УДК 681.142.2

# АВТОМАТИЧЕСКОЕ РАСПАРАЛЛЕЛИВАНИЕ АЛГОРИТМОВ СПЕЦИАЛЬНОГО ВИДА

В.Г.Кербель, Н.Н.Миренков

Описываются язык программирования для однородных вычислительных систем [1], позволяющий записывать алгоритмы специального вида, и методы получения в результате трансляции с этого языка параллельных программ в виде совокупности взаимодействующих ветвей [2].

Рассматриваемые алгоритмы представляют собой совокупность крупных блоков - подалгоритмов. Каждый блок есть тройка: $A_i=\{\alpha_i, \bar{A}_i, b_i\}$, где $\bar{A}_i$ - ядро блока, для которого $\alpha_i$ - непосредственно предшествующий, $b_i$ - следующий операторы; $\alpha_i$ - характеризует входные функционально-временные связи $\bar{A}_i$, $b_i$ - выходные, причем $\alpha_i$ и $b_i$ могут быть пустые.

Предложенный язык есть набор операторов, позволяющих описывать указанные функционально-временные связи, определенные операторами $\alpha_i$ и $b_i$. Его синтаксис задается в следующем виде:

⟨ программа ⟩ ::= ⟨ оператор ⟩ | ⟨ программа ⟩ ⟨ оператор ⟩

⟨ оператор ⟩ ::= ⟨ имя ядра блока ⟩ , ⟨ тип оператора ⟩ | ⟨ оператор конца ⟩

⟨ имя ядра блока ⟩ ::= ⟨ буква ⟩ | ⟨ имя ядра блока ⟩ ⟨ буква ⟩ | ⟨ имя ядра блока ⟩ ⟨ цифра ⟩

⟨ тип оператора ⟩ ::= ⟨ оператор входа ⟩ | ⟨ оператор выхода ⟩

⟨ оператор конца ⟩ ::= КОНЕЦ

⟨ буква ⟩ ::= А | Б | В | Г | Д | Е | Ё | Ж | З | И | Й | К | Л | М | Н | О | П | Р | С | Т | У | Ф | Х | Ц | Ч | Ш | Щ | Ъ | Ы | Ь | Э | Ю | Я

⟨ цифра ⟩ ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

⟨ оператор входа ⟩ ::= ПРИЕМ ( ⟨ простой вход ⟩ ) | ПРИЕМ ⟨ обобщенный вход ⟩

⟨ простой вход ⟩::=⟨ конструкция ⟩|⟨ простой вход ⟩⟨ конструкция ⟩

⟨ конструкция ⟩::=⟨ список имен ⟩;⟨ набор массивов ⟩|⟨ список имен ⟩;⟨ набор массивов ⟩;⟨ конструкция⟩

⟨ список имен ⟩::=⟨ имя ядра блока ⟩|⟨ имя ядра блока ⟩,⟨ список имен⟩

⟨ набор массивов ⟩::=⟨ имя массива ⟩|⟨ имя массива ⟩,⟨ набор массивов⟩

⟨ обобщенный вход⟩ ::=⟨ строка⟩|⟨ обобщенный вход⟩ ⟨ строка⟩

⟨строка ⟩::=⟨ номер ветви ⟩(⟨ спецификация ⟩)

⟨ спецификация ⟩::=⟨ список обобщенных имен ⟩;⟨ набор массивов ⟩|⟨ список обобщенных имен ⟩;⟨ набор массивов ⟩;⟨ спецификация ⟩

⟨ номер ветви ⟩ ::=⟨ цифра ⟩|⟨ цифра ⟩⟨ номер ветви ⟩

⟨ список обобщенных имен ⟩::=⟨ имя ядра блока ⟩,⟨ указатель ветви ⟩|⟨ имя ядра блока ⟩,⟨ указатель ветви ⟩,⟨ список обобщенных имен ⟩

