TALLINNA POLÜTEHNILISE
INSTITUUDI TOIMETISED

ТРУДЫ ТАЛЛИНСКОГО
ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

№ 426

ВОПРОСЫ СОЗДАНИЯ СРЕДСТВ ОБРАБОТКИ ИНФОРМАЦИИ (ОБРАБОТКА ДАННЫХ, ПРОГРАММИРОВАНИЕ)

Труды экономического факультета XXVIII

ТАЛЛИН 1977

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED
ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА
№ 426 1977

УДК 519.24
681.142.2
681.3.06:518.5
518.512.25
519.1

ВОПРОСЫ СОЗДАНИЯ СРЕДСТВ ОБРАБОТКИ ИНФОРМАЦИИ (ОБРАБОТКА ДАННЫХ, ПРОГРАММИРОВАНИЕ)

Труды экономического факультета XXVIII

Таллин 1977

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED
ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА
№ 426 1977

УДК 519.24

Л.К.Выханду, Х.О.Крусберг

О НЕЛИНЕЙНОМ ФАКТОРНОМ АНАЛИЗЕ

В факторном анализе [I] применяется главным образом линейная модель

$$Z = AF, \qquad (I)$$

где $Z_{m \times n}$ – матрица данных наблюдений;
m – число параметров;
n – число объектов;
$A_{m \times m}$ – матрица (в общем случае) факторных нагрузок;
$F_{m \times n}$ – матрица факторных значений.

На возможность отхода от гипотезы линейности указал уже Т.Бартлетт [2]. Конкретные модели предложены Р. Макдональдом [3] и им же развито начало теории нелинейного факторного анализа в статьях [4,5].

Нами проверяется обоснованность введенных Макдональдом требований к факторной модели и предлагается некоторое ослабление их.

Один из возможных подходов к нелинейному случаю следующий. Из матрицы A выбираем любые три линейно независимых столбца

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & a_{m3} \end{pmatrix}.$$

В этом случае матрица факторных значений должна иметь вид

$$F = \begin{pmatrix} f_{11} & f_{12} & \cdots & f_{1n} \\ f_{21} & f_{22} & \cdots & f_{2n} \\ f_{31} & f_{32} & \cdots & f_{3n} \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix}.$$

Как известно, для матрицы F выполнены условия ортогональности и нормированности [I]:

$$E(f_p) = 0,$$
$$E(f_p f_q) = 0, \quad p \neq q, \tag{2}$$
$$E(f_p^2) = 1,$$

(где E оператор математического ожидания).

Пусть имеется ортогональная 3x3 – матрица T. Введением обозначений

$$B = AT, \quad G = T'F \tag{3}$$

можем модели (I) придать следующую форму

$$Z = BG.$$

Выбираем по предложению Макдональда [5] матрицу T так, чтобы матрица G имела вид

$$G = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ x_{11}x_{21} & x_{12}x_{22} & \cdots & x_{1n}x_{2n} \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ x_1x_2 \end{pmatrix}. \tag{4}$$

Пусть строки матрицы G ортогональные и нормированные, то есть выполнены следующие отношения

$$\begin{aligned} E(x_1 x_2) &= E(x_1)E(x_2) = 0, \\ E(x_1 \cdot x_1 x_2) &= E(x_1^2)E(x_2) = 0, \\ E(x_2 \cdot x_1 x_2) &= E(x_1)E(x_2^2) = 0, \\ E\left[(x_1 x_2)^2\right] &= E(x_1^2)E(x_2^2) = 1. \end{aligned} \tag{5}$$

При определении матрицы T можно исходить из геометрических соображений и провести вращение координатных осей поочередно на углы φ_{12}, φ_{13} и φ_{23} в факторном пространстве (f_1, f_2, f_3):

$$T_{12} = \begin{pmatrix} c_{12} & -s_{12} & 0 \\ s_{12} & c_{12} & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad T_{13} = \begin{pmatrix} c_{13} & 0 & -s_{13} \\ 0 & 1 & 0 \\ s_{13} & 0 & c_{13} \end{pmatrix},$$

$$T_{23} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_{23} & -s_{23} \\ 0 & s_{23} & c_{23} \end{pmatrix},$$

где ради простоты использованы обозначения

$$c = \cos\varphi, \quad s = \sin\varphi.$$

Тогда матрица T получается как произведение

$$T = T_{12} T_{13} T_{23}$$

и формула (3) принимает вид

$$G = T'_{23}\, T'_{13}\, T'_{12}\, F.$$

Р.Макдональд в статье [5] выводит две альтернативные группы формул для вывода угла вращения. Мы вначале следуем его ходу мыслей. Наряду с матрицей (4) рассмотрим матрицу

$$\bar{G} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}. \qquad (6)$$

Вычислим на плоскости (f_1, f_2) произведение

$$\bar{G}_{12} = T'_{12} F = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} c_{12} & s_{12} & 0 \\ -s_{12} & c_{12} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix} =$$

$$= \begin{pmatrix} c_{12} f_1 + s_{12} f_2 \\ -s_{12} f_1 + c_{12} f_2 \\ f_3 \end{pmatrix} \qquad (7)$$

и составим выражение

$$\Phi_{12} = E\left[(x_3 - x_1 x_2)^2\right] = E(x_3^2) + E(x_1^2 x_2^2) - 2E(x_1 x_2 x_3) =$$
$$= 2 - 2E(x_1 x_2 x_3) =$$
$$= 2 - 2 s_{12} c_{12} E(f_2^2 f_3 - f_1^2 f_3) - 2(c_{12}^2 - s_{12}^2) E(f_1 f_2 f_3),$$

так как по (5) $E(x_1^2 x_2^2) = 1$. Кроме того Макдональд предполагает выполнение равенства $E(x_3^2) = 1$.

Нас интересует минимальное значение полученного выражения Φ_{12}, тогда x_1 и x_2 лучше всего определяют x_3 как произведение $x_1 x_2$. После несложных вычислений подходящий угол вращения определяется через

$$\tan 2\varphi_{12} = \frac{E(f_2^2 f_3) - E(f_1^2 f_3)}{2E(f_1 f_2 f_3)}.$$

Аналогичные вычисления дают для других плоскостей углы поворота

$$\tan 2\varphi_{13} = \frac{E(f_2 f_3^2) - E(f_1^2 f_2)}{2E(f_1 f_2 f_3)}$$

и

$$\tan 2\varphi_{23} = \frac{E(f_1 f_3^2) - E(f_1 f_2^2)}{2E(f_1 f_2 f_3)}.$$

Эти три угла и определяют преобразование Макдональда. Так как Макдональд использует условие $E(x_3^2) = 1$, которое автоматически не выполняется, то нами предпринята попытка исследовать этот вопрос глубже.

Если матрица $\bar{G}$ должна иметь вид (6), то требования (2) ортогональности и нормированности, подробно выписанные,

следующие:

$$E(x_1)=0,\quad E(x_2)=0,\quad E(x_3)=0,$$
$$E(x_1x_2)=0,\quad E(x_1x_3)=0,\quad E(x_2x_3)=0,$$
$$E(x_1^2)=1,\quad E(x_2^2)=1,\quad E(x_3^2)=1.$$

Легко проверить, что эти требования выполнены при каждом повороте осей. Например, на плоскости (f_1,f_2) нужный результат получается при помощи формулы (7), если учитывать условия (2) для матрицы F.

Если дополнительно потребовать для каждого столбца, чтобы

$$x_{3j}=x_{1j}x_{2j},$$

то прибавляются еще условия

$$\begin{aligned} &E(x_1x_3)=E(x_1^2x_2)=0,\\ &E(x_2x_3)=E(x_1x_2^2)=0,\\ &E(x_3^2)=E(x_1^2x_2^2)=1. \end{aligned} \tag{8}$$

Эти условия выполняются необязательно точно, их невозможно проверить при помощи формул (7) на плоскости (f_1,f_2). Поэтому исследуем их подробнее.

Ради простоты введем следующие обозначения:

$$\begin{aligned} &K_1=E(f_1^3), && K_2=E(f_2^3), && K_3=E(f_1^2f_2),\\ &K_4=E(f_1^2f_3), && K_5=E(f_1^2f_2^2), && K_6=E(f_1f_2^2),\\ &K_7=E(f_2f_3^2), && K_8=E(f_2^2f_3^2), && K_9=E(f_1f_3^2),\\ &L_1=E(f_1^2f_3^2), && L_2=E(f_2^2f_3), && L_3=E[(f_2^2-f_1^2)^2],\\ &L_4=E(f_1f_2f_3), && L_5=E[f_1f_2(f_2^2-f_1^2)], && L_6=E(f_1f_2^2f_3),\\ & && L_7=E(f_1^2f_2f_3). \end{aligned} \tag{9}$$

Выпишем первое требование из (8) при повороте на плоскости $(f_1\ f_2)$:

$$\begin{aligned} E(x_1^2x_2)&=E[(c_{12}^2f_1^2+2c_{12}s_{12}f_1f_2+s_{12}^2f_2^2)(-s_{12}f_1+c_{12}f_2)]=\\ &=s_{12}c_{12}^2E(2f_1f_2^2-f_1^3)+c_{12}^3E(f_1^2f_2)-s_{12}^3E(f_1f_2^2)+\\ &+s_{12}^2c_{12}E(f_2^3)-2s_{12}^2c_{12}E(f_1^2f_2)=0. \end{aligned}$$

Разделим полученное уравнение на $\cos^3\varphi_{12}$, учитывая обозначения (9). В результате получается соотношение

$$K_6\tan^3\varphi_{12}+(2K_3-K_2)\tan^2\varphi_{12}+(K_1-2K_6)\tan\varphi_{12}-K_3=0.$$

Следующие два условия из (8) дают соответственно уравнения

$$K_3\tan^3\varphi_{12}+(K_1-2K_6)\tan^2\varphi_{12}+(K_2-2K_3)\tan\varphi_{12}+K_6=0,$$
$$(L_3-4)\tan^2 2\varphi_{12}+4L_5\tan 2\varphi_{12}+4(K_5-1)=0.$$

Если провести замену $\tan\varphi_{12}=t$ **(тогда** $\tan 2\varphi_{12}=\frac{2t}{1-t^2}$**), получим систему**

$$z_1(t)=K_6t^3+(2K_3-K_2)t^2+(K_1-2K_6)t-K_3=0$$
$$z_2(t)=K_3t^3+(K_1-2K_6)t^2+(K_2-2K_3)t+K_6=0$$
$$z_3(t)=(K_5-1)t^4-2L_5t^3+(L_3-2K_5-2)t^2+2L_5t+(K_5-1)=0.$$

Так как не существует угла φ_{12}**, тангенс которого удовлетворял бы одновременно всем трем уравнениям, будем пользоваться методом наименьших квадратов. Для минимизации выражения**

$$z_1^2(t)+z_2^2(t)+z_3^2(t)$$

или

$$\Phi_{12}=Q_1t^8-4Q_2t^7+Q_3t^6+2Q_4t^5+Q_5t^4+4Q_6t^3+$$
$$+Q_7t^2+2Q_8t+Q_9, \qquad (10)$$

где

$$Q_1=(K_5-1)^2,$$
$$Q_2=L_5(K_5-1),$$
$$Q_3=K_3^2+K_6^2+4L_5^2+2L_3(K_5-1)-4(K_5-1)(K_5+1),$$
$$Q_4=K_1K_3-K_2K_6+6K_5L_5-2L_5(L_3-1),$$
$$Q_5=K_1^2+K_2^2+6K_5^2+L_3^2-8L_5^2-2K_1K_6-2K_2K_3-$$
$$-4L_3-4K_5(L_3-1)+6.$$
$$Q_6=L_5[(L_3-1)-3K_5],$$
$$Q_7=K_1^2+K_2^2+4L_5^2-2K_1K_6-2K_2K_3+2L_3(K_5-1)-$$
$$-4(K_5-1)(K_5+1),$$
$$Q_8=K_2K_6-K_1K_3+2L_5(K_5-1),$$
$$Q_9=K_3^2+K_6^2+(K_5-1)^2,$$

дифференцируем по t **и получаем уравнение**

$$4Q_1t^7-14Q_2t^6+3Q_3t^5+5Q_4t^4+2Q_5t^3+6Q_6t^2+Q_7t+Q_8=0.$$

Уравнение нечетного порядка, следовательно, имеется по меньшей мере одно действительное решение.

Решение t **, при котором выражение (10) имеет наименьшее значение, и определяет угол поворота** φ_{12}.

Дальше делаем поворот на плоскости (f_1,f_3):

$$\bar{G}_{13} = T'_{13}\,\bar{G}_{12} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} c_{13} & 0 & s_{13} \\ 0 & 1 & 0 \\ -s_{13} & 0 & c_{13} \end{pmatrix} \begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix} = \begin{pmatrix} c_{13}f_1 + s_{13}f_3 \\ f_2 \\ -s_{13}f_1 + c_{13}f_3 \end{pmatrix}$$

и требования (8) теперь следующие:

$$K_7 t^2 + 2L_4 t + K_3 = 0,$$
$$L_2 t + K_6 = 0,$$
$$(K_8 - 1)t^2 + 2L_6 t + (K_5 - 1) = 0.$$

При помощи метода наименьших квадратов получим сперва выражение

$$\Phi_{13} = Q_1 t^4 + 4Q_2 t^3 + Q_3 t^2 + 2Q_4 t + Q_5, \qquad (11)$$

где $t = \tan\varphi_{13}$

и

$$Q_1 = K_7^2 + (K_8 - 1)^2,$$
$$Q_2 = K_7 L_4 + L_6(K_8 - 1),$$
$$Q_3 = L_2^2 + 4L_4^2 + 4L_6^2 + 2K_3 K_7 + 2(K_8 - 1)(K_5 - 1),$$
$$Q_4 = 2L_4 K_3 + L_2 K_6 + 2L_6(K_5 - 1),$$
$$Q_5 = K_3^2 + K_6^2 + (K_5 - 1)^2,$$

а потом уравнение для определения угла φ_{13}:

$$2Q_1 t^3 + 6Q_2 t^2 + Q_3 t + Q_4 = 0. \qquad (12)$$

При повороте на плоскости (f_2, f_3) выражение для Φ_{23}:

$$\Phi_{23} = Q_1 t^4 + 4Q_2 t^3 + Q_3 t^2 + 2Q_4 t + Q_5,$$

где

$$t = \tan\varphi_{23},$$
$$Q_1 = K_9^2 + (L_4 - 1)^2,$$
$$Q_2 = K_9 L_4 + L_7(L_1 - 1),$$
$$Q_3 = K_4^2 + 4L_4^2 + 4L_7^2 + 2K_6 K_9 + 2(L_1 - 1)(K_5 - 1),$$
$$Q_4 = 2L_4 K_6 + K_3 K_4 + 2L_7(K_5 - 1),$$
$$Q_5 = K_3^2 + K_6^2 + (K_5 - 1)^2.$$

Уравнение для определения угла поворота φ_{23} получается аналогично предыдущему случаю:

$$2Q_1 t^3 + 6Q_2 t^2 + Q_3 t + Q_4 = 0.$$

Опыт практического использования изложенных в статье методов рассматривается в отдельной статье.

Литература

I. Харман Г. Современный факторный анализ. М., 1972.

2. B a r t l e t t, M.S. Factor analysis in psychology as a statistician sees it. - Uppsala Symposium on Psychological Factor Analysis. Uppsala, 1953.

3. M c D o n a l d, R.P. Numerical methods for polynomical models in nonlinear factor analysis. - "Psychometrika", 1967, 32.

4. M c D o n a l d, R.P. A general approach to nonlinear factor analysis. - "Psychometrika", 1962, 4.

5. M c D o n a l d, R.P. Factor interaction in nonlinear factor analysis. - "Brit. J. Math. and Statist. Psychol.", 1967, 20.

L. Vyhandu, H. Krusberg

About Nonlinear Factor Analysis

Summary

In this paper the results of MacDonald [5] in the field of nonlinear factor analysis are summarized and some of his conditions concerning factor scores are reconsidered.

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED
ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

№ 426 1977

УДК 519.24

Л.К.Вяханду, Х.О.Крусберг

МЕТОД ПРЯМОГО ФАКТОРНОГО АНАЛИЗА

В многопараметрической статистике имеются различные методы факторного анализа [2,3]. В данной статье рассматривается новый метод получения факторной матрицы непосредственно из корреляционной матрицы. Доказана связь полученной факторной модели с методом главных компонент.

I. Описание метода

Дана симметрическая корреляционная матрица параметров размером $m \times m$

$$R = \begin{pmatrix} 1 & r_{12} & \cdots & r_{1m} \\ r_{21} & 1 & \cdots & r_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ r_{m1} & r_{m2} & \cdots & 1 \end{pmatrix}.$$

Вычислим для каждого столбца величину

$$\rho_j = \frac{\sum_{i=1}^{m} r_{ij}^2}{r_{jj}}. \qquad (I.1)$$

Наибольшая из них $\rho^{(k)} = \max_j \rho_j$

определяет номер "важнейшего" в среднем параметра. Индекс k указывает номер итерации ($k = 1, \ldots, m$). Корреляционные коэффициенты этого параметра (столбец матрицы) делим на квадратный корень диагонального элемента r_{jj} матрицы R. Полученные числа составляют первый столбец b_1 новой факторной матрицы B. Величина

$$\frac{1}{m}\rho^{(k)}$$

указывает на удельный вес параметра k в общем разложении. Если вместо корреляционной матрицы имеется ковариационная, то нужно дополнительно делить ее на след матрицы.

Дальше составим остаточную корреляционную матрицу

$$ {}_1R = R - b_1 b_1' $$

и повторим процесс с нею. Для полного разложения в общем случае нужно сделать m шагов. (Если ранг матрицы R равен r, $r \leqslant m$, то требуется r шагов метода).

Для иллюстрации хода вышеописанных вычислений приведем два примера.

