В Ы Ч И С Л И Т Е Л Ь Н Ы Е С И С Т Е М Ы

Сборник трудов

1965 г. Института математики СО АН СССР Выпуск I7

О МЕТОДИКЕ РЕШЕНИЯ ЗАДАЧ НА УНИВЕРСАЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ

Ю.Г. Косарев

Универсальные вычислительные системы (УВС) по сравнению с обычными электронными вычислительными машинами (ЭВМ) обладают двумя основными особенностями, существенным образом сказывающимися на решении задач: одновременностью выполнения большого числа операций и возможностью программного изменения структуры как самих элементарных машин, так и связей между ними [I].

Первая особенность налагает, вообще говоря, ограничения на круг задач, решаемых на УВС. Однако на практике это ограничение не имеет сколько-нибудь большого значения, так как на УВС не могут решаться только такие задачи, в процессе решения которых требуется выполнить последовательность длиной более чем 10^{10} - 10^{13} операций, каждая из которых не может начаться до окончания предыдущей. В настоящее время автору не известно ни одной практической задачи подобного рода. Трудно ожидать, что значительное число таких задач появится в будущем. С меньшим же числом операций (чем 10^{10} операций) могут справиться и обычные ЭВМ. Следует отметить, что поскольку для каждой задачи существует обычно не единственный метод решения и тем более не один алгоритм, то указанное ограничение можно пытаться обойти за счет выбора соответствующего алгоритма.

Вопрос о свойствах алгоритмов с точки зрения возможности представления процесса их реализации в виде большого числа параллельно выполняемых операций до сих пор изучался мало. Некоторые подходы к рассмотрению алгоритмов с параллельными ветвями вычислений (параллельных алгоритмов) и особенностей их реализации на вычислительных системах были рассмотрены ранее в [2]. В данной и последующих статьях сборника [3], [4] содержится детализация и дальнейшее развитие этих подходов, иллюстрируемых примерами решения на УВС некоторых важных классов задач.

Вторая особенность УВС, программное изменение ее структуры, мало применяется в существующей практике. Какой-либо теории использования подобных устройств еще нет. Программное управление структурой УВС открывает широкие возможности для увеличения производительности УВС, делает возможным приспосабливать УВС под любую задачу и существенно облегчает программирование задач (в нынешнем понимании этого термина), благодаря возможности записывать программу каждой задачи на наиболее подходящем для нее языке. Преимущества переменной структуры: возможность произвольно увеличивать число элементарных машин (ЭМ), увеличивать объем памяти ЭМ, либо усложнять их логические схемы для выполнения более сложных команд, менять коммутации между ЭМ и т.п. - несомненны и могут давать существенный эффект даже при составлении программ настройки вручную простейшим, не претендующим на оптимальность способом [3].

Безусловно разработка оптимальных методов настройки структуры под данный алгоритм решения задачи может дать значительно больший эффект. В этой области, однако, трудно ожидать быстрых результатов, так как возникающая проблема в качестве составных элементов содержит проблему оптимального программирования и проблему оптимального синтеза вычислительных систем, каждая из которых еще далека от завершения.

В качестве задач, возникающих при этом, следует в первую очередь назвать: 1) выбор набора операций для каждой ЭМ, исходя из конкретного алгоритма, 2) составление плана распределения частей вычислительного процесса между ЭМ и их объединениями на различных иерархических уровнях, 3) распределение потоков информации по каналам связи, 4) составление программы работы УВС.

В конечном счете все четыре указанные задачи должны решаться самой УВС. Для решения первой задачи должны быть разработаны алгоритмы, которые производили бы анализ решаемой зада-

чи и на этой основе определяли оптимальный набор команд каждой ЭМ. Вторая задача по своему характеру аналогична задачам линейного и динамического программирования. Для ее решения, по-видимому, можно будет в значительной мере использовать готовый математический аппарат [5], [6]. Задача распределения потоков информации между каналами связи сходна с транспортными задачами, рассматриваемыми в математической экономике. Для этих задач также имеется развитый математический аппарат [7]-[9]. Четвертая задача включает в себя программу настройки, программу работы каждой ЭМ и УВС в целом, программу обмена информацией между ЭМ и внешними объектами.

Все четыре указанные задачи тесно связаны друг с другом и должны решаться совместно, как это делается в сложных системах [10].

В настоящее время, когда мы еще далеки от создания оптимальных методов решения задач подобного рода, приходится ограничиваться теми приемами и методами, которые выработаны в практике ручного программирования и ручного синтеза вычислительных машин и систем. Однако, как говорилось выше, этот путь уже позволяет достаточно эффективно решать на УВС практические задачи.

§ I. Основные понятия и определения

I. Согласно А.А. Ляпунову [11] любой алгоритм может быть представлен в операторной форме или в терминологии Ю.И. Янова [12] в виде логической схемы. Логическая схема алгоритма представляет собой конечную строчку, составленную из заданного набора операторов $(A_1, A_2, \ldots, A_n)$, предикатов и двух вспомогательных символов $\underset{i}{\lfloor}, \underset{i}{\rfloor}$ $(i = 1, 2, \ldots)$, называемых соответственно левой и правой полускобками. При этом строчка $A_{i_1}, A_{i_2}, \ldots, A_{i_s}$ означает, что должны быть последовательно выполнены операторы $A_{i_1}, A_{i_2}, \ldots, A_{i_s}$. Строчка $A_{i_1} \alpha \underset{m}{\lfloor} A_{i_2} \ldots \underset{m}{\rfloor} A_{i_s} \ldots$, где α - некоторый предикат, означающий, что после выполнения оператора A_{i_1}, при $\alpha = 1$, должен быть выполнен оператор A_{i_2}, стоящий непосредственно за левой полускобкой, при $\alpha = 0$ - оператор A_{i_s}, стоящий за правой полускобкой.

Часто употребляют более наглядную запись логической схемы $A_{i_1} \overset{\alpha}{\underset{1}{\lfloor}} A_{i_2} \ldots \underset{1}{\rfloor} A_{i_s} \ldots$. При этом стрелку, соответствующую $\alpha = 1$, помещают над строчкой, а стрелку, соответствующую $\alpha = 0$, -

под строчкой. Стрелку, соответствующую переходу к соседу справа, обычно опускают. Если переход от какого-либо оператора к его соседу справа отсутствует, то их принято разделять точкой с запятой.

2. На операторы $A_1, A_2, \ldots, A_n$ и предикаты не налагаются какие-либо ограничения, кроме их выполнимости.[x] Вопрос о выполнимости оператора и алгоритма в целом требует уточнения. В теории алгоритмов под требованием выполнимости алгоритма понимается, что он может быть реализован за конечное число шагов, на каждом из которых выполняется какая-либо элементарная операция из заданного набора. При этом вопрос о самом числе шагов игнорируется. На несовершенство такого подхода применительно к теории автоматов указал еще Дж. Нейман [13]. Конечность числа шагов еще не означает практической реализации данного алгоритма, так как необходимо, чтобы это число было не только конечно, но и меньше некоторого числа N_{np}, определяемого предельным количеством элементарных операций, которое может быть осуществлено на данном уровне вычислительной техники. Кроме того, в теории алгоритмов [14] обычно не налагается каких-либо ограничений на объем памяти, потребный для запоминания исходной и промежуточной информации, т.е. считается, что память может быть сколь угодно большой. На практике эта величина V_{np} ограничена достигнутым техническим уровнем. Величины N_{np} и V_{np} непостоянны и увеличиваются по мере появления более производительных ЭВМ и ВС. Эти величины нельзя рассматривать оторванно друг от друга, поэтому называя величины предельного числа операций N_{np} и предельный объем памяти V_{np}, необходимо указывать конкретное техническое средство, их реализующее. Кроме N_{np}, требуется также указать допустимое время работы над одним алгоритмом. Таким образом, можно для каждого вычислительного устройства M_i указать N_{npi} – предельно выполнимое им число операций, содержащихся в списке команд, и V_{npi} – предельный объем хранимой информации. Записав данный алгорити в виде последовательности команд данного вычислительного устройства (программы) и подсчитав общее число команд N, которое должно быть выполнено для реализации данного алгоритма, и наибольший объем хранимой информации V, а также

---

x) Далее для краткости мы будем пользоваться термином „оператор", подразумевая под этим и предикаты, если это не оговаривается особо.

сравнив их соответственно с N_{npi} и V_{npi}, можно сказать, может ли данный алгоритм быть реализован данным вычислительным устройством. Условимся в этом случае говорить, что данный алгоритм выполним данным вычислительным устройством, если

$$N \leqslant N_{npi} \; ; \; V \leqslant V_{npi} \; , \qquad (I)$$

и не выполним - в противоположном случае. Если же данный алгоритм практически невыполним ни одним из существующих вычислительных устройств, то назовем его практически невыполнимым.

3. В некоторых случаях удобно, чтобы в схемах алгоритмов правая полускобка всегда была правее соответствующей (с тем же индексом) левой полускобки, т.е. чтобы логические схемы алгоритмов не имели петель.

Очевидно, что любая логическая схема практически выполнимого алгоритма может быть преобразована в беспетлевую. Действительно, для этого достаточно повторить всю часть схемы, заключенную между полускобками, принадлежащими данному предикату, столько раз, сколько это требуется по условию задачи. Из практической выполнимости данного алгоритма вытекает, что это число повторений должно быть заведомо меньше N_{np}, что и указывает на справедливость данного утверждения.

При построении беспетлевой схемы из схем с петлями каждая петля с известным числом циклов ее повторения выписывается столько раз, сколько имеется циклов. Если число циклов является функцией от получаемых результатов вычислений, то в этом случае на основании требования выполнимости должно быть известно предельно допустимое число циклов C_m, при котором не нарушаются условия (I). Для построения беспетлевой схемы достаточно написать данный цикл в схеме C_m раз. Случай, когда имеется ряд петель с переменным числом циклов, про которые известно, что они в совокупности не нарушают условия (I), может быть описан беспетлевой схемой, в которой каждый цикл повторен столько раз, сколько он максимально может встретиться при решении данной задачи. То, что при этом число записанных операций в схеме может превысить N_{np}, не имеет в данном случае существенного значения, так как нас интересует соблюдение условия (I) только для числа выполняемых операций.

4. Введем понятие зависимости между операторами. Будем говорить, что: (1) оператор B непосредственно информационно зависит от оператора A, если оператор B производится над результатом оператора A; (2) оператор A непосредственно управляет операторами B и C, если результат оператора A определяет, какой из этих двух операторов должен выполняться, и если при этом оператор A непосредственно предшествует операторам B и C.

Заметим, что нередко между операторами могут быть одновременно и информационные и управляющие связи. Пусть, например, требуется выполнить над числом z оператор B, если $z<a$, и оператор C — в противном случае. Оператор A (сравнение числа z с a) в этом случае предшествует оператору B по информации и непосредственно управляет операторами B и C.

Для каждого алгоритма можно построить две схемы (схему информационных связей и схему управляющих связей) либо одну объединенную схему с указанием типа связи. Такие схемы могут быть определены в виде графов. Условимся далее для большой наглядности рисовать эти граф-схемы в беспетлевой форме (в виде дерева).