⟨ указатель ветви ⟩::=⟨ номер ветви ⟩|⟨ указатель ветви ⟩,⟨ номер ветви ⟩

⟨ оператор выхода ⟩::= ПЕРЕДАЧА (⟨ простой выход ⟩)| ПЕРЕДАЧА ⟨ обобщенный выход ⟩

⟨ простой выход ⟩::=⟨ время ⟩|⟨ время ⟩;⟨ простой вход ⟩

⟨ обобщенный выход ⟩ ::= (⟨ ранг блока ⟩;⟨ время ⟩)|(⟨ ранг блока ⟩,⟨ время ⟩)⟨ обобщенный вход ⟩

⟨ ранг блока ⟩::=⟨ цифра ⟩|⟨ цифра ⟩⟨ ранг блока ⟩

⟨ время ⟩::=⟨ цифра ⟩|⟨ цифра ⟩⟨ время ⟩

Если в операторе ПРИЕМ массив собирается от различных ядер и ветвей, то сборка эта осуществляется встык (в порядке записи источников информации).

ПРИМЕР I. Рассмотрим алгоритм, структура которого представлена на рис.I.

Запись на предложенном языке имеет вид:

I. A2,ПРИЕМ(AI;B2)
2. A3,ПРИЕМ(AI;B3)
3. A4,ПРИЕМ(AI,A3;B4;A3;B7)
4. A5,ПРИЕМ(A2,A4;B5,B6)
5. AI,ПЕРЕДАЧА(TI;A2,A3;CI;A4;C6)
6. A2,ПЕРЕДАЧА(T2;A5;C2)
7. A3,ПЕРЕДАЧА(T3;A4;C3,C5)
8. A4,ПЕРЕДАЧА(T4;A5;C4)
9. A5,ПЕРЕДАЧА(T5)
IO. КОНЕЦ



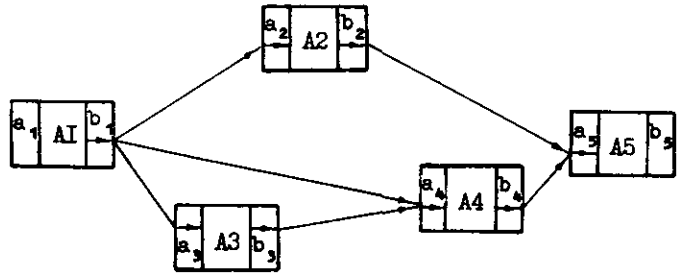



Рис.I

Здесь $Ai$ - имя ядра блока $i \in \{ I,2,3,4,5 \}$; $Ti$ - ориентировочное время реализации $Ai$; $Ci$ - имя передаваемого массива; $Bi$ - имя массива, в который записывается принимаемая информация. Операторы левого столбца соответствуют входным функционально-временным связям блоков, правого - выходным. Например, третий оператор означает следующее: блок с ядром A4 должен принять информацию от блоков с ядрами AI, A3 и поместить её в массив B4, затем принять от A3 следующую информацию и поместить её в массив B7.

ПРИМЕР 2. Рассмотрим алгоритм, структура которого приведена на рис.2.

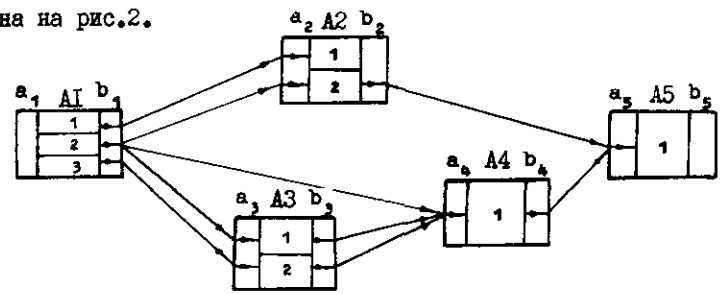



Рис. 2

Его запись имеет вид:

I. A2, ПРИЕМ
I (AI, I; L I)
2 (AI, 2; L I)

2. A3, ПРИЕМ
I (AI, 2; L 2)
2 (AI, 3; L 2)

6. A2, ПЕРЕДАЧА (2,T2)
2 (A5, I; M2)

7. A3, ПЕРЕДАЧА (2,T3)
I (A4, I; M3)
2 (A4, I; M3)

```
3. A4, ПРИЕМ
   I (A1,2; L3;A3,2,I; L4)
4. A5, ПРИЕМ
   I (A2,2; L5;A4,I; L6)
5. A1, ПЕРЕДАЧА (3,T1)
   I (A2,I; M1)
   2 (A2,2,A4,I;M1;A3,I;M5)
   3 (A3,2;M1)
8. A4, ПЕРЕДАЧА (I,T4)
   I (A5,I; M4)
9. A5, ПЕРЕДАЧА (I,T5)
10. КОНЕЦ
```

В данном случае описан алгоритм, ядра блоков которого являются р-алгоритмами; использованы операторы обобщенного входа и выхода. $Li$ и $Mi$ – имена массивов соответственно для $Bi$ и $Ci$, определенных в примере I. Здесь шестой оператор означает, что блок с ядром A2, ранг которого два и ориентировочное время решения T2, должен передать из своей второй ветви массив M2 в первую ветвь блока с ядром A5.

Данный язык предназначен для программирования функционально-временных связей сложных задач, структуры которых могут быть отражены сетевыми графиками или граф-схемами, в которых любой блок (вершина) есть подпрограмма, имеющая один вход и один выход. Эти подпрограммы могут составляться различными людьми и на произвольных языковых средствах, имеющихся в составе математического обеспечения машины, на базе которой построена однородная вычислительная система (ОВС). Условием запуска подпрограммы является готовность всех её входных массивов для выполнения оператора ПРИЕМ.

Проблема распараллеливания алгоритмов, структурные схемы которых подобны описанным выше, уже рассматривалась в литературе (так называемые схемы без ветвлений [3]). Однако эти исследования опирались на предположение о реализации параллельного счета на вычислительной системе с общим полем памяти, и, кроме того, ядра блоков рассматривались на уровне одного или нескольких операторов. Особенность данного подхода связана с крупноблочным представлением алгоритма, ядра блоков которого сами могут быть р-программами, и с необходимостью минимизации обменов между ветвями параллельного процесса.

Общая задача ставится следующим образом: дано $N$ работ $A_i = \{R_i, T_i\}$ и их функционально-временные связи $V_{ij}$; $R_i$ – ранг работы (количество машин, необходимых для её выполнения), причем $\max_i R_i = L_0$, $T_i$ – предполагаемое время выполнения работы



(включая время приема и передачи информации), $V_{ij}$ – объем информации, передаваемой работой $A_i$ работе $A_j$. Если между $A_i$ и $A_j$ существует только временная зависимость, то $V_{ij} = \sigma$ ($\sigma$ – константа реализации, равная минимально возможному объему передаваемой информации). В остальных случаях $V_{ij} = 0$. Число машин в ОВС – $L_1$. Требуется, удовлетворяя функционально-временные связи, выбрать такое число $L \in L_0, L_1$ машин и разбить все множество работ $G = \bigcup_{i=1}^{N} A_i$ на $L$ подмножеств $G_1, G_2, \ldots, G_L$, образующих ветви р-программы, чтобы все работы были выполнены за время не большее $T$ и при этом минимизировалась целевая функция

$$F = F_1 + F_2 = \sum_{i=1}^{N} \sum_{j=1}^{N} \delta_{ij} \cdot V_{ij} + \sum_{i=1}^{k} r_i \cdot t_i,$$