Пример I.I. Дана корреляционная матрица

$$ R = \begin{pmatrix} 1 & \frac{1}{20}\sqrt{2} & \frac{1}{5} \\ \frac{1}{20}\sqrt{2} & 1 & -\frac{1}{5}\sqrt{2} \\ \frac{1}{5} & -\frac{1}{5}\sqrt{2} & 1 \end{pmatrix}. $$

Вычислим значения выражений (I.I)

$$ \rho_1^{(1)} = \left(1 + \frac{1}{200} + \frac{1}{25} \right) : 1 = 1{,}045, $$

$$ \rho_2^{(1)} = \left(\frac{1}{200} + 1 + \frac{2}{25} \right) : 1 = 1{,}085, $$

$$ \rho_3^{(1)} = \left(\frac{1}{25} + \frac{2}{25} + 1 \right) : 1 = 1{,}12. $$

Наибольшим из них является третье, значит

$$ b_1 = \begin{pmatrix} \frac{1}{5} \\ -\frac{1}{5}\sqrt{2} \\ 1 \end{pmatrix} : 1 = \begin{pmatrix} \frac{1}{5} \\ -\frac{1}{5}\sqrt{2} \\ 1 \end{pmatrix}. $$

Вычислим остаточную матрицу

$$ {}_1R = R - b_1 b_1' = \begin{pmatrix} \frac{24}{25} & \frac{9}{100}\sqrt{2} & 0 \\ \frac{9}{100}\sqrt{2} & \frac{23}{25} & 0 \\ 0 & 0 & 0 \end{pmatrix} $$

и оценим ее столбцы:

$$ \rho_1^{(2)} = \left(\frac{576}{625} + \frac{162}{10\,000} \right) : \frac{24}{25} = 0{,}98, $$

$$ \rho_2^{(2)} = \left(\frac{162}{10\,000} + \frac{529}{625} \right) : \frac{23}{25} = 0{,}94. $$

В матрицу B следующим входит первый столбец:

$$b_2 = \begin{pmatrix} \frac{24}{25} \\ \frac{9}{100}\sqrt{2} \\ 0 \end{pmatrix} : \sqrt{\frac{24}{25}} = \begin{pmatrix} \frac{2}{5}\sqrt{6} \\ \frac{3}{40}\sqrt{3} \\ 0 \end{pmatrix}.$$

Последняя итерация дает

$$b_3 = \begin{pmatrix} 0 \\ \frac{17}{40}\sqrt{5} \\ 0 \end{pmatrix}.$$

Таким образом, получается матрица

$$B = \begin{pmatrix} \frac{1}{5} & \frac{2}{5}\sqrt{6} & 0 \\ -\frac{1}{5}\sqrt{2} & \frac{3}{40}\sqrt{3} & \frac{17}{40}\sqrt{5} \\ 1 & 0 & 0 \end{pmatrix}.$$

Вклад каждого из факторов в описание суммарной дисперсии определяется как сумма квадратов факторных нагрузок по столбцу, отнесенная к полной суммарной дисперсии признаков, которая равна m. Следовательно,

$$\rho_3^{(1)} = 1{,}12/3 = 0{,}373; \quad \rho_1^{(2)} = 0{,}98/3 = 0{,}327; \quad \rho_2^{(3)} = 0{,}300.$$

Пример I.2. Пусть в матрице

$$R = \begin{pmatrix} 1 & r_{12} & r_{13} \\ r_{21} & 1 & r_{23} \\ r_{31} & r_{32} & 1 \end{pmatrix} \qquad (I.2)$$

переменные упорядочены в порядке исключения (в смысле данной статьи). Тогда искомая матрица имеет вид

$$B = \begin{pmatrix} 1 & 0 & 0 \\ r_{21} & \sqrt{1-r_{12}^2} & 0 \\ r_{31} & \frac{r_{23}-r_{12}r_{13}}{\sqrt{1-r_{12}^2}} & \sqrt{\kappa} \end{pmatrix}, \qquad (I.3)$$

где

$$\kappa = \frac{(1-r_{12}^2)(1-r_{13}^2)-(r_{23}-r_{12}r_{13})^2}{1-r_{12}^2}.$$

2. Природа матрицы B

В линейной алгебре доказывается [I], что симметрическую положительно определенную $m \times m$ – матрицу R можно представить в виде

$$R = BB', \qquad (2.I)$$

где матрица $B = (b_{ij})$ невырожденная размером $m \times m$.

Матрица коэффициентов корреляции R есть матрица Грама, то есть положительно полуопределенная с рангом $r (r \leq m)$. Следовательно, матрица B в (2.I) должна быть размером $m \times m$. Так как матрица B в этом представлении не однозначно определена, то берется нижней треугольной матрицей R

$$B = \begin{pmatrix} b_{11} & 0 & 0 & \dots \\ b_{21} & b_{22} & 0 & \dots \\ b_{31} & b_{32} & b_{33} & \dots \\ \cdot & \cdot & \cdot & \dots \end{pmatrix},$$

где $b_{ii} \neq 0$, если R невырожденная, т.е. положительно определенная матрица. Если же R положительно полуопределенная, то без уменьшения общности переставляем переменные так, чтобы отличный от нуля минор матрицы R стоял в верхнем углу. Тогда $b_{ii} \neq 0$ при $i \leq r$ и матрица B получается размером $m \times r$.

Для нахождения элементов матрицы B можно использовать уравнения, вытекающие из соотношения (2.I):

$$b_{i1}^2 + b_{i2}^2 + \dots + b_{ii}^2 = 1, \quad i = 1, 2, \dots, m,$$

$$b_{i1} b_{j1} + b_{i2} b_{j2} + \dots + b_{ii} b_{ji} = r_{ij}, \quad i < j.$$

Решение B этой системы можно получить методом квадратного корня [I].

Например, если R есть матрица размером 3x3 (I.2), то получим B в виде (I.3).

Значит, матрицу B, сконструированную по методу первого пункта данной статьи, можно получить и методом квадратного корня.

3. Анализ факторной модели, определенной данным методом

Формула (2.I) показывает, что матрица B удовлетворяет фундаментальной факторной теореме Тэрстоуна [2] и поэтому может рассматриваться факторной матрицей. Составим при помощи нее факторную модель

$$Z = BG, \qquad (3.I)$$

где Z – $m \times n$ – матрица исходных данных;

G – $m \times n$ – матрица факторных значений.

В этой модели член с характерным фактором опущен. В факторном анализе доказывается [2], что

$$R = \frac{1}{n} Z Z'. \qquad (3.2)$$

Для исследования свойств модели (3.I) определим из нее

$$G = (B'B)^{-1} B'Z,$$

откуда, учитывая соотношение (3.2) и (2.I), получим

$$\frac{1}{n} GG' = \frac{1}{n} (B'B)^{-1} B' Z Z' B (B'B)^{-1} = E.$$

Итак, строки матрицы G ортогональны и поэтому факторное пространство ортогонально. Непосредственной проверкой можно убедиться, что столбцы B в (I.3) не ортогональны.

Исследуем дальше, имеется ли связь между моделью (3.I) и моделью главных компонент [2]

$$Z = AF,$$

где

$$\begin{aligned} A A' &= R, \\ A'A &= \Lambda, \\ \frac{1}{n} F F' &= E \end{aligned} \qquad (3.3)$$

(здесь через Λ обозначена диагональная матрица, на главной диагонали которой собственные значения матрицы R).

Для этой цели выбираем невырожденную $m \times m$-матрицу T так, чтобы

$$\begin{aligned} BT &= A, \\ T^{-1} G &= F \end{aligned} \qquad (3.4)$$

Какие свойства должна иметь матрица T? Как ее вычислить?

Для выяснения свойств матрицы T получим сперва из (3.4) матрицу

$$T = (B'B)^{-1} B'A,$$

а потом, учитывая соотношения (3.3) и (2.I),

$$TT' = (B'B)^{-1} B'AA'B(B'B)^{-1} = E,$$

т.е. она ортогональна.

Если матрица R и тем самым матрица B невырожденные, то предыдущие выражения упрощаются.

Значит, при помощи ортогонального преобразования можно из модели (3.I) получить известную модель главных компонент.

Для нахождения матрицы T рассмотрим вопрос сначала в двухмерном пространстве факторов с осями g_1 и g_2. Выпишем модель (3.I) подробнее:

$$z_1 = b_{11} g_1 + b_{12} g_2,$$
$$z_2 = b_{21} g_1 + b_{22} g_2,$$
$$\cdots\cdots\cdots\cdots$$
$$z_n = b_{m1} g_1 + b_{m2} g_2. \qquad (3.5)$$

Коэффициенты этой системы образуют на плоскости точки с координатами

$$(b_{i1}; b_{i2}), \quad i = 1, 2, \ldots, m.$$

Так как матрица поворота осей на плоскости следующая

$$T = \begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix},$$

то результат поворота имеет вид

$$(b_{i1}; b_{i2}) \begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix} = (b_{i1}\cos\varphi + b_{i2}\sin\varphi; -b_{i1}\sin\varphi + b_{i2}\cos\varphi).$$

Для ортогональности коэффициентов полученное выражение должно равняться нулю, откуда

$$\tan 2\varphi = \frac{2b_1 b_2}{b_1^2 - b_2^2}.$$

Например, для 2x2-матрицы

$$\tan 2\varphi = \frac{\sqrt{1 - r_{12}^2}}{r_{12}}$$

или

$$\cos 2\varphi = r_{12}.$$

Элементы матрицы T теперь следующие:

$$\cos\varphi = \sqrt{\frac{1 + r_{12}}{2}}, \quad \sin\varphi = \sqrt{\frac{1 - r_{12}}{2}}.$$

В общем случае метод для нахождения матрицы поворота T указан в книге Х.Хармана [2], а именно: столбцами матрицы T являются собственные векторы матрицы B'B.

4. Пример

В качестве примера приведем в таблице I корреляционную матрицу (ее верхнюю часть) измерений черно-белых телок, взятую из книги Вебера [3].

В таблице 2 приведены для сравнения методов два столбца коэффициентов, полученные по центроидному методу (цифры заимствованы из книги Вебера); два столбца нагрузок, соответствующие наибольшим собственным числам (метод главных компонент, где на главной диагонали корреляционной матрицы единицы), и, наконец, три первые столбца матрицы B, полученные по методу данной статьи.

Таблица I

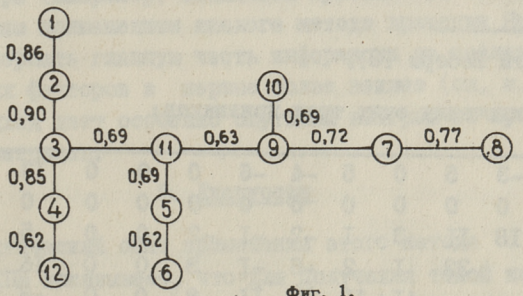
Корреляционная матрица из книги Вебера [3]
(ключ к таблице: 86 $\longrightarrow r_{12} = 0{,}86$)

I,00	86	76	76	58	56	44	50	57	60	63	59
	I,00	90	80	60	52	49	57	58	54	67	46
		I,00	85	62	53	48	56	56	55	69	52
			I,00	58	52	43	54	59	55	67	62
				I,00	62	65	69	68	68	69	42
					I,00	47	53	60	6I	6I	53
						I,00	77	72	56	70	42
							I,00	69	67	68	42
								I,00	69	72	46
									I,00	63	59
										I,00	60
											I,00

Вычисления по предлагаемому методу дали следующий порядок исключения признаков:

II 2 I0 7 I2 6 5 9 8 4 I 3.

Для более подробного анализа представим матрицу R в виде максимального корреляционного пути [4] на фиг. I.



Фиг. 1.

Если проследить порядок включения переменных в матрицу B , то видно, как метод выбирает признаки из разных групп признаков. Сперва исключается признак II, который является как будто центральным, затем - признаки 2 и I0 из разных групп. Интересно проследить остаточную матрицу

Таблица 2

Факторные коэффициенты и нагрузки

Параметр	Центроидное решение[x]		Факторные нагрузки		Первые столбцы матрицы B		
I	0,8II4	0,3785	0,82I8	0,3872	0,63	0,5899	0,I439
2	0,8388	0,3692	0,8399	0,3909	0,67	0,7424	0
3	0,8388	0,3662	0,8422	0,3866	0,69	0,5896	0,0285
4	0,8I83	0,3723	0,8288	0,3970	0,67	0,4730	0,0694
5	0,7955	0,2369	0,8I56	-0,2300	0,69	0,I855	0,2839
6	0,7089	-0,0923	0,7355	-0,0947	0,6I	0,I499	0,2656
7	0,7293	-0,4I85	0,7405	-0,48I3	0,70	0,0283	0,I506
8	0,7806	-0,3292	0,7938	-0,3759	0,68	0,I54I	0,2856
9	0,8050	-0,283I	0,8I98	-0,2995	0,72	0,I3I5	0,2835
I0	0,7795	-0,I723	0,7972	-0,2006	0,63	0,I588	0,7602
II	0,8528	-0,I046	0,8659	-0,I0I3	I,00	0	0
I2	0,6595	0,0954	0,683I	0,I523	0,60	0,078I	0,2626
	Вклад фактора		Собств. числа		Оценки		
	7,4290	I,0305	7,6856	I,2080	5,8467		I,0090
					I,6004		
	%		%		%		
	6I,9	8,6	64,0	I0,I	48,7	I3,3	8,4

[x] Из книги Вебера [3].

после исключения этих трех признаков:

23	0	-3	5	0	5	-4	-6	0	0	0	I3
	0	0	0	0	0	0	0	0	0	0	0
		I8	II	3	I	-2	-I	-2	0	0	5
			32	I	2	-6	-I	3	0	0	I6
				4I	I0	I2	II	8	0	0	-8
					53	0	2	7	0	0	8
						49	25	I7	0	0	-7
							43	I0	0	0	-8
								38	0	0	-6
									0	0	0
										0	0
											56

Для сравнения приводим и остаточную матрицу метода главных компонент:

I7	2	8	7	0	-I	2	-I	I	2	-4	-3
	I4	4	-5	0	-6	6	5	I	-5	-2	-I7
		I4	0	-2	-5	4	4	-I	-4	0	-II
			I6	0	-5	I	3	3	-3	-I	-I
				28	0	-6	-4	-6	-2	-4	-I0
					45	-I2	-9	-3	0	-4	4
						22	0	-3	-I3	I	-I
							23	-7	-4	-5	-6
								24	-2	-2	-5
									32	-8	8
										24	2
											5I

Для сравнения практического значения двух приближений надо напомнить, что метод главных компонент требует знания значений всех признаков. Предложенный метод требует знания значений только признаков II,2,I0. Конечно, результат восстановления матрицы R хуже, чем при методе главных компонент (на это указывают вклады факторов в дисперсию), но вычислительной работы намного меньше.

Можно предположить еще следующий, оправдавший себя в практике прием. Найдем предложенным методом еще один-два фактора (например, исключаем признак 7). После этого однократным применением аналога метода вращения Якоби [5] можем исчерпать главную часть информации из последних более слабых факторов в первые более важные (см. п. 3.5). Такой подход дает особенно ощутимый выигрыш во времени при больших матрицах.

5. Дискуссия

Практический опыт применения этого метода с I974 года в ВЦ ТПИ показывает, что для получения такой же точности (в смысле суммы квадратов компонент), как при методе главных компонент, требуется в среднем вычислить на один фактор больше. Объем вычислений при этом существенно меньше.

На практике наилучшим оказался такой ход вычислений.

Если доля какого-то фактора в общей дисперсии меньше, чем $1/m$, то найдем еще один фактор. Ввиду неортогональности получаемых факторов можно использовать с успехом аналог метода Якоби [5].

С успехом можно применить и модификацию данного метода, где каждый признак получит заданный вес p_i. Тогда правило выбора исключаемого признака примет вид

$$\rho^{(k)} = \max_j \frac{\sum p_i r_{ij}}{r_{jj} \cdot \sum p_i}.$$

В нашей практике использовались веса, обратные коэффициенту вариации признака (более стабильные признаки имеют больший вес) и субъективные веса, которые измеряют стоимость получения значения признака. Последний вариант особенно важен при массовых наблюдениях, где существенное значение имеет сокращение количества измеряемых признаков вместе с возможно "дешевой стоимостью" их получения. После получения взвешенного факторного решения можно легко сравнить это решение с невзвешенным или с решением, полученным методом главных компонент.

Если меньшая стоимость опыта существенно не вредит точности результатов, то взвешенный вариант оправдывает себя полностью. Иногда потеря точности вынуждает проводить более "грубые" измерения показателей.

В факторном анализе и в методе главных компонент трудным вопросом является интерпретация полученных коэффициентов, т.е. описание факторов в терминах наблюденных параметров. Если применить описанный здесь метод, то интерпретация существенно облегчается.

Литература

1. Рао С.Р. Линейные статистические методы и их применение. М., 1968.

2. Харман Г. Современный факторный анализ. М., 1972.

3. Weber, E. Einführung in die Faktorenanalyse. Jena, 1974.

4. Выханду Л.К. Об исследовании многопризнаковых биологических систем. Применение математических методов в биологии. Сб. 3. ЛГУ, 1964, с. 19–22.

5. Вихандуу Л.К. Некоторые проблемы теории анализа данных. "Тр. Таллинск. политехн. ин-та". Обработка информации,№ 366, 1974, с. 3-15.

L. Vyhandu, H. Krusberg

A Direct Factor Analysis Method

Summary

In this paper a factor analysis model Z = BG is considered, where B is defined as a special factoring of the correlation matrix R = BB'.

The connection between the new method and principal component method has been proven.

New method is computationally quick and the results are easy to interpret.

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED
ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

№ 426 1977

УДК 681.142.2

А.О. Вооглайд

РАСШИРЕННЫЙ АНАЛИЗАТОР ПРЕДШЕСТВОВАНИЯ СО СМЕШАННОЙ СТРАТЕГИЕЙ

В настоящей статье мы рассматриваем один метод синтаксического анализа "снизу вверх", который основывается на методе анализа предшествования со смешанной стратегией [5].

У читателя может возникнуть оправданный вопрос: почему рассматривается еще один метод анализа из большого, хорошо известного семейства методов анализа? Этот вопрос оправдан тем более, что согласно Гризу [9], большинство на практике используемых анализаторов сравнимы как по скорости, так и по объему требуемой памяти. Эффективность анализатора зависит больше от качества составления программ, чем от применяемого метода анализа. Так же известно, что анализатор, работающий по методу предшествования и использующий при редуцировании один символ слева от основы, позволяет анализировать все детерминированные языки [13].

1. Потребительский аспект использования системы построения трансляторов

При переходе от теории к практике возникает целый ряд трудностей. Поэтому мы намерены тщательно рассмотреть проблемы, которые возникают при использовании синтаксического анализатора в рамках системы построения трансляторов (СПТ). Мы ставим своей целью доказать, что выбор метода анализа и класса допустимых грамматик зависит не от прихоти отдельных лиц, а от требований пользователя СПТ.

Предположим, что начинаем пользоваться в своей практической работе некоторым анализатором, работающим по од-

ному или другому методу анализа. Каким бы методом анализа мы не пользовались, вначале необходимо контексто-свободную грамматику нашего языка преобразовать в эквивалентную ей грамматику, которая должна принадлежать к классу грамматик, допускаемых используемым методом анализа. Как правило, используемые при этом преобразования значительно изменяют структуру исходного дерева анализа входного текста. Тем самым становится невозможным хотя бы частичная стандартизация семантической обработки дерева анализа, а изменение самого языка требует переработки не одной ранее составленной семантической подпрограммы.

Имея опыт разработки СПТ для трех разных ЭВМ ("Раздан-3", "Минск-32","ЕС-1020"), мы находим, что класс допустимых грамматик и соответствующий анализатор должны удовлетворять следующим требованиям.

Требование I. Класс допустимых грамматик должен быть выбран так, чтобы преобразования из класса грамматик LR (к) в класс данных грамматик сохраняли прежнее синтаксическое дерево [I], т.е. ту часть дерева анализа,в которой каждая вершина несет семантическую информацию.

Класс грамматик LR(к) выбран потому, что тест LR(к) является одним из самых мощных критериев определенности для контексто-свободных грамматик. Если выполнено требование I, мы называем анализаторы, работающие на соответствующих классах грамматик, семантически эквивалентными [I2].

Требование 2. Синтаксический анализатор должен строить синтаксическое дерево, не создавая предварительно дерева анализа.

