Г. М. Адельсон-Вельский, Ф. М. Филлер, Программа вычисления сетевых графиков, *Ж. вычисл. матем. и матем. физ.*, 1965, том 5, номер 1, 144–148

Используя линейную интерполяцию по ξ и квадратичную по λ для аналитического представления подынтегральных выражений внутри области S, получим разностное представление уравнения с точностью 2-го порядка по ξ и 4-го по λ. Интеграл в правой части уравнения можно записать так:

$$2\int\limits_S\int \xi\,(f_\lambda F_\xi - f_\xi F_\lambda)\,d\lambda\,d\xi = 2\xi\int\limits_\Omega\int df\,dF, \tag{2.7}$$

где Ω — площадь в пространстве f, F. Интеграл в правой части (2.7) находится по значениям f и F в вершинах четырехугольника Ω.

Второе и третье уравнения системы (2.5) представлены приближенно в виде

$$F_{j+1} = F_{j-1} - \int\limits_{j-1}^{j+1}\frac{\Phi}{l}\,d\lambda = F_{j-1} + \left[\left(\frac{\Phi}{l}\right)_{j-1} + 4\left(\frac{\Phi}{l}\right)_j + \left(\frac{\Phi}{l}\right)_{j+1}\right]\frac{\Delta\lambda}{3};$$

аналогично и третье уравнение.

Получающаяся разностная схема по виду похожа на схему, рассмотренную в § 1, однако содержит большее количество неизвестных функций, что затрудняет счет с использованием этой схемы на машинах с малой памятью.

Рассмотренный здесь подход позволяет повысить точность аппроксимации по переменной ξ, для этого необходимо рассмотреть дополнительно $i-1$-й слой.

Поступила в редакцию
11.09.1964

Цитированная литература

1. С. В. Иорданский, Ю. Д. Шмыглевский. Сублимация осесимметричного тупого тела вблизи точки торможения набегающего потока. Инж. сб., 1960, 28, 26—35.
2. Р. Д. Рихтмайер. Разностные методы решения краевых задач. М., Изд-во ин. лит., 1960.
3. И. Ю. Браиловская, Л. А. Чудов. Решение уравнений пограничного слоя разностным методом. В сб. «Вычисл. методы и программирование». М., Изд-во МГУ, 1962, 167—182.
4. В. В. Щенников. Расчет ламинарного пограничного слоя у сублимирующей поверхности. Ж. вычисл. матем. и матем. физ., 1961, 1, № 5, 869—883.
5. А. Н. Тихонов, А. А. Самарский. Об однородных разностных схемах. Ж. вычисл. матем. и матем. физ., 1961, 1, № 1, 5—63.

УДК 51:681.14

ПРОГРАММА ВЫЧИСЛЕНИЯ СЕТЕВЫХ ГРАФИКОВ

Г. М. АДЕЛЬСОН-ВЕЛЬСКИЙ, Ф. М. ФИЛЛЕР

(*Москва*)

Рассматривается ориентированный граф Г без контуров (см. [1]); z_i — вершины графа, (z_i, z_j) — звенья. Звено (z_i, z_j) называется выходящим из вершины z_i и входящим в вершину z_j. Последовательность вершин графа Г $(z_{i_1}, z_{i_2}, \ldots, z_{i_n})$ называется путем, если для любой пары соседних вершин этой последовательности существует звено графа Г, выходящее из первой из этих вершин и входящее во вторую. Так как в графе Г отсутствуют контуры, а число его вершин конечно, то число различных путей также конечно.

Пусть каждому звену (z_i, z_j) графа Г поставлено в соответствие неотрицательное число t_{ij} — длина звена. Длиной пути $(z_{i_1}, z_{i_2}, \ldots, z_{i_n})$ называется сумма длин звеньев, образованных парами соседних вершин этого пути:

$$d(z_{i_1}, z_{i_2}, \ldots, z_{i_n}) = t_{i_1, i_2} + t_{i_2, i_3} + \ldots + t_{i_{n+1}\, i_n}. \tag{1}$$

Если путь состоит только из одной вершины, его длина равна нулю.

Сетевым графиком называется действительная функция, определенная на вершинах z_l графа Г формулой

$$f(z_l) = \max d(z_a, \ldots, z_l), \tag{2}$$

где максимум берется по всем путям, заканчивающимся вершиной z_l. Можно доказать, что значения этой функции $f(z_l)$ удовлетворяют равенству

$$f(z_l) = \max_{z_i \swarrow z_j} [f(z_i) + t_{i,l}]. \tag{3}$$

Здесь символ $z_i \swarrow z_j$ означает, что в графе Г существует звено (z_i, z_l).