Рассмотрим в качестве примера умножение прямоугольных матриц A и B.

$$\begin{bmatrix} a_{11} a_{12} & a_{1n} \\ a_{21} a_{22} \cdot & a_{2n} \\ \cdots & \\ a_{m1} a_{m2} \cdot & a_{mn} \end{bmatrix} \times \begin{bmatrix} b_{11} b_{12} \cdot & b_{1p} \\ b_{21} b_{22} & b_{2p} \\ \cdots & \\ b_{n1} b_{n2} & b_{np} \end{bmatrix} = \begin{bmatrix} c_{11} c_{12} \cdot \cdot & c_{1p} \\ c_{21} c_{22} & \cdot c_{2p} \\ \cdots & \\ c_{m1} c_{m2} \cdots & c_{mp} \end{bmatrix}, \tag{2}$$

где $$c_{ij} = a_{i1} b_{1j} + a_{i2} b_{2j} + \quad + a_{in} b_{nj},$$

$$i = 1, 2, \quad , m; \quad j = 1, 2, \quad , p \tag{3}$$

Для простоты будем считать, что $n = 2^s$ (s — целое). Обозначим $d_{ijk} = a_{ik} b_{kj}$ ($i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, p$, $k = 1, 2, \ldots, n$). Тогда

$$c_{ij} = d_{ij1} + d_{ij2} + \cdots + d_{ijn}. \tag{4}$$

Граф-схема информационных связей при вычислении элемента матрицы c_{ij} показана на рис. I. На этом рисунке вершинами являются либо исходные данные (обозначены точками), либо результаты умножения или сложения двух чисел (обозначены прямоугольниками со знаком соответствующей операции).

Заметим, что хотя по форме граф-схемы связей близки к граф-схемам алгоритмов, предложенных Л.А.Калужниным [I5], но они отражают другие свойства алгоритмов. Граф-схемы Л.А.Калужнина устанавливают п о р я д о к выполнения операторов (в терминологии автора-операторов и распознавателей). В то же время граф-схемы связей показывают, какие о п е р а т о р ы д о л ж н ы б ы т ь п р е д в а р и т е л ь н о в ы п о л н е н ы д л я т о г о, ч т о б ы с т а л о в о з м о ж н ы м в ы п о л н е н и е данного оператора. В граф-схеме Л.А.Калужнина возможны, вообще говоря, такие изменения порядка выполнения отдельных операторов, которые никак не отражаются на результате.Граф-схема связей как раз и устанавливает возможные границы подобных изменений. Граф-схемы связей аналогичны сетевым графикам, применяемым в системе планирования типа PERT [I6] - [I8]. Как те, так и другие указывают, какие операции (работы) должны быть окончены к началу данной операции (работы).

Для ввода в ЭВМ удобно связи между операторами записывать в виде таблицы, в которой для каждого оператора указываются непосредственно предшествующие операторы. Естественно, что если таких операторов несколько, то они не должны зависеть друг от друга, т.е. ни один из них не предшествует и не управляет ни одним другим.

5. При заданном наборе операторов данный алгоритм может быть представлен различными логическими схемами. Например,выражения

$$\begin{aligned} &p_1 \underset{1}{\perp} \underset{2}{\perp} A_1 p_2 \underset{2}{\perp} p_3 \underset{3}{\perp} \underset{1}{\perp} A_2 \underset{3}{\perp} A_3 A_4 A_5 Я, \\ &\bar{p}_1 \underset{1}{\perp} A_2 p_3 \underset{2}{\perp} \underset{1}{\perp} \underset{3}{\perp} A_1 p_2 \underset{3}{\perp} \underset{2}{\perp} A_3 A_5 A_4 Я \end{aligned} \qquad (5)$$

с предикатами $p_1, \bar{p}_1$ и p_2, тождественно равными нулю предикатом p_3 и оператором конца $Я$ являются различными схемными записями одного и того же алгоритма.

Для того, чтобы отличать различные схемные записи одного алгоритма от схем различных, но эквивалентных по результатам

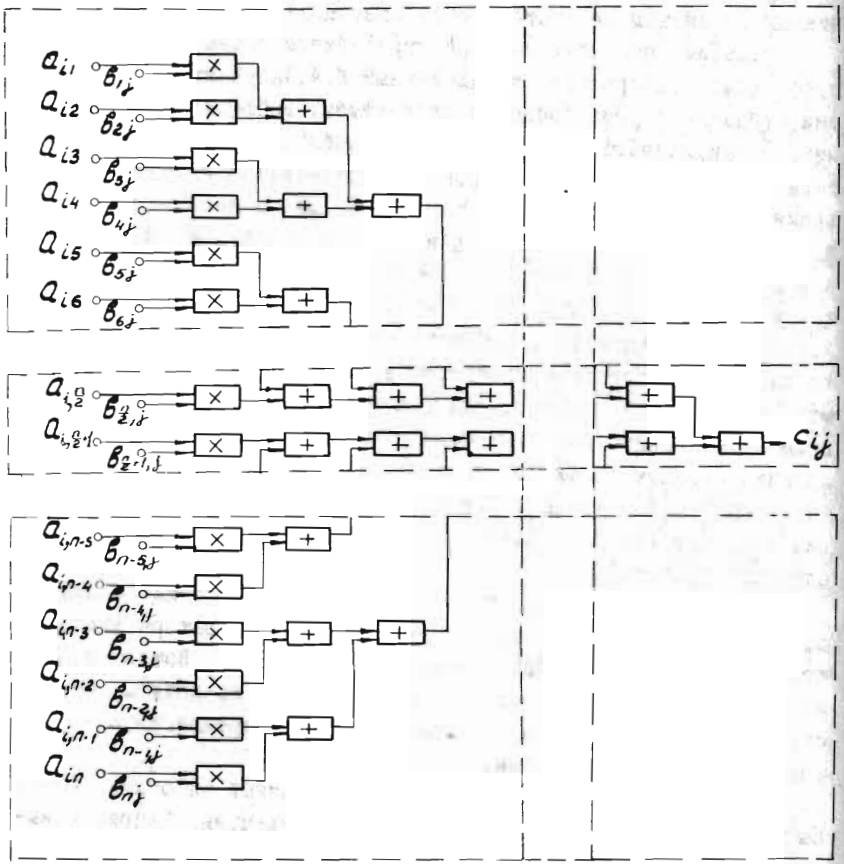
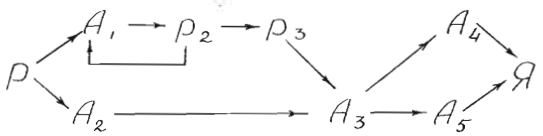

a_{i1} b_{1j}
a_{i2} b_{2j}
a_{i3} b_{3j}
a_{i4} b_{4j}
a_{i5} b_{5j}
a_{i6} b_{6j}
$a_{i,\frac{n}{2}}$ $b_{\frac{n}{2},j}$
$a_{i,\frac{n}{2}+1}$ $b_{\frac{n}{2}+1,j}$
c_{ij}
$a_{i,n-5}$ $b_{n-5,j}$
$a_{i,n-4}$ $b_{n-4,j}$
$a_{i,n-3}$ $b_{n-3,j}$
$a_{i,n-2}$ $b_{n-2,j}$
$a_{i,n-1}$ $b_{n-1,j}$
a_{in} b_{nj}
$n\,mp$ $\frac{n}{2}mp$ $\frac{n}{4}mp$ $\frac{n}{8}mp$ . . . $2mp$ mp
1 2 3 4 . . . S $S+1$

Рис. I.

алгоритмов, условимся в том случае, если двум схемным записям соответствует одна и та же схема связей, считать их эквивалентными схемами одного и того же алгоритма и схемами различных алгоритмов–в противоположном случае. Нетрудно убедиться, что схемы в приведенном выше примере не противоречат этому правилу, так как им соответствует одна и та же схема связей:



(6)

где p равно p_i , либо $\bar{p}_i$.

Как видно из указанного примера,схемные записи одного и того же алгоритма могут отличаться друг от друга значениями предикатов, а также порядком выполнения во времени взаимно независимых операторов.

Обычно используемые наборы операторов являются избыточными, то есть последовательность операторов может быть заменена другой эквивалентной по результатам последовательностью операторов, принадлежащих этому же набору. Условимся, что при такой замене получается новый алгоритм. Этот случай не следует смешивать с заменой некоторых последовательностей операторов о д н и м оператором, который обычно называют о б о б щ е н н ы м, что нужно рассматривать как подробную и сокращенную формы записи одного и того же алгоритма.

В соответствии с установившейся традицией будем различать схему алгоритма и схему программы. Принципиальное отличие между ними,состоит в том, что первая явно не зависит от конкретного устройства ее реализующего, а вторая - приспособлена к данной ЭВМ (или ВС) и может содержать операторы, учитывающие ее особенности, например, операторы обмена информацией между оперативной и вспомогательной памятями, изменение содержимого индекс-регистров и т.п. Применительно к УВС условимся в схемы алгоритмов включать операторы обмена информацией между элементарными машинами, операторы настройки и обобщенного условного перехода [I].

6. Введем некоторые определения. При этом будем использовать термин "операция", подразумевая, что все сказанное может быть отнесено к оператору, реализуемому данной операцией.

П р о с т о й о п е р а ц и е й, или просто о п е р а-

ц и е й, будем называть одно- или двуместную операцию над m-разрядными двоичными числами (или кодами), которая не превышает по сложности операции умножения либо деления двух m-разрядных чисел. Под с л о ж н о с т ь ю операции будем понимать наименьшее число функций $f_1(x_1,x_2)=\bar{x}_1\vee\bar{x}_2$ (штрих Шеффера), либо $f_2(x_1,x_2)=\bar{x}_1\&\bar{x}_2$ (стрелка Пирса), необходимых для её реализации за m рабочих тактов. Под рабочим тактом понимается дискретное автоматное время $t=0,1,2,\ldots,$ под m- наибольшее количество двоичных разрядов в числах, над которыми производятся операции. В дальнейшем рассмотрении будем предполагать, что величина m фиксирована. Возможность представления любой операции с помощью функции штрих Шеффера либо стрелки Пирса, следует из свойства полноты этих функций. В число простых операций попадают все операции, входящие в список команд обычных ЭВМ, за исключением $\sqrt{x}$, $\log x$ и т.п.

Множество простых операций содержит подмножества, элементы которых образуют полный набор логических функций. (Например, функция отрицания и конъюнкция двух чисел, функция отрицания и дизъюнкция двух чисел, штрих Шеффера, стрелка Пирса). Следовательно, л ю б о й а л г о р и т м м о ж е т б ы т ь п р е д с т а в л е н с п о м о щ ь ю р а з л и ч н ы х н а б о р о в п р о с т ы х о п е р а ц и й.

При заданном наборе назовем к о р т е ж е м п р о с т ы х о п е р а ц и й, или просто к о р т е ж е м, любую последовательность простых операций, в которой каждая операция может зависеть только от исходных данных и результатов предшествующих операций.

Очевидно, что при заданном наборе простых операций данный алгоритм может быть представлен различными кортежами простых операций, отличающимися порядком выполнения во времени некоторых взаимно независимых операций.

7. Будем называть ρ - о п е р а ц и е й совокупность о д н о в р е м е н н о и н е з а в и с и м о друг от друга выполняемых простых операций.

Назовем высотой ℓ ρ-операции число входящих в нее простых операций.