Для выполнения требования 2 необходимо, чтобы в результате своей работы анализатор выдавал семантически управляемое дерево анализа. Использование такого технического приема гарантирует независимость от преобразований грамматики [I4] и дает значительную экономию машинных ресурсов. Так, например, в работе [6] на странице 634 приведен в виде дерева результат обработки синтаксическим анализатором одной программы на "PICTURE-ALGOL". Указанное дерево состоит из 58 вершин, а использование выше-

указанной техники позволяет уменьшить число вершин этого дерева до 18. Кроме этого следует отметить, что данная методика допускает конструкции языка, которые с помощью только контексто-свободных грамматик сгенерировать невозможно [14].

Требование 3. Синтаксический анализатор должен быть снабжен мощной аппаратурой локализации и исправления ошибок, примерно такой, какая описана в [4].

Требование 4. Синтаксический анализатор должен быть снабжен средствами для отладки создаваемого транслятора, т.е. должна иметься возможность в ходе трансляции программы с объектного языка в какой-то мере изменять синтаксис и семантику этого языка.

Требование 5. Синтаксический анализатор должен проводить первичный семантический анализ, используя при этом сам процесс создания синтаксического дерева, который является своеобразным алгоритмом прохождения дерева. Под первичным семантическим анализом мы понимаем разбиение вершин дерева на классы, причем элементы одного класса могут быть объединены в линейный список. В дальнейшем эти списки, определяя некоторую частичную упорядоченность вершин дерева, позволяют проводить направленную семантическую обработку без прохождения всего дерева. Так, например, в один список могут быть собраны определяющие вхождения переменных (декларации), или, скажем, все составные имена, встречающиеся в программе и т.д. При обработке таких списков можно с большим успехом использовать стандартные семантические подпрограммы [6].

К примеру, рассмотрим реализацию языка переструктурирования файлов [10]. При помощи заданного языка один файл дополняется данными другого или преобразуется в новый. Переносимые единицы определяются списками данных и логическими выражениями. Во время образования дерева синтаксиса составные имена типа ПЛ/I, встречающиеся в программе, распределяются в пять классов эквивалентности следующим образом:

а) определяющие место в первом файле;

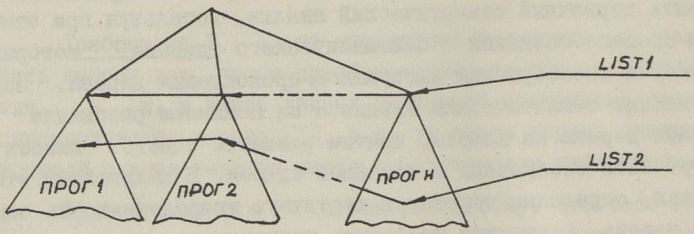
б) определяющие место во втором файле;

в) определяющие места в обоих файлах;

г) составные имена первого файла из логического выражения;

д) составные имена второго файла из логического выражения.

Если предположить, что описания файлов содержат распознаватель составных имен в виде конечного автомата [3] и на входящие в логическое выражение составные имена наложены дополнительные требования в зависимости от того, ссылаются ли они на элементы первого или второго файла, то легко увидеть, что такая большая по объему работа покрывается стандартными подпрограммами, если только имеется соответствующее разбиение и соответствующие списки. При этом можно предположить, что подобные семантические подпрограммы уже имеются в трансляторе с языка запросов [I5].



Фиг. 1.

В качестве второго примера рассмотрим следующий случай. Предположим, что одновременно транслируется несколько разных (но логически связанных)программ на некотором языке программирования и что соответствующее синтаксическое дерево имеет структуру, изображенную на фиг. I.

Здесь список I индуцирует частичную упорядоченность списка 2. Если предположить, что список 2 охватывает декларации переменных, то наличие двух рассматриваемых списков позволяет обрабатывать внутри каждой программы определения и переопределения переменных, учитывать независимость этих программ по определениям от совокупности программ.

Отметим, что последний пример взят из практики (транслятор с языка БЕЙЗИК ЭВМ "WANG-2200" на языке PL/1 DOC EC).

Все вышеперечисленные требования поставлены пользователями, которые проектируют и реализуют с помощью СПТ семантическую обработку конкретных проблемно-ориентированных языков. Особое значение для них имеет требование 5, выполнение которого, как показывают и рассмотренные нами два примера, в значительной степени упрощают и стандартизируют процесс семантического анализа. Именно это требование побудило нас разработать новый класс грамматик предшествования, описываемый в настоящей статье.

Существующая СПТ для ЭВМ "Минск-32" базируется на синтаксическом анализаторе, работающем по методу анализа предшествования со смешанной стратегией, и отвечает большинству вышеприведенным требованиям. Однако оказалось, что пользователи заинтересованы в более мощном механизме разбиения на классы, чем в этой СПТ. А именно - мощность механизма разбиения на классы должна быть равна мощности редуцирования [I2]. Равенство мощностей вызвано тем, что разбиение на классы, по существу, означает разное именование (с помощью понятий грамматики) одинаковых синтаксических единиц в зависимости от того, в каком месте в тексте программы (или сентенциальной форме) они встречаются. Так как мы используем универсальный метод детектирования, то задача разбиения на классы порождает новую проблему редуцирования [I2]. Учитывая понятие семантической эквивалентности (требование I), можно заключить, что мощность разбиения соответствует мощности LR(к).

С помощью указанной СПТ в Таллинском политехническом институте и Тартуском государственном университете создано более двадцати трансляторов с проблемно-ориентированных языков разной сложности и разного объема. Этот опыт показал, что довольно часто требуется мощность разбиения, превышающая мощность LR(к). Обычно таких "критических" мест в реализуемом языке немного, но от того, удается ли для таких конструкций провести стандартное разбиение на классы или нет, во многом зависит простота и эффективность семантического анализа.

Примечательно, что обычно в таких "критических" местах разбиение определяется правым контекстом, который находится за рекурсией (например, за арифметическим или логическим выражением, за списком некоторых элементов и т.д.) - т.е. на неопределенном расстоянии от основы. Отсюда возникает проблема: нельзя ли дополнить анализатор предшествования со смешанной стратегией так, чтобы это поставленное пользователем требование было выполнено.

Оказывается, что эта проблема решается положительно. Идея усовершенствованного метода анализа заключается в следующем. Дерево анализа строится не непрерывно параллельно с разбором входного текста слева направо - в местах, где редуцирование не определено LR (к) контекстом, построение текущего поддерева приостанавливается и начинается построение следующего поддерева, которое уже и устраняет возникшую проблему редуцирования.

В следующем параграфе мы введем новый класс грамматик, допускающих такую технику анализа, и изучим его свойства.

I. Основные понятия

В настоящей статье все используемые понятия определены так, как в статье [I2]. Под теоремой I подразумевается теорема предшествования.

Для редуцирования используемого LR (к) контекст [I6], который обозначим $C_A^{\#k}$

$$C^{\#k}=\left\{(x,y)/\#S\#^k \overset{*}{\Rightarrow} xAyv,\ x\in V_T^{*},\ y,v\in V_T^{*},\ |y|=k\right\}.$$

Кроме того известно, что по любой λ-свободной КС-грамматике G можно построить грамматику предшествования G', такую, что

а) $\mathfrak{L}(G)=\mathfrak{L}(G')$,

в) G' является LR(к) грамматикой, если ею является G [II].

КС-грамматика G называется неопределенной, если в языке L(G) найдется по меньшей мере одна цепочка, порождаемая двумя различными левосторонними выводами (из аксиомы S). КС-грамматика, которая не является неопределенной, называет-

ся определенной. Доказано, что проблема определенности для произвольной КС-грамматики алгоритмически не разрешима [8] также, как проблема LR(к) [I6].

2. Грамматика предшествования, редуцируемая при помощи рекурсивного контекста

При генерации канонического анализа КС-формы X основное внимание падает на редуцирование, так как мы используем универсальный метод детектирования. Опираясь на теорему 9 и определение семантического равенства анализаторов из [I2], мы можем предположить, что базой мы имеем класс грамматик предшествования смешанной стратегии, где для редуцирования используется контекст LR (к). Нашей целью является расширение данного класса грамматик так, чтобы в него входила часть таких грамматик, которые не являются грамматиками предшествования LR(к) при любом к, или они не являются грамматиками предшествования LR(к) при данном фиксированном к. Следуя [I2], введем следующие обозначения.

Пусть (#, к) PK - грамматика предшествования смешанной стратегии, где для редуцирования используется LR(к) контекст. В дальнейшем мы будем пользоваться понятием конфликта редуцирования (КР). Под КР понимается ситуация, где для выполнения редуцирования нужен контекст, т.е. множество правил грамматики содержит по меньшей мере два правила с одинаковой правой частью: $A \to x$ и $B \to x$ ($x \in V^+$, $A \neq B$). Обозначим данный КР через (A, B, x). Пусть данная КСГ $G = (V_N, V_T, P, S)$ и $A, B \in V_N$. Пусть H (A, B) - пересечение LR (к) контекстов нетерминальных символов A и B:

$$H(A, B) = C_A^{\#,k} \cap C_B^{\#,k}.$$

Пусть G - произвольная грамматика предшествования, и к - фиксированное целое число.

Грамматика $G \in (\#,\#)$РК тогда, когда

I) $G \in (\#, K)$РК или

2) $G \in (\#, K)$РК, если для каждого такого КР (A, B, x), где $H(A, B) \neq \emptyset$, существуют КС-грамматики G_1 и G_2, такие, что

а) $\mathfrak{L}(G_1) = \{x \mid S \overset{*}{\Rightarrow} yAx\}$

$\mathfrak{L}(G_2) = \{x \mid S \overset{*}{\Rightarrow} yAx\}$;

б) $\mathfrak{L}(G_1) \cap \mathfrak{L}(G_2) = \emptyset$;

в) $G_1, G_2 \in (\#,\#)\,PK$.

При описании анализатора мы предполагаем, что умеем найти отношения предшествования и по ним детектировать, а также по LR (к) контексту определить соответственные действия. Используем следующие действия:

(1.) Двигаясь слева направо в магазине анализа определяем основу.

(2.) Редуцируем основу.

3. Пропускаем основу.

4. Изменяем направление движения в магазине.

5. Двигаясь в магазине анализа справа налево, определяем основу.

С каждой основой свяжем одно из следующих состояний:

а) конфликта нет;

в) основа редуцируется при помощи LR (к) контекста;

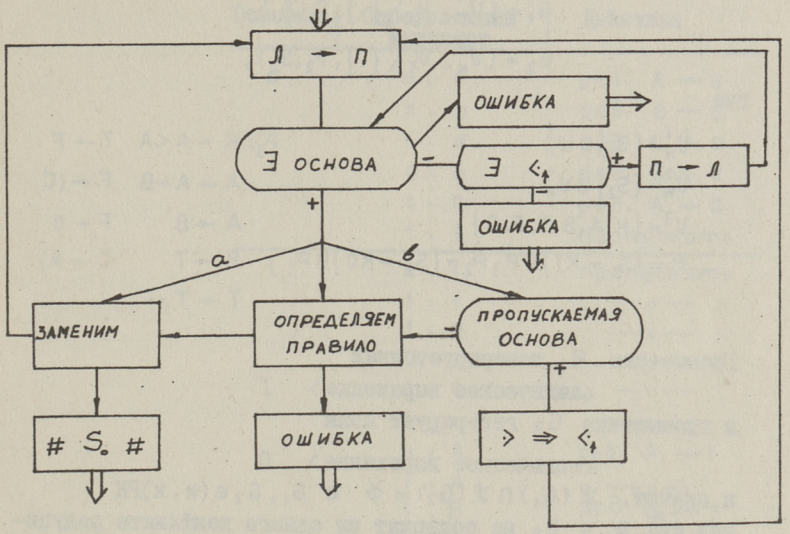
с) основа пропускается (или редуцируется) при помощи LR(к) контекста.

Действие I определяется теорией $\mathfrak{M}$ I и действие 2 производится стандартно [5].

Действие 3 означает, что соответственно LR (к) контексту данная основа пропускается или редуцируется стандартным образом. Если данная основа пропускается, то отношение $\dot{>}$ в конце основы заменяется на отношение $\lessdot_1$, которое при детектировании отождествляют с отношением $\lessdot$. Отношение $\dot{>} = \rho^{+}\alpha\lambda_{T}^{*}$ используем в более общем виде $\dot{>} = \rho^{+}\alpha\lambda^{*}$, так как $\dot{>} \in V \times V$. Отношение $\lessdot_1$ учитывается как элемент контекста, пропускаемая основа элементом контекста не учитывается. Принципиальная схема анализатора приведена на фиг. 2.

Пример I. Рассмотрим такую КС-грамматику, которая генерирует два типа предложений:

I) <составное имя из входа> <логическое выражение> <вход>;



Фиг. 2.

2) <составное имя из выхода> <логическое выражение> <выход>.

Пусть соответствующая грамматика будет следующей:

$G=(\{S_0,S,S_1,S_2,L,L_1,L_2,K,A,B,T,F,C\},\{a,\cdot,I,O,<,+,*,(,)\},P,S_0)$.

$P: S_0 \to \#S\#$	$L \to a$	$A \to A+B$	$F \to (C$
$S \to L_1S_1$	$L \to L.a$	$A \to B$	$F \to a$
$S \to L_2S_2$	$S_1 \to KI$	$B \to T$	$C \to A)$.
$L_1 \to L$	$K \to A<A$	$T \to T*F$	
$L_2 \to L$	$S_2 \to KO$	$T \to F$	

Здесь a обозначает идентификатор, O – операцию выхода, I – операцию входа. Для упрощения примера используем только две арифметические и две логические операции. В данной грамматике имеется конфликт редуцирования (L_1, L_2, L), причем $H(L_1,L_2)\neq\phi$ ни при одном к, так как редуцирование определяется контекстом, который находится за логическим выражением, т.е. за рекурсией. В то же время

$$G_1 = (V_N^1,\ V_T \setminus \{\cdot\},\ P_1, S_1)$$

$$G_2 = (V_N^2,\ V_T \setminus \{\cdot\},\ P_2, S_2),$$

где

$V_N^1 = \{S_1\} \cup V_N^3$

$V_N^2 = \{S_2\} \cup V_N^3$

$V_N^3 = \{K, A, B, T, F, C\}$

$P_1 = \{S_1 \to KI\} \cup P_3 \quad P_2 = \{S_2 \to K0\} \cup P_3$

P_3: $K \to A<A$ $T \to F$; $A \to A+B$ $F \to (C$; $A \to B$ $F \to a$; $B \to T$ $C \to A)$; $T \to T \times F$

Грамматика G_1 генерирует язык

<логическое выражение> I

и грамматика G_2 генерирует язык

<логическое выражение> 0

и, значит, $\mathfrak{L}(G_1) \cap \mathfrak{L}(G_2) = \phi$ и $G_1, G_2 \in (\#, к)$РК так как G_1 и G_2 не содержат ни одного конфликта редуцирования и являются грамматиками предшествования. Следовательно, $G \in (\#, \#)$РК.

Рассмотрим генерацию анализа слова $a \cdot aa + a < a1$. Учитывая простоту примера, покажем только последовательные состояния магазина анализа:

$$\#\lessdot a \gtrdot . \Leftarrow \#\lessdot L \doteq \cdot \doteq a \gtrdot a \Leftarrow \#\lessdot L \gtrdot a \Leftarrow \#\lessdot L \lessdot_1 a$$

$$+ \Leftarrow \#\lessdot L \lessdot_1 F \gtrdot + \Leftarrow \ldots \Leftarrow \#\lessdot L \lessdot_1 A < A \gtrdot I \Leftarrow$$

$$\#\lessdot L \lessdot_1 K = I \gtrdot \# \Leftarrow \#\lessdot L \lessdot_1 S_1 \gtrdot \# \Leftarrow \#\lessdot L \gtrdot S_1 \gtrdot \#$$

$$\Leftarrow \#\lessdot L_1 \doteq S_1 \gtrdot \# \Leftarrow \# S \# .$$

Пример 2. Предлагаем пример, где контекст, определяющий действия в середине слова, находится за рекурсивной частью. Приводим действия, связанные с основой, и контексты, определяющие эти действия.

$$G = (\{S, A, B, C, D, A, B\}\{1, a, b\}, P, S)$$

P: $S \to AC$ $B \to B_1 B1$ $A \to a$

$S \to BD$ $A_1 \to 1$ $B \to a$

$A \to A_1 A1$ $B_1 \to 1$ $C \to a$

$D \to b$

Основа	Определяющий контекст	Действие
a	#, a	ред. A → a
	#, b	ред. B → a
	1, #	ред. C → a
	a, #	ред. C → a
	1, c	ред. A → a
	1, 1	пропускать
1	#, 1	пропускать
	1, 1	—"—
	1, a	—"—
	a, 1	—"—
	#, a	—"—
	1, c	—"—
	#, A	ред. $A_1 \to 1$
	1, A	—"—
	1, B	ред. $B_1 \to 1$
	#, B	—"—

Рассмотрим анализ слова 11a11

$$\#\lessdot 1\gtrdot 1 \Leftarrow \#\lessdot 1\lessdot_1 1\gtrdot_1 a \Leftarrow \#\lessdot 1\lessdot_1 1\lessdot_1 a\gtrdot_1 1 \Leftarrow \#\lessdot 1\lessdot_1 1\lessdot_1 a\lessdot_1 1\gtrdot_1 1$$

$$\Leftarrow \#\lessdot 1\lessdot_1 1\lessdot_1 a\lessdot_1 1\lessdot_1 1\gtrdot_1 a \Leftarrow \#\lessdot 1\lessdot_1 1\lessdot_1 a\lessdot_1 1\lessdot_1 1\lessdot_1 a\gtrdot_1 \# \Leftarrow$$

$$\Leftarrow \#\lessdot_1 1\lessdot_1 1\lessdot_1 a\lessdot_1 1\lessdot_1 1\gtrdot_1 C\# \Leftarrow \#\lessdot_1 1\lessdot_1 1\lessdot_1 a\lessdot_1 1\gtrdot_1 1\gtrdot_1 C\# \Leftarrow \#\lessdot_1 1\lessdot_1 1\lessdot_1 a\gtrdot_1 1\gtrdot_1 1\gtrdot_1 C\gtrdot \#$$

$$\Leftarrow \#\lessdot 1\lessdot_1 1\gtrdot_1 A \doteq 1\gtrdot 1\gtrdot C\gtrdot \# \Leftarrow \#\lessdot 1\lessdot_1 A_1 \doteq A \doteq 1\gtrdot 1\gtrdot C\gtrdot \# \Leftarrow \#1\lessdot_1 A 1\gtrdot C\gtrdot \# \Leftarrow$$

$$\Leftarrow \#\lessdot 1\gtrdot A \doteq 1\gtrdot C\gtrdot \# \Leftarrow \#\lessdot A_1 \doteq A \doteq 1\gtrdot C\gtrdot \# \Leftarrow \#\lessdot A \doteq C\gtrdot \# \Leftarrow \#S\#$$

С практической точки зрения грамматики из класса $(\#,\#)$РК обладают двумя плохими свойствами:

I) класс КС-языков не замкнут относительно операции пересечения;

2) это свойство выражено в следующей теореме.

Теорема 3. Пусть V_T -алфавит, содержащий по меньшей мере два символа. Не существует алгоритма, позволяющего по произвольной грамматике предшествования с терминальным алфавитом V_T установить, принадлежит данная грамматика к классу $(\#,\#)$РК или нет.

