и теоремы Джаффару [9] требование определенной однородной регулярности функции $f(x)$ на всей вещественной оси при положительных нецелочисленных значениях $\alpha$ выражается в терминах вейвлет-коэффициентов в виде неравенства [8, 9]

$$|\underline{\theta}_{j,k}| \leqslant C 2^{-j(1/2+\alpha)}, \qquad (13)$$

где $C$ — положительная константа; $\alpha$ — показатель (или экспонента) Липшица—Гельдера, характеризующий иррегулярность функции, $\alpha_{\min} < \alpha < \alpha_{\max}$. Задача состоит в том, чтобы показать определение фрактальной размерности с использованием коэффициентов вейвлет-преобразования.

В [10] показана возможность такой реализации в виде

$$\Delta(f) = \frac{3}{2} + \lim_{j \to \infty} \frac{\log_2(\Sigma_k |\underline{\theta}_{j,k}|)}{j}, \qquad (14)$$

где $\Delta(f)$ — фрактальная размерность, варьируемая в зависимости от коэффициентов вейвлет-разложения $\underline{\theta}_{j,k}$.

Таким образом, выражение (14) реализовано в обобщенном компактном метрическом пространстве $\Re$, $\Delta(f)$ с фрактальной мерой $\Delta(f)$, образованном вложением фрактального метрического пространства $(X, d_H)$ в пространство Гильберта $L^2$, т. е.

$$(X, d_H) \subseteq L^2 \in (\Re, \Delta(f)), \qquad (15)$$

где $\Re$ — обобщенное компактное фрактальное метрическое пространство.

### Обобщенные фрактальные размерности

Грассбергер, Хэнтчел и Прокачиа ввели в нелинейной динамике семейство обобщенных размерностей $D_q$, называемых размерностями Реньи [11, 16].

Пусть построено покрытие аттрактора ячейками размера $\varepsilon$ и вероятность пребывания в $i$-й ячейке покрытия, т. е. соответственная инвариантная мера ячейки, есть $p_i$.

Тогда для любого действительного числа $q$, $-\infty < q < \infty$, размерность $D_q$ определяется как предел

$$D_q = \frac{1}{q-1} \lim_{\varepsilon \to \infty} \frac{\log \sum_{i=1}^{N(\varepsilon)} p_i^q}{\log \varepsilon}. \qquad (16)$$

Из выражения (16) видно, что при $q = 0$ оно сводится к определению емкости, при $q = 2$ — корреляционной размерности. При $q = 1$ возникает неопределенность типа 0/0, но ее раскрытие по правилу Лопиталя приводит к определению информационной размерности.

Таким образом, емкость, корреляционная и информационная размерности выступают как представители семейства Реньи [16].

Однако для описания гетерогенных фракталов имеет место характеристика, называемая $\alpha$-спектром. Тогда выражение для $D_q$ имеет вид

$$D_q = \frac{1}{q-1} [q\alpha(q) - f(\alpha(q))]. \qquad (17)$$

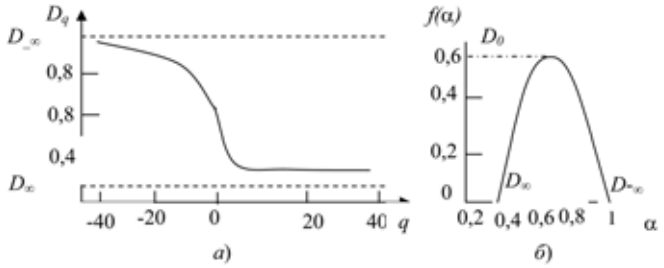



**Рис. 5. Зависимость обобщенных размерностей $D_q$ от $q$ для двухмасштабного канторова множества (*а*); α-спектр для этого множества (*б*)**

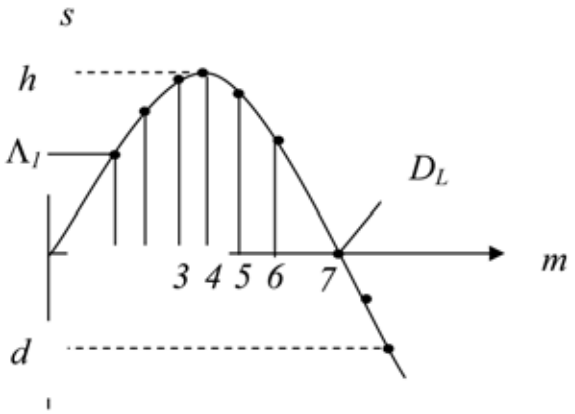



**Рис. 6. Зависимость суммы ляпуновских показателей $S_m = \sum_{i=1}^{m} \Lambda_i$ от числа слагаемых**

Зная зависимость $\alpha(q)$, можно вычислить все обобщенные размерности. И наоборот, зная все обобщенные размерности, можно найти α-спектр:

$$\alpha(q) = \frac{d}{dq}[(q-1)D_q]. \qquad (18)$$

На рис. 5 представлены зависимость величины $D_q$ от $q$ и α-спектр для двухмасштабного канторова множества.

Обобщенные размерности и α-спектр дают *геометрическое описание странных аттракторов*. Однако возникает необходимость охарактеризовать динамические свойства системы (процесса). Известно, что существует взаимосвязь между спектром ляпуновских показателей, характеризующих динамические свойства системы (показателей аттрактора), и его фрактальной размерностью [15, 16].

### Ляпуновская размерность

В [15, 16] предложено выражение, характеризующее взаимосвязь между спектром ляпуновских показателей аттрактора и его фрактальной размерностью:

$$D_L = m + \frac{\sum_{i=1}^{m} \Lambda_i}{|\Lambda_{m+1}|}, \qquad (19)$$

где $\Lambda_i$ — ляпуновские показатели; $m$ — размер фазового пространства; $D_L$ — ляпуновская размерность Каплана—Йорке.

Здесь, если аттрактор странный хаотический, то старший показатель положителен: $\Lambda_1 > 0$. Вместе с тем, сумма всех ляпуновских показателей отрицательна:

$$S_N = \sum_{i=1}^{N} \Lambda_i < 0.$$

Необходимо отметить, что $D_L$ в выражении (19) называют *новой фрактальной размерностью*. На рис. 6 показан спектр ляпуновских показателей.

В частности, $S_1$ есть старший ляпуновский показатель $\Lambda_1$, $\max S_m$ отвечает КС (Колмогоров—Синай) энтропии $h$. Величина $S_N$ — отрицательная, характеризующая сжатие фазового объема при конденсации облака изображающих точек на аттрактор.

Точка пересечения графика (рис. 6), образованного отрезками прямых с осью абсцисс дает ляпуновскую размерность по формуле (19).

Таким образом, имеет место выражение, описывающее как геометрическое, так и хаотическое состояния аттрактора, что демонстрирует интегральную составляющую новой фрактальной (ляпуновской) размерности $D_{KY} = D_L$ в $N$-мерном фазовом пространстве.

Однако численное определение эволюции $D_{KY}$ представляется проблематичной. Поэтому в [17, 18] показана полиномиальная интерполяция (на примере параболы) $D_{KY}$, в результате которой имеет место выражение вида

$$D_{KY} = \{l_2 + 3l_3 + \\ + [9l_2^2 + 6l_2l_3 - 8l_1l_3 + 8l_1l_2 + l_3^2]^{1/2}\}/2(l_3 - l_2), \qquad (20)$$

где при $l_2 = 0$ формула (20) принимает вид

$$D_{KY} = 1{,}5 + 0{,}5[1 - 8l_1/l_3]^{1/2}. \qquad (21)$$

Важно также отметить, что в долгосрочном прогнозе используется выражение из [18]:

$$D_{KY} = 2 - \lambda_1/\lambda_3, \qquad (22)$$

где $\lambda_1$ и $\lambda_3$ — старший и младший показатели экспонент; $D_{KY}$ — фрактальная размерность Каплана—Йорке.

Необходимо заметить, что в комплексном фазовом пространстве, характеризующем поведение среды, выделяют состояния, которые притягива-

ют к себе возможные траектории эволюции информационных систем.

Эти состояния определяются как информационные аттракторы, по аналогии с понятием странный аттрактор в теории динамического хаоса.

Таким образом, информационный аттрактор выступает в роли структуры, которая в направлениях, характеризуемых положительными показателями Ляпунова, создает разнообразие, увеличивает погрешности эволюции информационной технологии, т. е. порождает порядок, и, наоборот, уменьшает погрешности ее развития в направлениях, характеризуемых отрицательными показателями.

### Фрактальность отношения сигнал/помеха

Представляется интересным оценка соотношения сигнал/помеха в виде отношения фрактальных (нечетко-фрактальных) размерностей в контексте оценки динамической информации.

В случае наличия в знаменателе отношения сигнал/помеха хаотической составляющей ее фрактальная размерность определяется выражением (19), т. е.

$$D_{\text{сигнала}}/D_{ky} = \xi. \qquad (23)$$

Важно отметить, что динамика информационных процессов вызывает вариацию фрактальных размерностей. Тогда для более "тонкого" анализа распределения фрактальных размерностей можно использовать вейвлет-преобразование.

### Заключение

Характерная особенность таких данных, как гетерогенность, нестационарность, хаотичность, неопределенность (нечеткость) и т. д., требует адекватной формализации, которая невозможна без введения новых качественных и количественных методов измерений в ИИС. Рассмотренные в работе интегральные размерности, идеологической основой конструирования которых является синергетический подход и IT-технологии, позволяют получать репрезентативную информацию о системе (процессе) в контексте принятия корректного решения.

