ЭКОНОМИКА И МАТЕМАТИЧЕСКИЕ МЕТОДЫ 1969 том V, вып. 6

ОПТИМИЗАЦИЯ МАШИННЫХ ПРОГРАММ ПРИ ПОМОЩИ ТАБЛИЦ РЕШЕНИЙ

О. И. АВЕН, В. А. ДУШСКИЙ

(Москва)

Создание автоматизированных систем управления (АСУ) требует, в частности, разработки методов работы персонала и наборов машинных программ, обеспечивающих решение задач по обработке информации и подготовке решений на ЭВМ. На основе правил функционирования АСУ разрабатываются алгоритмы работы персонала и решения задач на ЭВМ.

Очевидно, что от качества этих алгоритмов зависит как характер работы персонала АСУ, так и сложность машинных программ, например, время, необходимое для их выполнения на ЭВМ.

Известно, что одна и та же задача может быть решена несколькими алгоритмами, поэтому при разработке АСУ надо выбрать наилучшие. Для этого можно использовать так называемые таблицы принятия решений [1], где даются описания правил функционирования дискретных управляющих систем или, как говорят, логики таких систем. При этом предполагается, что действие системы детерминированно определяется набором значений конечного числа ее входных параметров, каждый из которых может принимать одно из конечного множества значений. В зависимости от значений входных параметров система совершает одно из конечного числа действий. Иными словами, работа такой системы определяется функцией $A = F(C)$, где C — вектор входных параметров, A — вектор выходных параметров. Каждая из координат векторов A и C принимает значения из соответствующего этой координате конечного множества. Таблица принятия решений состоит из четырех квадрантов (табл. 1). В левых квадрантах записывают-

Таблица 1

	R_1	R_2	R_3	R_4	R_5	R_6
C_1	0	1	1	1	0	0
C_2	—	0	1	1	1	0
C_3	0	—	0	1	—	1
A_1	1	1	1	0	1	0
A_2	1	0	0	1	1	0

ся: в верхнем — названия координат входного вектора C, в нижнем — названия координат выходного вектора A. Правые два квадранта служат для записи функциональной зависимости A от C. Они разделяются на ряд вертикальных столбцов $R_1, R_2, \ldots, R_n$, каждый из которых называется правилом. Правила обычно нумеруются по порядку. На пересечении столбцов со строкой, соответствующей координате C_i входного вектора C, проставляется одно из возможных значений C_i или прочерк. На пересечении столбца-правила со строкой, соответствующей координате A_j выходного вектора A, проставляется одно из возможных значений A_j. Смысл такого правила определяется по столбцу так: если входные переменные приняли одновременно указанные в данном столбце-правиле значения (при этом прочерк

в строке C_i означает, что значение переменной C_i для рассматриваемого правила несущественно), то выходные переменные принимают значения, указанные в этом же столбце.

Например, правило R_2 (табл. 1) означает, что если $C_1 = 1$, $C_2 = 0$, вне зависимости от того, какое значение принимает C_3, $A_1 = 1$, $A_2 = 0$ или $F(1, 0, 0) = F(1, 0, 1) = (1, 0)$. Если, как в приведенном примере, каждый из входных параметров и выходных действий может принимать только одно из двух значений (0 или 1), то говорят, что имеется таблица с ограниченным входом. Таблицы, не удовлетворяющие этим ограничениям, называются таблицами с расширенным входом.

Понятно, что по всякой таблице с расширенным входом можно построить таблицу с ограниченным входом, описывающую работу той же системы (хотя и в других параметрах). Для этого, например, достаточно вместо каждой переменной C_i, принимающей одно из m_i значений, ввести $M_i =]\log_2 m_i[$ новых двоичных переменных $C_{i1}, \ldots C_{i, M_i}$ ($]a[$ есть наименьшее целое число, не меньшее чем a). Двоичные наборы длины M_i упорядочиваются обычным образом:

$$\begin{array}{c} 0\,0\,\ldots\,0\,0\,0 \\ 0\,0\,\ldots\,0\,0\,1 \\ 0\,0\,\ldots\,0\,1\,0 \\ \ldots\ldots\ldots \\ 1\,1\,\ldots\,1\,1\,1 \end{array}$$