К о р т е ж е м ρ - о п е р а ц и й, или ρ - к о р т е ж е м, будем называть любую последовательность ρ-операций, в которой каждая операция, входящая в данную ρ-операцию, может зависеть только от исходных данных и результатов операций, входящих в предшествующие ρ-операции. Каждой ρ-операции

присвоим номер, соответствующий ее месту в последовательности.

Назовем **длиной** h ρ-кортежа число входящих в него ρ-операций, а его **шириной** L - наибольшую высоту образующих его ρ-операций.

$$L = max(l_1, l_2, \ldots, l_h) \qquad (7)$$

Рис. I без изображения связей можно рассматривать как некий ρ-кортеж, состоящий из $s+1$ ρ-операций (длина ρ-кортежа), первая из которых состоит из n простых операций (ширина ρ-кортежа), вторая из $n/2$ и т.д. и $s+1$-ая содержит одну операцию. (В полной задаче умножения матриц число операций в каждой из ρ-операций больше в mp раз).

Назовем **объемом ρ-кортежа** суммарное число простых операций, входящих в ρ-операции.

$$\mathcal{L} = \sum_{j=1}^{h} l_j . \qquad (8)$$

Будем называть **функцией пошагового распределения** зависимость между высотой ρ-операции и ее номером в ρ-кортеже.

$$l_j = \varphi(j), \qquad (9)$$

где j пробегает последовательность значений $1, 2, \ldots, h$. Для рассмотренного примера функция φ ρ-кортежа, показанного на рис. I, изображена на рис. 2.

Будем называть ρ-кортеж **сжатым**, если в нем первая ρ-операция образована всеми операциями, зависящими только от исходных данных, а любая из последующих ρ-операций состоит из тех и только тех операций, которые зависят хотя бы от одной операции, входящей в непосредственно предшествующую ρ-операцию. Нетрудно видеть, что на рис. I изображен именно такой ρ-кортеж. Как видно из рис. I, сжатый ρ-кортеж удобно совмещать с граф-схемой связей между операциями данного алгоритма.

Будем называть ρ-кортеж **вырожденным**, если в каждой из его ρ-операций содержится только по одной простой операции.

8. Алгоритм, представленный в виде кортежа ρ-операций, будем называть **параллельным алгоритмом**, или **ρ-алгоритмом**.

Алгоритм может быть представлен различными ρ-кортежами, множество которых будем называть **семейством** ρ-кортежей данного ρ-алгоритма.

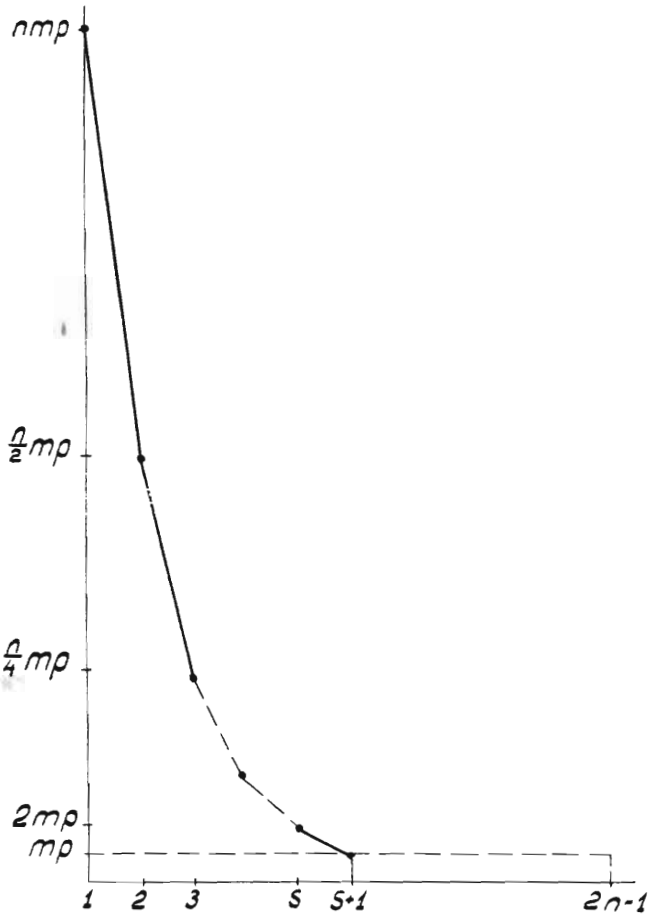


Рис.2.

В соответствии с определением алгоритма все ρ -кортежи семейства могут отличаться друг от друга только иным распределением операций между ρ -операциями и числом самих ρ -операций. Ясно, что все ρ -кортежи семейства имеют один и тот же объем.

Очевидна справедливость следующего утверждения: **каждое семейство ρ -кортежей содержит единственный сжатый ρ -кортеж.** Ему соответствует наименьшая длина h_m.

Наибольшая длина $h_{n\rho}$ из всех ρ -кортежей семейства у вырожденного ρ -кортежа.

Семейство ρ -кортежей отражает особенности соответствующего ρ -алгоритма, поэтому можно говорить о наименьшей длине h_m ρ-алгоритма, наибольшей его длине $h_{n\rho}$, совпадающей с его объемом $\mathcal{L}$, ширине ρ -алгоритма при заданной его длине и т.п.

Если семейство состоит только из одного вырожденного ρ -кортежа, то такой ρ -алгоритм будем называть **вырожденным.**

9. Поставим в соответствие первой ρ -операции ρ -кортежа некоторую исходную информацию v_0 , второй ρ -операции - информацию v_1 , образованную из результатов выполнения первой ρ -операции и той части исходной информации v_0 , которая потребуется для выполнения всех последующих ρ -операций. Аналогично v_j есть сумма объема информации, полученного при выполнении ρ -операции Q_j , и той части объема информации v_{j-1}, которая потребуется для последующих ρ -операций, считая с Q_{j+1}. Назовем соответствующие объемы информации v_j ,измеряемые в двоичных единицах, **существенными объемами информации ρ-операций.**

Наибольший из существенных объемов информации среди всех ρ-операций ρ -кортежа назовем **существенным объемом информации ρ-кортежа** (V)

$$V = max\ (v_0, v_1, \dots, v_h). \qquad (10)$$

10. Рассмотрим некоторый ρ -кортеж длиной h и шириной L , состоящий из ρ -операций высотой $l_1, l_2, \dots, l_h$ Дополним ρ -операции пустыми операциями с тем, чтобы все они стали высотой L , а ρ -кортеж - прямоугольным. Под **пустой** операцией будем понимать операцию, результат которой для выполнения данного алгоритма безразличен.

Операции, входящие в ρ-кортеж, расположены теперь в виде прямоугольной матрицы с h перенумерованными столбцами и L строками, которые условимся нумеровать сверху вниз.

Каждая **строка** такой матрицы представляет собой **кортеж** простых операций.

Из определения ρ -операции следует, что образующие её простые операции могут переставляться между собой произвольным образом, тем не менее ρ -операции, а значит и ρ -кортеж, считаются одними и теми же. Очевидно, что число возможных представлений ρ -операции высотой L с числом пустых операций $L-l_j$ равно $L!/(L-l_j)!$. При этом пустые операции не различаются между собой.

Все кортежи простых операций образуют **множество** кортежей данного ρ -кортежа. Число элементов множества, как нетрудно видеть, равно

$$\prod_{j=1}^{h} b_j \quad , \quad b_j = \begin{cases} l_j & \text{при } l_j = L \\ l_j + 1 & \text{при } l_j < L \end{cases} \tag{II}$$

Из этого множества выберем подмножества из L кортежей, таких, что каждая операция ρ -операций входила бы в данное подмножество один и только один раз. Общее число подмножеств, очевидно, равно

$$\frac{(L!)^h}{L \prod_{j=1}^{h} (L - l_j)!} \tag{I2}$$

Нетрудно видеть, что каждое такое подмножество определяет рассматриваемый ρ -кортеж. В дальнейшем будем называть эти подмножества кортежей **покрытиями** данного ρ -кортежа.

II. Рассмотрим кортежи κ_i и κ_j , принадлежащие некоторому покрытию. Будем говорить, что κ_i не **зависит** от κ_j , если ни одна операция кортежа κ_i непосредственно не зависит ни от одной из операций кортежа κ_j . В качестве **меры зависимости** κ_i от κ_j возьмем число операций кортежа κ_j , от которых непосредственно зависят операции кортежа κ_i . Это число обозначим символом c'_{ij} .

Если рассматривается все покрытие, то числа c'_{ij} образуют квадратную матрицу порядка L . Примем, что диагональные элементы матрицы $c'_{ij} = 0 \ (i = j)$. Сумма

$$C'_i = \sum_{j=1}^{L} c'_{ij} \tag{I3}$$

дает для κ_i общее число операций остальных кортежей покрытия, от которых непосредственно зависят операции данного кортежа. Сумму

$$C' = \sum_{i=1}^{L} C'_i \qquad (I4)$$

назовем связностью покрытия по операциям, а матрицу (c'_{ij}) - матрицей связности по операциям.

Распределим исходную информацию $\mathcal{I}_0$ между кортежами покрытия. В процессе вычислений, возможно, придется передавать исходную информацию из кортежа κ_j в кортежи κ_i. Обозначим количество различных передаваемых кодов c''_{ij}. Числа c''_{ij} образуют квадратную матрицу порядка L, у которой элементы $c''_{ij} = 0$ $(i=j)$. Матрицу (c''_{ij}) условимся называть матрицей связности покрытия по исходной информации, а

$$C'' = \sum_{i,j=1}^{L} c''_{ij} \qquad (I5)$$

связностью покрытия по исходной информации.

Матрицу

$$(c_{ij}) = (c'_{ij}) + (c''_{ij}) \qquad (I6)$$

будем называть просто матрицей связности, а

$$C = C' + C'' \qquad (I7)$$

связностью покрытия.

Представление ρ-кортежа в виде покрытия эквивалентно представлению процесса выполнения данного ρ-алгоритма в виде множества одновременно (вообще говоря, зависимо) выполняемых последовательных алгоритмов, обмен информацией между которыми описывается матрицей связности.

I2. Хотя предыдущее рассмотрение велось без учета предикатов и, следовательно, возможных разветвлений в процессе выполнения ρ-алгоритма, но оно может быть легко распространено и на этот случай. Действительно, пусть предикат α обусловливает выполнение либо ρ_1-подкортежа с шириной L_1 и длиной h_1, либо ρ_2-подкортежа с шириной $L_2 = L_1$ и длиной $h_2 \leqslant h_1$, каждый из которых уже не содержит предикатов (под ρ-подкортежем понимается последовательность ρ-операций, входящая в

ρ-кортеж). Тогда условимся брать в качестве основного ρ-подкортежа ρ_1-подкортеж с большей длиной. Иными словами, общая длина ρ-кортежа, содержащего разветвления, будет определяться более длинными ветвями. Объем ρ-кортежа будет при этом определяться числом операций, входящих в длинные ветви. Число выполняемых ρ-операций в этом случае будет обычно меньше длины ρ-кортежа. Поправочный коэффициент можно при необходимости вычислить, если знать вероятности, с которыми предикаты принимают то или другое значение, либо, что бывает зачастую проще, определять его путем моделирования.