#### Список литературы


1. **Песин Я. Б.** Характеристики размерностного типа для инвариантных множеств динамических систем // УФН. 1988. Т. 43. Вып. 4(262). С. 95—128.
2. **Ахромеева Т. С., Малинецкий Г. Г.** Синергетика и проблемы измерений // Измерительная техника. 2008. № 11. С. 9—13.
3. **Кроновер Р. М.** Фракталы и хаос в динамических системах. М.: Техносфера, 2006. 488 с.
4. **Pedrycz W., Bargiela A.** Fuzzy fractal dimensions and fuzzy modeling // Information Sciences. 2003. **153**. P. 199—216.
5. **Толмачев С. А., Григоренко М. Ю.** Один подход к решению задачи о нечетком покрытии // Кибернетика и системный анализ. 1992. № 6. С. 164—165.
6. **Дубовиков М. М., Крянев А. В., Старченко Н. В.** Размерность минимального покрытия и локальный анализ фрактальных временных рядов // Вестник РУЭН. Сер. Прикладная и компьютерная математика.2004. Т. 3 № 1. С. 30—44.
7. **Пригожин И., Стенгерс И.** Время. Хаос. Квант. К решению парадокса времени. М.: Комкнига. 2005. 232 с.
8. **Дремин И. М., Иванов А. В., Нечитайло В. А.** Вейвлеты и их использование // УФН. 2001. Т. 171. № 5. С. 465—501.
9. **Добеши И.** Десять лекций по вейвлетам. М.-Ижевск: НИЦ "Регулярная и хаотическая динамика", 2004. 464 с.
10. **Wanq Y.** Fractal function estimation via Wavelet Shrinkage // Journal of Royal Statistical Society. 1997. Vol. 59. P. 603—613.
11. **Малинецкий Г. Г., Потапов А. Б.** Современные проблемы нелинейной динамики. М.: Едиториал УРСС, 2002. 360 с.
12. **Тагиев Р. А., Владимирский Э. И.** Разработка новых методов определения прямых индикационных признаков при дистанционном зондировании природных объектов // Изв. высших техн. учебн. заведений Азербайджана. Баку. 2002. № 6. С. 61—64.
13. **Павлов А. Н., Анищенко В. С.** Мультифрактальный анализ сложных сигналов // УФН. 2007. Т. 177. № 8. С. 859—876.
14. **Муратов И. Х., Владимирский Э. И., Агаев Ф. У.** Нечетко-фрактальная мера + вейвлет-преобразование в контексте анализа, реконструкции и принятия решений по сигналам в открытой информационной системе // Тр. IX Междун. научно-техн. конф. "Кибернетика и высокие технологии XXI века". 13—15 мая 2008, Воронеж. 2008. Т. 1. С. 277—287.
15. **Потапов А. А.** Фракталы в радиофизике и радиолокации: Топология выборки. М.: Университетская книга, 2005. 848 с.
16. **Кузнецов С. П.** Динамический хаос. М.: Изд. физ.-мат. лит. 2006. 356 с.
17. **Sprott J. C.** Time-Series Analysis. Oxford University Press. 2003. 528 p.
18. **Chloverakis K. E., Sprott J. C.** A comparizon of correlation and Lyapunov dimensions. Physica D 200. 2005. P. 156—164.


## ПАМЯТИ КОЛЛЕГИ И ТОВАРИЩА

### Нариньяни Александр Семенович
### (2.11.1937—26.04.2010)

Редколлегия журнала "Информационные технологии" понесла тяжелую утрату... На 73-м году ушел из жизни известный ученый, блестящий преподаватель, талантливый организатор и замечательный человек — **Александр Семенович Нариньяни.**

Большую утрату понесла отечественная наука в лице бывшего директора Российского НИИ искусственного интеллекта, академика РАЕН, вице-президента консорциума "Российские Речевые Технологии", первого заведующего лабораторией искусственного интеллекта ВЦ СО АН СССР, одного из основателей отечественной компьютерной лингвистики и "недоопределенной" математики.

**Светлая память об Александре Семеновиче навсегда останется в наших сердцах.**

*Редколлегия и редакция журнала "Информационные технологии".*

УДК 681.5

**В. Д. Чертовской,** д-р техн. наук, проф.,
Санкт-Петербургский государственный
электротехнический университет "ЛЭТИ",
e-mail: vdchertows@mail.ru

# Анализ процесса согласованного автоматизированного планирования в иерархической системе управления производством

*Отмечена специфика внешней рыночной среды, в которой работает производство. Процесс планирования в системе управления производством становится относительно самостоятельным, а структура — иерархической. Рассмотрен метод математического описания иерархического планирования с согласованием экономических интересов, что хорошо согласуется с процедурным представлением. Теоретические положения подтверждены результатами прикладной компьютерной реализации.*

***Ключевые слова:*** *управление производством, иерархическая система, анализ, планирование, экономический интерес*

## Введение

Современные производства работают в рыночной среде с динамичными параметрами (цена, ресурсное обеспечение, спрос) при изменении вектора цели. Для отработки таких изменений необходима автоматизированная система управления, работающая в оптимальном режиме. Обобщенная модель такой системы [1, 2], построенная на основе авторской методологии структурно-алгоритмического моделирования и системного анализа, показала:

1) процесс планирования становится самостоятельным процессом;

2) планирование проводится в универсальной трехуровневой структуре (рис. 1) системы. Универсальность этой базовой структуры определяется тем, что она учитывает при переходе с одного уровня на другой возможные изменения масштабов по времени и координатам.

Описанию многоуровневых систем посвящено значительное количество публикаций, которые могут быть разделены на следующие группы.

1. С использованием высокоагрегированного математического аппарата [3], привязка которого к реальной структуре в целях получения прикладного результата затруднительна.

2. С рассмотрением структуры только с вертикальными связями [4].

3. С исследованием традиционных двухуровневых неоптимальных систем [5].

4. С изучением двухуровневых двухэлементных структур [6], далеких от реальных.

Вместе с тем, необходимо математически описать и исследовать процесс планирования в системе с трехуровневой структурой с учетом и согласованием экономических интересов, которые отражаются целевыми функциями.

## Постановка задачи

Рассмотрим класс адаптивных систем, удовлетворяющих условию $N_j[t_i] = [t_j]/[t_i] \gg 1$, где $N_j$ — количество выпускаемой продукции вида $j$ ($j = 1, J$); $[t_j]$ и $[t_i]$ — время изготовления единицы продукции и минимальный интервал времени моделирования. Это условие определяет серийный тип производства. В процессе планирования системы со структурой рис. 1 следует выделить прежде всего специфический общий случай оперативного перехода на выпуск новой продукции. Изменение состава вектора цели (спроса) имеет вид

$$R(t) = \Delta R \cdot 1(t - \theta), \qquad (1)$$

где $R$ — вектор спроса на новую продукцию; $\Delta R$ — изменения вектора; $1(t)$ — единичная функция; $\theta$ — сдвиг во времени.

В этом случае необходимо применение адаптивного автоматизированного режима, поскольку процедура разработки плана и процедура его выполнения разделены небольшим промежутком времени и не учитывать динамичность процесса планирования нельзя.

Этот случай может быть описан с помощью аппарата динамического линейного программирования, сочетающего в себе линейное программирование (статическое) и дифференциальные уравнения:


**Рис. 1. Связь процесса планирования по горизонтали и по вертикали**

$$P(T) \geqslant R(T); \quad (2)$$

$$P(t) = P(t-1) + p(t),\ P(0) = P_0; \quad (3)$$

$$z^{\text{п}}(t) = \mathbf{A}^{\text{п}} z^{\text{п}}(t) + \mathbf{B}^{\text{п}} p_1(t),\ z^{\text{п}}(0) = z_0^{\text{п}}; \quad (4)$$

$$p(t) = \mathbf{C}^n z^n(t); \quad (5)$$

$$\mathbf{D} p_1(t) \leqslant b(t); \quad (6)$$

$$J = -< G,\ P(T) > \to \min, \quad (7)$$

где $z^n$, $p$ — векторы незавершенного производства и ежедневного плана; $p_1$ — вектор запуска комплектов материалов в производство; $R$ — вектор спроса; $\mathbf{D}$ — матрица норм расходов; $b$ — вектор наличного количества ресурсов; $P$ — вектор плана с накоплением; $G$ — прибыль от выпуска единицы продукции, $< .,\ . >$ — скалярное произведение; $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ — матрицы соответствующей размерности; $t$, $T$ — минимальный интервал времени и время моделирования; $J$ — критерий задачи. К задаче (2)—(7) с помощью $P$- и $M$-задач [1, 2] может быть сведен и случай статистического спроса. В работе [2] показано, что описание (2)—(7) за счет увеличения размерности задачи может быть приведено к аппарату статического линейного программирования:

$$\mathbf{A} P[\tau] \leqslant b(\tau); \quad (8)$$

$$R^-[\tau] \leqslant P[\tau] \leqslant R^+[\tau]; \quad (9)$$

$$F(P[\tau] = < C,\ P[\tau] > \to \max, \quad (10)$$

где $\mathbf{A}$ — матрица норм расходов; $P$, $b$, $R$ — вектор-столбцы искомого плана, ресурсов, спроса; $C$ — вектор-строка прибыли за готовую продукцию; $F$ — целевая функция; $\tau$ — интервал времени; $< .,\ . >$ — скалярное произведение. В силу сказанного процедуру согласования экономических интересов в процессе планирования исследуем с помощью выражений (8)—(10).

### Согласование экономических интересов

Будем полагать, что задача (8)—(10) имеет решение, в том числе и при векторном критерии. Тогда задачу решим в два этапа: для отдельного элемента; для взаимодействия элементов.

*Описание отдельных элементов.* Для уровня $h = 3$ используем описание (8)—(10) при $\tau = T$. Для уровня $h = 1$ описание примет следующий вид:

$$F_k(P_k[T]) < C_k,\ P_k[T] > \to \max; \quad (11)$$

$$\mathbf{A}_k P_k[T] \leqslant b_k(0),\ k = 2,\ K; \quad (12)$$

$$P_k[T] \geqslant P[T],\ k = K; \quad (13)$$

$$\mathbf{A}_1 P_1[T] \leqslant b(0),\ k = 1, \quad (14)$$

где $F_k$ — критерий задачи; $C_k$ — стоимость единицы выпускаемой продукции; $b_k$ — вектор ресурсов на входе $k$-го элемента; $\mathbf{A}_k$ — матрица удельных расходов ресурсов; $P_k[T]$ — вектор искомого плана; $P[T]$, $b(0)$ и $b_k(0)$ — векторы плана и наличного количества ресурсов на уровнях $h = 1$ и $h = 2$.