Их число равно $2M_i \geqslant m_i$. Первые m_i из них сопоставляются значениям старой переменной C_i. Равенство переменной C_i какому-либо из ее значений равносильно равенству вектора $C_{i1}, \ldots, C_{iM^i}$, соответствующему двоичному вектору. Аналогичные преобразования проделываются с выходными переменными. То обстоятельство, что в правилах допускаются прочерки, дает возможность описания одной и той же функциональной зависимости $A = F(C)$ с помощью различных таблиц (таблицы, различающиеся только перестановкой строк и столбцов, считаются одинаковыми). В самом деле, если в табл. 1 заменим правила R_1 и R_5 на правила R' и R'' или на правила $R^{(1)}$, $R^{(2)}$ и $R^{(3)}$ табл. 2, оставив все остальные правила без изме-

Таблица 2

	R'	R''	$R^{(1)}$	$R^{(2)}$	$R^{(3)}$
C_1	0	0	0	0	0
C_2	—	1	0	1	1
C_3	0	1	0	0	1
A_1	1	1	1	1	1
A_2	1	1	1	1	1

нения, то получим таблицы, описывающие ту же самую зависимость A от C.

Этот факт имеет простое геометрическое объяснение. Один и тот же выходной вектор A получается при нескольких сочетаниях значений входных параметров. Например, в табл. 1

$$(1, 1) = F(0, 0, 0) = F(0, 1, 0) = F(0, 1, 1).$$

Остановимся пока на таблице с ограниченным входом.

Если дана таблица с n входными параметрами, то каждый набор значений входных переменных можно интерпретировать как вершину n-мерного единичного куба. Каждому возможному выходному вектору $\bar{A}$ отвечает множество $F^{-1}(\bar{A})$ вершин n-мерного куба, на которых функция F принимает значение A. Нетрудно проверить, что если правило R приводит к выходному вектору $\bar{A}$, то множество вершин куба $R^{-1}(\bar{A})$, к которым приме-

нимо правило R, есть k-мерная грань куба, где k — число прочерков в правиле R.

Таким образом, чтобы показать с помощью таблицы, что система дает выход $\bar{A}$ на множестве $F^{-1}(\bar{A})$, необходимо выбрать покрытие $F^{-1}(\bar{A})$ гранями куба, составить для каждой из граней покрытия соответствующее правило и все эти правила включить в таблицу. Хорошо известно, что такое покрытие может быть выбрано неоднозначно, что приводит к неоднозначности выбора таблицы, описывающей данную функциональную зависимость.

Возможная неоднозначность в задании функциональной зависимости $A = F(C)$ с помощью таблиц принятия решений ставит вопрос о выборе в каком-то смысле наиболее простой из возможных таблиц. Предлагаемые ниже меры сложности таблицы таковы, что чем проще (относительно них) таблица, тем она компактнее и, следовательно, тем проще обращение с ней. Если принять за меру сложности таблицы число содержащихся в ней единиц и нулей, то задача об отыскании таблицы минимальной сложности сводится к хорошо изученной задаче о минимизации дизъюнктивных нормальных форм булевых функций. Перенумеруем все фигурирующие в таблице векторы действий: $A^{(1)} = (A_1^{(1)}, \ldots, A_p^{(1)}), \ \ldots, A^{(l)} = (A_1^{(l)}, \ldots, A_p^{(l)})$, $p_{(i)}$ — число координат векторов A. Каждому возможному вектору A сопоставим булеву функцию n переменных

$$F^{(i)}(C_1, \ldots, C_n) = \begin{cases} 1, & \text{если } F(C_1, \ldots, C_n) = A^{(i)}, \\ 0, & \text{в противном случае.} \end{cases}$$