при следующих предположениях

$$\sum_{i=1}^{N} R_i \gg L_1; \quad \sum_{i=1}^{N} R_i T_i \le L_1 \cdot T,$$

где $t_i$ – $i$-й интервал времени выполнения работ, в течение которого функциональное состояние ОВС остается неизменным [4], $\sum_{i=1}^{k} t_i \le T$; $r_i$ – число машин, не занятых выполнением работ множества $G$ в течение интервала $t_i$;

$$\delta_{ij} = \begin{cases} \alpha 1 \text{ или } 0, \text{ или } \alpha 2, \text{ если } A_i \in C_k \cap G_\ell,\ A_j \in C_{k+\nu} \cap G_\ell,\ \nu \ge 1, \text{ и} \\ \text{соответственно требуется использование внешней памяти, не требуется использования внешней памяти, необходима пересылка данных в пределах оперативной памяти (ОП) одной машины;} \\ \alpha 3 \text{ или } \alpha 1 + \alpha 3, \text{ если } A_i \in C_k \cap G_\ell,\ A_j \in C_{k+\nu} \cap G_m,\ \nu \ge 1,\ \ell \ne m, \\ \text{и соответственно не требуется, требуется использование внешней памяти,} \end{cases} \quad (1)$$

где $\alpha 1$ – время записи и чтения слова для внешней памяти; $\alpha 2$ – время пересылки слова в пределах одной ОП; $\alpha 3$ – время передачи слова в другую машину; $C_k$ – множество работ, выполняемых в $k$-й интервал времени.

Для реализации этой задачи предлагается использовать эвристические алгоритмы, позволяющие получать удовлетворительные для практики решения.

Функции $F_1$ и $F_2$ независимы и неотрицательны, поэтому для минимизации $F$ достаточно поочередно минимизировать $F_1$ и $F_2$.

В силу этого, поставленную задачу предлагается решать в два этапа.

На первом этапе, считая все $V_{ij}$ равными $\delta$, т.е. полагая существование только временных связей между работами, выбираем такое $L_0$, чтобы время $t$ выполнения всех работ не превышало $T$, а затем, варьируя $L$ в $\overline{L_0', L_1}$, добиваемся максимальной загрузки машин, что приводит к минимизации $F_2$ [4]. Значение $t$ при этом может существенно уменьшиться. Результат первого этапа есть матрица $\|X\|$, отражающая разбиение множества $G$ на соответствующее число $L$ подмножеств $G_\ell : G = \bigcup_{\ell=1}^{L} G_\ell$. Матрица $\|X\|$ имеет размерность $t \times L$, её элементами $x(i,j)$ являются номера работ, которые выполняются на $j$-й машине ОВС в $i$-й квант времени.

На втором этапе, учитывая соответствующие $V_{ij}$, т.е. функциональную связь между работами, взаимозаменяем элементы подмножеств $G_\ell$ таким образом, чтобы минимизировать функцию $F_1$. Алгоритм минимизации $F_1$ опирается на следующие понятия и теоремы.

Работы (подпрограммы) $\mathcal{A}_i$ и $\mathcal{A}_j$ называются непосредственно связанными, если существует дуга (но не путь) в граф-схеме, соединяющая вершины, которые соответствуют этим работам.

Преобразование $\gamma$ матрицы $\|\Omega\|$ назовем п р о с т ы м, если матрица $\|\gamma(\Omega)\|$ отличается от $\|\Omega\|$ перестановкой двух строк. Две квадратные матрицы $\|\Omega^1\|$ и $\|\Omega^2\|$ с элементами соответственно $\omega^1(i,j)$ и $\omega^2(i,j)$ называются э к в и в а л е н т н ы м и ($\|\Omega^1\| \approx \|\Omega^2\|$), если существует последовательность простых преобразований $\gamma_1, \dots, \gamma_k$, преобразующих матрицу $\|\Omega^1\|$ в матрицу $\|\Omega^2\|$. Р а з б и е н и е матрицы задается разбиением множества $M$ её строк на подмножества $M_i : M = \bigcup_{i=1}^{k} M_i$ и $M_i \cap M_j = \phi$, $i \neq j$. Мощность $d(M_i)$ множества $M_i$ есть число строк, содержащихся в этом множестве.

