ЭКОНОМИКА
И МАТЕМАТИЧЕСКИЕ МЕТОДЫ

1965
том I, вып. 3

ЭКОНОМНЫЙ АЛГОРИТМ ВЫДЕЛЕНИЯ КРИТИЧЕСКИХ ПУТЕЙ В ОРИЕНТИРОВАННОМ ГРАФЕ БЕЗ КОНТУРОВ

И. Б. ЗАДЫХАЙЛО

(Москва)

В статье рассматривается простой алгоритм выделения на вычислительной машине критических путей в ориентированном графе без контуров *. Алгоритм описывается так, что переход от этого описания к программе на конкретную вычислительную машину весьма прост. Количество тактов машины, необходимых для выделения критических путей по этому алгоритму, равно $p \cdot c$, где p — число дуг в графе, а c — константа, зависящая от системы команд машины. Автор составил вариант программы, в котором $c \approx 26$ (см. раздел 4).

Алгоритм, излагаемый в настоящей статье, представляет вариант известного алгоритма Форда [1]; поэтому автор подробно останавливается лишь на специфических вопросах реализации его на ЭВМ. Данный алгоритм отличается от известных вариантов алгоритма тем, что в процессе выполнения каждого этапа вычислений просматриваются не все дуги подряд, а после рассмотрения каждой дуги определяются очередные дуги, подлежащие рассмотрению. Указанное обстоятельство позволяет существенно сократить программу всей задачи и соответственно уменьшить количество необходимых машинных операций.

В отличие от алгоритмов, описанных в работах [2] и [3], программа, реализующая предлагаемый алгоритм, может эффективно использовать внешнюю память вычислительной машины.

1. Постановка задачи. Пусть задан произвольный ориентированный граф без контуров, имеющий n вершин и p дуг, и каждой дуге поставлено в соответствие некоторое неотрицательное число, которое будем называть длиной дуги. Обозначим через x_0 множество вершин, в каждую из которых не заходит ни одна дуга. Для каждой из оставшихся вершин среди всех путей, ведущих к ней от элементов множества x_0, нужно выделить такой путь, чтобы сумма длин дуг, составляющих его, была максимальной. Требуется вычислить также соответствующую сумму длин дуг, которую мы будем называть максимальной длиной пути до данной вершины **

2. Величины, используемые в алгоритме. Введем в рассмотрение следующие величины.

1) Вектор A из p компонент.

Каждая компонента A_i вектора A соответствует некоторой дуге i и, в свою очередь, состоит из трех величин: a_i, b_i, c_i.

Величины a_i и b_i задают соответственно номер вершины, из которой исходит дуга i, и номер вершины, в которую заходит дуга i.

Величина c_i — длина дуги i.

---

* Автор придерживается в дальнейшем терминологии, принятой в работе [1].

** Из дальнейшего будет видно, что предлагаемый алгоритм можно, слегка изменив, применять и для определения минимальной длины пути.

В дальнейшем предполагается, что компоненты вектора A перенумерованы так, что они образуют группы: сначала идут все дуги, исходящие из какой-либо одной вершины, затем все дуги, исходящие из любой другой вершины, и т. д. Нумерация внутри группы произвольная.

2) Вектор B из n компонент.

Каждая компонента B_j вектора B состоит, в свою очередь, из шести величин: l_j, f_j, g_j, h_j, s_j, m_j.

Величина l_j — число дуг, заходящих в вершину с номером j.

Величина f_j — номер i_j, с которого начинаются компоненты вектора A, соответствующие дугам, исходящим из вершины j (т. е. A_{ij} — первая компонента, для которой $a_{i_j} = j$).

Переменная величина g_j в конце вычислений станет равной максимальной длине пути до вершины с номером j. Вначале $g_j = p$, $j = 1, 2, \ldots, n$.

Переменная величина h_j к концу работы станет равной номеру той вершины, из которой исходит дуга максимального пути до вершины j, заходящая в вершину j.