Условие (3) является, по существу, принципом оптимальности Беллмана.

Если в некоторую вершину графа Г не входит ни одно звено, эта вершина называется начальной вершиной графа Г. Если из вершины не выходит ни одно звено, она называется конечной вершиной графа Г. Легко видеть, что для любой начальной вершины z_l выполняется равенство $f(z_l) = 0$.

Пусть дана вершина z_l графа Г. Путь максимальной длины, оканчивающийся вершиной z_l, называется лимитирующим путем для этой вершины. Если $(z_{i_1}, z_{i_2}, \ldots, z_j, z_l)$ — лимитирующий путь, то

$$d(z_{i_1}, z_{i_2}, \ldots, z_j, z_l) = f(z_l). \tag{4}$$

Предпоследняя в лимитирующем пути вершина z_j называется лимитирующим соседом для вершины z_l. Очевидно, имеет место равенство

$$f(z_l) = f(z_j) + t_{jl}. \tag{5}$$

Путь $(z_{i_1}, z_{i_2}, \ldots, z_{i_s})$, имеющий максимальную длину среди всех путей графа Г, называется критическим путем.

В настоящей статье описываются принципы работы программы вычисления сетевого графика и определения критического пути.

Для работы программы должна быть задана информация, определяющая граф Г, и длины его звеньев. Чтобы задать информацию о графе Г, его вершины z_i должны быть занумерованы натуральными числами. Нумерация может быть произвольной, однако в описываемой программе максимальный номер определяет величину массива, отведенного для информации о вершинах. Поэтому желательно, чтобы они были занумерованы от единицы подряд.

Исходная информация для программы разбита на элементы, соответствующие звеньям графа Г. Звену (z_i, z_j) длины t_{ij} соответствует элемент информации, занимающий одну ячейку памяти трехадресной автоматической вычислительной машины:

	i	j	t_{ij}

В первом и втором адресах ячейки записаны, соответственно, номера начальной и конечной вершин звена, а в третьем адресе — длина звена (масштаб выбран так, что длины звеньев — целые числа).

Эта информация вводится перед началом работы программы в некоторый массив ячеек памяти. Элементы информации могут быть расположены в произвольном порядке. Программа допускает задание информации с пропусками (в пропущенных ячейках должно стоять стандартное слово — нули во всех разрядах), однако это увеличивает время работы программы.

Первый этап работы программы — перегруппировка звеньев и создание начальной информации о вершинах.

Для каждой вершины z_i отыскиваются все входящие в нее звенья. Эти звенья переносятся в рабочий массив ячеек памяти, где они располагаются подряд, причем в ячейку s_i, соответствующую вершине z_i, записывается число входящих в эту вер-

шину звеньев и начальный адрес информации о них. Кроме того, определяются начальные и конечные вершины графа Г. Расположение начальной информации в ячейке s_i таково:

n	π_1	π_2	a_1	0	0

Здесь n — число входящих в z_i звеньев a_1 — адрес первого из этих звеньев, $\pi_1 = 1$, если z_i — начальная вершина, и 0 в остальных случаях; $\pi_2 = 1$, если z_i — конечная вершина, и 0 для остальных.

В результате первого этапа для каждой вершины z_i входящие в нее звенья оказываются некоторым образом перенумерованными; в ячейке a_1 лежит первое из них, в следующей ячейке a_2 — второе, и т. д.; в ячейке a_n лежит последнее звено, входящее в вершину z_i.

Второй этап работы — вычисление функции $f(z)$ и нахождение лимитирующих соседей для всех вершин графа Г. В ходе этого этапа работы информация о вершинах z_i в ячейках s_i имеет следующий вид:

n	π_1	π_2	a_1	s_γ	T_i

Здесь n, π_2 и a_1 сохраняют первоначальные значения. Пока $f(z_i)$ не найдено, $\pi_1 = 0$, s_γ — адрес информации о вершине, являющейся кандидатом в лимитирующие соседи, а T_i — оценка снизу для $f(z_i)$. Когда $f(z_i)$ вычислено, π_1 делается равным 1, s_γ — адресом ячейки лимитирующего соседа, а $T_i = f(z_i)$.

Заметим, что к началу второго этапа в ячейках начальных вершин графа стоит правильное значение $f(z) = 0$ и правильное значение $\pi_1 = 1$.

Вычисление функции $f(z_i)$ и определение лимитирующих соседей производится методом полного перебора (см. [2]).