Нетрудно заметить, что минимальная в смысле введенной сложности таблица получается, если каждую из функций представить в минимальной дизъюнктивной нормальной форме (д.н.ф) и соответствующим образом по совокупности минимальных д.н.ф. для функций $F^{(i)}$ построить таблицу. При этом правила строятся следующим образом: каждое правило соответствует одному дизъюнктивному члену некоторой д.н.ф. какой-то из $F^{(i)}$; число правил, приводящих к действию $A^{(i)}$, равно числу дизъюнктивных членов минимальной д.н.ф. для $F^{(i)}$. По дизъюнктивному члену $C_{k_1}^{\varepsilon_1} C_{k_2}^{\varepsilon_2} \ldots C_{k_s}^{\varepsilon_s}$ минимальной д.н.ф. для $F^{(i)}$ соответствующее правило строится так: в строках с номерами $K_1, K_2, \ldots, K_S$ проставляются $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_S$ соответственно; остальные строки, соответствующие входным параметрам, заполняются прочерками; в строках действий проставляются $A_1^{(i)}, \ldots, A_p^{(i)}$. Легко видеть, что на некоторых из построенных описанным образом таблицах (известно, что булева функция может обладать и не одной минимальной д.н.ф.) минимизируется также число правил в таблице. Наконец, максимальное число значащих (содержащих ноль или единицу) ячеек в правилах также минимизируется на одной из таблиц, соответствующих минимальным д.н.ф. функций.

Таким образом, задача об упрощении таблиц принятия решений с ограниченным входом по представляющимся разумным определениям сложности сводится к нахождению минимальных д.н.ф. для системы функций $F^{(i)}$.

Отыскание минимальных д.н.ф. булевых функций представляет собой хорошо изученную проблему (хотя весьма трудоемкую в общем случае) и может быть осуществлено с помощью известных алгоритмов Куайна — Мак-Класки или Блэка [2]. Упрощение таблиц с расширенным входом происходит аналогично.

Очевидно, что таблица принятия решений, адекватно описывая функциональную зависимость, может быть преобразована в машинную программу, реализующую эту зависимость. Однако полезно ввести еще один

промежуточный этап — составление граф-схемы. Дело в том, что, хотя в таблице указано, какие условия должны быть проверены для принятия решения и как решение зависит от выполнения и невыполнения их, таблица не содержит прямых рекомендаций о порядке проверки условий. В то же время ясно, что, организуя проверку указанных условий в различной последовательности, можно получить программы, отличающиеся друг от друга как по своей сложности (требуемому объему памяти для их хранения), так и по среднему времени работы. Граф-схема представляет собой план, скелет программы, описывающий данную функциональную зависимость действий от условий вместе с некоторым порядком выполнения проверок. Граф-схемой (потокограммой) называется ориентированный граф (дерево), от каждой вершины которого, не являющейся конечной, отходит несколько ребер. Каждая неконечная вершина отмечена каким-либо условием, конечные — указаниями о предпринимаемых действиях. Если параметр C_i может принимать m_i значений, то от каждой вершины C_i отходит m_i ребер. Пусть пока все $m_i = 2$ (ограниченный вход). Будем считать, что левое ребро, отходящее от вершины C_i, соответствует результату проверки $C_i = 0$, правое ребро — результату $C_i = 1$. Процесс выполнения проверок и определения решений в соответствии с данной граф-схемой происходит так: начальный момент процесса соответствует корню дерева; шаг процесса — в вершине C_i, проверяем выполнение условия C_i и, если $C_i = 0$, сдвигаемся вниз по левому ребру, если же $C_i = 1$, сдвигаемся по правому. Решение, отвечающее данному распределению значений C_i, находится в конечной вершине описанного пути по граф-схеме.

Граф-схема называется соответствующей данной таблице, если они описывают одну и ту же зависимость $A = F(C)$. Например, табл. 3 соответствуют (среди прочих) граф-схемы рис. 1 и 2. Аналогично устанавливается соответствие между таблицами с расширенным входом и граф-схемами.