Величина $s = \{s_1, s_2, \ldots, s_n\}$ — переменная. В процессе вычисления в последовательные компоненты этой величины попадают номера вершин, подлежащих обработке (см. ниже описание алгоритма). Вначале заполняются лишь k первых компонент $s_1, s_2, \ldots, s_k$ этой величины (k — число вершин, принадлежащих множеству x_0, а $s_1, s_2, \ldots, s_k$ — номера этих вершин).

Величина m_j — это число дуг, исходящих из вершины с номером j.

Рассмотрим также две скалярные величины q и $q1$, положив вначале $q = 0$ и $q1 = k + 1$.

Следует отметить, что задача полностью описывается заданием лишь вектора A. Остальные величины носят вспомогательный характер и либо нужны, по существу, для организации алгоритма, либо введены для упрощения понимания алгоритма. В дальнейшем будет рассмотрен вопрос о том, как лучше организовать программу по данному алгоритму и какие величины можно исключить.

3. Описание алгоритма. Алгоритм может быть записан в следующем виде.

1. Прибавляем к q единицу. Если величина q стала равной $n + 1$, то алгоритм закончен. В противном случае полагаем вспомогательную скалярную величину r равной f_{s_q}.

2. Пользуясь полученным значением r, полагаем вспомогательную величину $r1$ равной b_r, а вспомогательную величину $r2$ равной a_r.

3. Уменьшаем на единицу значение величины l_{r1}.

4. Если значение величины l_{r1} стало равным нулю, то полагаем величину s_{q1} равной $r1$ и увеличиваем $q1$ на единицу.

5. Если выполнено неравенство $g_{r1} \geqslant g_{r2} + c_r$, то сразу переходим к п. 6, в противном случае выполняем следующие вычисления: $g_{r1} = g_{r2} + c_r$, $h_{r1} = r2$.

6. Прибавляем единицу к r и вычитаем единицу из m_{r2}.

7. Если величина m_{r2} стала равной нулю, то переходим к п. 1, если нет, то переходим к п. 2.

В результате выполнения данного алгоритма величины g_j принимают значения длин соответствующих максимальных путей, а сами пути описываются значениями h_j. Действительно, значение h_j есть номер той вершины, из которой осуществляется переход по критическому пути в очередную вершину с номером j. Таким образом, значение h_j представляет номер предыдущей вершины, лежащей на критическом пути. Аналогичным образом могут быть получены номера других вершин. Процесс продолжается

до тех пор, пока не будет совершен переход к вершине, номер которой принадлежит множеству x_0.

Заметим, что величины a_i $(i = 1, 2, \ldots, p)$ и m_j $(j = 1, 2, \ldots, n)$ использовались в алгоритме лишь для определения конца обработки группы дуг, исходящих из данной вершины. Вместо этого для той же цели достаточно ввести двоичную величину $\bar{a}_i$, которая равна единице, если данная дуга — последняя в группе, и равна нулю в противном случае.

Приведенный выше алгоритм можно теперь записать в следующем виде.

1. Прибавляем к q единицу. Если величина q стала равной $n + 1$, то алгоритм закончен, в противном случае полагаем вспомогательную величину $r2$ равной s_q, а вспомогательную величину r равной f_{r2}.
2. Полагаем вспомогательную величину $r1$ равной b_r.
3. Уменьшаем величину l_{r1} на единицу.
4. Если величина l_{r1} стала равной нулю, то полагаем $s_{q1} = r1$ и прибавляем к $q1$ единицу. В противном случае переходим к п. 5.
5. Если выполнено неравенство $g_{r1} \geqslant g_{r2} + c_r$, то переходим к п. 6, в противном случае выполняем вычисления: $g_{r1} = g_{r2} + c_r$, $h_{r1} = r2$.
6. Если $\bar{a}_r = 1$, то переходим к п. 1, в противном случае прибавляем к r единицу и переходим к п. 2.

4. Оценка используемой памяти и количества машинных тактов. Подсчитаем сначала число двоичных разрядов, необходимых для размещения в памяти машины введенных выше векторных величин.

Число разрядов, необходимых для записи всех величин b_i и $\bar{a}_i$ $(i = 1, 2, \ldots, p)$, примерно равно $(\lg_2 n + 1) \cdot p$.