Доказательсто: предположим противоположное, что существует алгоритм, распознающий по произвольной КС-грамматике предшествования G $G \in (\#,\#)$РК или $G \notin (\#,\#)$РК.

Очевидно, что КС-грамматика G является определенной тогда и только тогда, когда для любой сентенциальной формы в G детектирование и редуцирование определены однозначно, т.е. если $S \overset{*}{\Rightarrow} x_1 A y_1 \Rightarrow x_1 z_1 y_1$ и $S \overset{*}{\Rightarrow} x_2 B y_2 \Rightarrow x_2 z_2 y_2$.
В этом случае всегда $x_1 = x_2$ $y_1 = y_2$ $z_1 = z_2$ (условие однозначности дектирования) и A = B (условие однозначности редуцирования).

Пусть мы имеем произвольную грамматику предшествования $G \in (\#,\#)$РК. Поскольку мы имеем дело с грамматикой предшествования, то однозначность условия детектирования выполнена соответственно теореме I. Пусть $G \in (\#,\#)$РК и (A, B, x) произвольный конфликт редуцирования в грамматике G. Имеются две возможности:

а) $H(A, B) = \phi$.

Согласно конструкции множества H, для всех выводов

$$S \overset{*}{\Rightarrow} x_1 A y_1 w_1 \Rightarrow x_1 x y_1 w_1;$$
$$S \Rightarrow x_2 B y_2 w_2 \Rightarrow x_2 x y_2 w_2,$$

где $x_1, x_2, y_1, y_2, w_1, w_2 \in V^*$; $|y_1| = k$, $|y_2| = k$;

справедливо $(x_1, y_1) \neq (x_2, y_2)$, т.е. условие однозначности редуцирования выполнено.

б) $H(A, B) \neq \phi$.

Согласно конструкции класса грамматик $(\#,\#)$РК для всех выводов

$$S \overset{*}{\Rightarrow} x_1 A w_1 \Rightarrow x_1 x w_1$$
$$S \overset{*}{\Rightarrow} x_2 B w_2 \Rightarrow x_2 x w_2$$

справедливо $w_1 \neq w_2$, так как $w_1 \in \mathcal{L}(G_1)$, $w_2 \in \mathcal{L}(G_2)$ и $\mathcal{L}(G_2) \cap \mathcal{L}(G_1) = \phi$.
Это означает, что условие однозначности редуцирования выполнено.

Пусть мы имеем произвольную грамматику предшествования $G \in (\#,\#)$РК. Тогда, согласно определению, должен существовать по меньшей мере один конфликт редуцирования (A, B, x), такой, что $H(A, B) \neq \phi$ и $\mathcal{L}(G_1) \cap \mathcal{L}(G_2) \neq 0$.
Это означает, что существуют следующие выводы:

$$S \overset{*}{\Rightarrow} x_2 B y_2 z_2; \quad |y_1| = k,$$
$$|y_2| = k, \qquad x_1, x_2 \in V^*$$
$$y_1, y_2, z_1, z_2 \in V_T^*,$$

где из условия $H(A, B) \neq \phi$ следует, что существуют такие (x_1, y_1) и (x_2, y_2), что $(x_1, y_1) = (x_2, y_2)$, и из условия $\mathfrak{L}(G_1) \cap \mathfrak{L}(G_2) \neq \phi$ следует, что существуют такие z_1 и z_2, что $z_1 = z_2$. Следовательно, в грамматике G существуют правила $S \overset{*}{\Rightarrow} xAy$ и $S \overset{*}{\Rightarrow} xBy (x \in V^*, y \in V_T^*)$ и соответственно определению КР, $A \neq B$.

Итак, мы доказали, что грамматика предшествования G однозначна тогда, когда $G \in (\#,\#)$РК. Учитывая теорему I.8.I из работы [8], получаем следующий результат: произвольная КС-грамматика преобразуется в грамматику G' так, что $\mathfrak{L}\{G\} = \mathfrak{L}(G') \setminus \{\lambda\}$ и сохраняется КС-грамматика. Также из [II] следует, что преобразование предшествования сохраняет свойство определенности. Следовательно, оказывается, что для любой КС-грамматики можно выяснить, является ли она однозначной или нет. Это, однако, противоречит теореме 4.5.I [8], и мы пришли к противоречию. Следовательно, не существует алгоритма, определяющего для произвольной грамматики предшествования $G \in (\#,\#)$РК или $G \notin (\#,\#)$РК. Теорема доказана.

Ввиду этих вышеописанных двух причин следует наложить ограничения на грамматики G_1 и G_2, описывающие правый контекст. Одна такая возможность указана в следующем параграфе.

3. Грамматики предшествования, редуцируемые при помощи ограниченного КС-языка

Множество $\mathfrak{X} \subseteq V_T^*$ называется ограниченным, если существуют цепочки $x_1, \dots, x_n$, такие, что $\mathfrak{X} \subseteq x_1^* \dots x_n^*$.

При помощи данного понятия определяем класс грамматик предшествования, редуцируемых при помощи ограниченных КС-языков. Класс таких грамматик обозначаем через $(\#,\#)$ОЯР.

Пусть G будет любая грамматика предшествования, а k – фиксированное целое число.

Мы говорим, что $G \in (\#,\#)$ОЯР, если

I) $G \in (\#,K)\,KP$ или

2) $G \in (\#,K)\,KP$, то при каждом таком (A, B, x), при котором H (A, B) $\neq \phi$ существуют КС-грамматики G_1 и G_2, так, что

а) $\mathfrak{L}(G_1) = \{x \mid S' \overset{*}{\Rightarrow} yAx,\ y \in V^*,\ x \in V_T\}$,

$\mathfrak{L}(G_2) = \{x \mid S \overset{*}{\Rightarrow} yBx,\ y \in V^*,\ x \in V_T\}$;

б) $\mathfrak{L}(G_1) \cap \mathfrak{L}(G_2) = \phi$;

в) $\mathfrak{L}(G_1)$ и $\mathfrak{L}(G_2)$ ограниченные КС-языки;

г) G_1 и G_2 определенные КС-грамматики.

Далее мы хотим показать, что существует алгоритм, распознающий по произвольной КС-грамматике предшествования, принадлежит ли она к классу грамматик предшествования, редуцируемых при помощи ограниченного КС-языка. Для этого нам необходимо доказать следующие утверждения.

I. Все конфликты редуцирования грамматики G можно разделить на два множества $\mathcal{M}_1$ и $\mathcal{M}_2$ следующим образом:

$$\mathcal{M}_1 = \{(A,B,x) / H(A,B) = \phi\};$$

$$\mathcal{M}_2 = \{(A,B,x) / H(A,B) \neq \phi\}.$$

II. Для каждого конфликта редуцирования (A, B, x) $\in \mathcal{M}_2$ можно найти грамматики G_1 и G_2 соответственно определению $(\#,\#)$ОЯР.

III. Можно определить, генерирует ли данная грамматика ограниченный КС-язык.

IУ. Можно определить, является ли пересечение языков пустым.

У. Можно определить, являются ли грамматики G_1 и G_2 однозначными.

Чтобы показать справедливость утверждения пункта I, необходимо для каждого A, B $\in V_N$ найти их LR (к) контексты и пересечение этих контекстов. Алгоритмы для нахождения аналогичных множеств приведены в работе [I6].

Для доказательства утверждения II сконструируем регулярную грамматику G_A^I, беря за основу грамматику предшествования G и $A \in V_N$ следующим образом:

$$G_A^I = (V_N^I, V_T^I, P_T, [S]),$$

где $V_N^I = \{[x] / x \in M_A\}$, $M_A = \{x / x \overset{*}{\Rightarrow} uAv,\ u, v \in V^*\}$

$V_T^I = \{ x / x \in V_T \cup V_N \}$

и из каждого правила $У \to X_1 \dots X_m \in P$ $У \in M_A$ получаем следующие правила грамматики G_A'

$$[Y] \to [x_i] x_{i+1} \dots x_m ;\ 1 \le i \le m,\ x_i \in M_A.$$

Язык, генерируемый регулярной грамматикой G_A^I, состоит из всех тех предложений грамматики G, которые образовали бы правый контекст $A \in V_N$, если вместо канонического вывода использовать левосторонний вывод ($x \overset{л}{\Rightarrow} у$, если $x = z_1 A z_2$ $у = z_1 z z_2$ и $A \to z \in P$, $z_1 \in V_T^*$, $z, z_2 \in V^*$).

Далее проведем в грамматике G_A^I замену $\mathfrak{X} \Rightarrow G_x$ [8, с. 57], где

$$\mathfrak{X} \in \{Y \mid [z] \to [w] x_1 \dots x_m \in P^I,\ Y = x_i,\ 1 \le i \le m\};$$

$$G_x = (V_N, V_T, P, x).$$

(Согласно лемме I.4.4 [8], можно считать, что G_x является приведенной грамматикой).

Учитывая теорему I.7.I [8], получаем КС-грамматику, которая генерирует правый контекст $A \in V_N$.

Аналогично сконструируем регулярную грамматику G_B^I и проведем в ней такую же замену. Таким образом, мы получили приведенные КС-грамматики G_1 и G_2.

Доказательством утверждения Ш является теорема 5.5.2 [8]:

Теорема 4. Существует алгоритм, распознающий по произвольной КС-грамматике G, порождает ли она ограниченный КС-язык. Если множество $L(G)$ ограничено, то могут быть эффективно найдены цепочки $w_1, \dots, w_n$ из множества V_T^*, такие, что $L(G) \subseteq w_1^* \dots w_t^*$. Для доказательства утверждения IУ докажем более общее условие, учитывая возможности расширения класса грамматик (#,#)ОЯР.

Теорема 5. Существует алгоритм, распознающий по произвольным КС-грамматикам G_1 и G_2, одна из которых порождает ограниченный язык, пусто ли множество $L(G_1) \cap L(G_2)$.

Доказательство. Обозначим $M_1 = \mathfrak{L}(G_1)$ и $M_2 = \mathfrak{L}(G_2)$. Согласно теореме 3, мы можем определить, какой из КС-языков M_1 или M_2 является ограниченным. Изменив, если нужно, обозначения, мы можем считать, что ограниченным является КС-язык M_1. Кроме того, по теореме 4 мы можем найти цепочки $w_1, \dots, w_n$, такие, что $M_1 \subseteq w_1^* \dots w_n^*$. Согласно теореме 2.I.I [8], разность $w_1^* \dots w_n^* - M$ является регулярным множеством, а согласно теореме 3.2.I [8], пересечение $M_2 \cap w_1^* \dots w_n^*$ и разность M_3 ($M_3 = M_2 \cap w_1^* \dots w_n^* - (w_1^* \dots w_n^* - M_1)$) являются КС-языками.

Пусть $x \in M_3$ произвольный элемент, тогда

$$x \in M_2 \cap w_1^* \dots w_n^* \quad \text{и} \quad x \bar{\in} w_1^* \dots w_n^* - M_1.$$

Из этого, в свою очередь, следует, что $x \in M_1 \cap w_1^* \dots w_n^*$ и следовательно, $x \in M_2 \cap M_1$.

Пусть $x \in M_2 \cap M_1$ произвольный элемент.

Так как $x \in M_1$ и $M_1 \subseteq w_1^* \dots w_n^*$, то из этого следует, что $x \bar{\in} w_1^* \dots w_n^* - M_1$, а из этого, что $x \in M_2$, получаем $x \in M_2 \cap w_1^* \dots w_n^*$. Всё это показывает, что

$$x \in M_2 \cap w_1^* \dots w_n^* - (w_1^* \dots w_n^* - M_1)$$

и следовательно, $M_3 = \mathfrak{L}(G_1) \cap \mathfrak{L}(G_2)$ является КС-языком. Согласно теореме 4.I.2 [8] существует алгоритм, определяющий, пусто ли множество M_3.

Следовательно, теорема 5 доказана.

Справедливость утверждения У показывает теорема 5.6.4 [8]:

Теорема 6. Существует алгоритм, распознающий по произвольной КС-грамматике, порождающей ограниченный КС-язык, является ли она определенной.

Утверждения I-У и теорема 2 дают нам следующую теорему:

Теорема 7. Существует алгоритм, распознающий по произвольной λ-свободной грамматике G, является ли $G \in (\#,\#)$ОЯР.

Этот результат важен тем, что исходя из произвольной грамматики класса $(\#,\#)$ ОЯР, можно механически создать эффективный анализатор для языка, генерируемого этой грамматикой.

Далее рассмотрим некоторые свойства класса грамматик (#,#)ОПР.

В теореме 3 содержится следующий результат.

Теорема 8. Каждая $G \in (\#,\#)$ОПР является определенной.

Также представляет интерес то, что в рассматриваемом классе грамматик проблема $LR(k)$ решается положительно.

Теорема 9. Проблема, существует ли для данной грамматики $G \in (\#,\#)$ОПР, такое $k > 0$, что G является $LR(k)$ - грамматикой, разрешима.

Доказательство. Пусть имеется произвольная $G \in (\#,\#)$ ОПР. Предположим, что при данном фиксированном к КС-грамматика G не является $LR(k)$ грамматикой. Так как G -грамматика предшествования, то должен существовать хотя бы один конфликт редуцирования (A,B, x) такой, что $H(A, B) \neq \phi$.

Согласно теореме 4 можно эффективно найти $w^*_{11}, \ldots, w^*_{1n}$ и $w^*_{21}, \ldots, w^*_{2n_2}$ такие, что $\mathcal{L}(G_A) \subseteq w^*_{11} \cdots w^*_{1n_1}$ и $\mathcal{L}(G_B) \subseteq w^*_{21} \cdots w^*_{2n_2}$.

Пусть $u \barwedge v$ префиксальное пересечение регулярных множеств u и v.

Пусть даны два слова $x, y \in V_T^*$. Определим максимальный префикс этих двух слов

$$Sf(x,y) = z, \quad x = zu, \quad y = zv.$$
$$Sf(u,v) = \lambda, \quad vu \neq \lambda, \quad u,v \in V_T^*$$
$$u \barwedge v = \{Sf(x,y) / x \in u \,\&\, y \in v\}.$$

Поскольку $\mathcal{L}(G_A)$ и $\mathcal{L}(G_B)$ ограниченные множества и $\mathcal{L}(G_A) \barwedge \mathcal{L}(G_B) \neq \phi$ (так как $H(A, B) \neq 0$), то $M = \mathcal{L}(G_A) \barwedge \mathcal{L}(G_B)$ также является ограниченным множеством. Необходимо проверить, существуют ли в данном множестве элементы, генерируемые рекурсивно. Покажем, что если во множестве M существует элемент, который содержит часть, генерируемую рекурсивно, то эта часть принадлежит только одной w_i. Для этого используем понятие коммутативного множества. Множество X называется коммутативным, если для любых двух цепочек u и v, принадлежащих множеству X, имеет место равенство $uv = vu$.

Пусть $G = (V_N, V_T, P, S)$ и $A \in V_N$.

Полагаем

$$L_A(G) = \{u \in V_T^* / A \overset{*}{\Rightarrow} uAv \quad v \in V_T^*\};$$

$$R_A(G) = \{v \in V_T^* / A \Rightarrow uAv \quad u \in V_T^*\}.$$

Из лемм 5.5.5, 5.5.8 и 5.5.7 [8] известно, что для всякого $A \in V_N$ множества $L_A(G)$ и $R_A(G)$ коммутативны, (если $L(G) \neq \emptyset$) и ограничено, и существует алгоритм, позволяющий эффективно найти цепочки w_1, w_2, такие, что $L_A(G) \subseteq w_1^*$ и $R_A(G) \subseteq w_2^*$. Из этого видно, что действительно, если во множестве M существует элемент, который содержит часть, генерируемую рекурсивно, то эта часть принадлежит только одной w_i. Кроме того, можно найти такие w_i, которые генерируются рекурсивно. Так как $\mathcal{L}(G_A)$ и $\mathcal{L}(G_B)$ ограничены, то можно найти цепочки $w_{31}, \ldots, w_{3n_3}$, такие, что $M \subseteq w_{31}^* \ldots w_{3n_3}^*$.

Чтобы проверить, не является ли данная КС-грамматика G ни при одном $k > 0$ $LR(k)$ -грамматикой, необходимо проверить, существуют ли такие i_1, i_2, i_3 $1 \leq i_j \leq n_j$ $j = 1,2,3$, что $w_{1i} = w_{2i_2} = w_{3i_3}$ среди рекурсивно генерируемых w_i. Если такие i_1, i_2 и i_3 не существуют, то можно принять $K_{AB} = \max(P_A, P_B)$, где P_A и P_B целые числа, которые можно найти соответственно лемме 3.1.1 [8].

Проверяя таким образом все конфликты редуцирования данной грамматики, и если ни при одном из них пересечение правых контекстов не содержит рекурсии, то можно принять K равным $K_1 = \max\limits_{(A,B,x)} K_{AB}$, и данная грамматика является $LR(K_1)$-грамматикой.

Литература

1. AHO, A. V., DENMING, P. I.. ULLMAN, I.D. Weak and mixed strategy precedence parsing. J. ACM 1972, 19, 2, 225-243.

2. AHO. A. V., ULLMAN, I. D. The theory of parsing, translation and compiling. II. Compiling. USA, New Jersey. Prentice-Hall 1973.

3. GATES, G. W., POPLAWSKI, D. A. A simple technique for variable look-up. Communications ACM, 1973, 16, 561-565.

4. G R A H A M, S. L., R H O D E S, S. P. Practical syntactic error recovery. Communications ACM, 1975, 18, 5, 639-650.

5. G R A Y, J. N., H A R R I S O N, M. A. On the covering and reduction problems for context-free grammars. J. ACM 1972, 19, 4, 675-698.

6. L E W I, J., D E V A L M I N C K, K. SLS/1: A translator writing system. Lecture Notes in Computer Science. 1975, 34, 4, 627-641.

7. W I R T H, N., W E B E R, H., E U L E K - a generalization of ALGOL, and its formal definition. Pt.I. Comm. ACM. 1966, 9, 1, 13-25.

8. Гинзбург С. Математическая теория контекстосвободных языков. М., "Мир", 1970.

9. Грис Д. Конструирование компиляторов для цифровых вычислительных машин. М., "Мир", 1975.

10. Тедерсоо К.И. Реорганизация базы данных. Всесоюзная научная конференция "АСУ ВШ". Тезисы докладов. Таллин, 1977.

11. Томбак М. Об устранении конфликтов предшествования. "Тр. Тартуского гос.университета", 1976, вып. 37, с. 60-91.

12. Вооглайд А.О. Семантическое равенство распознавателей, работающих на грамматике LR(к) и грамматике предшествования с (1/1) ограниченным каноническим контекстом. "Тр. Таллинск. политехн. ин-та", 1976, № 411, с.39-55.

13. Вооглайд А.О., Томбак М.О. О проблемах редуцирования в грамматиках предшествования. "Тр. Таллинск. политехн. ин-та", № 386, 1975, с. 23-37.

14. Вооглайд А.О., Томбак М.О. Система построения трансляторов с LR(к) семантикой. Программирование. М., 1976, 5.