Для уровня $h = 2$ с учетом (13) и (14) справедливы следующие выражения:

$$F(P_k[T]) = \sum_{k=1}^{K} F_k(P_k[T]) \to \max; \quad (15)$$

$$\mathbf{A}_k^2 P_k[T] \leqslant b_k^2(0),\ k = 1,\ K; \quad (16)$$

$$\mathbf{A}_k^1 P_k[T] \leqslant P_{k-1}[T],\ k = 2,\ K; \quad (17)$$

$$P_K[T] \geqslant R^-[T],\ k = K; \quad (18)$$

$$\mathbf{A}_k^1 P_k[T] \leqslant b(0),\ k = 1, \quad (19)$$

где $b_k^2$ — количество нематериальных (ненакапливаемых) ресурсов; $\mathbf{A}_k^1$, $\mathbf{A}_k^2$ — матрицы удельных расходов накапливаемых (материальных) и ненакапливаемых ресурсов; $P[T]$ и $P_k[T]$ — решение, полученное на уровне $h = 2$; $\mathbf{A}_k = (\mathbf{A}_k^1,\ \mathbf{A}_k^2)^T$.

Выражения (17) характеризуют (см. рис. 1) горизонтальные связи (класс 1), а выражения (16), (18), (19) — вертикальные связи (классы 2а и 2б). Ограничения в выражении (16) будем называть локальными.

*Взаимодействие элементов.* В задаче согласования экономических интересов в принятой трехуровневой системе могут быть выделены два класса задач (рис. 2).

1. Горизонтальное согласование элементов $k = 1,\ K$ на уровне $h = 2$ — класс 1.

2. Вертикальное согласование элементов $k = 1,\ K$ уровня $h = 1$ с элементами уровня $h = 2$ — класс 2а; элементов уровней $h = 2$ и $h = 3$ — класс 2б.

В задаче класса 1 (выражения (15)—(17)) рассмотрим два случая.

Локальные ограничения (16) соблюдаются и не влияют на решения. Тогда ими можно пренебречь и предложить следующий алгоритм.

*Шаг 1.* Не снижая общности, положим $P'_K[T] = R^-[T]$, где $R^-[T]$ задается уровнем $h = 3$.

*Шаг 2.* Для каждого $k$ (от $k = K$ до $k = 1$) рассчитаем описанными ранее методами (с учетом векторного критерия $F_l$, $l = 1,\ L$, с определением чувствительности решений) планы $P_k^1[T]$, $F_k(P_k^1[T])$, $P'_{k-1}[T] = \mathbf{A}_k P'_k[T]$, $b'(0) \leqslant\geqslant b(0)$.

*Шаг 3.* Для заданного — с уровня $h = 3$ — значения вектора $b(0)$ определим планы $P_k^2[T]$, $F_1(P_k^2[T])$ при изменении $k$ от 1 до $K$.

*Шаг 4.* Найдем значения $\Delta F_k = F_k(P_k^1[T]) - F_k(P_k^2[T])$.

*Шаг 5.* Возможны два варианта: $\Delta F_k$ имеет одинаковые знаки для всех $k$ ($k = 1,\ K$) —переход к шагу 6; $\Delta F_k$ различные по знакам — переход к шагу 7.

*Шаг 6.* В первом частном варианте элементы уровня $h = 2$ согласованы и можно использовать понятие эквивалентности и монотонности функций. Целевые функции $F_l = F(F_{lk},\ k = 1,\ K)$ в выражении (17) монотонны относительно $F_{lk}(F(F_{l1},\ ...,\ F_{lk-1},$


**Рис. 2. Согласование экономических интересов**

$F_{lk}, F_{lk+1}, ..., F_{lK}) > F(F_{l1}, ..., F_{lk-1}, F'_{lk}, F_{lk+1}, ..., F_{lK})$, если $F_{lk} > F'_{lk}$.

*Шаг 7.* Во втором частном варианте интересы отличаются (не согласованы). Здесь возможно использовать теорию игр либо равновесие по Нэшу; либо метод решения задачи с векторным критерием.

При использовании теории игр можно, не снижая общности, считать, что для элементов $k = 1, n$ значения $\Delta F_k < 0$, а для элементов $k = n + 1, K(n, n + 1 \in 1, K)$ — значения $\Delta F_k > 0$.

Очевидно, что при плане $P_k^2[T]$ потери несут элементы $k = n + 1, K$ и система в целом, при плане $P_k^1[T]$ потери характерны для элементов $k = 1, n$. Тогда в интересах системы в целом целесообразно дополнительный выигрыш $\sum_{r=n+1}^{K} \Delta F_r$ при плане $P_k^1[T]$ распределить с учетом элементов $k = 1, n$ по правилу

$$\delta F_k = \left( \sum_{r=n+1}^{K} |\Delta F_r| \, |\Delta F_k| \right) / \sum_{k=1}^{K} |\Delta F_k|. \qquad (20)$$

Можно показать, что решение (20) существует. Тогда компромиссным (равновесным) решением будет $P'_k[T]$, а значения целевых функций

$$F_k(P'_k[T]) = \begin{cases} F_k(P_k^1[T]) + \delta F_k, \; k = 1, n, \\ F_k(P_k^1[T]) - \delta F_k, \\ k = n + 1, K, \end{cases} \qquad (21)$$

где $P'_k[T]$ — компромиссное решение.

Несложно показать, что решение (21) — устойчивое равновесие по Нэшу.

Недостатками использования равновесия по Нэшу являются итеративный характер решения и потребность для $k$-го элемента в информации о всех других элементах.

Эти недостатки не имеют места, как показано автором, при использовании метода решения векторных задач. Введем обозначения

$$F_{k\min} = \begin{cases} F_k(P_k^1[T]), \; k = 1, n \\ F_k(P_k^2[T]), \; k = n + 1, K \end{cases} \qquad (22)$$

$$F_{k\max} = \begin{cases} F_k(P_k^2[T]), \; k = 1, n \\ F_k(P_k^1[T]), \; k = n + 1, K. \end{cases} \qquad (23)$$

При использовании метода идеальной точки целевая функция

$$F' = \sum_{k=1}^{n} \{F_k(P_k^2[T]) - F_k(P'_k[T])\}^2 + \sum_{k=n+1}^{K} \{F_k(P_k^1[T]) - F_k(P'_k[T])\}^2 \to \min.$$

В силу выпуклости области ограничений и целевой функции решение, если оно существует, единственно.

Задачи (11), (12) и (15), (16), (17) вертикального согласования (класс 2а) решаются наиболее просто. Напомним, что элементы являются согласованными, если целевые функции монотонны, а области определения решений совпадают. Монотонность функции $F = F(F_1, ..., F_K)$ уже отмечалась. Область определения решений (12) является частью области определения решений задачи (16)—(19), если $b_k^2(0) = b_k'^2(0)$, и величина $b_k^1(0)$ определяется значением $b_k'^1(0)$, где $b_k^1(0)$, $b_k'^1(0)$ — величины на уровнях $h = 2$ и $h = 1$. Целевая функция (11) есть часть целевой функции (15). Таким образом, интересы согласованы.

Перейдем к решению задач (18), (19) вертикального согласования (класс 2б). В работе [2] показано, что и эти задачи согласованы.

Если условия (16) не соблюдаются, необходимо предварительно определить минимум дополнительных ресурсов $\Delta b_k^2(0)$, обеспечивающих условия (16). Если таких ресурсов нет, необходимо найти новое значение плана $P''_k[T] \leqslant P'_k[T]$.

Следует особо отметить, что задача динамического линейного программирования вида (2)—(7) дает возможность описывать и динамические процессы управления. Действительно, динамическая задача примет вид для объекта управления

$$\dot{z}(t) = \mathbf{A}z(t) + \mathbf{B}u(t), \; z(0) = z_0; \qquad (24)$$

$$y(t) = \mathbf{C}z(t); \qquad (25)$$

$$\mathbf{D}u(t) \leqslant b(t); \qquad (26)$$

для управляющей части

$$\varepsilon(t) = p(t) - y(t); \qquad (27)$$

$$J = \int_0^T \{C_1\varepsilon(t) + C_2u(t)\}dt \to \min, \qquad (28)$$

или в дискретной форме

$$J = \{C_1z(t) - C_2u(T)\} \to \max, \qquad (29)$$

где $u(t)$, $p(t)$, $y(t)$, $\varepsilon(t)$ — вектор-столбцы управления, плана, выхода, отклонения; $C_1$, $C_2$ — вектор-

строки потерь за счет отклонения от плана и затрат на дополнительные ресурсы. Выражения (2)—(7) и (24)—(29) представляют собой однородное системное описание разнородных процессов планирования и управления, образуя однородный метод исследования процесса адаптивного автоматизированного управления.

Система (24)—(29) сводится [7, 8] к описанию (8)—(10). Здесь задача согласования экономических интересов решается описанными ранее методами адаптивного планирования, а задача координации динамических свойств системы требует специального рассмотрения.

## Компьютерная реализация

Прикладную иллюстрацию рассмотренных теоретических положений проведем на примерах малой размерности, поскольку они хорошо иллюстрируют идею и не требуют для своего изложения, как выполненные автором высокоразмерные примеры, значительного места в ограниченной по объему работе.

Рассмотрим числовой пример

$$F = 60x_1 + 70x_2 + 120x_3 + 130x_4 \to \max;$$

$$x_1 + x_2 + x_3 + x_4 \leqslant 16;$$

$$6x_1 + 5x_2 + 4x_3 + 3x_4 \leqslant 110;$$

$$4x_1 + 6x_2 + 10x_3 + 13x_4 \leqslant 100;$$

$$x = \{x_j\},\ x_j \geqslant 0,\ j = 1, 4.$$

Для реализации возможно использовать различные программные средства. При необходимости оперативного решения задач (1)—(3) небольшой размерности можно использовать в простейшем случае электронные таблицы Excel. В более сложных случаях целесообразнее применять пакет Optimization Toolbox программного продукта MatLab, который хорошо стыкуется с таблицами Excel, СУБД Access. В последнем случае программа для решения приведенного числового примера имеет вид

$$\text{function } x = 1p111(f, A, b, lb);$$

$$f = [-60; -70; -120; -130];$$

$$A = [1\ 1\ 1\ 1; 6\ 5\ 4\ 3; 4\ 6\ 10\ 13];$$

$$b = [16; 110; 100];$$

$$1b = \text{zeros}(4, 1);$$

$$[x] = \text{linprog}(f, A, b, [], [], lb);$$

Результат ее выполнения представлен на экране (рис. 3).

Из него видно, что пакет Optimization Toolbox имеет неудобный интерфейс ввода—вывода.