Алгоритм полного перебора запрограммирован, как упорядоченная последовательность шагов двух типов, называемых в дальнейшем шаг влево и шаг вправо. Каждый из этих шагов начинается из некоторой вершины графа Г и определяет вершину, из которой будет произведен следующий шаг. Первый шаг делается из конечной вершины z_k графа Г. Из любой вершины z_i мы прежде всего пытаемся произвести шаг влево. Для этого рассматриваются звенья, входящие в вершину z_i, и выбирается звено, которое не рассматривалось на предыдущих шагах алгоритма. Пусть это звено (z_l, z_i); тогда следующий шаг будет произведен из вершины z_l. Кроме того, запоминается, что в вершину z_l мы пришли из z_i.

Пусть из рассматриваемой в данный момент вершины z_i шаг влево произвести нельзя. Это означает, что либо нет звеньев, входящих в z_i, т. е. z_i — начальная вершина графа Г, либо все звенья уже были рассмотрены на предыдущих шагах алгоритма. Очевидно, в ходе алгоритма первая возможность реализуется раньше, чем вторая.

Если из вершины z_i нельзя произвести шаг влево, из нее производится шаг вправо, причем $f(z_i)$ оказывается уже вычисленным. Справедливость последнего утверждения следует из описания алгоритма. Для первого шага вправо оно верно, так как он производится из начальной вершины графа Г, для которой $f(z) = 0$.

Пусть шаг вправо производится из вершины z_l, тогда $f(z_l)$ уже вычислено. Согласно сказанному выше, в ячейку s_l необходимо записать $\pi_1 = 1$. Рассматриваемый шаг вправо приводит в ту вершину z_i, из которой мы попали в z_l при шаге влево. Функция шага вправо состоит в определении T_i, являющегося оценкой снизу для $f(z_i)$, и кандидата в лимитирующие соседи. В начале второго этапа для всех вершин $T = 0$. При шаге вправо из вершины z_l в вершину z_i вычисляется

$$\tau = f(z_l) + t_{i,\,l}. \tag{6}$$

Из (3) следует, что $f(z_i) \geqslant \tau$. Кроме того, согласно индуктивному предположению, $f(z_i) \geqslant T_i$ к данному моменту, которое в дальнейшем будет обозначаться $T_i^{\text{ст}}$. Следовательно,

$$f(z_i) \geqslant \max(\tau,\ T_i^{\text{ст}}), \tag{7}$$

т. е. оценка $T_i^{\text{нов}}$ после шага вправо определяется по формуле

$$T_i^{\text{нов}} = \max(\tau, T_i^{\text{ст}}). \tag{8}$$

Если τ больше, чем $T_i^{\text{ст}}$, адрес s_l, соответствующий z_l, записывается в ячейку s_i в качестве информации о кандидате в лимитирующего соседа для z_i.

Таким образом, в любой момент алгоритма

$$T_i = \max_l [f(z_l) + t_{l,\,i}], \tag{9}$$

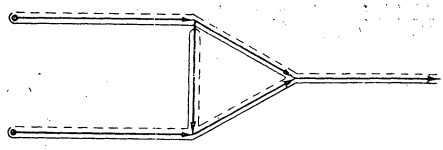
где максимум берется по всем $z_l \leftarrow z_i$, которые рассмотрены к данному моменту; значение s_ν соответствует вершине z_ν, для которой

$$T_i = f(z_\nu) + t_{\nu,\,i}. \tag{10}$$

После этого шага вправо производится очередной шаг влево из вершины z_i, если он возможен. Можно доказать, что если из вершины z_i произведен шаг влево, то через некоторое число шагов алгоритма мы вернемся в эту вершину шагом вправо. Таким образом, к некоторому моменту будут рассмотрены все звенья, входящие в z_i. Из формул (9) и (10) следует, что тогда $f(z_i) = T_i$, s_ν — адрес лимитирующего соседа и из точки z_i следует произвести шаг вправо. Алгоритм заканчивается, когда вычислено значение $f(z_k)$ для конечной вершины графа и определен ее лимитирующий сосед.

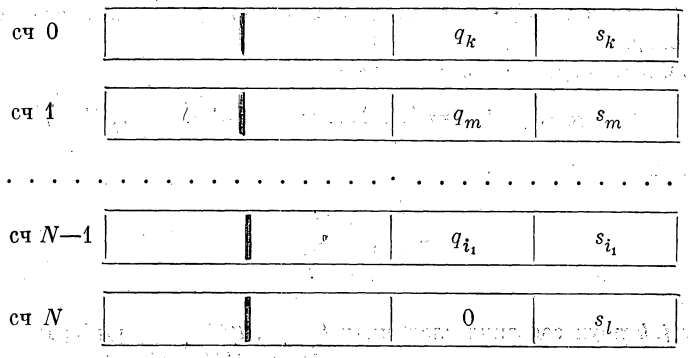
Если в данном графе Г имеется не одна конечная вершина, то необходимо повторить тот же алгоритм для каждой конечной вершины, после чего функция $f(z)$ будет вычислена во всех вершинах графа.