15. Нунапуу Э.Х.-Т. Функция создания и обновления базы данных. Всесоюзная научная конференция АСУ ВШ. Тезисы докладов, Таллин, 1977.

16. Кнут Д. О переводе языков слева направо. М., "Мир", 1975, с. 9-42.

A. Vooglaid

Extensive Parser of Mixed Strategy Precedence

S u m m a r y

The paper tackles a new class of grammars where one can solve the following algorythmic problems:

a) could the integer k be found when the given grammar is LR(k) grammar,

b) is the given grammar ambiguous.

For the parser of examined class of grammars any mixed strategy precedence parser is suitable with a few modifications.

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED
ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

№ 426 1977

УДК 681.3.06.:518.5

Е.Б.Бернштейн, А.Л.Шмундак

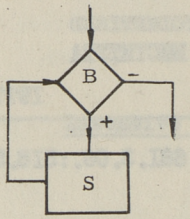
ПАКЕТ МАКРОКОМАНД ДЛЯ СТРУКТУРНОГО ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ АССЕМБЛЕР

I. Введение

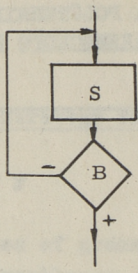
Поиски средств для облегчения отладки программ привели к появлению в середине шестидесятых годов понятия структурного программирования (см. [1]). Как показали Бём и Якопини, любой алгоритм может быть реализован путем использования конкатенации, итеративной группы с одним входом и одним выходом и предложения выбора. На практике в структурном программировании для управления программой используются следующие операторы:

Примерные форматы операторов и соответствующие им блок-схемы приведены на фиг. 1–4.

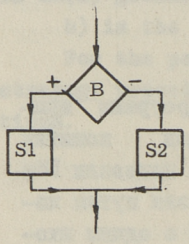
Вообще говоря, структурное программирование ориентируется на работу с языками высокого уровня (АЛГОЛ, ПЛ/I и т.д.). В Ассемблере ДОС/ЕС перечисленные операторы могут быть реализованы с помощью макрокоманд [2]. В вычислительном центре ТПИ написан набор макроопределений для структурного программирования. Дальнейшая часть работы посвящена описанию использования этих макрокоманд. В разделе 2 мы опишем макрокоманды для конструкции IF-THEN-ELSE. Раздел 3 посвящен макрокоманде SELECT, раздел 4 – макрокомандам DO-WHILE и REPEAT-UNTIL, раздел 5 – макрокоманде, предназначенной для реализации часто встречающегося случая конструкции DO-WHILE. В разделе 6 рассматриваются некоторые ограничения, налагаемые на программы, использующие данный пакет, и сообщения об ошибках, печатаемые макрогенератором Ассемблера.



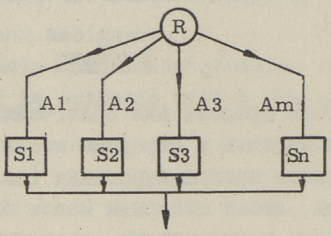
WHILE B DO
S
ENDWHILE



REPEAT
S
UNTIL B



IF B THEN
S1
ELSE
S2
ENDIF



SELECT R
CASE A1
S1
CASE A2
S2

CASE An
Sn
ENDSEL

2. Конструкция IF-THEN-ELSE

Конструкция IF реализуется с помощью макрокоманд IF, ELSE и ENDIF. Таким образом, текст

IF предикат THEN
группа I
ELSE
группа 2
ENDIF

транслируется в последовательность команд, обеспечивающих выполнение операторов, входящих в группу I или в группу 2, в зависимости от того принимает предикат значение истины или лжи. Как всегда, ELSE и группа 2 могут быть опущены.

Предикат представляет собой совокупность позиционных параметров макрокоманды. Параметры бывают трех типов:

1) базисный предикат,
2) "команда",
3) связка AND, OR, ANDIF, ORIF.

Предикат есть набор параметров этих трех видов, в котором обязательно присутствует хотя бы один базисный предикат.

В простейшем случае предикат состоит только из базисного предиката, который может записываться в одной из трех форм:

1) условие,
2) команда сравнения, операнд 1, условие, операнд 2,
3) код команды, операнд 1, операнд 2, условие.

В третьем случае возможна любая команда, вырабатывающая признак результата, кроме команд сравнения. Условие может быть любым числом от 1 до 14 или одним из мнемонических обозначений из таблицы 1.

Таблица 1

Тип команды	Мнемоника	Код	Значение	Дополнение	Код
Арифметический	P	2	плюс	NP	13
	M	4	минус	NM	11
	Z	8	нуль	NZ	7
	O	1	переполнение	NO	14
Проверить по маске	O	1	единицы	NO	14
	M	4	смесь	NM	11
	Z	8	нули	NZ	7
Сравнение	H, GT	2	больше	NH, LE	13
	L, LT	4	меньше	NL, GE	11
	E, EQ	8	равно	E	7

Базисный предикат форм 2 и 3 должен заключаться в скобки, если только весь предикат не состоит из одного - единственного базисного предиката.

Примеры.

1. IF 2 THEN
   S1
   ELSE
   S2
   ENDIF

Предикат состоит из базисного предиката формы 1. Признак результата выработан до макрокоманды.

2. IF CLC,FIELD,EQ,25(R7) — Предикат базисный формы I
S1
ELSE
S2
ENDIF

В этом случае макрорасширение выглядит так:

```
        CLC FIELD, 25 (R7)
        BC 15-8,  #L1
        S1
        B       #L2
#L1 EQU     *
        S2
#L2 EQU     *
```

3. IF TM,FLAG,X'80',Z — Предикат базисный формы 3
S
ENDIF

В некоторых случаях для выработки признака результата требуется более одной команды. В таком случае перед базисным предикатом должен стоять параметр типа "команда", который имеет вид:

(код команды, операнд I, операнд 2) или
(код команды, операнд I).

Код команды и операнды записываются по правилам Ассемблера. (Заметим, что нельзя использовать команды более чем с двумя операндами: BXLE, BXH, STM, LM).

Пример.

```
IF (LH,R1,SIZE),(CH,R1,LE,MAXSIZE) THEN
S1
ELSE
S2
ENDIF
```

Кроме простых предикатов, которые состоят из одного базисного предиката или из группы команд, за которыми следует базисный предикат, можно использовать составные предикаты, которые получаются из простых с помощью булевых связок AND и OR. В этом случае проверяется столько простых компонент, сколько необходимо для вычисления значения всего предиката. Связки имеют одинаковый приоритет, и скобки расставляются справа налево: A AND B OR C вычисляется

как A AND (B OR C). Для изменения порядка скобок используются операторы ANDIF и ORIF. Например, A OR B ANDIF C OR D вычисляется как (A OR B) AND (C OR D).

Пример.

```
IF (TM,FLAG,X'80',O),AND,(LH,R3,C),                    *
   (LTR,R3,R3,P),ORIF,(CH,R1,H,MAXSIZE),OR             *
   (CH,R1,L,MINSIZE) THEN

 S
 ENDIF
```

транслируется в

```
          TM FLAG,X'80',
          BC 15-1,#L1
          LH R3,C
          LTR R3,R3
          BC  2,#L2
#L1.      EQU *
          CH  R1,MAXSIZE
          BC  2,#L2
          CH  R1,MINSIZE
          BC  15-4,  L3
#L2       EQU *
          S
#L3       EQU *
```

Для удобства введен также оператор ELSEIF который эквивалентен следующим друг за другом операторам ELSE и IF, но не увеличивает глубину вложенности (т.е. для него не требуется отдельный оператор ENDIF). Например, эквивалентны следующие конструкции:

```
IF B1 THEN
S1
ELSEIF B2
S2
ELSE
S3
ENDIF
```

```
IF B1 THEN
S1
ELSE
IF B2 THEN
S2
ELSE
S3
ENDIF
ENDIF
```

3. Конструкция SELECT

Конструкция SELECT реализуется с помощью мак-

рокоманд SELECT, CASE, DEFAULT и ENDSEL. Общий вид этой конструкции на Ассемблере таков:

```
SELECT      Rx
CASE        a11, a12, ... a1k, (b11, c11) ... (b1k1, c1k1)
S1
CASE        a21, a22, ... a2k, (b21, c21) ... (b2k2, c2k2)
S2
:
CASE        an1, ...      ank, (bn1, cn1) ...
Sn
DEFAULT
SO
ENDSEL
```

Здесь Rx – общий регистр (может быть указан как его номер, так и его символическое имя), а все a_{ij}, b_{ij}, c_{ij} - целые числа, SO, S1, ... Sn – как и прежде, наборы операторов Ассемблера.

Sk будет выполняться, если:

1. $\langle Rx \rangle = a_{ki}$ ($\langle Rx \rangle$ – содержимое регистра Rx)

или

2. $b_{ki} \leqslant \langle Rx \rangle \leqslant c_{ki}$

SO выполняется, если содержимое Rx не совпадает ни с одним из a_{ij}, указанных во всех макрокомандах, и не попадает ни в один из интервалов (b_{ij}, c_{ij}).

Пример.

```
SELECT   5
CASE     3,16(8,12),5,284,(18,101)
S1
CASE     1024,(842,849),1
S2
DEFAULT
SO
ENDSEL
```

Если содержимое пятого регистра совпадает с одним из чисел 3,5,I6,284 или попадает в один из интервалов (8,I2) или (I8,I0I), то будет выполнено S_1, если же содержимое пятого регистра есть число I или I024 либо попадает в интервал (842,849), будет выполнено S_2. Во всех осталь-

ных случаях будет выполнено SO. Оператор DEFAULT и SO могут быть опущены. В этом случае Ассемблером будет напечатано предупредительное сообщение:

DEFAULT IS MISSING, SUPPOSED TO BE UNNECESSARY

4. Конструкции DO-WHILE и REPEAT-UNTIL.

Конструкция DO-WHILE реализуется при помощи макрокоманд WHILE и ENDWHILE и на Ассемблере выглядит так:

```
WHILE предикат DO
S
ENDWHILE
```

Конструкцию REPEAT-UNTIL реализуют макрокоманды REPEAT и UNTIL . Запись на Ассемблере следующая:

```
REPEAT
S
UNTIL предикат
```

Предикат подробно описан в разделе 2, а блок-схемы этих конструкций даны во введении.

5. Конструкция ITERATIVE-DO

Конструкция ITERATIVE-DO реализуется с помощью макрокоманд DO и ENDDO.

Простейшая форма итеративного цикла - выполнение некоторого набора операторов N раз. Такая форма определяется с помощью ключевого параметра TIMES и выглядит так:

```
DO TIMES = (Rx,N)
S
ENDDO
```

где N определяет, сколько раз будет выполняться цикл, а Rx - общий регистр, в который загружается число N. Число N можно не указывать, если регистр Rx уже загружен нужным значением. В этом случае запись такова:

```
DO TIMES = Rx
S
ENDDO
```

Регистр Rx необходимо указывать всегда, в противном случае будет выдано предупреждающее сообщение и цикл выполняться не будет.

В наиболее общей форме итеративный цикл на Ассемблере представляется так:

```
DO  FROM= (Rx,i),BY = (Ry,j), TO = (Rz,k)
S
ENDDO
```

где Rx, R_y и R_z – общие регистры, а i,k и j –границы изменения и шаг соответственно. В зависимости от значений i,j,k будет порождаться возрастающий или убывающий цикл. i и k могут быть или десятичными константами, или адресными константами. j всегда задается (если оно вообще задано) в виде десятичной константы.

Примеры.

I.
```
DO FROM = (Rx,= A(LAST)), BY = (Ry, - 4) TO=(Rz,=A(FIRST))
S
ENDDO
```

по этим макрокомандам будет сгенерирован убывающий цикл:

```
        L   Rx, = A(LAST)
        L   Ry,= F'-4'
        L   Rz,= A(FIRST)
#M1     EQU *
        S
        BXH Rx,Ry, #M1
        CR  Rz,Rx
        BE  #M1
```

2.
```
DO  FROM = (3,0), BY = (6,1),TO = (7,10)
S
ENDDO
```

сгенерируется возрастающий цикл,

```
        L     3, = F'0'
        L     6, = F'1'
        L     7, = F'10'
#M1     EQU   *
        S
        BXLE  3,6, #M1
```

Некоторые из значений ключевых параметров можно опускать. При этом необходимо придерживаться следующих правил:

I. Регистр Rx всегда должен быть указан.

2. Регистры Ry и Rz либо оба указываются, либо оба опускаются.

Если хотя бы одно из этих правил не выполнено, то цикл выполняться не будет, о чем выдается предупреждающее сообщение.

3. i, j, и k могут быть опущены. При этом считается, что регистры уже загружены нужными значениями. При этом нужно учитывать следующее:
- если Ry и Rz опущены, то j и k опускать нельзя: в противном случае совсем не определены параметры BY и TO;
- если i, j и k опущены и при этом невозможно определить возрастающий цикл или убывающий, то требуется задавать параметр DIR (DIR=DOWN или DIR=UP), указывающий направление цикла. По умолчании будет принято направление UP.

Ввиду использования команд BXH и BXLE (в расширении макрокоманды DO) регистры Ry и Rz, если они оба указаны, должны удовлетворять следующим ограничениям:

- Ry - регистр с четным номером
- Rz=Ry+1.

Примеры.

1.
```
DO FROM = ( 3,1),BY=(,1),TO=(,5)
S
ENDDO
```
Будет генерироваться возрастающий цикл. Ry и Rz опущены.

2.
```
DO FROM=6,BY=8, TO=9,DIR=DOWN
S
ENDDO
```
Сгенерируется убывающий цикл i,j и k опущены.

3.
```
DO FROM=3, BY=(4,-1),TO=5
S
ENDDO
```
Сгенерируется убывающий цикл i и k опущены.

6. Некоторые особенности

При использовании макрокоманд структурного программирования запрещается использовать метки, начинающиеся с символа #.

Макрокоманды структурного программирования вводят в язык Ассемблера блочную структуру. Глубина вложенности не должна превышать 100. Если при закрытии некоторой конструкции оказывается, что вложенная в нее конструкция не закрыта, в тексте программы печатается сообщение:

1, CONFLICT

Если в предикате макрокоманд IF.ELSEIF, WHILE, UNTIL обнаружена ошибка, печатается сообщение:

2. PARAMETER NR. K OR K-1 INVALID

и анализ предиката заканчивается.

Литература

1. Дал У., Дейкстра Э., Хоор К. Структурное программирование. М., 1975.

2. McGowan, C., Kelly, J. Top-down structured programming techniques. N.-Y., 1975.

E. Bernstein, A. Shmundack

A Set of Macros for Structured Programming in Assembly Language

Summary

A set of macros facilitating structured programming in assembly language is presented. The rules for using the set and error diagnostics are described and a number of samples are given.

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED
ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

№ 426 1977

УДК 518.512.25

Э.А.-Ю. Юби

ЗАДАЧИ СО СРЕДНИМИ ЗНАЧЕНИЯМИ СЛУЧАЙНЫХ ВЕЛИЧИН

Рассматриваются две системы линейных алгебраических уравнений. Коэффициенты первой системы – математические ожидания случайных величин, коэффициенты второй системы – статистические оценки этих математических ожиданий. Найдены условия, когда решение второй системы является состоятельной оценкой решения первой системы.

I. Постановка задачи

Рассмотрим систему линейных алгебраических уравнений по средним

$$\begin{aligned} &\bar{a}_{11} x_1 + \dots + \bar{a}_{1n} x_n = \bar{b}_1, \\ &\dots\dots\dots\dots\dots\dots \\ &\bar{a}_{n1} x_1 + \dots + \bar{a}_{nn} x_n = \bar{b}_n, \end{aligned} \tag{I}$$

или, сокращенно,

$$MAx = Mb,$$

где $\bar{a}_{ij}, \bar{b}_i$ — средние значения случайных величин a_{ij}, b_i, а $x^* = (x_1^*, \dots, x_n^*)$ искомое детерминированное решение системы, $i, j = 1, \dots, n$.

В практике, как правило, вместо средних значений MA, Mb в нашем распоряжении имеются их оценки, скажем

$$\hat{a}_{ij}(N) = \frac{\sum_{k=1}^{N} a_{ij}^k}{N}, \quad \hat{b}_i(N) = \frac{\sum_{k=1}^{N} b_i^k}{N}, \tag{2}$$

где a_{ij}^k, b_i^k $(k = 1, \dots, N)$ независимые реализации случайных величин a_{ij}, b_i. Тогда вместо детерминированного решения x^* системы (I) мы получим некоторый случайный вектор $\hat{x}(N)$, решение системы

$$\hat{A}(N)x = \hat{b}(N), \tag{3}$$

где N – число реализаций случайных величин. Если $\det\hat{A}(N) = 0$, то доопределим решение, приравняем его для определенности нулю, $\hat{x}(N) = 0$. Докажем простую лемму.

Лемма I. Пусть $(\xi_N, \eta_N) \to (a,b)$ с вероятностью I при $N \to \infty$ (a и b вещественные числа). Дана $f(x,y): R^2 \to R^1$, причем $f(x,y)$ непрерывна в некоторой окрестности точки (a,b), тогда $f(\xi_N, \eta_N) \to f(a,b)$ с вероятностью I при $N \to \infty$. Доказательство. По условию для любого $\varepsilon > 0$

$$P_N = P\left\{\sqrt{(\xi_M - a)^2 + (\eta_M - b)^2} < \varepsilon \text{ для всех } M \geq N\right\} \to 1, \quad N \to \infty.$$

Надо доказать, что для любого $\varepsilon > 0$

$$\pi_N = P\left\{|f(\xi_M, \eta_M) - f(a,b)| < \varepsilon \text{ для всех } M \geq N\right\} \to 1, \quad N \to \infty.$$

По непрерывности $f(x,y)$ в точке (a,b) для любого $\varepsilon > 0$ существует $\delta > 0$ такая, что из $\sqrt{(x-a)^2 + (y-b)^2} < \delta$ следует, что $|f(x,y) - f(a,b)| < \varepsilon$. Зафиксируем произвольный $\varepsilon > 0$. Тогда существует $\delta > 0$, что $P\left\{\sqrt{(\xi_M - a)^2 + (\eta_M - b)^2} < \delta\right\} \leq P\left\{|f(\xi_M, \eta_M) - f(a,b)| < \varepsilon\right\}$ и, следовательно, и $P\left\{\sqrt{(\xi_M - a)^2 + (\eta_M - b)^2} < \delta \text{ для всех } M \geq N\right\} \leq P\left\{|f(\xi_M, \eta_M) - f(a,b)| < \varepsilon \text{ для всех } M \geq N\right\}$. По условию вероятность слева стремится к единице, следовательно, и вероятность справа стремится к единице при $N \to \infty$, ч.т.д.

2. Состоятельность и несмещенность оценки

Теорема I. Если математические ожидания MA, Mb существуют, $\det MA \neq 0$, то $P(\det\hat{A}(N) = 0) \to 0$ и $\hat{x}(N) \to x^*$ с вероятностью I при $N \to \infty$.