Задачу размерности, приемлемой для промышленного применения, предпочтительнее решать в пакете LINDO. Для автоматизации вычислений целесообразно либо пакет LINDO интегрировать с СУБД InterBase, работающей в среде Delphi, либо решать задачу только в рамках InterBase с использованием алгоритма работы [6]. В последнем случае результат решения автором приведенного выше числового примера показан на рис. 4.

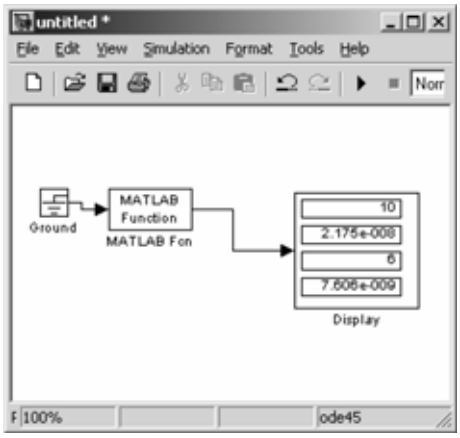



**Рис. 3. Решение задачи линейного программирования в Optimization Toolbox**

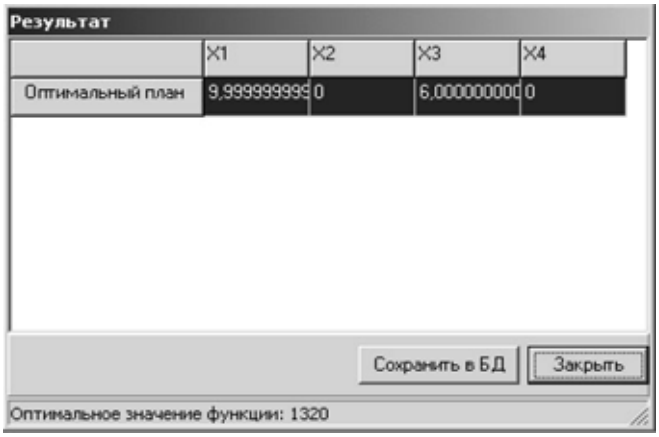



**Рис. 4. Решение примера**

## Заключение

Рыночные отношения характеризуются изменением не только величин, составляющих векторы спроса, но и изменением состава самого вектора цели при переходе на выпуск новой продукции. В последнем случае требуется адаптивное планирование с учетом его инерционности. Для описания процесса адаптивного планирования предложен и исследован метод динамического линейного программирования. Этот метод позволяет изучать как отдельный структурный элемент, так и систему в целом, при этом разнородные процессы планирования и управления описываются одним и тем же методом. Автором осуществлена компьютерная реализация процесса адаптивного планирования.

### Список литературы

1. **Системное** проектирование интегрированных АСУ ГПС машиностроения / Ю. М. Соломенцев, В. Я. Полыскалин, В. Д. Чертовской и др. М.: Машиностроение, 1988. 488 с.

2. **Чертовской В. Д.** Теоретические основы автоматизированного управления: процедурное представление. М.: МГУП, 2007. 218 с.
3. **Голенко-Гинзбург Д., Кац В., Синяковский С., Ицкович Э. Л.** Управление трехуровневой производственной системой типа "человек—машина" // Автоматика и телемеханика. 2000. № 5. С. 166—184.
4. **Findeisen M. J. et al.** Control and coordination in hierarchical systems. N. Y.: Wiley, 1980. 467 p.
5. **Ширяев В. И., Ширяев Е. В., Головин И. Л., Смолин В. В.** Теория и алгоритмы для идентификации, адаптации и управления фирмой в условиях изменения ситуации на рынке // Информационные технологии. 2002. № 4. Приложение. 25 с.
6. **Альсевич В. В., Габасов Р., Глушенков В. С.** Оптимизация линейных экономических моделей. Статические задачи. Минск: БГУ, 2000. 210 с.
7. **Чертовской В. Д.** Интеллектуализация автоматизированного управления производством. СПб.: С.-Петербург. ун-т, 2007. 164 с.
8. **Chertovskoy V. D.** Production Planning end Control in Intelligent Manufacturing System. // Intelligent Manufacturing System IMS 2008. Poland; Szczecin: University of Technology. P. 111—115.

УДК 51-77:330.322.011

**Е. М. Бронштейн,** д-р физ.-мат. наук, проф., e-mail: bro-efim@yandex.ru,
**Г. Р. Муслимова,** аспирант,
Уфимский государственный авиационный технический университет

# Формирование оптимальных портфелей, состоящих из инвестиционных проектов, с учетом групповых выплат

*Рассматривается задача формирования оптимального инвестиционного портфеля, состоящего из инвестиционных проектов, когда предусмотрены потоки платежей по некоторым группам проектов. В частности, учтены возможность групповых потоков разных знаков и возможность заимствования средств. Предложен эвристический метод снижения размерности задачи булевского линейного программирования, основанный на предварительном решении соответствующей задачи линейного программирования. Также рассмотрены некоторые существующие методы для решения поставленной задачи, представлены результаты численного эксперимента.*

***Ключевые слова:*** *инвестиционный проект, дискретная оптимизация, метод ветвей и границ*

## Введение

Задачи оптимизации инвестиционных решений занимают важное место в математической теории финансов. В данной работе рассматривается задача формирования портфеля, состоящего из инвестиционных проектов. Впервые задачи формирования инвестиционного портфеля были поставлены в работе [1], в работе [2] для их решения применены методы линейной оптимизации. В работах [3—6] сформулированы задачи формирования оптимальных портфелей при различных предположениях, где в качестве целевой функции принимается чистая приведенная (NPV) или накопленная (NFV) стоимость портфеля. В работе [7] обсуждаются задачи, где в качестве целевой функции принимается модифицированная рентабельность, при этом учитываются и другие характеристики проекта, в работе [8] — время финансирования портфеля взаимозависимых проектов.

Под (дискретным) инвестиционным проектом (потоком платежей) понимается вектор $C = (c_0, c_1, ..., c_n)$, компонента $c_k$ которого это платеж (при $c_k < 0$) или возврат средств (при $c_k > 0$) в момент времени $k$. Предположим, что инвестору предложен набор проектов, из которых необходимо выбрать несколько инвестиционных проектов для финансирования. Предполагается, что в качестве альтернативного используется вложение средств с известным годичным коэффициентом накопления. В работах [4, 9] поставлена и проанализирована задача формирования оптимального портфеля инвестиционных проектов, если для некоторых групп проектов предусмотрены групповые платежи. Например, необходим офис, если финансируется хотя бы один проект в некотором регионе. Другой пример связан с приобретением специального транспорта. В работах [4, 9] предполагалось, что для групп проектов могут осуществляться только платежи. В то же время, в приведенных примерах офис или транспортное средство на некоторое время может сдаваться в аренду, а это приведет и к поступлениям средств.

## Постановка задачи

Имеется набор проектов $C_1$, $C_2$, ..., $C_m$, из которых инвестору необходимо выбрать некоторые инвестиционные проекты для финансирования. При этом предполагается некоторая зависимость между проектами, т. е. выбор проекта для включения в портфель зависит от того, с какими проектами он связан. Зависимость можно представить в виде ориентированного графа, где вершины соответствуют проектам, а ребра определяют зависимость между проектами таким образом, что включение в портфель следующего проекта воз-

можно, только если в портфель включен его предшественник.

Пусть выделены некоторые подмножества $U_j \subset \{1, ..., m\}$ ($j = 1, ..., s$) проектов, для каждого из которых определен вектор платежей $P_j = (P_{j0}, P_{j1}, ..., P_{jn})$, реализуемый при финансировании хотя бы одного проекта из множества $U_j$. Множества $U_j$ могут пересекаться. Также для проектов заданы годичный коэффициент накопления по банковским вкладам в течение $n$ лет (за год вклад возрастает в $q$ раз) и ставка $r > q$, под которую инвестор при необходимости занимает средства, т. е. долг за год возрастает в $r$ раз. Начальный капитал инвестора обозначим через $F_{-1}$.

Необходимо выбрать для финансирования проекты, которые будут обеспечивать максимально возможный капитал на конечный момент времени.

Для формализации задачи введем следующие обозначения:

$F_k$ — капитал инвестора (или долг, если эта величина отрицательная) в момент $k$ после всех выплат;

$x_i$ — булевская переменная, равная 1, если $i$-й проект включается в портфель, и равная 0 в противном случае;

$y_j$ — булевская переменная, равная 1, если в портфель включен хотя бы один проект из множества $U_i$, и равная 0 в противном случае.

Имеют место следующие ограничения:

$$y_j \geqslant x_i \text{ при } i \in U_j; \tag{1}$$

$$y_j \leqslant \sum_{i \in U_j} x_i \text{ при } i \in U_j; \tag{2}$$

$$x_i \geqslant x_k, \tag{3}$$

если для выполнения $k$-го проекта должен выполняться $i$-й проект.

Неравенства (1) и (2) в совокупности означают, что $y_j = 1$ тогда и только тогда, когда $x_i = 1$ хотя бы для одного $i \in U_j$.

Неравенство (3) означает, что из $x_k = 1$ следует $x_i = 1$.

Имеем $F_0 = F_{-1} + \sum_{i=1}^{m} x_i c_{i0} + \sum_{j=1}^{s} y_j p_{j0}$.

Если $F_0 \geqslant 0$, то сумма на счете возрастает за год в $q$ раз, с учетом платежей в 1-й момент времени получим

$$F_1 = qF_0 + \sum_{i=1}^{m} x_i c_{i1} + \sum_{j=1}^{s} y_j p_{j1}.$$

Если $F_0 \leqslant 0$, то долг за год возрастет в $r$ раз, откуда имеем

$$F_1 = rF_0 + \sum_{i=1}^{m} x_i c_{i1} + \sum_{j=1}^{s} y_j p_{j1}.$$

Если значение $F_k$ ($k = 1, ..., n-1$) известно, то аналогично можно получить

$$F_{k+1} = \begin{cases} qF_k + \sum\limits_{i=1}^{m} x_i c_{i(k+1)} + \\ + \sum\limits_{j=1}^{s} y_j p_{j(k+1)} \text{ при } F_k \geqslant 0 \\ rF_k + \sum\limits_{i=1}^{m} x_i c_{i(k+1)} + \\ + \sum\limits_{j=1}^{s} y_j p_{j(k+1)} \text{ при } F_k \leqslant 0. \end{cases} \tag{4}$$

Необходимо найти значения $x_i, y_j \in \{0, 1\}$, при которых величина $NFV = F_n$ является максимальной.