I3. Будем говорить, что с х е м а ρ-алгоритма решения задачи построена, если определены:

I) ρ-кортеж, представляющий данный алгоритм, в том числе функция пошагового распределения,

2) покрытие ρ-кортежа простыми кортежами,

3) распределение исходной информации между кортежами,

4) матрицы связности (c_{ij}) и связности по операциям (c'_{ij}).

§ 2. Характеристическая функция ρ-алгоритма

I. В § I было указано на неоднозначность представления ρ-алгоритма ρ-кортежами, множество которых мы назвали семейством ρ-кортежей. Рассмотрим этот вопрос подробнее.

Будем называть h - п о д с е м е й с т в о м ρ-кортежей все ρ - кортежи семейства, имеющие длину h.

ρ-кортеж h - подсемейства назовем м и н и м а л ь н ы м п о ш и р и н е, если он имеет наименьшую ширину из всех ρ-кортежей подсемейства

$$L^* = \min_{\tau} L_\tau , \qquad (I8)$$

где τ - индекс ρ-кортежа h-подсемейства.

L^* будем называть м и н и м а л ь н о й ш и р и н о й h - п о д с е м е й с т в а.

Назовем L - п о д с е м е й с т в о м ρ-кортежей все ρ-кортежи семейства, имеющие ширину L.

ρ-кортеж L-подсемейства будем называть м и н и м а л ь н ы м п о д л и н е, если он имеет наименьшую длину из всех ρ-кортежей подсемейства

$$h^* = \min_{q} h_q , \qquad (I9)$$

где q -индекс ρ-кортежа L -подсемейства.

h^* назовем минимальной длиной L-подсемейства.

ρ -кортеж будем называть оптимальным, если он одновременно минимален и по длине и по ширине.

Мы уже видели, что длина ρ -кортежей не может быть меньше длины h_m сжатого ρ -кортежа. В h_m-подсемействе выделим ρ -кортежи с минимальной шириной L_m^* и максимальной шириной $\hat{L}_m$. Очевидно, что все L -подсемейства при $L_m^* \le L \le \hat{L}_m$ имеют минимальную длину h_m. L -подсемейства с $L > \hat{L}_m$, хотя, вообще говоря, и могут существовать, но рассматривать их практически не интересно.

Минимальным по ширине в $h_{n\rho}$-подсемействе является ρ-кортеж с $L = 1$, который будет вообще единственным в подсемействе $L = 1$, а, следовательно, является оптимальным.

Таким образом, на плоскости h, L имеются два множества точек $\mathcal{M}_1 = \{(h, L^*)\}$ и $\mathcal{M}_2 = \{(h^*, L)\}$, где $h_m \le h \le h_{n\rho}$, $\hat{L}_m \ge L \ge 1$. Рассмотрим их объединение

$$\mathcal{M} = \mathcal{M}_1 \cup \mathcal{M}_2 .$$

ТЕОРЕМА I. Множество $\mathcal{M}$ на плоскости h, L представляет собой монотонную последовательность точек

$$a_1(h_1, L_1) = a_m(h_m, \hat{L}_m), \ldots, a_i(h_i, L_i), a_{i+1}(h_{i+1}, L_{i+1}), \ldots, a_n(h_n, L_n) = a_{n\rho}(h_{n\rho}, 1),$$

где $h_{i+1} \ge h_i$; $L_{i+1} \le L_i$

Докажем сначала некоторые леммы.

ЛЕММА I. Любую ρ -операцию, состоящую более чем из одной операции можно заменить последовательностью из двух или более ρ -операций, составленных из тех же операций, что и исходная. Справедливость леммы очевидна и непосредственно вытекает из взаимной независимости операций, входящих в одну ρ -операцию.

ЛЕММА 2. В каждом h -подсемействе, $h_m \le h \le h_{n\rho}$ имеется не менее одного ρ-кортежа. Действительно, рассмотрим сжатый ρ -кортеж с длиной h_m и шириной L_m. В согласии с леммой I заменим какую-либо из его ρ -операций, содержащую более одной операции, двумя и тем самым увеличим длину ρ -кортежа на I.

Применяя эту процедуру многократно получим последовательность ρ-кортежей, начиная от сжатого и кончая вырожденным, каждый из которых будет иметь длину, большую на единицу, чем у предыдущего, что и доказывает лемму.

ЛЕММА 3. В каждом L-подсемействе, $1 \leqslant L \leqslant \hat{L}_m$ имеется не менее одного ρ-кортежа. Возьмем за исходный ρ-кортеж с шириной $\hat{L}_m$. В соответствии с леммой I все ρ-операции с высотой $\hat{L}_m$ заменим на две, так, чтобы хотя бы одна из ρ-операций осталась высотой $\hat{L}_m - 1$. Повторяя эту процедуру многократно, получим последовательность ρ-кортежей с шириной от $\hat{L}_m$ до I, каждый из которых будет иметь ширину на единицу меньшую, чем у предыдущего, что и требовалось доказать.

ЛЕММА 4. L-подсемейство с минимальной длиной h^* содержит ρ-кортежи длиной $h^*, h^*+1, h^*+2, \ldots, h_{np}-L+1$, где h_{np}-длина вырожденного ρ-кортежа. Действительно, выделим в минимальном по длине ρ-кортеже ρ-операцию высотой L и заменим любую другую ρ-операцию с высотой больше I двумя в согласии с леммой I. Это даст нам ρ-кортеж длиной h^*+1. Повторяя эту процедуру, получим ρ-кортеж длиной h^*+2 и т.д. до тех пор, пока все ρ-операции, кроме выбранной, не станут высотой I. Очевидно, что длина ρ-кортежа при этом будет равна $h_{np}-L+1$, и это и есть наиболее длинный ρ-кортеж в L-подсемействе. Лемма доказана.

ЛЕММА 5. h-подсемейство с минимальной шириной L^* содержит ρ-кортежи с шириной $L^*, L^*+1, L^*+2, \ldots, L^*+\kappa$, где $L^*+\kappa = \min\{\hat{L}_m, h_{np}-h+1\}$. Возьмем за исходный минимальный по ширине ρ-кортеж h-подсемейства. Перенесем в его первую ρ-операцию одну за другой операции, входящие в первую ρ-операцию ρ-кортежа с шириной $\hat{L}_m$, пока ее высота не станет равной L^*+1. Если этого не произойдет, а все возможные операции уже перенесены, то начинаем строить таким же способом вторую ρ-операцию и т.д. Если при этом где-либо образуется пустая ρ-операция, то в согласии с леммой I разделим на две любую ρ-операцию, кроме достраиваемой. Подобное деление, очевидно, возможно пока имеются ρ-операции, кроме выбранной, с высотой больше I, что, очевидно, будет пока $L^*+\kappa < h_{np}-h+1$. Наибольшая ширина ρ-кортежа h-подсемейства будет при этом равна $L^*+\kappa = \min\{\hat{L}_m, h_{np}-h+1\}$. Лемма доказана.

ЛЕММА 6. Множеству $\mathcal{M}$ не могут одновременно принадлежать три точки $a_1(h,L)$, $a_2(h',L)$ и $a_3(h,L')$, где $h'<h$, $L'<L$. Действительно, предположим противное. Тогда, поскольку a_2 принадлежит множеству $\mathcal{M}$ ($a_2\in\mathcal{M}$), то a_1 не принадлежит множеству $\mathcal{M}_2$ ($a_1 7\in\mathcal{M}_2$). Аналогично, так как $a_3\in\mathcal{M}$, то $a_1 7\in\mathcal{M}_1$, а следовательно, $a_1 7\in\mathcal{M}$.

ЛЕММА 7. Множеству $\mathcal{M}$ не могут одновременно принадлежать точки $a_1(h,L)$ и $a_0(h',L')$, где $h'<h\leqslant h_{np}$; $L'<L\leqslant\hat{L}_m$. В самом деле, предположим противное. Тогда по лемме 4 в L'-подсемействе должен быть ρ-кортеж с длиной $h>h'$, если только $h\leqslant h_{np}-L'+1$. Последнее неравенство имеет место, так как из того, что $a_1\in\mathcal{M}$, следует, что $h\leqslant h_{np}-L+1$. Отсюда $h<h_{np}-L'+1$, то есть указанный ρ-кортеж существует. Это значит, что $a_1 7\in\mathcal{M}_1$.

Далее, по лемме 5 в h' - подсемействе должен быть ρ-кортеж шириной $L>L'$, если только $L\leqslant\min\{\hat{L}_m, h_{np}-h+1\}$. Последнее неравенство имеет место, так как из того, что $a_1\in\mathcal{M}$, следует, что $L\leqslant\hat{L}_m$ и $L\leqslant h_{np}-h+1$. Отсюда $L<h_{np}-h'+1$, то есть и такой ρ-кортеж существует, а это означает, что $a_1 7\in\mathcal{M}_2$. Следовательно, $a_1 7\in\mathcal{M}$. Лемма доказана.

Приступим теперь к доказательству теоремы.

Пусть точка $a_{11}(h^o,L^o)\in\mathcal{M}$, где $h^o<h_{np}$, $1<L^o$ (рис. 3). Тогда по лемме 7 точки $a_{22}(h^o+1,L^o+1)7\in\mathcal{M}$ и $a_{00}(h^o-1,L^o-1)7\in\mathcal{M}$. Кроме того, по лемме 6 множеству $\mathcal{M}$ не могут одновременно принадлежать точка a_{21} и точка a_{20}, а по лемме 7 - точка a_{21} и точка a_{10}. Действительно, по лемме 2 множеству $\mathcal{M}$ должна принадлежать хотя бы одна точка с $h=h^o+1$, а по лемме 7 в этой точке $L\leqslant L^o$. Тогда это либо одна из точек a_{21} или a_{20} (то есть утверждение верно), либо точка с $L<L^o-1$, причем при $L^o=2$ возможно только первое. С другой стороны, по лемме 3 множеству $\mathcal{M}$ должна принадлежать хотя бы одна точка с $L=L^o-1$, а по лемме 7 в этой точке $h\geqslant h^o$. Тогда это либо одна из точек a_{10} или a_{20}, либо обе вместе (в этих случаях утверждение верно), либо $h>h^o+1$, причем при $h^o=h_{np}-1$ возможно только первое. Очевидно, что утверждение может быть неверно только в случае одновременной принадлежности множеству $\mathcal{M}$ точек $(h^o+1, L<L^o-1)$ и $(h>h^o+1, L=L^o-1)$, но это невозможно по лемме 7.