Таблица 3

C_1	0	1	0	—
C_2	0	0	0	1
C_3	0	—	1	—
A	0	1	1	0

Следует отметить, что граф-схемы используются наряду с таблицами принятия решения для описания самих функций $A = F(C)$. По-видимому, здесь предпочтение должно быть отдано таблицам, так как они не несут в себе избыточной информации об организации процесса проверок и потому их построение при описании систем происходит легче; в случае, когда мы имеем дело с системой, логика которой подвержена частым изменениям, внесение соответствующих изменений в правила таблиц решений происходит легче и менее трудоемко, нежели исправление граф-схем; наконец, обладая таблицей, можно (с помощью приводимых ниже алгоритмов) оптимизировать организацию работы системы в смысле объема программы или среднего времени ее работы.

Рассмотрим задачу о построении по данной таблице граф-схемы минимального объема. Эта задача исследовалась рядом авторов [3—10], предложивших несколько алгоритмов упрощения программ с использованием таблиц решений. Большинство из предложенных алгоритмов не обеспечивает минимальности программ и дает лишь возможность выбрать из имеющихся вариантов сравнительно более простой. Минимальная программа по этим алгоритмам получается лишь в некоторых частных случаях. Исключение составляет алгоритм, рассмотренный в [3], который рассчитан на переработку таблиц с ограниченным входом. Ниже дается

обобщение этого алгоритма применительно к таблицам с расширенным входом, которые более полно соответствуют реальным условиям. Алгоритм, описанный в [3], является частным случаем предлагаемого ниже итеративного алгоритма.

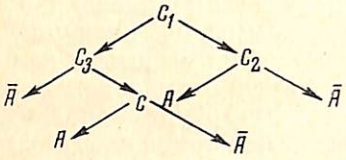

Рис. 1

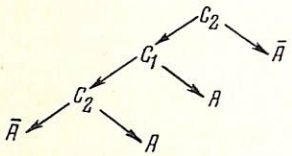

Рис. 2

Пусть Ω — совокупность всех наборов значений параметров $C_1, \ldots . C_n$ (совокупность всех целочисленных точек некоторого n-мерного параллелепипеда), точки Ω будут обозначаться ω_r;

$$\Omega(i_1 j_1, i_2 j_2, \ldots) = \{\omega: C_{i_1}(\omega) = j_1, C_{i_2}(\omega) = j_2, \ldots\},$$

$A(r)$ — набор действий, соответствующих элементу ω_r; b_i — стоимость проверки условия C_i;

$$D(r, s) = \{i: C_i\omega_r) \neq C_i(\omega_S)\}$$

$$\Delta_i{}^{rs} = \begin{cases} 1, \text{ если } D(r,S) = \{i\} \text{ и } A(r) = A(S), \\ 0, \text{ в противном случае;} \end{cases}$$

$$\nu_i{}^r = \begin{cases} 0, \text{ если существует такое } \omega_s, \text{ что } \Delta_i{}^{rS} = 1, \\ 1, \text{ в противном случае;} \end{cases}$$

$$\delta_i(i_1 j_1, \ldots) = \begin{cases} 0, \text{ если } \sum\limits_{\omega_r \in \Omega(i_1 j_1, \ldots)} \nu^r_i = 0, \\ 1, \text{ в противном случае;} \end{cases}$$

$$v_i(i_1 j_1, \ldots) = \sum_{k=1}^{n} b_k \Bigg[\sum_{1 \leqslant l_1 < l_2 \leqslant m_i} \delta_k(i_1 j_1, \ldots, il_1)\, \delta_k(i_1 l_1, \ldots, il_2) -$$

$$- \sum_{1 \leqslant l_1 < l_2 < l_3 \leqslant m_i} \delta_k(i_1 j_1, \ldots, il_1)\, \delta_k(i_1 j_1, \ldots il_2)\, \delta_k(i_1 j_1, \ldots, il_3) + \ldots (-1)^{m_i} \cdot$$

$$\cdot \delta_k(i_1 j_1, \ldots, i1)\, \delta_k(i_1 j_1, \ldots, i2) \ldots \delta_k(i_1 j_1, \ldots, i m_i) \Bigg] + b_i[1 - \delta_i(i_1 j_1, \ldots)].$$