Возможна ситуация, когда заимствование средств невозможно. В этом случае необходимо обеспечить неразорение инвестора в любой момент времени. Условие неразорения имеет вид

$$F_k \geqslant 0 \text{ при } k = 0, 1, ..., n. \tag{5}$$

При выполнении условия (5) в условии (4) актуальна только верхняя строка.

Задача (1)—(6) является задачей булевского линейного программирования. Отметим, что она всегда допустима: значения $x_i = 0$, $y_j = 0$ удовлетворяют всем ограничениям.

Задача (1)—(7) сводится к задаче частично булевского линейного программирования, которая также всегда допустима. Для этого введем вспомогательные переменные $V_i$ ($i = 0, ..., n$). Заметим, что поскольку $r > q$, из (4) следует равенство

$$F_{k+1} = \max\left\{qF_k + \sum_{i=1}^{m} x_i c_{i(k+1)} + \sum_{j=1}^{s} y_j p_{j(k+1)}, \right.$$
$$\left. rF_k + \sum_{i=1}^{m} x_i c_{i(k+1)} + \sum_{j=1}^{s} y_j p_{j(k+1)}\right\}.$$

Задача принимает следующий вид.

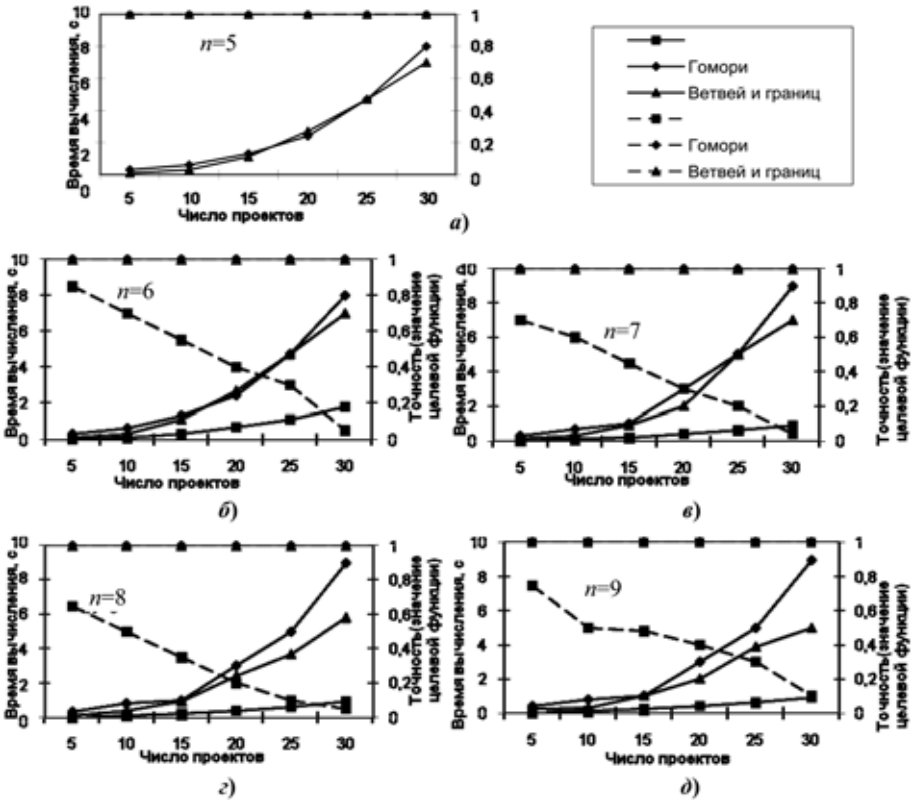
Найти числа

$$x_1, ..., x_m \in \{0, 1\};$$
$$y_1, ..., y_s \in \{0, 1\};$$
$$F_1, ..., F_n \in R;$$
$$V_1, ..., V_n \in R$$

такие, что

$$y_j \geqslant x_i \text{ при } i \in U_j;$$
$$y_j \leqslant \sum_{i \in U_j} x_i \text{ при } i \in U_j;$$
$$x_i \geqslant x_k \text{ при } i, k = 1, ..., m;$$
$$V_{k+1} \leqslant qF_k \text{ при } k = 0, 1, ..., n-1;$$
$$V_{k+1} \leqslant rF_k \text{ при } k = 0, 1, ..., n-1;$$
$$F_k = V_k + \sum_{i=1}^{m} x_i c_{ik} + \sum_{j=1}^{s} y_j p_{jk} \text{ при } k = 0, ..., n;$$
$$F_n \to \max.$$

При этом $V_0 = F_{-1}$.




**Зависимость времени вычисления и точности от числа проектов**

### Численные методы решения

Для решения задачи (1)—(8) применим метод ветвей и границ [10—12]. Этот метод дает точное решение, но при этом является трудоемким.

Для снижения размерности задачи применим следующий эвристический прием. Решим аналогичные задачам (1)—(9) и (1)—(10) задачи линейного программирования, сняв условие целочисленности, т. е. полагая $x_i, y_j \in [0, 1]$. Естественно считать более перспективными для включения в портфель те проекты, у которых $x_i$ больше. Для снижения размерности решаемой задачи зададим параметр $K \in [0, 1]$, портфель будем формировать только из тех проектов, у которых $x_i > K$. При уменьшении значения $K$ множество перспективных проектов расширяется. После отбора проектов оптимизация осуществляется методом ветвей и границ.

Также применялся метод отсечений (Гомори) [9—11].

### Численный эксперимент

Для анализа эффективности описанных алгоритмов был проведен численный эксперимент. При этом были заданы следующие параметры:

- $F_{-1} = 100$ — начальный капитал инвестора;
- $C_j$ и $P_j$ — платежи по проектам, генерировались случайным образом в диапазоне от −150 до 150;
- число подмножеств $U_j$ задавалось в зависимости от числа проектов от 3 до 8 (3 для 5 проектов, 4 для 10 проектов, 5 для 15 и т. д.);
- ставка банковского вклада 4 %;
- ставка банковского кредита 7 %;
- значения булевской переменной $y_j$ генерировались случайным образом;
- зависимость проектов (условие (4)) не предполагается;
- для метода ветвей и границ проекты упорядочивались по убыванию чистой приведенной стоимости с учетом групповых платежей;
- $n$ — длительность проектов, в годах.

Для каждого эксперимента было сгенерировано 100 задач.

Результаты численного эксперимента в зависимости от продолжительности проектов и их числа представлены на графиках (см. рисунок), где сплошные линии отображают временную зависимость, а штриховые — точность. Точность результатов, полученных методами ветвей и границ и Гомори, принята за 1.

В таблице приведены данные по среднему числу шагов метода ветвей и границ (ВГ) и модифицированного метода ветвей и границ (МВГ), которые необходимы для достижения 30 % точности относительно рекордного (для того или иного метода) результата. Эксперимент показал, что от продолжительности проектов эти величины практически не зависят.

### Заключение

В работе поставлена задача формирования оптимального инвестиционного портфеля, в которой учтен ряд дополнительных факторов. Рассмотрены некоторые методы решения данной задачи и предложен способ снижения размерности переборной задачи. Анализ полученных результатов позволяет сделать следующие выводы:

- трудоемкость вычислений слабо зависит от длительности проектов, но существенно зависит от числа проектов;
- модифицированный метод ветвей и границ целесообразно использовать для небольшого числа проектов, если весьма грубую оценку решения требуется получить быстро;

**Зависимость числа шагов алгоритмов, необходимых для достижения 30 % точности относительно оптимального результата, от числа проектов**

| Число шагов метода | Число проектов | | | | | |
|---|---|---|---|---|---|---|
| | 5 | 10 | 15 | 20 | 25 | 30 |
| ВГ | 1 | 1—2 | 4—5 | 6 | 7 | 8 |
| МВГ | 1 | 1 | 3 | 5 | 5—6 | 7 |

- при малом временном горизонте (до 5) применение метода Гомори для получения точного решения более эффективно по сравнению с методом ветвей и границ, при временных горизонтах порядка 8—9 — наоборот.

**Список литературы**


1. **Lorie J. H., Savage I. J.** Three Problems in Rationing Capital // Journal of Business. 1955. V. XXVIII. N 4.
2. **Weingartner H. M.** Mathematical Programming and the Analysis of Capital Budgeting Problems. Chicago: Markham publ, 1967.
3. **Бронштейн Е. М., Спивак С. И.** Как сформировать оптимальный портфель // Рынок ценных бумаг. 1997. № 14.
4. **Mamer J. W., Shogan A. W.** A Constrained Capital Budgeting Problem with Applications to Repair Kit Selection // Management Science. 1987. V. 33, N 6.
5. **Виленский П. Л., Лившиц В. Н., Смоляк С. А.** Оценка эффективности инвестиционных проектов. М.: Дело, 2002.
6. **Бронштейн Е. М.** Оптимальные инвестиционные портфели // Системное моделирование социально-экономических процессов. М.: ЦЭМИ РАН, 2004.
7. **Бронштейн Е. М., Олейник Т. Н.** Задача календарного планирования портфеля инвестиционных проектов // Информационные технологии. 2007. № 3.
8. **Balinski M. L.** On a Selection Problem // Management Science. 1970. V. 17, N 11.
9. **Деордица Ю. С., Нефедов Ю. М.** Исследование операций в планировании и управлении: учеб. пособие. Киев: Выща школа, 1991.
10. **Корбут А. А., Сигал И. Х., Финкельштейн Ю. Ю.** Об эффективности комбинаторных методов в дискретном программировании. М.: Наука, 1979.
11. **Сигал И. Х., Иванова А. П.** Введение в прикладное дискретное программирование. М.: Физматлит, 2007.


# ПРИКЛАДНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ


УДК 528.854.2



**А. В. Замятин,** канд. техн. наук, доц.,
Томский политехнический университет,
e-mail: zamyatin@tpu.ru


## Распределенные вычисления в задачах автоматизированной интерпретации аэрокосмических изображений


*Предложена обобщенная технология распределенной классификации многоканальных данных дистанционного зондирования Земли, учитывающая особенности классификаторов с линейным разделением и оценкой условной плотности распределения. Приведены результаты численных экспериментов, полученные с использованием модельных многоканальных данных на дорогостоящем суперкомпьютерном кластере и кластере из недорогих персональных компьютеров в локальной сети, позволяющие провести комплексный анализ предложенной технологии, включая оценку ее производительности, параллельного ускорения и параллельной эффективности.*



***Ключевые слова:*** *распределенные вычисления, многоканальные данные, классификация, суперкомпьютер, компьютерный кластер*


### Введение

С распространением данных дистанционного зондирования Земли (ДЗЗ), представленных терабайтами мульти- и гиперспектральных данных, а также совершенствованием процедур их автоматизированной интерпретации потребители увеличивают требования к оперативности процесса обработки данных и точности его результатов.

Наличие в любой современной организации достаточно большого объема доступных вычислительных ресурсов в виде персональных компьютеров (ПК), объединенных в локальной вычислительной сети, а также все большее распространение центров коллективного пользования, оснащенных суперкомпьютерной многопроцессорной техникой, предоставляют возможности по организации на базе подобного оборудования вычислительных кластеров с высокой производительностью обработки данных ДЗЗ за счет использования *распределенно-параллельных вычислений*, активно исследуемых отечественными и зарубежными учеными [1—6]. К сожалению, многие работы в этой области носят обзорную направленность, не содержат деталей реализации распределенных вычислений в зависимости от законов распределения данных, особенностей схем и технологий классификации, широко применяемых при последовательном решении задач автоматизированной интерпретации мульти- и гиперспектральных данных ДЗЗ [7—11]. Кроме того, подавляющее большинство исследований проводятся с использованием дорогостоящей высокопроизводительной суперкомпьютерной техники, а результаты, полученные с использованием кластеров из недорогих пользовательских ПК, объединенных в локальной сети с невысокими показателями пропускной способности, практически отсутствуют [3, 4]. Все это затрудняет практическую реализацию и анализ подходов к распределенной автоматизированной интерпретации данных ДЗЗ на вычислительных кластерах различной стоимости и конфигурации.

Принимая во внимание вышеизложенное, предлагается подход к эффективной организации распределенных вычислений на кластерах различной стоимости и конфигурации при решении задач автоматизированной интерпретации многоканальных (мульти- и гиперспектральных) аэрокосмических изображений (АИ) с учетом существующих технологий последовательной классификации данных ДЗЗ.

### Особенности распределенной классификации данных ДЗЗ

Рассмотрим особенности, оказывающие наиболее существенное влияние на простоту разработки программного обеспечения (ПО) кластера, степень рационального использования его ресурсов и производительность. Реализация распределенной классификации предполагает наличие управляющего (головного) вычислительного узла ($p_0$) и $N$ вычислительных узлов (ВУ) для параллельной обработки данных ($p_i$, $i = 1, 2, ..., N$).

**Разбиение данных.** Способ разбиения данных ДЗЗ для классификации на вычислительном кластере должен быть, с одной стороны, достаточно простым и не требовать затратного предварительного анализа данных, с другой — позволять использовать схожую процедуру классификации на каждом ВУ, что облегчит разработку и отладку ПО классификаторов, а также упростит процесс объединения результатов параллельной обработки. В подобных случаях широко используют разделение изображения $\mathbf{I} = \{\mathbf{I}_i, i = 1, 2, ..., N, N = C \times R\}$ на фрагменты $\mathbf{I}_i$ некоторой регулярной сетью с числом столбцов $C$ и строк $R$. Каждый из фрагментов содержит $Z$ каналов, где $Z$ — размерность пространства признаков исходного изображения $\mathbf{I}$ [1]. В общем случае при решении задачи построения такой регулярной сети, определяющей итоговое число фрагментов данных, целесообразно учитывать различную вычислительную мощность ВУ, пропускную способность каналов связи, возможные динамические изменения архитектурных параметров вычислительного кластера. В такой постановке эта задача требует отдельного детального рассмотрения, поэтому для простоты предположим, что мощности $p_i$, $i = 1, 2, ..., N$, и пропускная способность каналов связи кластера одинаковы и не изменяются с течением времени. В этом случае фрагменты $\mathbf{I}_i$ имеют идентичный размер, что позволяет использовать на каждом ВУ сходную процедуру классификации в соответствии с моделью параллельной обработки SPMD (Single Process, Multiple Data) [5]. При многоядерной архитектуре ВУ возможно увеличение вычислительной эффективности за счет параллельной многопроцессной SMP-обработки (Symmetrical Multi Processing) фрагмента $\mathbf{I}_i$ средствами операционной системы (ОС) ВУ, что следует учитывать при разработке ПО кластера [5].

**Классификация.** Известно, что наиболее распространенные подходы к контролируемой классификации данных ДЗЗ основаны на методах линейного разделения, не предполагающих оценку условной плотности распределения (далее — плотности распределения), и методах максимального правдоподобия с оценкой условной плотности распределения [7, 10, 12].

В первом случае для интерпретации мульти-спектральных АИ применяют классификаторы с использованием искусственных нейронных сетей (ИНС) или опорных векторов (*SVM*) [2, 7, 13, 14]. Для классификации гиперспектральных АИ применяют подходы, основанные на поиске минимального расстояния (*Hypermin*) или спектрального угла (*Hypersam*) между признаковым вектором и векторами средних обучающих выборок [2]. Все подобные классификаторы предполагают предварительное обучение по значениям обучающих выборок $\Lambda = \{\lambda_i, i = 1, 2, ..., M\}$ классов $\omega_i$ и сохранение параметров обучения классификатора $\Pi = \{\pi_i, i = 1, 2, ..., M\}$, $M$ — число классов [2]. Параметры $\Pi$, как правило, немногочисленны: для ИНС — это топология, веса синапсов, параметры пороговых функций, для метода опорных векторов — векторы весов $\boldsymbol{w}_j$ и свободные коэффициенты $b_j$, $j = 1, 2, ..., M'$, $M' = C_M^2$, определяемые для каждой пары классов, где $C_M^2$ — число сочетаний из $M$ по 2 [2, 13].

Во втором случае классификация с оценкой плотности распределения может быть параметрической и непараметрической [7, 9]. В качестве параметрического обычно используют классификатор максимального правдоподобия с оценкой плотности распределения гауссианом (ML), недостатком которого является невысокая точность при произвольном распределении признаков. Непараметрические классификаторы, отличающиеся высокими вычислительными затратами, часто основаны на методе $k$-го ближайшего соседа ($k$-$NN$) и позволяют получить более точные результаты при произвольном распределении признаков, а поэтому представляют значительный практический интерес [8, 9, 12]. Этот тип классификации предполагает восстановление значений условной плотности вероятности $p(\boldsymbol{x}|\omega_i)$ для каждого классифицируемого значения $\boldsymbol{x} = \{x_1, x_2, ..., x_Z\}$ изображения $\mathbf{I}$ по значениям обучающих выборок $\Lambda$ без сохранения параметров обучения.

Известно, что сокращение затрат на передачу данных при распределенной обработке может увеличить производительность кластера [15]. Непараметрическому классификатору с оценкой плотности распределения требуется передача с $p_0$ на $p_i$, $i = 1, 2, ..., N$, значений обучающих выборок $\Lambda$, в то время как классификатору с линейным разделением достаточно передать с $p_0$ на $p_i$ лишь параметры $\Pi$, что при значительном объеме обучающих данных и невысокой пропускной способности каналов связи кластера позволит существенно сократить затраты на передачу данных и увеличить производительность кластера.

Для повышения точности классификации мультиспектральных данных ДЗЗ применяют различные подходы с использованием текстурных характеристик и формированием расширенного пространства признаков. Наиболее простым способом применения этих характеристик является совместное использование спектральных (первичных) и текстурных (вторичных) признаков в объединенном признаковом векторе $\{\boldsymbol{x}\} = \{\boldsymbol{x}^{\text{пер}}\} + \{\boldsymbol{x}^{\text{втр}}\}$, однако в этом случае результаты классификации могут характеризоваться низким пространственным разрешением [8, 16]. Более точные результаты классификации получают при раздельном использовании компонент $\boldsymbol{x}^{\text{пер}}$ и $\boldsymbol{x}^{\text{втр}}$ и применении двухэтапного преобразования [8]. Суть преобразования заключается в том, что на первом этапе осуществляется расчет апостериорной вероятности $p(\omega_i|\boldsymbol{x}^{\text{втр}})$, а на втором этапе, в соответствии с методом максимального правдоподобия, выполняется поиск класса $\omega_i$ с максимальным значением $p(\omega_i) \times p(\boldsymbol{x}^{\text{пер}}|\omega_i)$, где $p(\omega_i) = p(\omega_i|\boldsymbol{x}^{\text{втр}})$ — априорная вероятность принадлежности к классу $\omega_i$ [10].

При распределенной обработке эта схема классификации АИ может быть реализована двумя способами — с одновременной передачей фрагментов данных $\mathbf{I}_i = \{\boldsymbol{x}^{\text{пер}}\}$ и $\mathbf{I}_i' = \{\boldsymbol{x}^{\text{втр}}\}$ с $p_0$ на $p_i$, $i = 1, 2, ..., N$, или с последовательной передачей с $p_0$ на $p_i$ (на первом этапе на $p_i$ передается фрагмент $\mathbf{I}_i'$ и рассчитываются значения $p(\omega_i) = p(\omega_i|\boldsymbol{x}^{\text{втр}})$, а затем передается фрагмент $\mathbf{I}_i$, $i = 1, 2, ..., N$, и осуществляется второй этап преобразований). Для более рационального использования оперативной памяти (ОП) узла $p_i$ следует применять последовательную передачу фрагментов данных $\mathbf{I}_i$ и $\mathbf{I}_i'$, что позволит не размещать их в памяти единовременно и увеличит вычислительную эффективность обработки.

### Технология распределенной классификации АИ

Рассмотрим обобщенную технологию распределенной классификации многоканальных данных ДЗЗ, учитывающую отмеченные выше особенности, которая реализуется как на дорогостоящем суперкомпьютерном кластере, так и на кластере из недорогих ПК в локальной сети.

*Этап 1.* Формирование на $p_0$ многоканального исходного массива данных, представленного либо каналами мультиспектрального АИ $\mathbf{I}$ и каналами текстурного блока данных $\mathbf{I}'$, либо только каналами гиперспектрального АИ $\mathbf{I}$ (без расчета текстурных каналов).