Таким образом, должна реализовываться одна из четырех возможностей, а именно: множеству $\mathcal{M}$ принадлежат либо одна

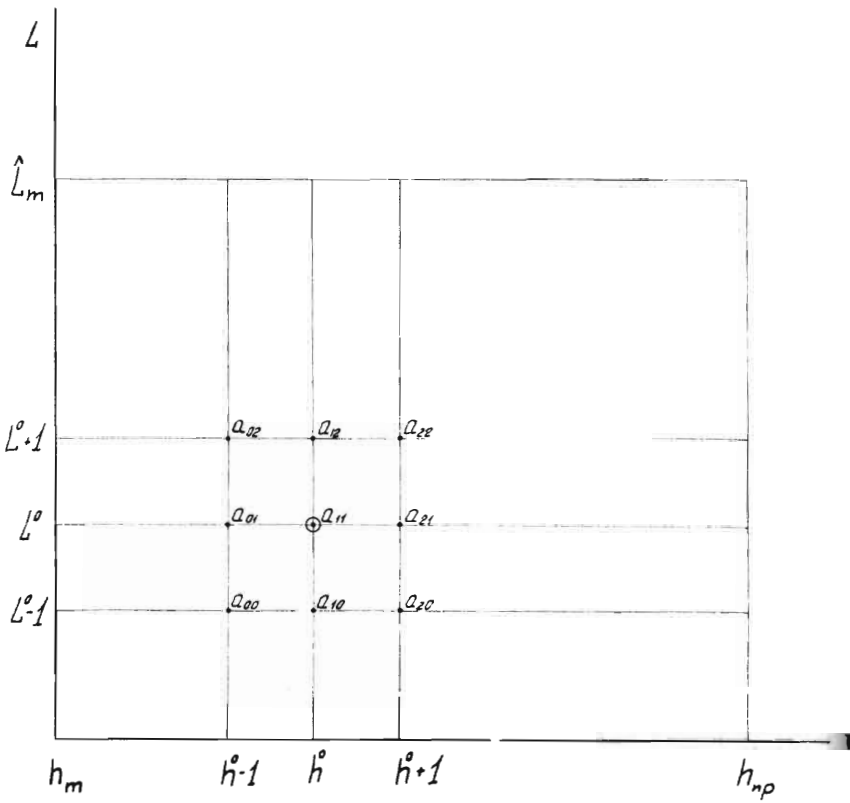

L
$\hat{L}_m$
$L^\circ+1$
L°
$L^\circ-1$
a_{02}
a_{12}
a_{22}
a_{01}
a_{11}
a_{21}
a_{00}
a_{10}
a_{20}
h_m
$h^\circ-1$
h°
$h^\circ+1$
h_{np}

Рис. 3.

из точек a_{21}, a_{20}, a_{10}, либо пара точек a_{10}, a_{20}. В последнем случае по леммам 6 и 7 точки $(h^0, L^0-2)\gamma \in \mathcal{M}$, $(h^0+1, L^0-2)\gamma \in \mathcal{M}$.

Условимся считать, что точка (h', L') следует за точкой (h, L), если $h' \geqslant h$, $L' \leqslant h$. Тогда будем рассматривать четыре указанных варианта как переходы $a_{11}a_{21}, a_{11}a_{20}, a_{11}a_{10}, a_{11}a_{10}a_{20}$. В этих переходах значения h не убывают, а значения L не возрастают.

Применим к точке $a_1 = a_m$ выводы, сделанные для точки a_{11}. В результате получаем переход из двух или трех точек, принадлежащих множеству $\mathcal{M}$. Затем применяем к последнему члену перехода ту же процедуру и т. д. (На первых шагах до точки (h_m, L_m^*) будут реализовываться переходы только типа $a_{11}a_{10}$). В результате получаем монотонную последовательность точек. Вследствие монотонности она дойдет либо до значения $h = h_{np} - 1$, либо до $L = 2$. Далее будут осуществляться или переходы типа $a_{11}a_{10}$ (при $h = h_{np} - 1$), или $a_{11}a_{21}$ (при $L = 2$) и так до точки $(h_{np} - 1, 2)$, ибо на линиях $h = h_{np}$ и $L = 1$ множеству $\mathcal{M}$ принадлежит только одна точка, которой и будет заканчиваться данная последовательность.

Очевидно, что все точки множества вошли в последовательность. Действительно, по леммам 2 и 3 на каждой линии h $(h_m \leqslant h \leqslant h_{np})$ и L $(1 \leqslant L \leqslant \hat{L}_m)$ содержится хотя бы одна точка последовательности. Таким образом, для любой точки $a(h, L)$, не принадлежащей последовательности, найдется по крайней мере одна точка $a'(h', L')$ такая, что либо $h' > h$, $L' > h$, либо $h' < h$, $L' < L$. И в том и в другом случае $a\gamma \in \mathcal{M}$ по лемме 7 или же найдется пара точек $a'(h', L')$ и $a''(h'', L'')$, таких, что $h = h'$, $L > L'$ и $h > h''$, $L = L''$, тогда $a\gamma \in \mathcal{M}$ по лемме 6.

Таким образом, теорема I доказана полностью.

Построенную последовательность можно рассматривать как дискретную функцию

$$L = F(h), \qquad (20)$$

определенную для $h = h_m, h_m + 1, \ldots, h_{np}$ и принимающую значения $\hat{L}_m, \hat{L}_m - 1, \ldots, 1$.

Назовем эту функцию характеристической функцией ρ-алгоритма.

Условимся изображать характеристическую функцию в виде ориентированного графа, ребра которого представляют собой отрезки, параллельные горизонтальной и вертикальной осям плос-

кости h, L , при этом каждый из четырех упомянутых переходов будет изображаться следующим образом:

а) $a_{11}\,a_{21}$ б) $a_{11}\,a_{20}$ в) $a_{11}\,a_{10}$ г) $a_{11}\,a_{10}\,a_{20}$.

Тогда характеристическая функция будет иметь вид, показанный на рис. 4, на котором кривая а) соответствует суммированию n слагаемых, а кривая б) - вычислению таблицы из n чисел, каждое из которых является результатом одной операции над исходными числами.

Из этих примеров видно, что в характеристической функции могут иметь место все четыре перехода, упомянутых выше. По лемме 6 после переходов а) и г) невозможны переходы в) и г) . Примеры остальных 12 пар переходов можно видеть на рис. 4.

Очевидны следующие свойства характеристической функции:

1) Вершины углов типа б) перехода не содержат точек характеристической функции.

2) Точки, соответствующие минимальным по длине ρ -кортежам, находятся на вертикальных участках, а точки, соответствующие минимальным по ширине ρ -кортежам - на горизонтальных.

3) Точки, соответствующие оптимальным ρ -кортежам, и только они находятся во всех вершинах углов типа перехода г).

4) Знания всех оптимальных точек и $\hat{L}_m$ достаточно для определения всех других точек характеристической функции.

Характеристическая функция полностью определяется объемом $\mathcal{L}$ ρ-алгоритма и граф-схемой связей. При одном и том же $\mathcal{L}$ можно указать некоторые особенности характеристических функций.

1) Вырожденный ρ -алгоритм имеет единственную точку $(h_m = h_{np}, \hat{L}_m = 1)$, которая, очевидно, оптимальна. Наибольшее число оптимальных точек у характеристической функции ρ -алгоритма, все операции которого не зависят друг от друга (рис. 4б). В последнем случае характеристическая функция симметрична относительно диагонали $L = h$, а оптимальными точками являются точки с $h \le d$ или $L \le d$, где d есть округленное до большего целого значение $\sqrt{\mathcal{L}}$.

2) Если $U-\rho$-алгоритм с независимыми операциями, то для произвольной точки $\alpha(h, L)$ характеристической функции любого другого A ρ-алгоритма (с зависимыми операциями)

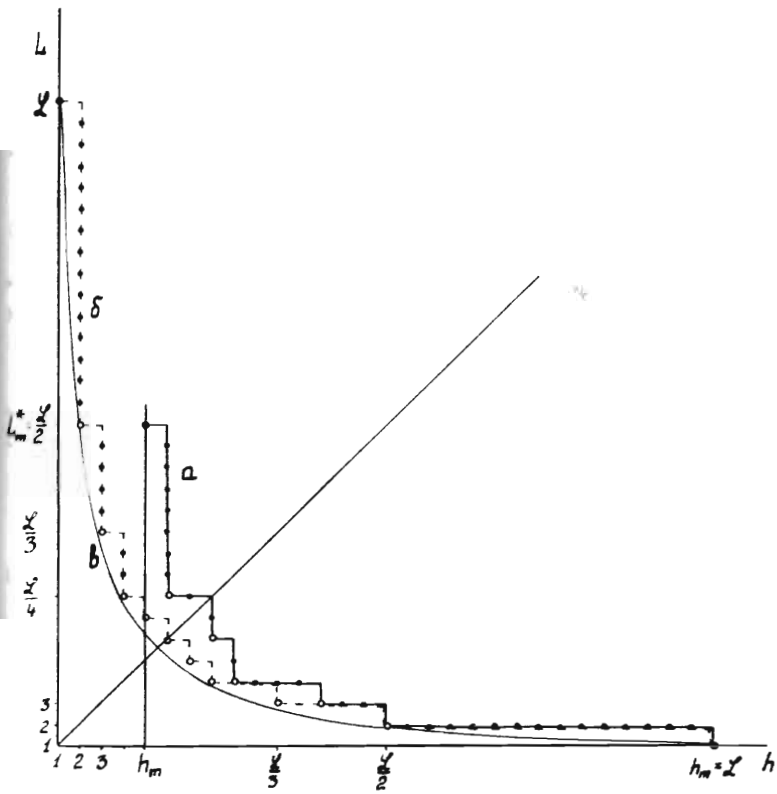


Рис. 4.

справедливы соотношения:

$$\text{при } h=h' \quad , \quad L \geqslant L' \,,$$
$$\text{при } L=L' \quad , \quad h \geqslant h'',$$

где $u'(h', L')$ и $u''(h'', L'')$ — точки характеристической функции ρ-алгоритма U. Справедливость этого утверждения следует из того, что при фиксированном h (или L) для ρ -алгоритма U может быть образован ρ -кортеж с L^* (или h^*) не большей, чем для ρ -алгоритма A (рис. 4).

2. Будем называть э ф ф е к т и в н о с т ь ю данного ρ -кортежа отношение объема ρ -алгоритма к произведению длины ρ -кортежа на его ширину

$$\delta = \frac{\mathcal{L}}{hL} \,. \tag{21}$$

Наибольшая эффективность $\delta=1$ будет при $L=\mathcal{L}/h$,когда все ρ -операции имеют одинаковую высоту. Такие ρ -кортежи и соответствующие им точки характеристической функции будем называть э ф ф е к т и в н ы м и.

У неэффективных ρ -кортежей имеются ρ -операции с высотой $\ell_j < L$. Если дополнять эти операции до высоты L пустыми, то, очевидно, $hL=\mathcal{L}+\Lambda$, где Λ - общее число пустых операций в ρ -кортеже. Тогда

$$\delta = \frac{\mathcal{L}}{\mathcal{L}+\Lambda} = \frac{1}{1+\Lambda/\mathcal{L}} \,. \tag{22}$$

Отсюда видно, что δ убывает по гиперболическому закону с ростом пустых операций.

Ясно, что все эффективные точки являются также и оптимальными. На характеристической функции (20) им соответствуют точки, лежащие на гиперболе $L=\mathcal{L}/h$ (Рис. 4,в).

Если характеристическую функцию изобразить в координатах $(L, \mathcal{L}/h)$

$$\mathcal{L}/h = f(L), \tag{23}$$

то эффективные точки располагаются на прямой $\mathcal{L}/h = L$.

При представлении характеристической функции в виде (23) ее точкам соответствует эффективность

$$\delta = \frac{1}{L} f(L). \tag{24}$$

Нетрудно видеть, что поскольку точки характеристической функции при заданном L соответствуют наименьшему h , а при за-

данном k - наименьшему L, то им будут отвечать наибольшие значения δ в k (или L) - подсемействах.

Для рассматриваемого примера умножения матриц функции $f(L)$ и $\delta(L)$ имеют вид, показанный на рис.5.