Пусть строки матрицы пронумерованы от I до $n$; $1 = j_1 < j_2 < \dots < j_{k+1} = n+1$ – точки деления промежутка $[1, n+1]$ на $k$ частей, которые задают разбиение множества $M$, причем $M_i$ содержит строки с номерами от $j_i$ до $j_{i+1}-1$, $d(M_i) = j_{i+1} - j_i$. С л е д о м множества $M_i$ называется величина $S(M_i) = \sum_{\alpha=0}^{j_{i+1}-j_i-1} \omega(\alpha+j_i, \alpha+j_i)$;



т е н ь ю этого множества на $M_\ell$ (при $d(M_i) = d(M_\ell)$, $i \neq \ell$) – величина $S^\ell(M_i) = \sum_{\alpha=0}^{j_{i+1}-j_i-1} \omega(\alpha+j_i, \alpha+j_\ell)$.

Для множества эквивалентных матриц

$$\Omega = \{\|\Omega^k\| / \|\Omega^i\| \approx \|\Omega^j\|;\ i,j = 1,2,\dots,n!\}$$

справедливы следующие теоремы.

ТЕОРЕМА 1. Е с л и $S(\Omega^k) = \min_{\alpha} S(\Omega^\alpha)$, т о

$$\forall i,j = 1,\dots,n \{\omega^k(i,i) - \omega^k(i,j) + \omega^k(j,j) - \omega^k(j,i) \le 0\} \quad (2)$$

ТЕОРЕМА 2. Е с л и д л я н е к о т о р о й м а т р и ц ы $\|\Omega^\alpha\| \in \Omega$ в ы п о л н е н о у с л о в и е (2), т о

$$\sum_{j=1}^{n} \min_{i} \omega^\alpha(i,j) \le S(\Omega^\alpha) \le \frac{\sum_{i=1}^{n} \left( \sum_{j=1}^{n} \omega^\alpha(i,j) - \omega^\alpha(j,j) \right)}{n-1}. \quad (3)$$

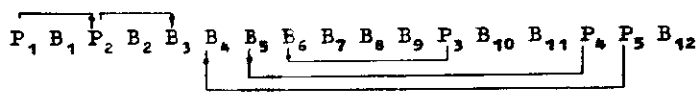
Идея минимизации функции $F_1$ заключается в следующем. Рассматривается случай $R_i = 1$, $T_i = 1$, $i \in \overline{1,N}$. Для определенности рассмотрим минимизацию обменов по истечении первого кванта времени, т.е. рассмотрим первую строку матрицы $\|X\|$, $x(1,i)$ – номер работы $\mathcal{A}^*$, выполняемой на $i$-й машине в первый квант времени. В матрице $\|Z\|$ (соответствующей матрице $\|\Omega\|$) размерности $L \times L$ элементу $z(i,j)$ присваиваем значение времени, необходимого для передачи информации из $\mathcal{A}^*$ работам, непосредственно связанным с $\mathcal{A}^*$ и назначенным на выполнение в $j$-ю машину. Таким образом, матрица $\|Z\|$ характеризует все возможные передачи информации по истечении первого кванта времени решения задачи. Элемент $z(j,j)$ есть время передачи информации внутри $j$-й ветви р-программы. В идеальном случае $z(j,j) = 0$, т.е. минимальное время обменов возникает при малых значениях диагональных элементов. Минимизация $S(Z)$ осуществляется последовательностью простых преобразований; строки $i$ и $j$ переставляются, если для них не выполнено условие (2). Критерием минимальности $S(Z)$ является выполнение условия (2) для матрицы $\|Z\|$. Ясно, что перестановке строк $i$ и $j$ матрицы $\|Z\|$ соответствует перестановка $i$-го и $j$-го элементов первой строки матрицы $\|X\|$.

Если не все $T_i = 1$, то соответствующую работу представляем последовательностью непосредственно связанных работ. Объем пе-

редаваемой информации между этими работами $\tilde{V} > \max_{i,j} V_{ij}$. Такое значение $\tilde{V}$ обеспечивает выполнение всей работы в одной машине, но за несколько квантов времени.