Можно доказать, аналогично тому, как это сделано в [3], что сложность граф-схемы, равная суммарной стоимости всех неконечных ее вершин (где первым вычисляется C_i), не может быть меньше чем

$$\sum_{k=1}^{n} b_k \delta_k + \nu_i.$$

На существовании этой нижней оценки сложности граф-схемы и основывается предлагаемый алгоритм. Сначала вычисляются нижние сложности для граф-схем, начинающихся всеми возможными способами (на этом этапе имеется всего n возможностей). Среди возможных начал выбирается такое (начальное условие), которое имеет минимальную нижнюю оценку. Пусть все $m_i = 2$, т. е. рассматривается таблица с ограниченным входом. Обобщение для таблиц с расширенным входом очевидно.

Если выбрано начальное условие $C^{(1)}$, то дальнейшее течение процесса определения решения заключается в проверке оставшихся $n-1$ условий и распадается на два пути: первый из них соответствует невыполнению начального условия $(C^{(1)}=0)$, второй — выполнению $(C^{(1)}=1)$. Оба пути описываются таблицами, легко извлекаемыми из исходной таблицы. В таблицу, соответствующую $C^{(1)}=0$, относятся все правила исходной, для которых $C^{(1)}=0$, причем строка $C^{(1)}$ вычеркивается; аналогично получается таблица, соответствующая $C^{(1)}=1$.

Например, если на рис. 2 принять $C^{(1)}=C_1$, то таблицы, соответствующие $C^{(1)}=0$ и $C^{(1)}=1$, будут иметь следующий вид (табл. 4).

Таблица 4

$C^1=0$					$C_1=1$			
	C^2	0	0	1		C_2	0	1
	C^3	0	1	—		C_3	—	—
	A	0	1	0		A	1	0

Понятно, что правила с прочерком в строке $C^{(1)}$ перейдут в обе таблицы с соответствующими изменениями. Для выбранного начального условия строятся всевозможные продолжения граф-схем на один ярус, и для них вычисляются нижние оценки. Снова сравниваются все оценки, находящиеся на конечных вершинах естественно получающегося графа. Выбирается вершина с минимальной оценкой, для нее строятся всевозможные продолжения на один ярус и т. д. После некоторого числа циклов описанного процесса будет построена полная граф-схема для рассматриваемой функции $A=F(C)$. Она и является искомой. Аналогично будет составляться алгоритм построения для данной таблицы с ограниченным входом граф-схемы с минимальным средним временем работы. Предложенный в [4] алгоритм построения для данной таблицы с ограниченным входом граф-схемы с минимальным средним временем работы также может быть обобщен на случай таблиц с расширенным входом.

При постановке этой задачи, естественно, предполагается, что задано распределение вероятностей на множестве Ω, т. е. заранее известны вероятности того или иного сочетания результатов, проверок условий C_i.

Итак, пусть $p(\omega_r)$ — вероятность сочетания ω_r

$$p(i_1j_1,\ \ldots)=P\{C_{i1}(\omega)=j_1,\ldots,\}=\sum_{\omega\in\Omega(i_1j_1,\ldots)}p(\omega);$$

$$h_i(i_1j_1,\ldots)=b_i\sum_{\omega_S\in\Omega(i_1j_1,\ldots)\ \exists t(\Delta S_i^t=1)}p(\omega_S);$$

$$g(i_1j_1,\ldots)=\begin{cases}\dfrac{h_i(i_1j_1,\ldots)}{p(i_1j_1,\ldots)}, & \text{если } p(i_1j_1,\ldots)>0,\\ 0, & \text{в противном случае.}\end{cases}$$

$$H(i_1j_1,\ldots)=\sum_{k\neq i_1,\ldots}b_k-\sum_k g_k(i_1,j_1,\ldots).$$

Можно показать (так же, как в [4]), что математическое ожидание времени работы программы, определяемой граф-схемой с начальной вершиной C_i (теперь b_i интерпретируется как время, необходимое для проверки условия C_i), не может быть меньше чем $H+g_i$. Далее алгоритм строится на основании этой оценки так же, как алгоритм построения граф-схемы с минимальным объемом на основании нижней оценки объема схемы. Именно: на первом шаге вычисляются оценки для всевоз-