Среди эффективных точек важную роль играет точка с наибольшим значением L, которую будем называть главной эффективной точкой, или просто главной точкой, и обозначать буквой g. Условимся часть характеристической функции, которой соответствуют $1 \leqslant L \leqslant L_g$, называть эффективной областью, а часть, которой соответствуют $L_g < L \leqslant \hat{L}_m$ - неэффективной.

Будем называть L-развертkой ρ_c-кортежа ρ-кортеж, образованный путем замены каждой ρ-операции ρ_c-кортежа последовательностью ρ-операций, которые все имеют длину L, а последняя не больше чем L.

Возможность применения L-развертки к ρ-кортежу с шириной $L_c > L$ вытекает из леммы I. Условимся в L-развертке дополнять все ρ-операции до длины L пустыми операциями.

Очевидна следующая важная

ТЕОРЕМА 2. Пусть дан ρ-кортеж и функция его пошагового распределения $\ell_j = \varphi(j), (j = 1,2,\ldots,k)$. Тогда, если существует общий делитель L чисел ℓ_j, то из данного можно построить эффективный ρ-кортеж.

Действительно, построим L-развертку заданного ρ-кортежа. Так как все ℓ_j кратны L, то каждая из ρ-операций заменится последовательностью, состоящей из равновысоких (высотой L) ρ-операций. Таким образом, будет получен прямоугольный ρ-кортеж с $\delta = 1$.

Из этого утверждения вытекает следствие: Если известна какая-либо эффективная точка $e(k_e^*, L_e^*)$ и $L_e^* = L_a L_c$, где L_a и L_c - целые, то характеристическая функция имеет эффективные точки $a(k_a = L_c k_e, L_a)$ и $c(k_c = L_a k_e, L_c)$.

В рассмотренном примере умножения матриц для $n = m = p = 2^{10}$ эффективные точки будут соответствовать точкам, координаты которых являются степенями двойки, а главной точке будет соответствовать $\mathcal{L}/k_g^* = L_g^* = mp = 2^{20}$ (рис. 5).

Указанным способом можно получать также значения коор-

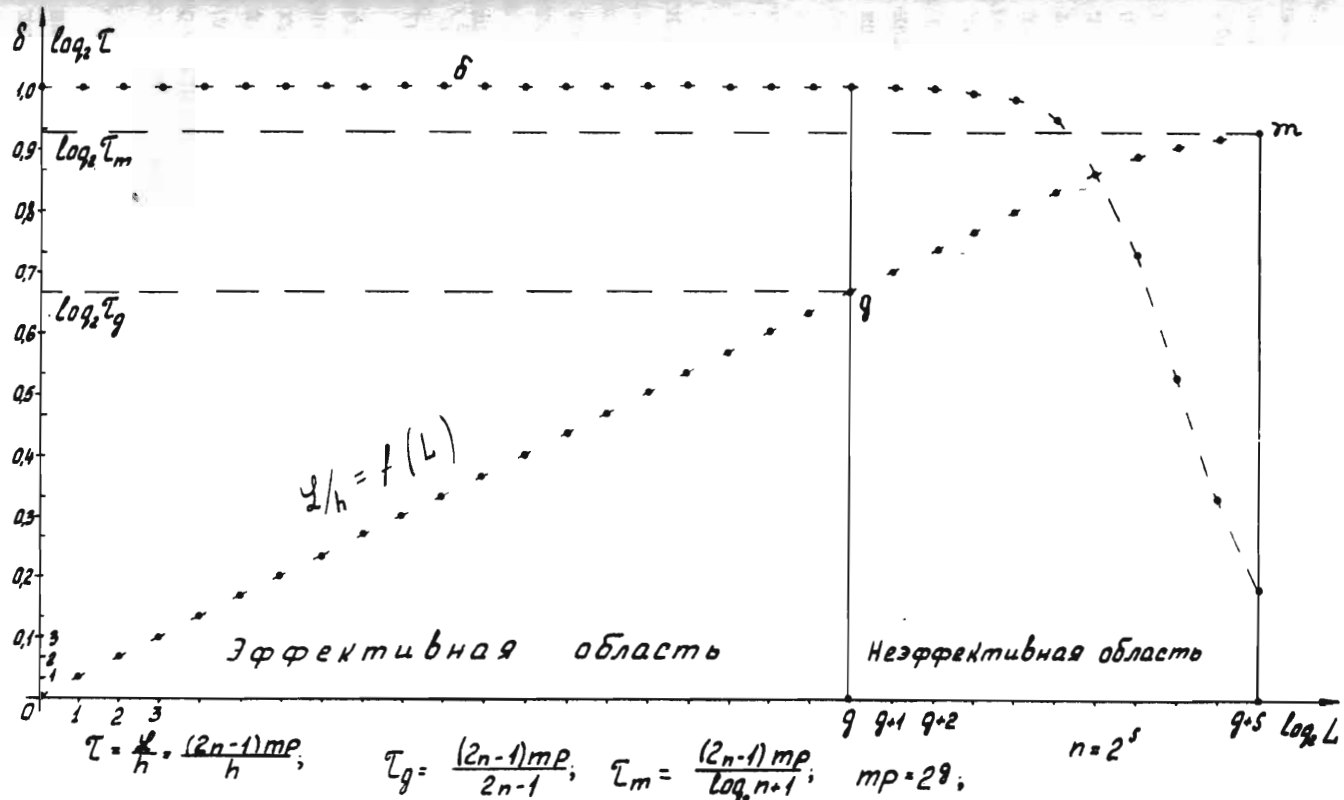

δ
log₂ τ
δ
1,0
log₂ τ_m
0,9
0,8
0,7
log₂ τ_q
0,6
0,5
0,4
0,3
0,2
0,1
m
q
L/h = f(L)
Эффективная область
Неэффективная область
0
1 2 3
q q+1 q+2
q+s
log₂ L
τ = L/h · (2n-1)mp/h;
τ_q = (2n-1)mp/(2n-1);
τ_m = (2n-1)mp/(log₂ n+1);
mp = 2^q;
n = 2^s

Рис. 5.

динат точек, для которых δ мало отличается от I (к в а з и э ф ф е к т и в н ы е т о ч к и). Действительно, дополним некоторые ρ -операции ρ -кортежа пустыми так, чтобы у нового набора ℓ_j появились общие делители. Тогда, если число дополненных операций $\Lambda \ll \mathcal{L}$, то согласно (22) $\delta \approx 1$

В частности, если исходный ρ_ℓ -кортеж был эффективным, а требуется найти ρ -кортеж с шириной L , которой не кратно $L^*_{\ell_i}$, то добавим ко всем ρ -операциям ρ_ℓ -кортежа λ пустых с тем, чтобы

$$L^*_\ell + \lambda = \alpha \cdot L \ ,$$

где $\alpha > 0$-целое, $1 \leqslant \lambda < L$. Тогда L -развертка содержит $\Lambda = \lambda h^*_\ell$ пустых операций и согласно (22)

$$\delta = \frac{1}{1 + \lambda / L^*_\ell} \ . \qquad (25)$$

Из (25) видно, что если L^*_ℓ лежит в области больших значений L , например $L^*_\ell = L^*_{\ell_j}$, так что $L \ll L^*_\ell$, то $\lambda / L^*_\ell \ll 1$ и δ будет близок к 1.

ТЕОРЕМА 3. П р и L - р а з в е р т к е ρ - к о р т е ж а с ш и р и н о й L_c , к о г д а L_c к р а т н о L, в е л и ч и н а к о э ф ф и ц и е н т а э ф ф е к т и в н о с т и н е у б ы в а е т, т о е с т ь $\delta \geqslant \delta_c$.

Действительно, при составлении L -развертки, так как L_c кратно L , то не нужно добавлять пустых операций, поэтому их число в новом ρ -кортеже по сравнению с ρ_c -кортежем может только уменьшиться за счет отбрасывания ρ -операций, состоящих сплошь из пустых операций. Откуда на основании (22) и следует справедливость неравенства $\delta \geqslant \delta_c$.

СЛЕДСТВИЕ. Ш и р и н а т а к и м о б р а з о м п о л у ч е н н о г о ρ - к о р т е ж а $h \leqslant \frac{L_c}{L} h_c$.

3. Рассмотренные свойства L -разверток можно использовать для определения (хотя бы приближенно) вида характеристической функции.

В практических задачах обычно нетрудно составить сжатый ρ -кортеж и из него получить другие ρ -кортежи h_m-подсемейства. Если существуют общие делители всех высот ρ -операций в каких-либо ρ -кортежах (либо этого можно добиться добавлением небольшого числа пустых операций), то согласно теореме 2 тем самым определяются эффективные (либо квазиэффективные)

точки. Среди этих точек для нас важны точки с большими значениями L (малыми h).Практически достаточно знать хотя бы одну такую точку, которую назовем б а з о в о й и обозначим точкой $\mathcal{B}$. В частности, базовой точкой может быть главная эффективная точка. В ряде задач выбор базовой точки непосредственно подсказывается видом вычислительного алгоритма. Так,например, в задачах линейной алгебры $L_{\mathcal{B}}$ обычно равно порядку матриц, с которыми приходится оперировать.

Из $\rho_{\mathcal{B}}$ -кортежа строим различные L -развертки. При этом для каждого L получается значение $h \geqslant h^*$, где h^* — значение характеристической функции при заданном L . Таким образом, получается верхнее значение координаты h^* . Нижней границей h^* является значение $(h^*)_u$ характеристической функции ρ -алгоритма U с независимыми операциями (см. конец п. I).

Таким образом, получаем полосу, в которой заключена характеристическая функция. Если в области интересующих нас значений L (или h) эта полоса слишком широка, то можно провести построение,взяв в качестве базовой точки другую точку и т.д.

4. ρ -кортежи, принадлежащие h_m -подсемейству, в отличие от всех остальных имеют хотя бы по одной цепочке операций длиной, равной длине h_m ρ -кортежа, в которой каждая операция непосредственно зависит от предшествующей (кроме, естественно, первой). Эта цепочка, собственно, и ограничивает дальнейшее уменьшение длины ρ-кортежей. Она аналогична к р и т и ч е с к о м у п у т и в системе планирования PERT ,и мы будем называть ее этим же термином. Обычно в ρ -кортеже таких цепочек бывает не одна, а несколько, причем у них, как правило, имеются общие операции. Так,например, в задаче умножения матриц все операции принадлежат критическому пути (рис. I). Очевидно, что операции, принадлежащие критическому пути (и только они), не могут быть перемещены из одной ρ -операции в другую при переходе от одного ρ -кортежа h_m -подсемейства к другому.

Будем каждую операцию характеризовать интервалом перемещения, равным разности номеров самой правой и самой левой ρ -операций, которым может принадлежать данная ρ -операция в ρ -кортежах h_m -подсемейства.

Для определения интервала перемещения каждой операции достаточно построить сжатый ρ -кортеж и аналогичный ему, с ж а т ы й в п р а в о. Последний строится так. Отнесем к h_m -й

ρ-операции все операции, от которых не зависит ни одна операция данного ρ -алгоритма; а к любой из предыдущих ρ -операций отнесем те и только те операции, от которых зависит хотя бы одна из операций, входящих в непосредственно следующую за ней ρ -операцию.

Знание интервала помогает целенаправленно вести перебор ρ-кортежей h_m - подсемейства.