*Этап 2.* Построение на $p_0$ обучающих выборок $\Lambda = \{\lambda_i, i = 1, 2, ..., M\}$ основным критерием качества которых является *репрезентативность*.

*Этап 3.* При использовании классификатора с линейным разделением его обучение на $p_0$ для распознавания каждого из $\omega_i$ классов по выборкам $\Lambda$ с сохранением соответствующих параметров обучения $\Pi = \{\pi_i, i = 1, 2, ..., M\}$.

*Этап 4.* Разбиение исходного массива данных $\mathbf{I}$ ($\mathbf{I}'$) некоторой регулярной сетью на $N$ фрагментов $\mathbf{I}_i$ ($\mathbf{I}_i'$) идентичного размера, где $i = 1, 2, ..., N$, $N$ — число доступных для параллельной обработки ВУ, а также формирование $N$ идентичных групп с параметрами $\Pi$ (для классификации с линейным разделением) или с выборками $\Lambda$ (для классификации с оценкой плотности распределения).

*Этап 5.* Полученные на этапе 4 $N$ групп данных $\{\mathbf{I}_i(\mathbf{I}_i'), \Pi\}$ или $\{\mathbf{I}_i(\mathbf{I}_i'), \Lambda\}$ передаются на $p_i$, где исполняются идентичные процедуры попиксельной классификации фрагментов $\mathbf{I}_i$ ($\mathbf{I}_i'$), $i = 1, 2, ..., N$. При использовании двухэтапной схемы и расширенного пространства признаков согласно способу последовательной передачи данных на первом этапе на $p_i$ передаются фрагменты данных $\mathbf{I}_i'$, а на втором, после окончания обработки, — фрагменты данных $\mathbf{I}_i$. При многоядерной архитектуре $p_i$ осуществляется дополнительное разделение фрагмента $\mathbf{I}_i = \{\mathbf{I}_{ij}, j = 2, ..., n\}$ ($\mathbf{I}_i' = \{\mathbf{I}_{ij}', j = 2, ..., n\}$) для SMP-обработки фрагментов $\mathbf{I}_{ij}$ ($\mathbf{I}_{ij}'$) на каждом из $n$ ядер узла $p_i$. Результатом обработки этапа являются фрагменты тематической карты $R_i'$, $i = 1, 2, ..., N$.

*Этап 6.* Фрагменты $R_i'$ пересылаются на $p_0$, где происходит их компоновка в единую тематическую карту $\mathbf{T} = \{R_i', i = 1, 2, ..., N\}$, а также оценка $\mathbf{T}$ критерием точности *каппа-индекс согласия* (КИС), вычисляемым путем сравнения по матрице ошибок эталонного и полученного в результате классификации изображений [7].

Таблица 1

**Характеристики вычислительных кластеров**

| Номер конфигурации | Число ВУ | Сеть | Характеристики ВУ | | |
|---|---|---|---|---|---|
| | | | Процессор | ОП | ОС |
| 1 | 1 | — | AMD Athlon X2 Core dual 3800+ | 1,6 Гбайт | MS Windows XP |
| 2 | 1—9 | Fast Etherhet, 100 Мбит/с | Intel Celeron 2,66 ГГц | 512 Мбайт | MS Windows XP |
| 3 | 1—24 | Gigabit Etherhet, 1000 Мбит/с | 2 × Intel XEON 5150 (2 × 2 ядра) | 8 Гбайт | Novell SLES 10 |

Таблица 2

**Характеристики модельных данных**

| Номер изображения | Размеры, пикс. | Размерность $Z$ | Число классов | Объем, Мбайт | Распределение |
|---|---|---|---|---|---|
| 1/2 | 128 × 128 | 3 | 5 | 0,05 | Унимод., бимод. |
| 3/4 | 128 × 128 | 5 | 5 | 0,08 | Унимод., бимод. |
| 5/6 | 128 × 128 | 7 | 5 | 0,115 | Унимод., бимод. |
| 7/8 | 128 × 128 | 50 | 5 | 0,826 | Унимод., бимод. |
| 9/10 | 128 × 128 | 100 | 5 | 1,61 | Унимод., бимод. |
| 11/12 | 128 × 128 | 200 | 5 | 3,22 | Унимод., бимод. |
| 13 | 3000 × 3000 | 10 | 5 | 85,8 | Унимод. |
| 14 | 3000 × 3000 | 20 | 5 | 171,6 | Унимод. |
| 15 | 3000 × 3000 | 100 | 5 | 858 | Унимод. |

### Исследования

Для оценки работоспособности и эффективности предложенная технология распределенной классификации данных ДЗЗ реализована в среде программирования *Microsoft Visual Studio 2005* с использованием стандарта параллельного программирования MPI (реализация MPICH 2.0), применимого в ОС семейств Windows и Unix [1, 5].

**Постановка задачи исследования.** Определим ключевые параметры технологии распределенной обработки данных ДЗЗ:

- тип классификатора для данных с различными характеристиками;
- производительность кластеров различной конфигурации (недорогие пользовательские ПК, объединенные в локальной вычислительной сети невысокой пропускной способности; суперкомпьютер петафлопной производительности; различное число $N$ доступных ВУ, в том числе многоядерной архитектуры);
- условное ускорение $S_N$ и эффективность $E_N$ параллельной классификации данных ДЗЗ на кластерах различной конфигурации [5].

Основные характеристики ПК и параметров вычислительных кластеров приведены в табл. 1.

Для проведения экспериментов с возможностью варьирования в широких пределах параметров обрабатываемых многоканальных данных использован оригинальный генератор модельных многоканальных данных с уни- и бимодальным законами распределения признаков и заданными границами классов, разработанный с помощью системы *Matlab 2007* (*The MathWorks*) [17]. Характеристики модельных данных приведены в табл. 2 (обозначение "унимод." — распределение признаков в классах изображений, близкое к гауссовскому, "бимод." — распределение с двумя явно выраженными модами).

В соответствии с моделью обработки SPMD данные каждого ВУ обрабатываются независимо. Поэтому для определения наиболее точного и вычислительно эффективного классификатора (т. е. требующего наименьшее время $t$ для обработки одного объема данных) на первом этапе достаточно провести исследования на отдельном ВУ без учета распараллеливания с использованием различных классификаторов и небольших тестовых изображений с варьируемыми в широких пределах характеристиками (табл. 1, конфиг. 1; табл. 2, изобр. 1÷12), а на втором этапе выполнить оценку распределенной обработки многоканальных данных значительного объема на кластерах различной конфигурации (табл. 1, конфиг. 2, 3; табл. 2, изобр. 13—15).

**Результаты экспериментов.** Оценка точности и вычислительной эффективности классификаторов при различных законах распределения данных (рис. 1) показала наиболее высокие и устойчивые результаты классифика-

тора *SVM*, что совпадает с результатами исследований, полученными другими авторами для случаев последовательной классификации [15, 18, 19].

С увеличением размерности признакового пространства точность классификатора *SVM* возрастает, что объясняется увеличивающейся разделимостью классов. Сравнение вычислительной эффективности классификаторов (рис. 2, *а*) показывает существенное преимущество *SVM* перед *k-NN*, лишь немного уступая более простым классификаторам *Hypermin*, *Hypersam* и *ML*. Следует отметить, что классификатор *ML* использовать в экспериментах при $Z > 7$ затруднительно, так как при этом необходимо оперировать громоздкой ковариационной матрицей размерности $Z$ [7, 10].

Учитывая отмеченное выше преимущество непараметрических классификаторов без оценки плотности, заключающееся в меньших требованиях к пропускной способности каналов связи кластера, на втором этапе исследований приведем результаты исключительно для *SVM*.

Оценка эффективности многоядерных ВУ показывает, что их наиболее рациональное использование достигается при одинаковом числе ядер и процессов. Меньшее число процессов не позволяет использовать возможности многоядерной архитектуры ВУ, большее — ведет к

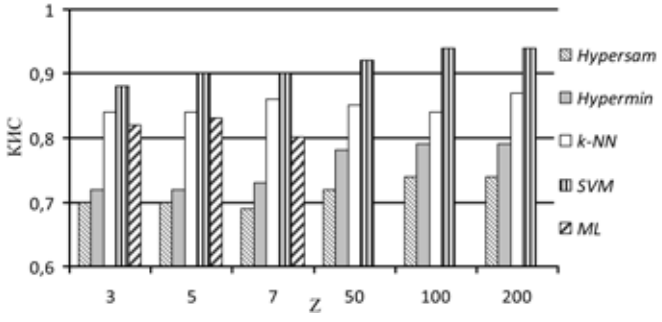



**Рис. 1. Точность классификаторов различных типов на данных различной размерности при бимодальном распределении**

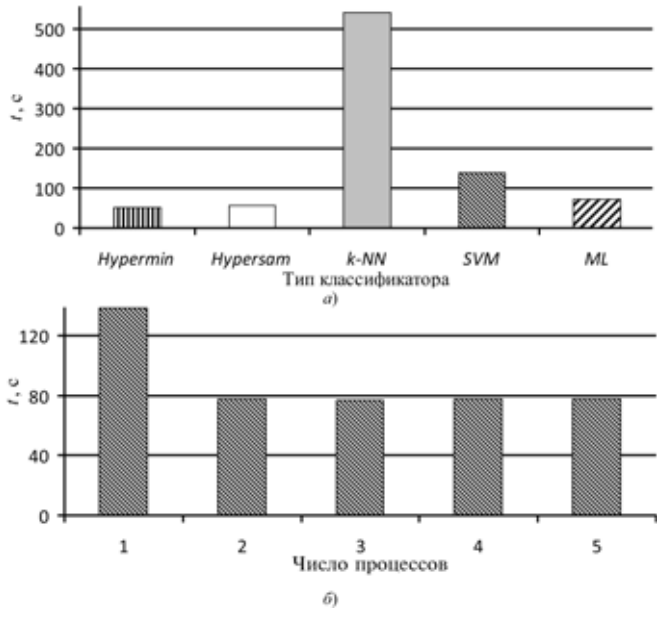



**Рис. 2. Вычислительная эффективность классификации (табл. 2, изобр. 13):** *а* — для одноядерного ВУ при различных типах классификаторов; *б* — для двухъядерного ВУ при различном числе параллельных процессов в ОС