Доказательство. По формуле Крамера $\hat{x}_i(N) = \dfrac{\det\hat{A}_i(N)}{\det\hat{A}(N)}$, если $\det\hat{A}(N) \neq 0$. По лемме I $\det\hat{A}_i(N) \to \det MA_i$ и $\det\hat{A}(N) \to \det MA$ с вероятностью I при $N \to \infty$. Тогда $P(\det\hat{A}(N) = 0) \to 0$, $N \to \infty$, так как $\det MA \neq 0$. Функция $f(x,y) = \frac{y}{x}$ непрерывна в некоторой окрестности точки $(x,y) = (\det MA, \det MA_i)$. Поэтому по лемме I

$$\hat{x}_i(N) = \frac{\det\hat{A}_i(N)}{\det\hat{A}(N)} \to \frac{\det MA_i}{\det MA} = x_i^*$$

с вероятностью I при $N \to \infty$.

Теорема доказана.

Рассмотрим вопрос несмещенности оценки $\hat{x}(N)$ и вычисление определителя из случайных величин. Вообще говоря, $M\hat{x}(N)$ не существует в предположениях теоремы I, а если и существует, то $M\hat{x}(N) \neq x^*$. В частном случае, когда только правые части b_i случайные, $M\hat{x}(N) = x^*$. Это очевидным образом получается из формулы Крамера. Как указывается в работе [I], до сих пор полностью не описан случай, когда формула Крамера дает несмещенное решение. Если $\hat{x}(N)$ ограниченная случайная величина, то имеет место асимптотическая несмещенность оценки $\hat{x}(N), M\hat{x}(N) \to x^*$, $N \to \infty$, если выполняются условия теоремы I. Это следует из теоремы Лебега [2].

Рассмотрим матрицу A из случайных элементов. Вообще говоря, $\det A$ случайная величина. Следует отметить, что если $\det MA \neq 0$, тем не менее $\det A$ может равняться тождественно нулю. Рассмотрим пример.

$$A = \begin{pmatrix} \xi & \eta \\ \frac{1}{\eta} & \frac{1}{\xi} \end{pmatrix},$$

$\det A \equiv 0$. Пусть ξ и η равномерно распределены в $[1,5]$ и $[1,2]$. Очевидно, $\det MA = \frac{3}{4} \ln \frac{5}{4} > 0$.

Итак, нам надо оценить $\det MA$ через $\det \hat{A}(N)$. Не умаляя общности, рассмотрим случай $n = 2$. По определению

$$\det MA = \bar{a}_{11}\bar{a}_{22} - \bar{a}_{21}\bar{a}_{12}, \quad \det \hat{A}(N) = \hat{a}_{11}(N)\hat{a}_{22}(N) - \hat{a}_{12}(N)\hat{a}_{21}(N).$$

Для несмещенности оценки $\det \hat{A}(N)$ достаточно, чтобы математические ожидания всех произведений

$$M\hat{a}_{i_1 j_1}(N)\hat{a}_{i_2 j_2}(N) = M\hat{a}_{i_1 j_1}(N) M\hat{a}_{i_2 j_2}(N) = \bar{a}_{i_1 j_1}\bar{a}_{i_2 j_2}.$$

Это выполняется всегда, если элементы a_{ij} и $a_{k\ell}$, $i \neq k$, $j \neq \ell$ независимы для всех i, j, k, ℓ. Достаточное условие для этого – независимость всех случайных элементов a_{ij}, $i, j = 1, \ldots, n$. Если в некоторых произведениях встречаются зависимые случайные величины, то для оценки их средних значений надо использовать независимые реализации для получения несмещенности оценки определителя. Так в рассмотренном примере, если использовать для оценки $M\xi \cdot M\frac{1}{\xi} = \frac{3}{4} \ln 5$ величины $\hat{\xi} = \frac{\sum_{k=1}^{N} \xi^k}{N}$, $\left(\widehat{\frac{1}{\xi}}\right) = \frac{\sum_{k=1}^{N} \frac{1}{\xi^k}}{N}$,

вычисленные по одним и тем же независимым реализациям ξ^k в обоих случаях, то $M\left[\hat{\xi}\cdot\left(\frac{1}{\hat{\xi}}\right)\right] \neq M\hat{\xi}\,M\left(\frac{1}{\hat{\xi}}\right)$. Если в оценках $\hat{\xi}$ и $\left(\frac{1}{\hat{\xi}}\right)$ использовать разные независимые реализации, то имеет место равенство $M\left[\hat{\xi}\cdot\left(\frac{1}{\hat{\xi}}\right)\right] = M\hat{\xi}\cdot M\left(\frac{1}{\hat{\xi}}\right)$. По формуле Крамера $\hat{x}_i(N)=\frac{\det\hat{A}_i(N)}{\det\hat{A}(N)}$. Если для определителей в числителе и знаменателе использовать независимые реализации (причем в числителе одни, а в знаменателе другие), то достаточное условие для несмещенности оценки $\hat{x}_i(N)$ в виде

$$M\frac{1}{\det\hat{A}(N)}=\frac{1}{M\det\hat{A}(N)} \quad \text{(при существовании } M\frac{1}{\det\hat{A}(N)}\text{)}.$$

Тогда $M\hat{x}_i(N)=M\frac{\det\hat{A}_i(N)}{\det\hat{A}(N)}=\frac{M\det\hat{A}_i(N)}{M\det\hat{A}(N)}=\frac{\det MA_i}{\det MA}=x^*_i$.

При практическом решении задачи $MAx=Mb$ может случиться, что оценка $\hat{x}(N)$ значительно отличается от x^*. Это возможно, если исходная задача $MAx=Mb$ не корректна в смысле Тихонова [3]. Тогда следует применить разработанные А.Н.Тихоновым методы регуляризации.

Полученный результат может быть использован в линейном стохастическом программировании. Если коэффициенты матрицы ограничений средние значения некоторых случайных величин, за реализацией которых мы можем наблюдать, то для решения такой задачи можно разработать прямые методы [4]. По терминологии Ю.М. Ермольева в данной работе разработан прямой метод решения системы линейных алгебраических уравнений.

В заключение автор выражает благодарность Т. Тобиасу за оказанную ему помощь.

Литература

I. Глоговский В.М., Кац С.А., Хургин Я.И. О системах алгебраических уравнений со случайными коэффициентами. "Журнал вычислительной математики и математической физики", 1971, т. II, № 6, с. 1574–1581.

2. Вазан М.Т. Стохастическая аппроксимация. М., "Мир", 1972. 295 с.

3. Тихонов А.Н., Арсенин В.Я. Метод решения некорректных задач. М., "Наука", 1974, 224 с. с ил.

4. Ермольев Ю.М. Методы стохастического программирования. М., "Наука", 1976, 240 с. с ил.

E. Ubi

Some Problems with the Mean Values

Summary

This article deals with two linear algebraic systems of equations. The coefficients of the first system are mean values of some random variables, the coefficients of the second one are the estimates of these mean values. Conditions are found for the solution of the second system to be the consistent estimate of the solution of the first one.

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED
ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

№ 426 1977

УДК 681.142.2

И.И. Амитан

ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ РЕШЕНИЯ ДВУХ ЗАДАЧ ОПТИМИЗАЦИИ

В связи с переходом к многопроцессорным вычислительным системам появилась необходимость в разработке параллельных алгоритмов и языка для их описания.

В данной работе представлены параллельные алгоритмы решения двух задач оптимизации: транспортной задачи и многопродуктовой задачи размещения производства. Исходя из специфики этих задач, алгоритмы представлены как совокупность логически завершенных частей, которые могут выполняться одновременно. Предполагается существование многопроцессорной вычислительной системы с неоднородным управлением и общей памятью; кроме того, каждый процессор имеет некоторую память для локальных величин.

I. Язык сигналов для описания параллельных процессов

В этой части данной статьи дается лишь краткое описание необходимых конструкций языка сигналов, при описании которого использовалась система понятий и обозначений АЛГОЛа-68 [I].

Для описания взаимодействия параллельных процессов наряду с физическими процессорами рассматриваются программные процессоры – actor, представляющие собой структурный вид

```
mode actor = struct(ref phispr i,sema sem,ref[0:1] flex
actor fol, ref role R),
```

где указывается семафор процессора, ссылки на физический процессор и на другие программные процессоры, а также ссылка на процессорную процедуру – role, исполняемую данным

actor. С каждым программным процессором связывается одна или несколько процессорных процедур, которые состоят из управляющей процедуры и разделенной программы и представляют собой расширенное понятие процедуры с параметрами, вырабатывающей "ничто" или некоторый вид. Для связи между программными процессами используются входные и выходные сигналы или команды – incomm и outcomm, вводятся процедуры give, подающие эти команды, и операции get, определяющие, подана ли команда. Управляющая процедура, описанная в каждой процессорной процедуре, в зависимости от поступления той или иной входной команды передает управление на соответствующий сегмент разделенной программы; поступившая команда гасится операцией get. В случае, если входного сигнала не поступало, управление передается в исходную точку разделенной программы данной процессорной процедуры. Разделенная программа – это программа, в которой определены точки проверки поступления входных команд, они отмечаются подчеркнутой ";". Для синхронизации работы процессоров вводится процедура WAIT. Одновременное выполнение процессорных процедур – role программными процессорами – actor осуществляется в параллельном предложении, где каждая его составляющая представляет собой отдельный параллельный процесс. Описания "актеров", их "ролей" и всех необходимых операций, данных, процедур и параллельное предложение составляют программу параллельного алгоритма на языке сигналов.

2. Параллельный алгоритм транспортной задачи

Для замкнутой модели транспортной задачи рассматривается метод потенциалов, алгоритм которого в обычной последовательной форме может быть записан следующим образом

```
begin prep;
      while newT do workT od;
      fin
end,
```

где prep – процедура подготовки расчетов – построение начального решения;

newT – процедура проверки критерия оптимальности, вырабатывающая булевское;

workT – процедура перестройки базиса;

fin – завершение расчетов и выдача необходимых результатов на печать.

Представленный выше алгоритм решения транспортной задачи преобразован для случая одновременной работы двух процессоров. Так как в языке сигналов программным представлением физических процессоров является программный процессор – actor, то задавая работу процессоров, их взаимодействие и связи между ними, мы будем иметь дело только с программными процессорами. Один из процессоров – главный, он выполняет всю основную часть вычислений: строит начальное решение, производит замену векторов базиса, выдает результаты. Второй процессор – вспомогательный, он начинает работу, получив сигнал от главного. Главный процессор призывает вспомогательный на помощь при определении максимума улучшения целевой функции, а также при пересчете перевозок во время перестройки базиса.

Прежде чем дать точное описание алгоритма, введем обозначения. Каждый пункт потребления или производства – вершина – задается структурным видом

mode ver = struct (ref ver up, int fl, c, pt, st),

где поле up содержит ссылку на вершину, с которой данная вершина v связана перевозкой,

fl – величина перевозки между вершинами v и up of v;

c – стоимость перевозки между вершинами v и up of v;

pt – потенциал;

st – уровень вершины – количество вершин в цепи, соединяющей вершину v с корнем дерева.

Тогда все вершины задаются массивом [1:m+n] ver v. Вычислительный алгоритм основного процессора теперь задается таким образом.

I. Строится начальный план prep.

2. Выбирается вершина-потребитель и до тех пор, пока не будет исчерпано их множество, подается сигнал help вспомогательному процессору, по которому тот начинает определять максимальное улучшение целевой функции для своей вершины – maxa; в это же время основной процессор выбирает следующую вершину-потребитель и сам определяет для нее максимальное улучшение – max; последнюю вершину-потребитель ос-

новной процессор обрабатывает сам; исчерпав все множество вершин-потребителей, переходит к пункту 8.

3. Определив свой max и получив maxa от вспомогательного процессора, выбирает из них больший и, если его значение больше нуля, определяет тем самым вектор, вводимый в базис.

4. Если значение полученного максимума меньше или равно нулю, то возвращается к пункту 2, где выбирает следующую по порядку не исследованную вершину-потребитель.

5. Изменяет базис-процедура workt0.

6. Для пересчета перевозок по правой ветви цикла призывает вспомогательный процессор командой start и пересчитывает сам перевозки по левой ветви цикла.

7. Заканчивается перестройка базиса - workt.

8. Если перестройка базиса произошла, то процессор проверяет полученное решение на оптимальность, перейдя снова к пункту 2; если полученное решение оптимально, то переходит к пункту 9.

9. Заканчиваются вычисления и выдаются результаты, печатая массив вершин и значение целевой функции.

Алгоритм вспомогательного процессора.

1. Получив команду help от основного процессора, определяет значение максимального улучшения целевой функции.

2. Закончив свою работу, посылает об этом сигнал ready1 главному процессору.

3. Получив команду start , начинает пересчет перевозок по правой ветви цикла; после окончания работы посылает сигнал ready2 .

Одновременное выполнение вычислений основным и вспомогательным программными процессорами задается в параллельном предложении, где определяются их значения.

```
par (P[1] :=(1, level 0, P[2] ,main(yes,c1,c2,s1,s2,
  sec,1)),
      P[2] :=(2,  level 0, P[1] ,aux(s1,s2,c1,c2,first,2))),
```

где main и aux - "роли" "актеров" first и sec, задающие их действия; c1, c2, s1, s2 - входные и выходные команды.

```
role main = (incomm beg, r1, r2, outcomm help, start,
             ref actor sub, int nmb ) void:
begin proc ent = void:
           if get beg then mk0
           elif get r1 then init(r); mkr1
           elif get r2 then init(r); mkr2 fi;
entry: skip ;
mk0 : prep; ref actor r:=P[nmb] ; nr:= m1;
mk01: nr+:= 1; rhf:= v[nr] ;
      if nr = m then mk02 elif nr> m then mkag fi;
      nra: = nr+1; give (help, sub);
mk02: maxs (lhf,nr,k1,max,v): if nr=m then mkm fi; WAIT ;
mkr1: if maxa> max then max:= maxa; lhf:=lhfa; k1:= k1a;
      rhf:=v[nra]; for i to m do v[i]:=va[i] od fi;
mkm: if max>0 then workt0; give(start, sub); ref ver l:=lhf;
     while l dif rhe do fl of l+:= min; l:=up of l od;
     WAIT; skip
                 else mkr3 fi;
mkr2: workt;
mkr3: rhf:=v[nra]; goto mk01;
mkag: if again then again:=false; nr:=1; goto mk01 fi;
      fin
end.
role aux = (incomm h,s, outcomm ready1, ready2,
            ref actor boss, int nmb) void:
begin proc ent = void:
           if get h then init(r) ; mkh
           elif get s then init(r) ; mks fi;
entry: skip ; WAIT;
mkh: maxs(lhfa,nra,k1a,maxa,va); give(ready1,boss); WAIT;
mks: ref ver l:=lhe, r:=rhf;
     while l dif r do fl of r -:= min; r:=up of r od;
     give(ready2,boss); WAIT; skip
end.
```

Полностью процедура параллельного алгоритма решения транспортной задачи приведена в приложении.

3. Параллельный алгоритм многопродуктовой задачи размещения производства

3.I. Постановка задачи и описание метода ее решения

Рассматривается многопродуктовая задача размещения, которая записывается следующим образом.

Задано множество продуктов $L = 1 : l$;

множество типовых проектов строительства предприятий $T = 1 : ntp$;

множество потребителей $N = 1 : n$.

Каждому пункту потребления j сопоставлен вектор объемов потребления по всем продуктам $b_j[L]$. Через $bs[L]$ обозначим $\sum bj[L]$, $j \in N$.

Каждому типовому проекту t сопоставлены вектор объемов производства по каждому продукту $a_t[L]$ и стоимость строительства cf_t.

Каждому пункту производства i сопоставлены вектор затрат по перевозке единицы груза до каждого пункта потребления $c_i[N]$ и некоторое подмножество T_i множества типовых проектов (проекты, осуществимые в данном пункте производства).

Требуется выбрать в каждом пункте производства по одному осуществимому типовому проекту $t_i \in T_i$ и составить план перевозки по каждому продукту таким образом, чтобы

а) сумма векторов объемов производства выбранных типовых проектов равнялась суммарному потреблению:

$$\sum a_{t_i}[L] = bs[L], \quad i \in M;$$

б) план перевозки по каждому продукту $k \in L$ удовлетворял условиям транспортной задачи:

$$X_k[M,N] \geqslant O[M,N],$$

$$\sum X_k[i,J] = a_{t_i}[K], \quad j \in N$$

$$X_k[i,j] = b_j[K], \quad i \in M$$

и достигался минимум затрат на строительство и перевозку

$$\sum_{i \in M} (cf_{t_i} + \sum_{K \in L} (c_t[N] \cdot X_K[i,N])) \longrightarrow \min.$$

Условие равенства суммарной производственной мощности и

спроса позволяет нам заранее вычислить затраты на производство и включить их в затраты на строительство.

Решать задачу будем методом ветвей и границ. Назовем частичным решением набор осуществимых типовых проектов для множества пунктов производства I: depth, где depth∈O:M называется глубиной частичного решения. Имея некоторое частичное решение P, мы будем увеличивать на единицу его глубину и рассматривать все допустимые одношаговые продолжения. Оценка целевой функции для частичного решения будет складываться из суммы затрат на строительство выбранных типовых проектов, минимума затрат на развозку продукции выбранных заводов, минимума целевой функции в многомерной задаче о ранце, решаемой для оставшихся пунктов производства и оставшейся для них потребности в производственных мощностях. Предварительное решение многомерной задачи о ранце позволяет сохранить информацию о том, какие продолжения частичного решения способны дать допустимое решение.

Рассматриваем набор частичных решений (кандидатов), которые выстроены в порядке возрастания их оценок. Каждый кандидат описывается структурным видом

```
mode cand = struct (int est, depth, ref cand next,
                    ref inf inf, [1:m] ref dec dec),
```

где est – оценка;
next – ссылка на кандидата, служит для организации цепного списка;
depth – глубина частичного решения,
dec[1:depth] – массив ссылок на выбранные решения.

Каждое решение описывается структурным видом

```
mode dec = struct ( int est, ref tp tp ),
```

где est – оценка решения;
tp – ссылка на типовой проект.

Типовой проект – структурный вид

```
mode tp = struct ( int cf, [1 : 2] int a]
```

где cf – стоимость строительства типового проекта;
a [L] – вектор объемов производства по каждому продукту;
inf – ссылка на информацию о решении многомерной задачи о ранце, которая представляется струк-

турным видом

```
mode inf = struct (int kr,min, [1:r] pd possdec),
```

где kr – число элементов r в векторе possdec при порождении данного элемента;

min – минимум целевой функции;

possdec – массив возможных одношаговых продолжений данного частичного решения, каждый элемент которого описывается видом

```
mode pd = struct(ref dec d, ref inf ad),
```

где d – ссылка на решение;

ad – ссылка на информацию о задаче, которая получится при добавлении к частичному набору решения d.

Для решения многомерной задачи о ранце потребуется еще структурный вид:

```
mode curstate = struct(int s, [1:k]int g).
```

Пункты производства задаются структурным видом

```
mode suppl = struct(int lst, [1:n]int ct),
```

а пункты потребления–

```
mode dem = struct ([1:m]int b).
```

Для транспортной задачи потребуются структурные виды:

```
mode row = struct(ref row pr,ref col qr,int ar,min,dr,i);
mode col = struct(ref row qc,ref col pc, int dc,ac,
                  [1:n] int c);
mode transinf = struct(int f,j, ref col tj,[1:n]row rw,
                  [1:m]col cl).
```

Метод ветвления теперь можно представить таким образом

```
prep B and B;
while est of first < rec do
      invest(first) od;
results,
```

где prep B and B – некоторая подготовительная часть;

first – ссылка на начало цепного списка частичных решений;

rec – значение целевой функции на рекордном решении;

invest – процедура изучения частичного решения.