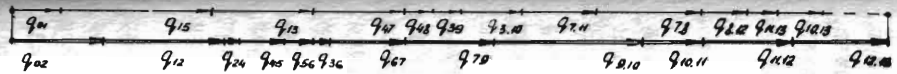
Существует аналогия между схемами параллельных алгоритмов и сетевыми графиками системы PERT , что естественно, так как и те и другие описывают процессы, состоящие из параллельно выполняемых ветвей. Основное отличие состоит в том, что параллельные алгоритмы предполагается реализовать на УВС, состоящей из одинаковых элементарных машин, в то время как сетевые графики выполняются, как правило, коллективами, специализирующимися на выполнении однотипных работ. В этом отношении сетевые графики имеют сходство с вычислительными системами, состоящими из разнородных машин.

Указанное отличие, хотя и существенно, но, по-видимому, не помешает применению одинаковых методов. Представляется полезным введение для системы PERT таких понятий, как "ширина", которая характеризует число одновременно выполняемых работ, "сжатый" график, "характеристическая функция" и т.д. Не углубляясь в этот вопрос, укажем, что аналогия становится более явной, если после того, как составлен сетевой график и выявлен критический путь, совместить его с временным графиком, то есть отложить вдоль горизонтальной оси время , а вдоль вертикальной - число параллельно выполняемых работ. Проиллюстрируем это на примере, взятом из [I8] (рис.6,а). На рис.6,б показан пример сжатого графика, аналогичного сжатому ρ-кортежу. Рис.6,в изображает оптимальный график с минимальной длиной. На рис. 6,г показан полученный при небольшом увеличении общего времени оптимальный график со значением коэффициента эффективности , близким к I.

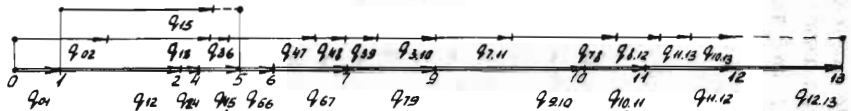
§ 3. Общая схема решения задач на УВС

Как уже упоминалось выше, решение задач на УВС имеет ряд характерных особенностей, заметно отличающих его от решения задач на обычных ЭВМ. Можно указать восемь основных этапов, которые необходимо выполнить (от формулировки задачи до составления программы)

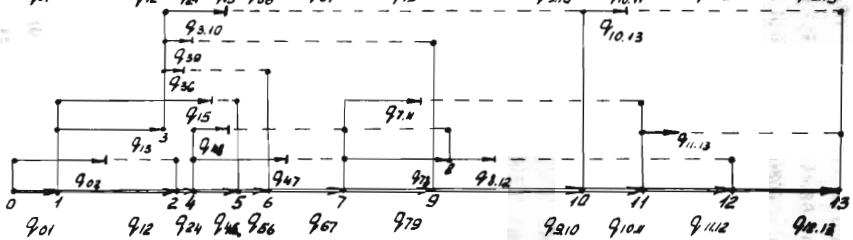
г)



в)



б)



а)

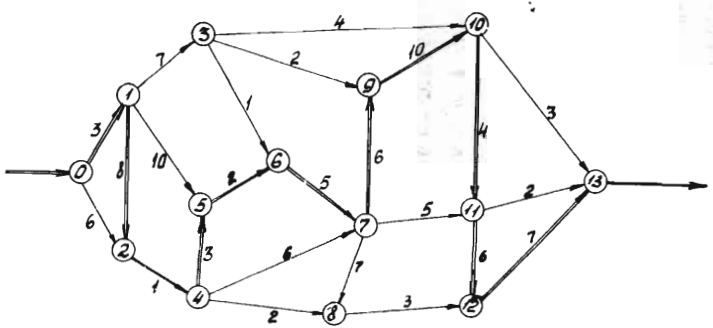


Рис. 6.

I. Формулировка задачи.

2. Выбор алгоритма решения задачи.

3. Построение граф-схемы связей алгоритма.

4. Исследование характеристической функции и выбор рабочей точки.

5. Определение существенного объема информации.

6. Выбор покрытия p -кортежа кортежами простых операций.

7. Составление схемы p -алгоритма.

8. Составление программы решения задачи на УВС.

Рассмотрим каждый из этапов.

I. Формулировка задачи. Формулировка задачи в основном не отличается от обычно употребляемых при решении задач на обычных ЭВМ и должна содержать:

а) Собственно формулировку задачи с указанием массива исходных данных и вида результирующих данных.

б) Метод или методы решения задачи.

в) Желаемое время решения задачи. Каждая задача является, как правило, частью какой-либо проблемы и время ее решения входит как слагаемое в общее время решения проблемы. На основании этого можно дать две оценки времени решения задачи: минимальное (t_{min}) и максимальное ($t_{пр}$). Под t_{min} понимается такое время, ниже которого его величина перестает существенно сказываться на общем времени решения проблемы. Под $t_{пр}$ понимается предельно допустимое время решения задачи. Желательно также знать промежуточные значения времени решения задачи $t_{min} \leq t \leq t_{пр}$ с указанием их веса.

2. Выбор алгоритма решения задачи. Каждый метод решения задачи может быть реализован многими алгоритмами, каждый из которых характеризуется временем его выполнения и материальными затратами. В зависимости от требований, предъявляемых к процессу решения задачи, должен быть выбран тот или иной алгоритм.

Вопрос о выборе рационального алгоритма весьма сложен. Он не решен даже для алгоритмов, выполняемых последовательно на одной ЭВМ; практически в настоящее время выбор алгоритма осуществляется в значительной степени интуитивно. Будем считать, что для УВС эта задача также будет решаться пока на таком же уровня ; этого, по-видимому , будет достаточно для первого периода использования УВС.

3. Построение граф-схемы связей алгоритма. Как уже указывалось, граф-схемы связей

удобно изображать в виде сжатого ρ-кортежа (рис. I). Для этого алгоритм записывается с помощью набора простых операторов, для каждого из которых указываются его непосредственные предшественники. Затем указанным в § I, п.7 способом строится сжатый ρ- кортеж, процесс построения которого может быть легко алгоритмизирован. Остается только нанести изображение связей между операторами.

Сжатый ρ-кортеж является одним из представлений ρ-алгоритма, а полученная граф-схема связей - удобной исходной позицией для анализа последнего.

4. Исследование характеристической функции и выбор рабочей точки. Поскольку каждый ρ-алгоритм может быть представлен многими ρ-кортежами, то нужно выбрать наиболее рациональный по времени и затратам. Для этого строится характеристическая функция, как это описывалось в § 2. Если при формулировке задачи задана функция зависимости между временем решения задачи и ее относительным весом, то выбор соответствующей рабочей точки не представляет больших трудностей, тем более, что известно предельное число машин в системе и выбор производится в сравнительно небольшой области. При этом нужно, конечно, в качестве рабочей точки взять оптимальную.

5. Определение существенного объема информации. Для рабочей точки (или точек) надо найти существенный объем информации ρ-кортежа, который определяет объем памяти УВС. Эта задача обычно не вызывает больших затруднений. Как можно видеть из работы [3], в ряде задач существенный объем практически совпадает с объемом исходной информации. В тех задачах, в которых существенный объем возрастает в процессе вычислений, обычно нетрудно найти закон, определяющий его увеличение.

6. Выбор покрытия ρ-кортежа кортежами простых операций. После того, как выбран ρ-кортеж, реализующий данный ρ-алгоритм, и определен существенный объем информации, необходимо найти рациональное покрытие ρ-кортежа простыми кортежами. Под рациональным будем понимать такое покрытие, при котором достигается некий компромисс между противоречивыми требованиями: уменьшением объема информации, соответствующей каждому кортежу покрытия, и уменьшением связности покрытия. Поясним это.

Прежде всего уточним вопрос об объеме информации. Вся ис-

ходная информация ϑ_0 должна быть распределена между кортежами покрытия. К кортежу κ_i отнесена информация ϑ_{i0} . При этом допускается вхождение исходных данных в разные кортежи. Тогда фактический объем

$$\vartheta_0' = \sum_{i=1}^{L} \vartheta_{i0}' \geq \vartheta_0 . \qquad (26)$$

Замена существенного объема ϑ_0 фактическим ϑ_0' потребует также замены остальных существенных объемов $\vartheta_1, \vartheta_2, \ldots, \vartheta_h$ фактическими $\vartheta_1', \vartheta_2', \ldots, \vartheta_h'$. Величина последних может измениться также, если результат какой-либо операции одновременно относить к нескольким кортежам. На практике это может иметь место, в частности, при работе на распределенных УВС, в которых, как показано в [I], выгоднее работать в режиме, когда передача информации производится сразу же после ее вычисления.

Аналогично существенному объему ρ-кортежа введем фактический объем ρ-кортежа

$$V' = max(\vartheta_0', \vartheta_1', \ldots, \vartheta_h') \geq V. \qquad (27)$$

В процессе распределения исходной информации и результатов операций между кортежами покрытия с каждым кортежем κ_i на каждом κ-м шаге его выполнения связывается некоторый фактический объем информации $\vartheta_{i,\kappa-1}'$. Разность наибольшего из этих объемов

$$\vartheta' = \max_{i,\kappa} \vartheta_{i,\kappa-1} \quad (i=1,2,\ldots,L; \kappa=1,2,\ldots,h) \qquad (28)$$

и среднего

$$\vartheta_{\varphi}' = V'/L \qquad (29)$$

является одной из характеристик данного покрытия.

Второй важной характеристикой покрытия является связность (I7). Очевидно, что увеличивая фактический объем можно в известных пределах уменьшать связность, и наоборот .

7. С о с т а в л е н и е с х е м ы ρ - а л г о р и т м а. После того как покрытие ρ-кортежа выбрано, остается ввести в схему ρ-кортежа β-операции обмена информацией между кортежами покрытия, обобщенного условного перехода и настройки и записать полученную схему ρ-алгоритма на формальном языке, например, матричном ρ-языке [4] или в виде граф-схемы. Для рассматриваемого примера умножения матриц пример граф-схемной записи для вычисления элементов первой строки матрицы (C_{ij}) показан на рис. 7.

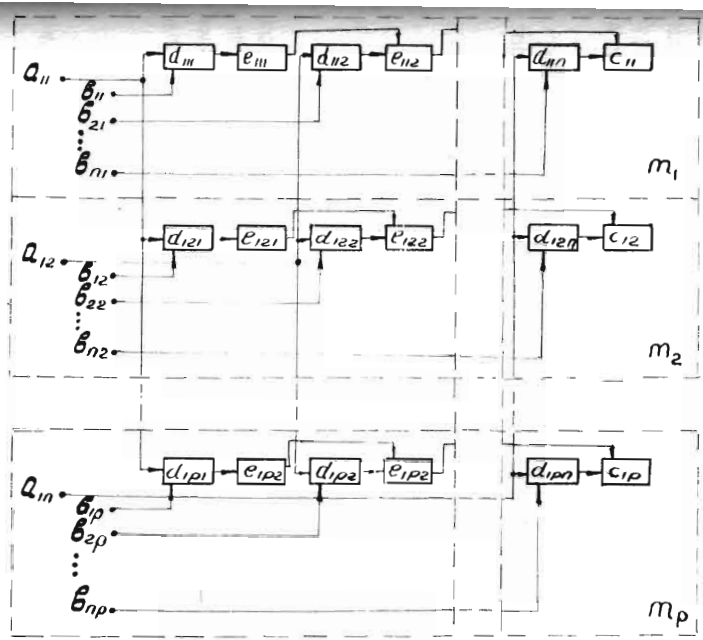

a_{11}
b_{11}
b_{21}
b_{n1}
d_{111}
e_{111}
d_{112}
e_{112}
d_{11n}
c_{11}
m_1
a_{12}
b_{12}
b_{22}
b_{n2}
d_{121}
e_{121}
d_{122}
e_{122}
d_{12n}
c_{12}
m_2
a_{1n}
b_{1p}
b_{2p}
b_{np}
d_{1p1}
e_{1p2}
d_{1p2}
e_{1p2}
d_{1pn}
c_{1p}
m_p

Рис. 7.