В описанном выше алгоритме каждое звено обрабатывается два раза: при шаге влево и при шаге вправо. Поэтому количество действий в алгоритме пропорционально первой степени числа звеньев в графе Г.



На фигуре изображен ход работы алгоритма в конкретном случае. Здесь пунктиром обозначены шаги влево, сплошной линией — шаги вправо.

Следует остановиться на одном важном обстоятельстве, связанном с осуществлением алгоритма полного перебора. Пусть в некоторый момент рассматривается вершина z_l. Для правильной работы алгоритма мы должны знать вершину z_{i_1}, из которой мы попали в z_l шагом влево, вершину z_{i_2}, из которой мы попали в z_{i_1} шагом влево, и т. д., до z_k, т. е. некоторый путь $z_l, z_{i_1}, z_{i_2}, \ldots, z_m, z_k$. Этот путь запоминается в системе счетчиков, сч 0, сч 1, ..., сч N. Информация, хранящаяся в этих счетчиках, имеет такой вид:



В каждом счетчике записан адрес s_ν вершины, из которой делается соответствующий шаг влево, и номер q_ν звена, входящего в эту вершину, по которому этот шаг производится. После каждого шага влево из данной вершины z_ν соответствующее q_ν

наращивается на 1. Если q_v стало равным числу звеньев, входящих в вершину z_v (информация о числе звеньев лежит в ячейке s_v), то после возврата в эту вершину из нее будет произведен шаг вправо.

Третий этап работы программы — определение критического пути. Прежде всего определяется вершина z_k с максимальным значением $f(z_k)$ и из ячейки s_k извлекается информация о ее лимитирующем соседе. Для этой вершины также находится лимитирующий сосед и т. д. до прихода в начальную вершину графа. Последовательность найденных вершин (в обратном порядке) и является критическим путем.

Последний этап работы программы — печать результатов.

Заметим, что исходная информация в описанном выше виде может определять граф с контурами. Тогда алгоритм полного перебора зацикливается. Поэтому в программе предусмотрена система контроля, которая обнаруживает и печатает имеющиеся контуры.

Можно составить программу по изложенным принципам, предусматривающую, что информация о графе лежит во внешних запоминающих устройствах. Однако в действующей программе предполагается, что вся информация лежит в оперативной памяти машины. По мнению авторов, любой алгоритм, оперативно использующий внешнюю память, приводит к столь большому времени работы программы, что если информация о графе не помещается в оперативной памяти машины, следует воспользоваться методом выделения частей сети.

Поступила в редакцию
24.01.1964

Цитированная литература

1. К. Берж. Теория графов и ее применения. М., Изд-во ин. лит., 1962.
2. А. Л. Брудно. Грани и оценки для сокращения перебора вариантов. В сб. «Проблемы кибернетики», № 10. М., «Наука», 1964.

УДК 51:681.14

МЕТОД МАТРИЧНОГО ПЕРЕБОРА И ЕГО ПРИМЕНЕНИЕ К ОДНОЙ ЗАДАЧЕ ТЕОРИИ ГРАФОВ

И. Х. СИГАЛ, В. А. ЧЕБАКОВ

(*Москва*)

В работе [1] дается метод упорядоченного перебора применительно к задачам оптимального управления. Но этот метод позволяет решать значительно более широкий круг задач. В настоящей заметке дается описание этого алгоритма без связи с задачами оптимального управления и приводится редукция к этой схеме одной задачи теории графов.

1. Пусть имеется трехмерная матрица

$$\{l_{jk}{}^i\}, \qquad j, k = 1, 2, \ldots, m, \qquad i = 1, 2, \ldots, N,$$

и необходимо найти

$$L = \min_{j,\,k} \sum_{i=1}^{N} l_{jk}{}^i,$$

где индексы j, k двух соседних слагаемых $l^i_{j_i, k_i}$, $l^{i+1}_{j_{i+1}, k_{i+1}}$ выбираются с соблюдением условия $k_i = j_{i+1}$.

Возьмем на плоскости $N+1$ вертикальных прямых и поместим на каждой из них по m точек, для определенности занумерованных сверху вниз. Будем теперь