Рассматриваемое частичное решение приводится в рабочее для алгоритма состояние, а затем развивается. Развитие заключается в том, что до тех пор, пока глубина частичного решения не достигнет m, изучаются все допустимые продолжения этого частичного решения, наилучшее из них выбирается для развития, а остальные пополняют список кандидатов. Кроме основного списка кандидатов строится рабочий список кандидатов – newlist, который сливается с основным списком после завершения изучения данного частичного решения. Когда глубина частичного решения достигает m, получаем (полное) решение, сравниваем его с рекордным и в случае надобности запоминаем новый рекорд и его значение целевой функции – rec (более конкретные подробности постановки задачи и метода решения см. в работе [2]).

3.2. Первый вариант параллельного алгоритма задачи размещения

Рассмотренный выше алгоритм мы представим для одновременной работы двух процессоров. Практическая реализация последовательного метода показала, что основное время счета уходит на решение транспортных задач при приведении информации о частичном решении в рабочее состояние и при получении уточненной оценки решения. В представленном параллельном алгоритме один из процессоров – основной, он выполняет все вычисления, а при решении транспортных задач призывает на помощь вспомогательный. Выполняя очередную транспортную задачу, главный процессор в определенных точках решения обращается к управляющему блоку с вопросом, есть ли сигнал от вспомогательного о том, что он готов решать следующую задачу. Если такой сигнал имеется, то главный процессор дает команду вспомогательному решать следующую транспортную задачу, если она не является последней; последнюю он всегда решает сам. Если сигнала о готовности помочь еще не поступало, основной процессор продолжает свою работу, т.е. решает свою транспортную задачу дальше. Алгоритм этот представляется эффективным, если вспомогательный процессор – специализированный процессор для решения транспортных задач.

Вычислительный процесс задается двумя процессорными процедурами – "ролями", которые выполняются одновременно.

Основной процессор – "актер", "исполняя роль" main выполняет такие действия алгоритма.

I. Подготовительные вычисления.

2. До тех пор пока оценка головы списка частичных решений меньше рекорда

а) частичное решение first приводится в рабочее состояние, т.е. уточняется его оценка, при этом для решения транспортных задач призывается на помощь вспомогательный процессор сигналом trdo;

б) рассматриваемое частичное решение развивается до глубины m или до тех пор, пока значение valuecand меньше рекордного значения; для решения транспортных задач снова происходит обращение за помощью к вспомогательному процессору;

в) если уточненная оценка решения valuecand меньше rec, то запоминается новое rec и новый рекорд;

г) сливаются основной и рабочие списки в процедуре correctlist;

д) выбирается следующее частичное решение из списка и начинается все сначала с пункта 2;

3. Выдача результатов.

Процессорная процедура main имеет следующий вид:

```
role main = (incomm beg,r, outcomm trdo,
             ref actor aux, int nmb) void:
begin proc ent = void:
    if get beg then mkb
    elif get r1 then init(r);
                if iter = 0 then mkr
                else do k+:=1;
                if (ak:=(a of curtp[k])≤ 0 then skip
                elif k=1∧(d = depth of cand∨d = m) then
                k -:=1; goto u1
                else curtri:=tri[k]; give(trdo,aux);
                goto u1 fi od;
        u1: skip fi;
entry: skip i
mkb: prep;
```

```
      while est of first < rec do
      ref cand cand:=first, newlist:=dummy; ref tp curtp;
      int valuecand:=0, it:=0, tr,iter,ak;
      [1:l] transinf tri ; ref transinf curtri;
      for k to l do j of tri[k]:=0; tj of tri[k]:= nil;
      for j to n do (rw of tri[k])[i]:=
      (skip,skip,skip,0, (b of dem[i])[k], i) od od;
      for d to depth of cand do down vi;
      valuecand+:=cf of (curtp:=tp of (dec of cand)[d]); up vi;
mkl:  for k to l do
      if (ak:= (a of curtp)[k]) >0 then curtri:=tri[k];
      iter:=0 ;
mkt:  f of curtri:=0; (cl of curtri) [ j of curtri+:=1]:=
      (skip, skip, skip, ac, ct of supp [d]);
      tr:= transiter (curtri); down vi; valuecand+:=tr;
      up vi; goto mkf;
mkr:  if k=1 ∧ (d=depth of cand ∨ d=m) then mkt fi;
      give (trdo,  aux);
mkf:  skip
      elif k=1∧ d=depth of cand then WAIT fi od;
      if it ≠ 0 then mkv fi od;
      for d from depth of cand + 1 to m while valuecand<rec do
      ref inf infcur:=inf of cand;
      ref pd pdnew,pdold:= (possdec of infcur)[1];
      ref dec main:=d of pdold; it+=1;
      int minest:=min of pdold + est of main, min1,
          d1:=depth of cand+:=1;
      for pd from 2 to kr of infcur do
          if minest > (min1:=min of (pdnew;=(possdec of
                       infcur)[pd]) + est of d of pdnew) then
          (registrate (cand,d1,minest + valuecand,pdold);
          minest:=min1; main:=d of (pdold:=pdnew)
          else registrate (cand,d1,min1 + valuecand,pdnew)fi od;
          (dec of cand)[d1]:=main;
          valuecand+:=cf of (curtp:=tp of main); goto mkl;
mkv:  skip od;
          if valuecand < rec then rec:=valuecand;
          ref cand (bestcand):=cand fi;
```

```
    correctlist (cand, newlist, freecand);
    freecand cons head cand od;
result
end.
```

Процедура registrate формирует нового кандидата и записывает его в рабочий список новых кандидатов – newlist. В процедура correctlist сливает основной и рабочий списки частичных решений.

В процессорной процедуре TRANS, выполняемой вспомогательным "актером", после получения команды do от главного процессора решается очередная транспортная задача. При завершении решения об этом посылается команда ready основному процессору, после чего происходит ожидание следующих распоряжений.

```
role TRANS = (incomm beg,do, outcomm ready,
              ref actor boss, int nmb) void:
begin proc ent = void: if get beg then mkb
                       elif get do then mkd fi;
entry: skip ;
mkb: int tr,ak1; ref transinf curtri1 ; WAIT;
mkd: curtri1:=curtri; ak1:=ak; f of curtri1:=0;
     (cl of curtri1)[j of curtri1+:=1]:=
     (skip ,skip, skip, ak1, ct of supp[d]);
     tr:=transiter (curtri1); down vi; valuecand+:=tr; up vi;
     give (ready, boss) ; WAIT ; skip
end.
```

Программа, описывающая этот параллельный алгоритм, содержит описания и одно параллельное предложение:

```
par (PO:= (0, level 0, PTR, main(yes,c1,s1,PTR,0)),
    PTR:= (1, level 0, PO, TRANS (yes,s1,c1,PO,1))).
```

Описания процедур transiter, registrate и correctlist приведены в приложении.

3.3. Второй вариант параллельного алгоритма многопродуктовой задачи размещения производства

Предполагается существование многопроцессорной вычислительной системы, состоящей из NP процессоров. Два процессора занимаются развитием частичных решений – [I:2] actor P, остальные NTR=NP-2 процессора –

[I:NTR] actor PTR решают транспортные задачи. Интересен случай, когда PTR - специализированные транспортные процессоры, но предложенный алгоритм работает также в случае, когда все NP процессоры одинаковые. Процессоры PTR организованы в цепной список, головой которого является ref actor , Pf , а концом - ref actor Pf in . Если вспомогательному процессору P [2] не пришлось тянуть ветку, то он также присоединяется к списку транспортных процессоров.

Опишем теперь алгоритмы всех трех разновидностей процессоров - "актеров". "Роль" main основного процессора определяет следующие его действия.

I. Описания требуемых величин и подготовительные вычисления.

2. До тех пор, пока оценка головы списка частичных решений меньше рекорда, исследует следующий элемент списка. Если он не nil и значение его оценки меньше rec, то вспомогательному процессору подается команда invest, чтобы он развивал ветвь с частичным решением first 1, в противном случае вспомогательный процессор устанавливается в цепной список транспортных процессоров, кроме того, он будет корректировать список частичных решений, сливая в один основной и рабочие списки.

3. Развивает частичное решение first:

а) приводит его в рабочее состояние; при решении транспортных задач дает "сообщения" транспортным процессорам и поручает им решать транспортные задачи;

б) развивает кандидата first при помощи одношаговых продолжений либо до полного решения глубины m, либо оставляет его, если значение оценки vc превысит значение rec ; для решения транспортных задач призываются на помощь свободные процессоры из массива PTR ;

4. Если вспомогательный процессор выступает в роли транспортного, то в этот момент ему посылается сигнал о корректировке основного и рабочего списков.

5. Если полученная оценка меньше rec , то изменяется значение рекорда на меньшее и запоминается решение, на котором оно достигается.

6. В зависимости от получения сигнала r2 об окончании развития ветви или r3 о завершении слияния списков от вспомогательного процессора основной процессор либо, если значение vc меньше rec, изменяет рекорд и рекордное решение bestcand, а затем корректирует основной и рабочий списки newlist и newlist1 и выбирает следующего кандидата, либо сразу переходит к следующему кандидату; затем все повторяется с пункта 2; если сигналов r2 и r3 не поступало, основной процессор ждет их.

7. Получив частичное решение в голове списка, оценка которого превышает значение rec, выдает результаты.

"Роль" main посылает транспортным процессорам уже не просто команды, а сообщения, в которых указывается и заказчик на транспортную задачу.

```
role main = (incomm beg,r1,r2,r3, outcomm invest, c1,
             outmess trdo, ref actor Pf, Pfin, aux, int nmb)
             void:
begin proc ent = void:
      if get beg then mkb
      elif get r2 then init(r); mkr2
      elif get r3 then init(r); mkc
      elif get r1 then init(r);
                  if iter = 0 then mkr else do k+:=1;
                  if (ak:=(a of curtp)[k]) < 0 then skip
                  else down q; if lin Pf then
                                  givem (trdo,boss,head Pf);
                                  up q; goto u1
                                  else up q; k-:=1;goto u1 fi fi od,
      u1: skip fi;
entry: skip ;
mkb: prep; ref actor r:=P[nmb] ; int VC, VC1; bool one;
     ref cand first1;
mkf: if est of first < rec then one:=true;
     if lin next of first then
           if est of (first1:=next of first) < rec then
           ref cand newlist1; one:=false; give(invest,aux)fi
     else fol of Pfin:=aux; Pfin:=aux fi;
     ref cand newlist:=dummy; w(k,ak,VC,curtri,first,vi);
     if VC < rec then rec:= VC;
                ref cand (bestcand):=first fi ; WAIT ;
```

```
mkr2: if VC1 < rec then rec:=VC1;
      freecand cons head first; ref cand (bestcand):=first
      else ref cand k:=first; next of first:=next of first1 fi;
      correctlist (first, newlist, freecand);
      correctlist (first, newlist1, freecand);
mkc: freecand cons head first; goto mkf fi;
     result
end.
```

Здесь W(k, ak, vc, curtri, first, vi) введено для сокращения записи и совпадает с соответствующей частью программы в предыдущем пункте 3.2.

Вспомогательный процессор либо тянет ветку, либо решает транспортные задачи и корректирует списки. Он выполняет такую последовательность действий.

I. Описывает требуемые величины и посылает сигнал о своей готовности работать.

2. Получив сигнал inv от главного процессора, начинает развивать частичное решение first1 (аналогично основному процессору).

3. Посылает сигнал ready основному процессору: работу закончил.

4. Получив сообщение do от главного процессора (значит, тянуть ветку на данном шагу алгоритма не пришлось), начинает решать очередную транспортную задачу; закончив ее решение, присоединяется к цепному списку свободных процессоров и посылает сигнал ready основному процессору.

5. Получив команду cl, корректирует основной и рабочий списки, построенные основным процессором.
Вспомогательный процессор имеет самый большой набор входных и выходных сигналов, в том числе входные и выходные сообщения, так как он может работать в двух режимах.

```
role branch = (incomm beg,inv,cl,r1, inmess do,
               outcomm ready,ready2,ready3, outmess trdo,
               ref actor Pf,Pfin,boss,int nmb) void:
begin proc ent = void:
      if get beg then mkb
      elif get inv then init(ra); mkin
      elif get cl then init(ra); mkc
      elif getm do then init(ra); r:=boss; mktr
      elif get r1 then init(ra);
            if iter=0 then mkr else do k1+:=1;
                  if (ak:=(a of curtp)[k1]) ≤ 0 then skip
                  else down q; if lin Pf then
                              givem (trdo,aux,head Pf);
                              up q; goto u1
                              else up q; k-:=1; goto u1 fi
            fi od;
      u1: skip fi;
entry: skip ;
mkb: int tr,ak1,k1; ref actor r,ra; ref transinf curtri1;
     give (ready,boss) ; WAIT ;
mkin: w (k1,ak1, vi1,VC1, curtria,first1);
      give (ready2, boss) ; WAIT ;
mktr: curtri1:=curtri; ak1:=ak; f of curtri1:=0;
      (cl of curtri1)[j of curtri1+:=1]:=
      (skip, skip, skip, ak1, ct of supp[d]);
      tr:=transiter (curtri1); down vi; VC+:=tr; up vi;
      if d = m then down q; Pfin cons aux; up q;
      give (ready, boss) fi ; WAIT ;
mkc: correctlist(first, newlist, freecand);
      give (ready3, boss) ; WAIT ; skip
end.
```

Вспомогательный процессор заведомо знает, что сигнал о решении транспортной задачи ему может прийти только от главного процессора, транспортные процессоры должны определить, от кого пришло сообщение. В "роли" TRANS задано: определение автора сообщения, решение транспортной задачи, отправка команды об окончании работы и присоединение к цепному списку свободных транспортных процессоров.

```
role TRANS = (incomm beg, inmess do, outcomm ready,
              ref actor aux, boss, Pfin, int nmb) void;
begin proc ent = void:
     if get beg then mkb
     elif getm do then init(ri);
          if Aof do dif boss then r:=aux else r:=boss fi;
          mkd fi;
entry: skip ;
mkb: int tr, ak1, t:=2; ref actor r,ri; ref transinf curtri1;
     ri:=PTR[nmb]; givek(ready,aux,boss,t) ; WAIT ;
mkd: curtri1:=(if r dif aux then curtri else curtria fi);
     ak1:=ak; f of curtri1:=0; (cl of curtri1)[j of curtri1+:=1]
     :=(skip, skip, skip, ak1, ct of supp[d]);
     tr:=transiter(curtri1);
     if r dif aux then down vi; VC+:=tr; up vi
                  else down vi1; VC1+:=tr; up vi1 fi;
     down q; Pfin cons PTR[nmb]; up q;
     if (k=1∧d=depth of first) v(k1=1∧d=depth of first1)
     then give( ready,r) else givek(ready,aux,boss,t) ; WAIT;
skip end.
```

Весь параллельный алгоритм задачи размещения записывается одним параллельным предложением, которое содержится в блоке из описаний и начальных определений:

```
par (P[1]:=(1,level 0,Pf,Pfin,P[2],
          main(yes,c1,c2,c3,s1,s2,ms3,Pf,Pfin,P[2],1)),
     P[2]:=(2,level 0,Pf,Pfin,P[1],
          branch(yes,s1,s2,c1,ms3,c1,c2,c3,ms3,Pf,Pfin,P[1])),
     for i to NTR do
     PTR[i]:=(i, level 0, P[2], P[1],
          (if i=NTR then nil else PTR[i+1] fi),
          TRANS(yes, ms3,c1,P[2],P[1], Pfin,i)))
```

Во внешнем блоке должны быть описаны и определены сигналы или команды, процессорные процедуры, операции и процедуры, которые используются при описании "родей" программных процессоров.

Литература

1. Revised report on the algorithmic language ALGOL-68, Springer-Verlag, Berlin, Heidelberg, New-York, 1976.

2. Грибов А.Б., Давыдова И.М., Романовский И.В. "Техника цепных списков, ее программная реализация и использование в алгоритмах оптимизации". Сб. "Исследование операций и статистическое моделирование", вып. 4, Л., изд-во ЛГУ, 1977.

Приложение

I. Параллельный алгоритм решения транспортной задачи

```
co par pot method co
begin int m,m1,j; read((m1,j)); m:=m1+j;
     [1:m]int b, [1:m1 × j]int c;
proc ppm = (int m,m1,[ ,]int cd,[ ]int bd) void:
begin int min,iter,cost,f1,fp,y,y1,nl,nr,k,kl,kr;
      bool again,rh;[1:2] actor P; ref ver v3;
mode ver = struct(ref ver up, int fl,c,pt,st);
ref ver rhf,lhf,ve,v1,v2,lhe,rhe; [1:m] ver v;
proc pot = (ref ver i) int:
     begin if st of i=0 then pt of i:=c of i+pot(up of i);
                             st of i:=1+st of up of i fi;
     pt of i end;
proc prep = void:
begin k:=m-m1; iter:= cost :=0; v[1]:=(nil,0,skip,0,1);
for i from 2 to m do v[i]:=(skip,-bd[i], skip,skip,0) od;
   fp:=-bd[1]; lhf:=v[1]; nl:=1;
mki: min:=maxint; f1:=nl × k-m;
    while fp < 0 do for j from m1+1 to m do
                  if st of v[j]=0 then
                   if cd[f1+j]< min then min:=cd[f1+j];
                                  rhf:=v[j]; nr:=j fi fi od;
    rhf:=(lhf,-bd[nr],min,p of lhf,-st of lhf-1);
    y:=fp; fp+:=fl of rhf od; fl of rhf:=-y;
mkj: min:=maxint; f1:=nr-m;
     for i to m1 do f1+:=k;  if st of v[i]=0 then
                                  if cd[f1] < min then min:=cd[f1];
                                        lhf:=v[i]; nl:=i fi fi od;
     if min < maxint then
        lhf:=(rhf,-bd[nl],-min,p,of rhf-min,st of rhf+1);
        y:=fp; fp+:=fl of lhf;
        if fp < 0 then fl of lhf:=-y; goto mki fi
                  else goto mkj fi; again:=false end;
```

```
proc maxs = (ref ver l, int i,m,k, ref[ ]ver v1) void:
begin ref[1:m]ver vm; for k to m do vm[k]:=v1[k]od;
ref ver i1:=vm[i]; int d,f:=i-m1; m:=0; s:=pot(i1);
for j to m1 do if (d:=s-cd[f]-pot(vm[j])) > m then
                    m:=d; l:=vm[j]; k:=f fi; f+:=m-m1 od;
for k to m do v1[k]:=vm[k]od end;
proc workt0 = void:
begin lhe:=lhf; rhe:=rhf; kl:=st of lhf; kr:=st of rhf;
rh:=kl+kr ≤ 0; y1:=cd [k1]; iter+:=1; min:=maxint;
if rh then while st of rhe ≠ kl do
            if st of rhe ≤ 0 then k1:=f1 of rhe;
            if k1 < min then min:=k1; ve:=rhe fi fi;
            lhe:=up of rhe od
    else while st of lhe ≠ kr do
         if st of lhe ≥ 0 then k1:=-f1 of lhe;
           if k1 < min then min:=k1; ve:=lhe fi fi;
         lhe:=up of lhe od fi;
mk1: while rhe ≠ lhe do k1:=-f1 of lhe;
         if st of lhe ≤ 0 ∧ k1 < min then min:=k1; ve:=lhe;
            rh:=false fi; lhe:=up of lhe; rhe:=up of rhe od;
     goto mk;
if rhe ≠ lhe then k1:=f1 of rhe;
   if st of rhe ≤ 0 ∧ k1 < min then min:=k1; ve:=rhe; rh:=true fi;
   lhe:=up of lhe; rhe:=up of rhe; goto mk1 fi;
mk: end workt0;
   proc work = void:
   begin kl:=abs(st of ve); st of ve:=0;
   if rh then v1:=lhf else v2:=lhf;v1:=rhf; min:=-min;
                                            y1:=-y1 fi;
   if kl > abs(st of v2) then kl:=abs(st of v2); st of v2:=0 fi;
mk2: v3:=up of v2; up of v2:=v1;y:=-f1 of v2; f1 of v2:=min;
   min:=y; y:=-c of v2; c of v2:=y1; y1:=y; v1:=v2;
   if v2 ≠ ve then v2:=v3; goto mk2 fi;
   for i from 2 to m do if abs(st of v[i])>kl then
                  st of v[i]:=0 fi od; again:=true end;
proc fin = void;
for s from 2 to m do cost+:=abs(c of v [s])x f1 of v [s]);
                  print(v[s]) od; print ((iter,cost)) end;
```

co "роли" main и aux здесь описывать не будем, их описания должны были бы быть здесь co

```
par (P[1]:=(1,level 0,P[2], main(yes,c1,c2,s1,s2,sec,1)),
     P[2]:=(2,level 0,P[1], aux(s1,s2,c1,c2,first,2)))
end ppm;
read((b,c)); ppm(m,m1,c,b)
end.
```