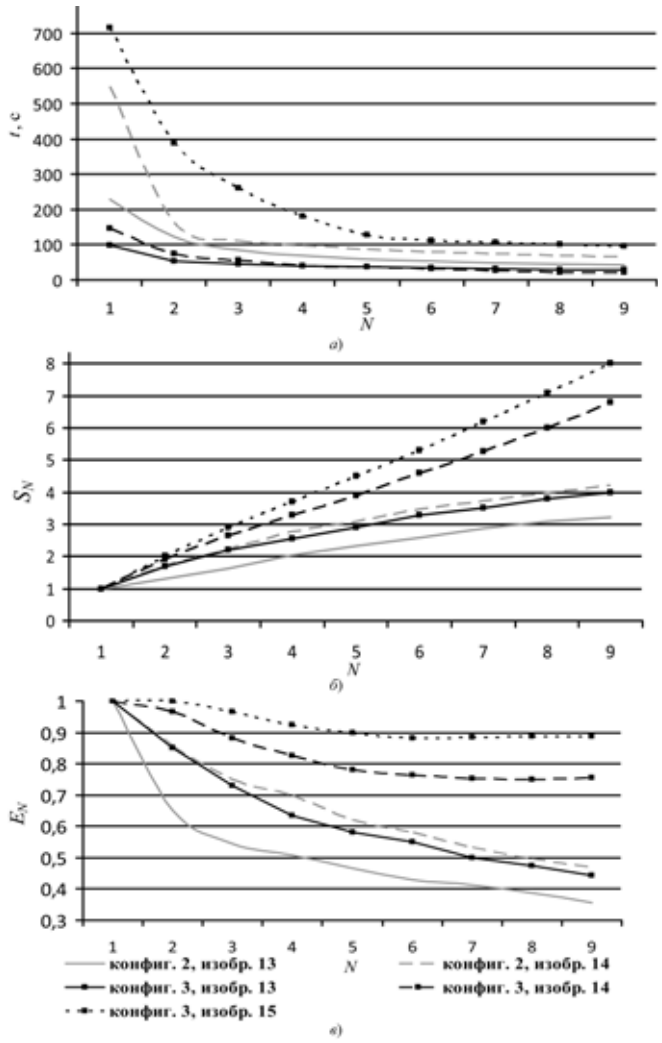



**Рис. 3. Результаты оценки распределенной классификации многоканальных данных:**
*а* — производительность; *б* — параллельное ускорение; *в* — параллельная эффективность

увеличению накладных расходов диспетчера задач ОС ВУ, что снижает пиковую производительность кластера (рис. 2, *б*), большее — ведет к увеличению накладных расходов диспетчера задач ОС ВУ, что снижает пиковую производительность кластера.

Анализ результатов экспериментов для различных конфигураций вычислительного кластера показывает, что при увеличении $N$ от 1 до 4—6 производительность возрастает в несколько раз, а при $N > 6$ этот рост практически прекращается (рис. 3, *а*). Результаты последовательной (см. рис. 2, *а*) и распределенной (рис. 3, *а*) *SVM*-классификации показывают сравнимое увеличение производительности для кластеров обеих конфигураций (в 2,5—3 раза при $N \approx 4$—6). При $N < 4$ видно большее (в 2,0—3,5 раза), а при $N > 5$ меньшее превосходство (в 1,5—2,5 раза) суперкомпьютера над кластером из ПК вследствие увеличивающегося влияния накладных расходов параллельной обработки данных ОС суперкомпьютера. Поэтому, определив достаточное $N$, можно снизить влияние накладных расходов параллельной обработки и увеличить степень рационального использования оборудования кластера. Если характеристики данных ДЗЗ и кластера близки к использованным в экс-

периментах (см. табл. 1, 2), то в предложенной выше технологии на этапе 4 целесообразно принять $N = 6$.

Следует отметить, что суперкомпьютерный кластер более подходит для работы с большими сценами гиперспектральных АИ (рис. 3, конфиг. 3, изобр. 15), так как он обычно обладает более существенными характеристиками доступного объема ОП $p_0$ и $p_i$ (недостаток ОП затрудняет выполнение этапов 1—4 технологии для изобр. 15 в конфиг. 2).

Из результатов оценки параллельной обработки данных ДЗЗ видно, что при увеличении объема данных показатели $S_N$ и $E_N$ также увеличиваются, приближаясь для случая с суперкомпьютером к идеальным ($S_N \approx N$, $E_N \approx 1$, рис. 3, *б*, *в*). Оценка $E_N$ для данных с различными характеристиками также показывает превосходство суперкомпьютерного кластера над кластером из ПК и подтверждает существенное снижение эффективности при $N > 6$ в обеих конфигурациях кластера (рис. 3, *в*).

## Заключение

Растущие объемы разновременной мульти- и гиперспектральной аэрокосмической информации требуют развития технологий ее автоматизированной обработки высокой производительности. На решение этой проблемы направлена предложенная обобщенная технология распределенной классификации данных ДЗЗ, реализуемая на дорогостоящих суперкомпьютерах и кластерах из недорогих ПК в локальной сети, которая учитывает особенности классификации многоканальных данных при использовании классификаторов с линейным разделением и оценкой условной плотности распределения, а также при применении расширенного признакового пространства.

Предложенная технология апробирована на модельных данных с варьируемыми в широких пределах характеристиками. Показано, что при параллельной обработке значительного объема обучающих данных с произвольным распределением следует использовать непараметрические классификаторы с линейным разделением. Для случая двухэтапной классификации с использованием текстурных признаков предложен способ последовательной передачи данных на вычислительные узлы кластера, позволяющий более рационально использовать оперативную память и увеличить производительность распределенной обработки.

На практике подтверждено, что при многоядерной архитектуре вычислительного узла наиболее целесообразно в его операционной системе использовать число процессов, равное числу доступных ядер.

Оценка параллельного ускорения и параллельной эффективности показывает значительное преимущество использованного суперкомпьютера при классификации аэрокосмических изображений объемом в сотни мегабайтов, в то время как на данных меньшего объема производительность кластера недорогих ПК и дорогостоящего суперкомпьютера сопоставимы.

## Список литературы


1. **Plaza A. J., Chang Ch.-I.** High Performance Computing in Remote Sensing. Chapman & Hall/CRC. 2008. 496 p.
2. **Jimenez L. O., Landgrebe D. A.** Supervised classification in high-dimensional space: geometrical, statistical, and asymptotical properties of multivariate data // IEEE Trans. Syst. Man Cybernet. Part C 28 (1). 1998. P. 39—54.
3. **Kalluri S., JaJa J., Bader D. A., Zhang Z., Townshend J., Fallah-Adl H.** High performance computing algorithms for land cover dynamics using remote sensing data // Int. J. Remote Sensing. 2000. Vol. 21. N 6 & 7. P. 1513—1536.
4 **Бучнев А. А., Пяткин В. П., Русин Е. В.** Распределенная высокопроизводительная обработка данных дистанционного зондирования Земли // Исследование Земли из космоса. 2007. № 4. С. 34—38.
5. **Воеводин В. В., Воеводин Вл. В.** Параллельные вычисления. СПб.: БХВ-Петербург. 2004. 608 с.
6. **Ефимов С. С.** Обзор методов распараллеливания алгоритмов решения некоторых задач вычислительной дискретной математики // Математические структуры и моделирование. 2007. № 17. С. 72—93.
7. **Richards J. A., Xiuping Jia.** Remote Sensing Digital Image Analysis: An Introduction. Berlin: Springer. 1999. 363 p.
8. **Замятин А. В., Марков Н. Г.** Непараметрическая классификация аэрокосмических изображений с использованием набора текстурных признаков // Исследование Земли из космоса. 2006. № 1. С. 25—34.
9. **Лапко А. В., Ченцов С. В.** Непараметрические системы обработки информации: Учеб. пос. М.: Наука. 2000. 350 с.
10. **Фукунага К.** Введение в статистическую теорию распознавания образов / Пер. с англ. М.: Наука. 1979. 368 с.
11. **Харалик Р. М.** Статистические и структурные подходы к описанию текстур // ТИИЭР. 1979. Т. 67. № 5. С. 38—45.
12. **Шапиро Е. И.** Непараметрические оценки плотности вероятности в задачах обработки результатов наблюдений // Зарубежная радиоэлектроника. 2000. № 2. С. 3—22.
13. **Support** vector machines: Theory and applications / Eds. L. Wang. 2005. 434 p.
14. **Bishop C. M.** Neural Networks for Pattern Recognition. Oxford Univ. Press. 1995. 508 p.
15. **Ермаков С. Г.** Метод устранения необходимости переключения вычислительных узлов при организации параллельной обработки информации // Информационные технологии. 2007. № 10. С. 65—68.
16. **Замятин А. В., Марков Н. Г.** Анализ динамики земной поверхности с использованием данных дистанционного зондирования Земли. М.: Физматлит. 2007. 176 с.
17. **The MathWorks** — MATLAB and Simulink for Technical Computing. — http://www.mathworks.com (1.09.2009)
18. **Camps-Vails G., Gómez-Chova L., Calpe J., Soria E., Martín J. D., Alonso L., Moreno J.** Robust support vector method for hyperspectral data classification and knowledge discovery // IEEE Trans. Geosci. Remote Sens. 2004. Vol. 42. N. 7. P. 1530—1542.
19. **Melgani F., Bruzzone L.** Classification of hyperspectral remote sensing images with support vector machines // IEEE Trans. Geosci. Remote Sens. 2004. Vol. 42. N. 8. P. 1778—1790.


# CONTENTS

**Адрес редакции:**

107076, Москва, Стромынский пер., 4

Телефон редакции журнала **(499) 269-5510**

E-mail: it@novtex.ru

Дизайнер *Т.Н. Погорелова*. Технический редактор *О. А. Ефремова*.

Корректор *Е. В. Комиссарова*

Сдано в набор 12.04.2010. Подписано в печать 19.05.2010. Формат 60×88 1/8. Бумага офсетная. Печать офсетная.
Усл. печ. л. 9,8. Уч.-изд. л. 11,18. Заказ 465. Цена договорная.

Журнал зарегистрирован в Министерстве Российской Федерации по делам печати,
телерадиовещания и средств массовых коммуникаций.
Свидетельство о регистрации ПИ № 77-15565 от 02 июня 2003 г.

Отпечатано в ООО "Подольская Периодика"
142110, Московская обл., г. Подольск, ул. Кирова, 15