В случае $R_i \neq 1$, т.е. наличия р-программ, в матрице $\|\mathcal{Z}\|$ каждой из них соответствует последовательное число строк, равное рангу этой р-подпрограммы. Каждая из этих строк характеризует время обмена соответствующей ветви р-подпрограммы, с л е д этого множества строк – время обмена внутри соответствующих подмножеств $G_\ell$, т е н ь – время обмена между ветвями р-подпрограммы равного ранга. Каждая р-подпрограмма может быть решена на связном множестве машины без наличия транзитных. Поэтому для минимизации $S(\mathcal{Z})$ в матрице $\|\mathcal{Z}\|$ можно переставлять только множество строк, а в качестве элементов (см.первый случай) используются значения следа и тени этих множеств.

Алгоритм минимизации функции $F_1$ может быть представлен схемой:




$P_1$ – передает управление на $P_2$, если все $R_i = 1$;

$B_1$ – представляет каждую работу $\mathcal{A}_i$ с $R_i > 1$ вектором работ

$$\mathcal{A}_i = \{\mathcal{A}_i^1, \mathcal{A}_i^2, \ldots, \mathcal{A}_i^{R_i}\};$$

$P_2$ – передает управление на $B_3$, если $T_i = T_j$ для всех $i$ и $j$;

$B_2$ – выбирает квант времени $\tau$: $\forall T_i \exists t_i \{T_i = t_i \cdot \tau\}$ и представляет каждую работу $\mathcal{A}_i$ как последовательность работ $\mathcal{A}_{i1}, \ldots, \mathcal{A}_{i,t_i}$, при этом объем передаваемой информации между $\mathcal{A}_{im}$ и $\mathcal{A}_{i,m+1}$ $(m = 1, \ldots, t_i - 1)$, считается равным $\tilde{V}$ ( $\tilde{V}$ – функция от $\tau$ и параметров ОВС);

$B_3$ – принимает от первого этапа число $t$ временных квантов, за которые спланирована реализация всех работ, и матрицу распределения работ $\|X\|$ ( $t$ равно числу строк матрицы $\|X\|$), причем $G_\ell = \{X(i,t) \in \|X\| \,/\, i = 1,2,\ldots,t\}$, $\ell = 1,2,\ldots,L$; $\Sigma' := C$, $C$ – максимальное число, представимое в элементарной машине ОВС;

$B_4$ – $\xi := 1$, $\Sigma := 0$;

$B_5$ – $\nu := 1$;



$B_6$ – выбирает работу $\mathcal{A}_{\xi,\nu} = x(\xi,\nu) \in \|X\|$;

$B_7$ – выбирает подмножество работ $\mathcal{A}_{i_1}, \ldots, \mathcal{A}_{i_k} \in G$, непосредственно связанных с $\mathcal{A}_{\xi,\nu}$;

$B_8$ – для $\ell = 1, \ldots, L$ выбирает подмножество работ $Y_\ell = \bigcup_{j=1}^{k} \mathcal{A}_{i_j} \cap G_\ell$;

$B_9$ – учитывая (I), вычисляет время взаимодействия работ $\mathcal{A}_{\xi,\nu}$ с $Y_\ell$ и присваивает это значение элементу $z(\nu, \ell)$ матрицы $\|\mathcal{Z}(L,L)\|$; если $x(\xi, \nu) = 0$, то $\nu$-я строка матрицы $\|\mathcal{Z}\|$ – нулевая;

$P_3$ – $\nu := \nu + 1$ и передает управление на $B_6$, если $\nu \leq L$;

$B_{10}$ – переставляет строки матрицы $\|\mathcal{Z}\|$ и элементы строки $\xi$ матрицы $\|X\|$, пока для всех $i,j = 1, \ldots, L$ не выполнится условие:

$$z(i,i) - z(i,j) + z(j,j) - z(j,i) \leq 0, \qquad (4)$$

причем перед каждой перестановкой элементы из (4) пересчитываются с помощью правил: $\alpha 1 \leftrightarrow \alpha 1 + \alpha 3, \alpha 2 \leftrightarrow \alpha 3, 0 \leftrightarrow \alpha 3$.