8. Составление программы решения задачи на УВС.

Программирование на УВС само состоит из нескольких этапов.

Для реализации выбранной схемы ρ-алгоритма необходимо прежде всего распределить кортежи покрытия между элементарными машинами. Будем предполагать, что число ЭМ в УВС равно ширине выбранного ρ-кортежа. В этом случае число возможных размещений, очевидно, равно $L!$. В УВС обмену информацией между двумя ЭМ m_i и m_j можно поставить в соответствие некоторый вес $p_{ij}=p_{ji}$; $p_{ij}=0$ при $i=j$. В сосредоточенных УВС, например, ЭМ, соединенным непосредственно друг с другом каналами связи, может быть придан вес I, а всем остальным парам машин - на единицу больший числа ЭМ, через посредство которых осуществляется обмен информацией между ними. В частности, в одномерных УВС p_{ij} может быть просто равно разности между номерами машин. В распределенных УВС величина p_{ij} должна учитывать в первую очередь время обмена между ЭМ и может быть выбрана пропорциональной последнему.

Характеристикой распределения кортежей между машинами с точки зрения обмена информацией будет не связность, а величина

$$PC = \sum_{i,j=1}^{L} p_{ij} C_{ij}, \qquad (30)$$

которую назовем связностью программы.

Задача заключается в выборе такого распределения кортежей между ЭМ, которое имело бы величину PC возможно меньшей и соответствовало бы возможностям системы коммутаций данной УВС. Для одномерных УВС, например, должны быть соблюдены условия, чтобы в каждый момент все ЭМ были разбиты на такие группы, в каждой из которых только одна ЭМ выводила бы информацию, а все другие ЭМ этой же группы могли только принимать ее. ЭМ, выводящая информацию, может одновременно принимать информацию от ЭМ, не входящей в данную группу. Соблюдение этого требования позволяет совместить пересылку кодов с операциями. Для распределенных ЭВМ этот этап имеет особо важное значение. Пример распределения кортежей операторов между ЭМ для задачи умножения матриц приведен на рис. 7. Соблюдение указанных выше правил упрощает также программы настройки и обмена информацией. Данная задача имеет сходство с задачей размещения стандартных элементов на плате, возникающей при проектировании ЭВМ и дру-

гой радиотехнической аппаратуры. При размещении стандартных элементов основное требование состоит в минимизации суммарной длины связей между элементами. В общем виде эта задача еще не решена, однако имеется ряд методов, значительно упрощающих её решение [19] - [21].

Для УВС, у которых переменной является только система коммутаций, после этого остается собственно программирование согласно схеме ρ-алгоритма, записанной, например, на матричном ρ-языке [4].

Программирование ведется для каждой элементарной машины отдельно. Однако в большинстве случаев (см. [3]) программы работы всех ЭМ либо полностью идентичны, либо отличаются лишь незначительными деталями.

Расширение списка команд каждой ЭМ за счет команд настройки, обмена информацией между ЭМ и обобщенного условного перехода не приведет к существенному усложнению процесса программирования. Благодаря этому программирование оказывается по сложности того же порядка, что и для одной ЭВМ.

В УВС с полностью переменной структурой, в которых имеется возможность изменения системы команд и других параметров, имеются большие возможности повышения эффективности решения задачи. При этом, однако, нужны более сложные методы программирования, обсуждение которых требует отдельного рассмотрения.

§ 4. Универсальная программа

I. В отличие от ЭВМ, у которых время решения задачи однозначно определяется алгоритмом ее решения и параметрами ЭВМ, у вычислительных систем имеется возможность выбирать время решения задачи путем изменения числа выполняющих ее ЭМ. Это свойство особенно важно при использовании УВС для управления какими-либо объектами, когда приходится одновременно решать несколько задач, срочность которых может изменяться в зависимости от ситуации. Решение этого же вопроса требуется также для обеспечения надежного управления объектами, когда функции вышедшей из строя ЭМ берут на себя остальные машины системы. При решении вычислительных задач это позволило бы сократить объем программирования, более оперативно решать задачи и т.п. Чтобы использовать эту возможность на практике, необходимо найти способы построения универсальных программ, которые при изменении числа ЭМ перестраивали бы себя либо сами, либо с помощью

некоторой преобразующей программы. Укажем на принципиальную возможность построения простых универсальных программ для ρ-алгоритмов, базовая точка которых лежит в области больших значений L, что имеет место для многих классов задач (см. [3]).

Пусть число ЭМ в УВС равно L. Тогда будем строить L-развертки ρ_B-кортежа. ρ_B - кортеж является прямоугольным с наличием возможно небольшого числа пустых операций. Его покрытие состоит из L_B кортежей. Дополним ρ_B-кортеж λ пустыми кортежами, так чтобы $L_B + \lambda = d \cdot L$, где $d > 0$ целое, $0 \le \lambda < L$, и разобьем дополненный ρ-кортеж на L полос шириной d. Строя теперь L-развертку, каждую полосу развернем в один простой кортеж. В результате получаем прямоугольный ρ-кортеж шириной L и длиной $h_B \cdot d$.

Покажем, что связность покрытия построенного ρ-кортежа не превышает связности ρ_B-кортежа, то есть $C \le C_B$.

Процессу построения нового ρ-кортежа шириной L и длиной $h = h_B \, d$ соответствует:

а) дополнение с двух сторон исходной матрицы связности λ нулевыми строками и столбцами;

б) разбиение матрицы на L^2 клеток размером $d \times d$ элементов;

в) замена клетки новым элементом. При этом диагональные клетки заменяются нулями, а каждый из остальных - элементом, значение которого не превышает суммы элементов соответствующей клетки исходной матрицы.

Отсюда следует, что связность покрытия нового ρ-кортежа не больше, чем у исходного.

То же самое можно сказать и о связности программы. Более того, если исходное покрытие было рациональным, то уменьшение связности программы может быть значительным, так как наиболее связанные друг с другом кортежи объединятся в один.

По аналогичным соображениям при переходе от ρ_B-кортежа к его L-развертке фактический объем исходной информации может только уменьшиться и приблизиться по значению к существенному объему. Естественно, что фактические объемы информации, приходящиеся на каждый кортеж покрытия, могут только увеличиваться. Это увеличение происходит из-за того, что общий фактический объем исходной информации делится между меньшим (в d раз) числом кортежей. Кроме того, увеличивается фактический объем промежуточной информации, приходящейся на каждый кортеж вследствие того, что результат каждой операции должен

храниться, по крайней мере, до тех пор, пока не начнется выполнение операций, входивших в ρ_β-кортеже в следующую по порядку ρ-операцию. При этом дополнительно придется хранить d кодов.

Таким образом, можно сделать общий вывод.

Если имеется рациональная программа решения задачи на УВС из L_β машин, соответствующим базовой точке, лежащей в области больших значений L , то программа для УВС с числом машин $L < L_\beta$ может быть получена путем сравнительно простых преобразований программы для базовой точки. При этом, если последняя была рациональной, то и полученная программа будет также рациональной для тех случаев, когда $L << L_\beta$, либо L_β кратно, либо близко к кратному L .

При этом, разумеется, объем информации, которую надо будет хранить в памяти любой из ЭМ, не должен превышать ее возможности, а время решения задачи выходить из допустимых границ из-за необходимости использования вспомогательных памятей.

Поступила в редакцию
I8.IX.I964 г.

Л И Т Е Р А Т У Р А

I. Евреинов Э.В. Универсальные вычислительные системы с частично переменной структурой. Данный сборник. стр.3-60.
2. Евреинов Э.В., Косарев Ю.Г. О возможности построения вычислительных систем высокой производительности. Новосибирск, Изд-во СО АН СССР, I962.
3. Евреинов Э.В., Косарев Ю.Г. О решении задач на универсальных вычислительных системах. Данный сборник,стр.I06-I6
4. Евреинов Э.В., Косарев Ю.Г. Матричный ρ -язык для описания параллельных алгоритмов. Данный сборник,стр.I00-I0
5. Канторович Л.В. Математические методы организации и планирования производства. Изд-во ЛГУ, I939.
6. Беллман Р. Динамическое программирование. Изд-во иностр. лит., I960.
7. Канторович Л.В. О перемещении масс. Докл. АН СССР, I942, т. 37, 7-8, 227-229.
8. Канторович Л.В., Гавурин М.К. Применение математических методов в вопросах анализа грузопотоков. В сб.:"Проблемы повышения эффективности работы транспорта". Изд-во АН СССР, I949, II0-I38.

9. Гольштейн Е.Г., Юдин Д.Б. Об одном классе задач планирования народного хозяйства. В сб."Проблемы кибернетики", 1961, вып. 5, 165-182.
10. Евреинов Э.В., Косарев Ю.Г. О методике разработки вычислительных систем. В сб.: "Вычислительные системы", 1963, вып. 6, 3-20 (Сиб. отд. АН СССР, Ин-т математики).
11. Ляпунов А.А. О логических схемах программ. Сб."Проблемы кибернетики", 1958, вып. 1, Физматгиз, 46-74.
12. Янов Ю.И. О логических схемах алгоритмов. Там же, стр. 75-127.
13. Neumann J. The general and logical theory of automata. В кн.: Cerebral Mechanisms in Behavior. The Hixon Symposium. L.A.Jeffress, New York-London, 1951, p.2070-2098. Русский перевод имеется в книге Тьюринг А. "Может ли машина мыслить?", Физматгиз, 1960, М., 59-101.
14. Марков А.А. Теория алгорифмов. Труды Матем. ин-та АН СССР им. Стеклова, XLII, 1954.
15. Калужнин Л.А. Об алгоритмизации задач. В сб."Проблемы кибернетики", 1959, вып. 2, 51-67.
16. Дятлов В.Л. и Мещанинов Л.С. Вычислительные системы и автоматизированные системы управления научными разработками. В сб.: "Вычислительные системы", 1964, вып. 11, 7-25 (Сиб. отд. АН СССР, Ин-т математики).
17. Авдеев Ю.А. и Николаева А.П. Управление сложными разработками по методу критического пути. Там же, 26-52.
18. Поспелов Г.С. и Тейман А.И. Метод логических диаграмм для планирования разработок сложных систем. Там же, 53-68.
19. Lee C.J. An algorithm for Path Connections and its Applications. IRE Trans.on EC, v.EC-10, N 3, 1961.
20. Харченко В.Л. О машинном методе проектирования соединений. В сб.: "Вычислительные системы", 1963, вып.6, 32-40 (Сиб. отд. АН СССР, Ин-т математики).
21. Тюренков В.А. Алгоритм нахождения кратчайшего пути. Там же, стр. 41-44.