Пусть для записи максимальной возможной длины дуги требуется N_1 двоичных разрядов; тогда для размещения всего вектора A потребуется примерно $[(\lg_2 n + 1) + N_1] \cdot p$ двоичных разрядов.

Далее, пусть для записи длины максимально возможного пути требуется N_2 двоичных разрядов. Тогда для размещения вектора B в памяти машины потребуется примерно $(\lg_2 p + N_2 + 3 \lg_2 n) \cdot n$ двоичных разрядов.

Общее число двоичных разрядов, нужных для размещения всех величин, $N_п$, может быть определено по формуле

$$N_п \approx p(N_1 + \lg_2 n + 1) + n(N_2 + 3 \lg_2 n + \lg_2 p).$$

Практически (на наиболее распространенных машинах и для типовых задач) обычно требуется p ячеек памяти для размещения вектора A и $2n$ ячеек для вектора B.

Оценим теперь число машинных тактов, необходимых для реализации предлагаемого алгоритма на вычислительной машине. Очевидно, что искомое число тактов пропорционально первой степени p. В зависимости от вычислительной машины и кодировки исходной информации коэффициент пропорциональности может быть различным. Автором составлена программа из 32 команд, реализующая данный алгоритм на машине М-20 при следующих предположениях: $n \leqslant 500$, $p \leqslant 3000$, $N_1 \leqslant 12$, $N_2 \leqslant 24$.

Внутренний цикл обсчета последовательных дуг содержит 26 команд, каждая из которых выполняется не более одного раза при обработке очередной дуги. Остальные 6 команд выполняются по одному разу при каждом переходе к обработке новой группы дуг, исходящих из очередной вершины (п. 1 алгоритма). Векторы A и B занимают в памяти машины $p + n$ ячеек. (Всего имеется два массива ячеек: в каждой ячейке одного из них содержатся компоненты векторных величин: b, c, $\bar{a}$, l, h, а в каждой ячейке другого — компоненты векторных величин s, g, u, f).

При реализации алгоритма на машине, имеющей четыре индекс-регистра и команды сборки разрядов ячейки по заданной шкале [4], число команд уменьшается более чем на одну треть.

Отметим также, что при реализации алгоритма можно (до известных пределов) эффективно использовать внешнюю память машины. При этом во внутренней памяти машины нужно иметь лишь $2n$ ячеек под величины вектора B и массив ячеек для одной группы дуг (величины $\bar{a}, b, c$), исходящих из какой-либо вершины. Вызов информации для любой другой группы дуг можно осуществлять по номеру вершины (подобно тому как производится вызов стандартной программы по ее номеру в интерпретирующей системе ИС-2, широко используемой при эксплуатации машины М-20 [5]). В соответствии с рассмотренным алгоритмом каждая группа вызывается во внутреннюю память машины только один раз и сразу используется.

5. О задании исходной информации. Удобно задавать в качестве начальных данных те величины, которые использовались в описании алгоритма. Однако в работах такого рода обычно принято считать, что исходную информацию задают в виде случайной перестановки компонент вектора $A(a, b, c)$. В этом случае легко построить дополнительную программу, преобразующую исходную информацию к виду, удобному для работы предлагаемого алгоритма (дополнительная программа может быть реализована за $c_1 \cdot p$ машинных тактов). После этого применяется алгоритм, рассмотренный в настоящей статье.

ЛИТЕРАТУРА

1. К. Берж. Теория графов и ее применения. М., Изд. иностр. лит., 1962.
2. Г. М. Адельсон-Вельский, Ф. М. Филлер. Программа вычисления сетевых графиков. Ж. вычисл. матем. и матем. фаз., 1965, т. 5, № 1.
3. О. К. Жданов, В. В. Кириллов, В. В. Коньков. Метод решения задачи управления (планирование и контроль) разработками. Ж. вычисл. матем. и матем. физ., 1965, т. 5, № 1.
4. Г. М. Адельсон-Вельский и др. О системе команд для трехадресной машины без регистра адреса. Докл. АН СССР, 1964, т. 154, № 3.
5. Библиотека стандартных программ. Под общей редакцией М. Р. Шура-Бура. М., ЦБТИ, 1961.

Поступила в редакцию
11 II 1965