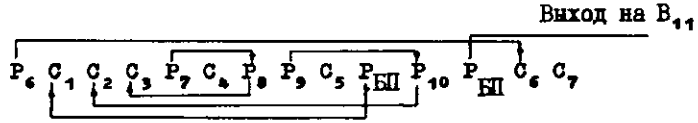
$B_{11}$ – $\xi := \xi + 1, \Sigma := \Sigma + S(\mathcal{Z})$;

$P_4$ – передает управление на $B_5$, если $\xi \leq t$;

$P_5$ – передает управление на $B_4$, если $\Sigma' > \Sigma$ (при этом $\Sigma' := \Sigma$);

$B_{12}$ – анализирует преобразованную матрицу $\|X\|$ и вставляет, по необходимости, операторы системных взаимодействий [2] и/или обращения к внешним устройствам.

Оператор $B_{10}$, в свою очередь, представляется схемой:




$P_6$ – передает управление на $C_6$, если хотя бы одна работа строки $\xi$ матрицы $\|X\|$ имеет ранг $R > 1$;

$C_1$, $C_2$, $C_3$ – соответственно $i := 1$; $\omega := \gamma := 0$, $j := i$; $j := j + 1$;

$P_7$ – передает управление на $P_8$, если

$$\beta = z(i,i) - z(i,j) + z(j,j) - z(j,i) \leq \omega; \qquad (5)$$

$C_4$ – $\gamma := j$, $\omega := \beta$;

$P_8$ – передает управление на $C_3$, если $j < L$;

$P_9$ – передает управление на $P_{10}$, если $\omega = 0$ ;

$C_5$ – переставляет строки $r$ и $i$ матрицы $\|Z\|$ и соответствующие этим строкам элементы строки $\xi$ матрицы $\|X\|$;

$P_{БП}$ – оператор безусловной передачи управления;

$P_{10}$ – $i := i + 1$ и передает управление на $C_4$, если $i < L$ ;

$C_6$ – используя подалгоритм $\{C_1 \div P_{10}\}$, минимизирует следы подмножеств, представляющих собой строки матрицы $\|Z\|$, соответствующие работам с рангом, бóльшим I; в каждом подмножестве число строк равно рангу работы;

$C_7$ – для каждого множества подмножеств равной мощности выполняет подалгоритм $\{C_1 \div P_{10}\}$, в котором роль элементов (строк матрицы $\|Z\|$) играют подмножества, а условие (5) заменяется условием:

$$S(M_i) - \min S^j(M_i) + S(M_j) - \min S^i(M_j) \le \omega,$$

где $\min S^j(M_i)$ – минимальная тень подмножества $M_i$ на $M_j$.

ЗАМЕЧАНИЕ. Оценку значения следа матрицы, получаемого алгоритмом, дает неравенство (3).

Эффективность предложенного подхода проверена с помощью специальных моделирующих программ.

Предложенная методика позволяет получать р-программы автоматически. Она может быть применена в операционной системе для решения задач планирования и пространственного размещения подсистем.

## Литература


I. ЕВРЕИНОВ Э.В., КОСАРЕВ Ю.Г. Однородные вычислительные системы высокой производительности. Новосибирск, "Наука", I966.

2. ГРИШАЕВА Н.К., КЕРБЕЛЬ В.Г., КОЛОСОВА Ю.И., КОНСТАНТИНОВ В.И., КОРНЕЕВ В.Д., ЛЕВАГИНА Т.А., МИРЕНКОВ Н.Н., ФИШЕРМАН С.Б. Язык параллельных алгоритмов. –В кн.: Вычислительные системы. Вып. 57. Новосибирск, I973, с. 33–54.

3. КОТОВ Е.В. Теория параллельного программирования. Прикладные аспекты. –"Кибернетика", I974, № I, с. 3–I7.

4. КРЫЛОВ Э.Г., МИРЕНКОВ Н.Н. Алгоритмы планирования функциональных состояний однородной вычислительной системы. –Настоящий сборник, с. 29–43.



Поступила в ред.-изд.отд.
II февраля I975 года