можных начал, затем для начала с минимальной оценкой строятся все продолжения на один ярус. На каждом из последующих шагов работы алгоритма вычисляются оценки для всех конечных вершин имеющегося графа «недостроенных граф-схем», среди конечных вершин выбирается вершина с минимальной оценкой, и для нее строятся всевозможные продолжения соответствующей этой вершине недостроенной граф-схемы на один ярус. Через конечное число шаров одна из граф-схем будет достроена. Эта граф-схема и является искомой, т. е. программа этой граф-схемы обладает минимальным средним временем работы.

В [11] предложен своеобразный способ вычисления решений по таблице принятия решений с ограниченным входом, минующий построение граф-схемы. По таблице принятия решений строятся две другие: маскирующая таблица M и таблица D. M получается из таблицы принятия решений заменой во всех правилах всех нулей и единиц на единицы, а прочерков — на нули. D получается из исходной таблицы сохранением всех единиц и заменой всех остальных ячеек нулями.

Отыскание решения для данного набора значений C_i производится следующим образом. Набор значений C_i поэлементно умножается на столбец матрицы, и вектор-произведение сравнивается с соответствующим столбцом D. Если вектор-произведение совпадает с ним, то принимается решение, соответствующее этому правилу, если нет — переходят к такому же испытанию следующего столбца. К сожалению, сравнение эффективности способа, предложенного в [11], с описанным в данной статье умозрительным образом представляется затруднительным.

Несмотря на то, что предложение об использовании таблиц решения для формализации описания управляющих систем было сделано несколько лет тому назад, методы их использования еще находятся в стадии разработки. В частности, в настоящее время ведутся работы по построению на основе таблиц принятия решений формализованных языков, предназначенных для описания функционирования систем управления АСУ (например, [12]). Некоторые возможности включения таблиц принятия решений в формальный язык содержатся в [13].

ЛИТЕРАТУРА

1. P. J. King. Decision tables. The Computer Journal., 1967, v. 10, № 8.
2. С. В. Яблонский. Функциональные построения в К-значной логике. Тр. Матем. ин-та АН СССР им. В. А. Стеклова, LI. М., Изд-во АН СССР, 1958.
3. L. T. Reinwald, R. M. Soland. Conversion of limited — entry decision tables to optimal computer programs. ACM Journal., 1967, v. 14, № 4.
4. L. T. Reinwald, R. M. Soland. Conversion of limited — entry decision tables to optimal computer program. ACM Journal., 1966, v. 13, № 2.
5. J. F. Egler. A procedure for converting logic table conditions into an efficient sequence of test instructions. Comm. ACM, 1963, v. 6, № 8.
6. M. Montalbano. Letter to editor (Egler's procedure refuted). Comm. ACM, 1964, v. 7, № 1.
7. S. L. Pollack. Conversion of limited entry decision tables to computer programs. Comm. ACM, 1965, v. 8, № 11.
8. L. I. Press. Conversion of decision tables to computer programs. Comm. ACM, 1965, v. 8, № 6.
9. V. G. Sprague, Letter to editor (On storage space of decision tables). Comm. ACM, 1966, v. 9, № 5.
10. J. R. Slagle. An efficient algorithm for finding certain minimum-cost procedures for making binary decisions. ACM Journal, 1964, v. 11, № 3.
11. H. W. Kirk. Use of decision tables in computer programming. Comm. ACM, 1965, v. 8.
12. O. Grindley. Systematics — a non-programming language for designing and specifying commercial systems for computers. The Computer Journal, 1967, v. 9.
13. И. О. Эванс. Описание структуры решения задачи при помощи логических таблиц. В сб. Современное программирование. Языки для экономических расчетов. М., «Сов. радио», 1967.

Поступила в редакцию
27 III 1969