2. Для многопродуктовой задачи размещения производства приведем процедуры transiter, registrate, correctlist. В программе подготовки определяются элемент dummy списки и другие элементы; соответствующая часть программы подготовки будет выглядеть следующим образом:

```
ref cand dummy:=heap(cand), bestcand:=dummy,
        freecand:=dummy, first:=heap(cand);
curstate init:=(0,bs); ref inf inf1;[0:(v:=abs(main)]ref inf br;
for i from 0 to v do br[i]:=nil od; rec:=est of dummy:=
maxint;
ref cand(first):=(value(init,inf1),0,dummy,inf1,skip).

proc correctlist = (ref ref cand main,new,free) void:
begin ref cand c0:=main,c1,c2; int r1;
while (r1:=est of (c1:=head new))< rec do
    while est of (c2:=next of c0)< r1 do c0:=c2 od;
c0 intr c1 od;
```

co закончено слияние существенных частей списков; те частичные решения, в которых оценка больше рекорда, не нужны и могут быть утилизированы co

```
while (r1:=est of c1) < maxint do
      free cons c1; c1:=head new od; new:=c1;
while est of c2 <rec do c2:=head c0 od;
while est of c2 <maxint do free cons head c2 od;
next of c0:=c2
end.
```

```
proc registrate = (ref cand c, int j,e, pd pd)void:
     if e <rec then
     ref cand c1:=(if freecand dif dummy then head freecand
                                          else heap cand fi);
```

co процедура registrate должна найти место в памяти для нового частичного решения, запомнить это место и поставить частичное решение в рабочий цепной список на подобающее место; рассматриваются частичные решения, у которых оценка меньше рекорда co

```
ref cand(c1):=(e,j,skip,ad of pd,dec of c);
(dec of c1)[j]:=d of pd;
if e≤est of newlist then newlist cons c1
else ref cand c0:=newlist,c2:=next of c0;
     while est of c2< e do c0:=head c2 od; c0 intr c1 fi fi.
```

```
proc transiter = (ref transinf tinf) int:
begin int m=upb rw of tinf, n=upb c1 of tinf, j=j of tinf;
      ref [1:m] row rw=rw of tinf; [1:n] col c1=c1 of tinf;
      ref col tj:=tj of tinf; int u,d,mini,ai,aj,dt,f:=0;
co mode transinf = struct(int f,j,ref col tj,[1:m]row rw,
                          [1:n] col c1)  co
ref row r,r0,r1,r2,r3,r4,r5, ref col t,t1,t2,t3;
if (aj:=ac of c1[j])=0 then mkf fi;
pc of c1 [j]:=tj; tj:=c1[j]; d:=maxint; r0:=nil; r4:=rw[1];
for i to m do
if (u:=(c of tj)[i]+dr of (r:=rw[i]))>d then
   r4:=pr of r4:=r else pr of r:=r0; qr of (r0:=r):=tj;
                        if u<d then d:=u; r2:=r fi fi od;
f+:=d×aj;
do r1:=nil; while r1 dif r2 do
   if qr of (r:=if lin r1 then pr of r1 else r0 fi) ≠ tj then
      pr of r1:=pr of r; r4:=pr of r4:=r elif
      (ai:=ar of (r3:=r)) ≠ 0 then
      if (u:=(c of tj)[i of r]>0 then u:=0 fi;
   if ai ≥aj then dc of tj:=d; (c of tj) [i of r]:=u-aj;
             ar of r:=ai-aj; ac of tj:=0; goto mkf fi;
```

```
   ar of r:=0; aj-:=ai; (c of tj)[i of r] :=u-ai fi; r1:=r od;
r2:=r3; r1:=nil; t2:=tj; t1:=nil;
mk1: while r1 dif r2 do
     r1:=(if lin r1 then pr of r1 else r0 fi); t3:=t2;
     while lin (t:=pc of t3) do
     if (u:=(c of t)[i of r1]<0 then
         r3:=r2; dt:=dc of t;  while r3 dif r4 do
         if (u:=(c of t[i of (r:=pr of r3)])≥0 then
             u+:=dr of r else u:=dt fi;
         if u ≠ dt then r3:=r else qr of r:=t;
         if (ai:=ar of r) ≠ 0 then goto mk2 fi;
     if r=r4 then r4:=r3 else pr of r3:=pr of r fi;
     pr of r:=pr of r2; r2:=pr of r2:=r fi od;
     qc of t:=r1; pc of t3:=pc of t; pc of t:=pc of t2;
     t2:=pc of t2:=t fi od od; r:=r2;
if lin t1 then while r dif r4 do r:=pr of r; min of r-:=
               mini od else
while r dif r4 do r:=pr of r; qr of r:=tj; min of r:=
               (c of tj)[i of r]-d od; t1:=tj fi;
while t1 dif t2 do dt:=dc of (t1:=pc of t1); r:=r2;
while r dif r4 do
if (u:=(c of t1)[i of (r:=pr of r)])< min of r then
   min of r:=u; qr of r:=t1 fi od od; mini:=maxint;
while r1 dif r4 do
if (u:=min of (r:=pr of r1)+dr of r)< mini then
   mini:=u; r3:=r1 fi; r1:=r od; d+:=mini; t:=tj;
   r1:=r0; f+:=mini×aj;
while t dif t2 do dc of (t:=pc of t)+:=mini od;
while r1 dif r2 do dr of r1+:=mini; r1:=pr of r1 od;
dr of r2+:=mini; r5:=r1;
while r3 dif r4 do
if min of (r:=pr of r3)+dr of r=mini then
    if ar of r ≠ 0 then r5:=r fi;
    if r=r4 then r4:=r3 else pr of r3:=pr of r fi;
pr of r:=pr of r2; r2:=pr of r2:=r else r3:=r fi od;
if r5=r1 then goto mk1 fi; ai:=ar of (r:=r5);
mk2:while (t:=qr of r) dif tj do
    if ai<(u:=-(c of t)[i of (r1:=qc of t)]) then
    ar of r:=0; (c of t)[i of r1]:=ai-u else
    ar of r:=ai-u; ai:=u;
```

```
    (c of t)[i of r1]:=dc of t-dr of r1 fi;
    if (u:=(c of t)[i of r])>0 then u:=0 fi;
    (c of t)[i of r]:=u-ai; r:=r1 od od;
mkf: f end.
```

I. Amitan

Parallel Algorithms to Solve Two Optimization Problems

Summary

In this paper we deal with nonhomogeneous multiprocessing systems which have a common core memory. Parallel algorithms of the classical transport problem and the multiproduct allocation problem are considered. For the latter problem an algorithm for the system of two processors as well as an algorithm for N processors are given. To describe the algorithms an extension of ALGOL-68 is used.

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED
ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

№ 426 1977

УДК 519.1

Г.А.Вейнер

ЕЩЕ РАЗ О ВСПОМОГАТЕЛЬНЫХ ФУНКЦИЯХ ПРИБЛИЖЕНИЯ ГРАФА ПОЛНЫМИ ПОДГРАФАМИ

В настоящей краткой статье автор продолжает изучение проблемы аппроксимации симметричного бинарного отношения R с помощью отношения эквивалентности E_0. В данной работе рассматривается одно следствие теоремы, которая доказана в статье [1]. В результате такого рассмотрения расширяется класс графов, для которого вспомогательные функции позволяют вполне решить задачу аппроксимации.

Напомним коротко поставленную перед нами проблему, которая, как и в [1], состоит в поиске аппроксимирующего отношения эквивалентности $E \subset S \times S$ для заданного симметричного бинарного отношения $R \subset S \times S$. Целью аппроксимации является поиск такого отношения эквивалентности E_0, которое в наименьшей степени отличается от заданного отношения R, причем под наименьшей степенью отличия понимается следующее: число несовпадающих элементов заданного бинарного симметричного отношения R и отношения эквивалентности E, равное

$$\rho(E) = |R - E| + |E - R|,$$

должно быть минимальным. Таким образом, отношение E_0 определяется из условия

$$\rho(E_0) = \min_{E \in \mathfrak{E}} \rho(E),$$

где $\mathfrak{E}$ – множество всех допустимых отношений эквивалентности на множестве S.

Далее мы рассматриваем отношение R в виде графа G. Предполагается, что структура графа G соответствует структуре в [1]. Предполагается также, что подграф G_a и остальные основные понятия, которые здесь используются,

определены таким же образом, как в [I].

Теорема [I]. Если для графа G_a выполняются следующие условия:

I) векторы функции $f(K_i, G_0)$ направлены в подграф G_0;

2) $L_i < \frac{1}{2} v_0 m_i$ и

3) $\sum_{i=1}^{k} (v_0 - \sum_{j=1}^{i} v_j) v_i \geq \sum_{i=1}^{k} L_i$,

для $i = 1, 2, \ldots, k$, то существует такое аппроксимирующее отношение E_0, которое имеет класс эквивалентности V_0.

Пусть $P = \{ K_i \mid i \in I_p, I_p \subset I_k \} \subset B$ – такое подмножество максимальных гроздей подграфа G_a, что векторы функции $f(K_i, G_0), i \in I_p$, направлены от подграфа G_0 в подграф K_i.

Следствие. Если для подграфа G_a выполняется:

I) $P \neq \phi$;

2) существует такое подмножество максимальных гроздей $P_0 = \{ K_j \mid j \in I_m, I_m \subseteq I_p \} \subseteq P$,

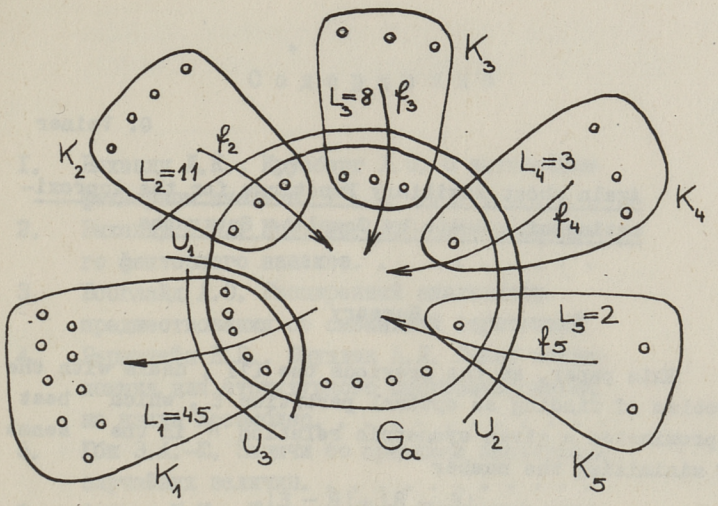
удаление которых полностью из подграфа G_a создает такой подграф $G'_a = (V'_a, U'_a) \subset G_a$, для которого выполняются условия предшествующей теоремы,

то существует такое E_0, при нахождении которого надо изъять ребра гроздей подграфа G'_a.

Доказательство. Предполагается, что удаление целых гроздей множества P_0 оптимально. Тогда и удаление ребер максимальных гроздей подграфа G'_a оптимально в силу утверждения теоремы.

Если предположить, что изъятие ребер по крайней мере одной грозди множества P_0 оптимально, тогда значения функции $f(K_i, G'_0)$ подграфа G'_a возрастают и условия теоремы выполнены. Это доказывает теорему.

Пример. Ребра на фигуре не изображены. Начальные числовые значения функции $f_1 = 1,2$; $f_2 = 2$; $f_3 = 2,3$; $f_4 = 3,9$ и $f_5 = 5$. Значения функции после удаления грозди K_1 будут $f'_2 = 1,5$; $f'_3 = 1,7$; $f'_4 = 2,9$ и $f'_5 = 3,8$. Значит, условия теоремы, наложенные на граф G'_a, выполнены, и оп-



Фиг. 1

тимальное разбиение определено сечениями U_2 и U_3.

Вид функций оценок позволяет без труда заключить, что для проверки условий теоремы достаточно проверить изменение функции наименьшего значения на подграфе G'_a. В примере такое значение представлено значением функции $f_2 = 2$.

Литература

I. Вейнер Г.А. Две вспомогательные функции оптимального приближения графа полными подграфами. – "Тр. Таллинск. политехн. ин-та", 1976, № 411.

G. Veiner

Again about Auxiliary Functions for the Approximating of a Graph by Complete Subgraphs

Summary

This paper, as the previous one [1] , deals with the problem of finding an optimal partition E , which best approximates a given symmetric relation R in the sense of minimizing the number

$$|E - R| + |R - E|.$$

The problem of using two auxiliary functions is also treated.

Содержание

Таллинский политехнический институт
Труды ТПИ № 426
ВОПРОСЫ СОЗДАНИЯ СРЕДСТВ ОБРАБОТКИ ИНФОРМАЦИИ
(ОБРАБОТКА ДАННЫХ, ПРОГРАММИРОВАНИЕ)
Труды экономического факультета ХХУШ
Сборник утвержден коллегией Трудов ТПИ 16 мая 1977 г.
Редактор И. Муллат. Технический редактор В. Ранник
Подписано к печати 21 ноября 1977 г. Бумага 60x90/16
Печ. л. 5,5 + 0,25 приложение. Уч.-изд. л. 4,6
Тираж 300. МВ-06295
Ротапринт ТПИ, Таллин, ул. Коскла, 2/9. Зак. № 1135

Цена 69 коп.

TALLINNA POLÜTEHNILISE INSTITUUDI TOIMETISED
ТРУДЫ ТАЛЛИНСКОГО ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА

№ 426 1977

ВОПРОСЫ СОЗДАНИЯ СРЕДСТВ ОБРАБОТКИ ИНФОРМАЦИИ

(Обработка данных, программирование)

Труды экономического факультета XXVIII

УДК 519.24

О нелинейном факторном анализе. Выханду Л.К., Крусберг Х.О. "Труды Таллинского политехнического института", № 426, 1977, с. 3-9.

В статье анализируются и получают дальнейшее развитие нелинейные модели факторного анализа второго порядка. Указывается возможность ослабления ограничений на факторные значения, что позволяет более точно учесть требования ортогональности и нормированности факторов.

Библ. наименований - 5.

УДК 519.24

Метод прямого факторного анализа. Выханду Л.К., Крусберг Х.О. "Труды Таллинского политехнического института", № 426, 1977, с. 11-21.

В статье рассматривается модель факторного анализа $Z = BG$, где B определяется через специальное разложение матриц корреляций $R = BB'$. Доказывается связь с методом главных компонент. Метод экономичен в вычислениях и удобен для интерпретации.

Фигур - 1, таблиц - 2, библ. наименований - 5.

УДК 681.142.2

Расширенный анализатор предшествования со смешанной стратегией. Вооглайд А.О. "Труды Таллинского политехнического института", № 426, 1977, с. 23–42.

Рассматривается новый класс грамматик, в пределах которого можно разрешить следующие алгоритмические проблемы: а) можно ли найти такое целое число K, что данная грамматика является LR(K) грамматикой, б) является ли данная грамматика однозначной.

Для анализатора рассмотренного класса грамматик годится любой анализатор предшествования со смешанной стратегией, в который требуется ввести лишь незначительные изменения. Необходимость исследования подобного класса грамматик возникает из практических соображений. Часто первоначальный синтаксический анализ удобно производить совместно с семантическим анализом.

Фигур – 3, библ. наименований – 12.

УДК 681.3.06:518.5

Пакет макрокоманд для структурного программирования на языке Ассемблер. Бернштейн Е.Б., Шмундак А.Л. "Труды Таллинского политехнического института", № 426, 1977, с. 43–52.

Описывается пакет макрокоманд, облегчающих структурное программирование на Ассемблере. Приводятся правила использования и печатаемые сообщения об ошибках. Дается ряд примеров.

Фигур – 4, таблиц – 1, библ. наименований – 2.

УДК 518.512.25

Задачи со средними значениями случайных величин. Юби Э.А.-Ю. "Труды Таллинского политехнического института", № 426, 1977, с. 53–57.

Рассматриваются две системы линейных алгебраических уравнений. Коэффициенты первой системы – математические

ожидания некоторых случайных величин, коэффициенты второй системы - статистические оценки этих математических ожиданий. Найдены условия, когда решение второй системы является состоятельной оценкой решения первой системы.

Библ. наименований - 4.

УДК 681.142.2

Параллельные алгоритмы решения двух задач оптимизации. Амитан И.И. "Труды Таллинского политехнического института", № 426, 1977, с. 59-82.

Предполагается существование многопроцессорной вычислительной системы с неоднородным управлением и общей памятью. Рассматриваются параллельные алгоритмы решения классической транспортной задачи и многопродуктовой задачи размещения производства, для последней задачи предлагается двухпроцессорный и многопроцессорный вариант. Для описания алгоритмов используется алгоритмический "язык сигналов", являющийся расширением языка АЛГОЛ-68.

УДК 519.1

Еще раз о вспомогательных функциях приближения графа полными подграфами. Вейнер Г.А. "Труды Таллинского политехнического института", № 426, 1977, с. 83-86.

В статье продолжается изучение проблемы аппроксимации симметричного бинарного отношения R с помощью отношения эквивалентности E. В качестве условия аппроксимации принимается минимизация числа

$$|R \setminus E| + |E \setminus R|.$$

Полученный результат расширяет множество случаев возможного использования ранее полученных автором вспомогательных функций.

Фигур - 1, библ. наименований - 1.

Цена 69 коп.