ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ

Сборник трудов
1964 г. Института математики СО АН СССР Выпуск II

ОБ ОДНОМ АЛГОРИТМЕ НАХОЖДЕНИЯ КРИТИЧЕСКОГО ПУТИ СЕТЕВОГО ГРАФИКА

Л.Т. Петрова, Н.Н. Карнаухова

I. Постановка задачи и алгоритм решения

При рассмотрении вопросов планирования комплексных исследовательских и производственных работ и управления ими возникает следующая задача.

Имеется ориентированный граф без контуров, содержащий M вершин и N дуг. Каждой дуге u_{ij}, ведущей из вершины i в вершину j, однозначно соотносится число $t_{ij} \geq 0$, которое будем называть длиной дуги. Путем в графе называется такая последовательность дуг, в которой конец каждой предыдущей дуги совпадает с началом следующей. Под длиной пути будем понимать сумму длин всех дуг, принадлежащих данному пути. Такой граф назовем сетевым графиком. Для данного сетевого графика требуется найти путь (или пути) с наибольшей длиной - критический путь и определить его длину.

Эта задача, с одной стороны, легко сводится к транспортной задаче линейного программирования, с другой стороны -близка вопросам, рассматривавшимся при разработке схемной символики.*) Поэтому она может решаться методами, разработанными в

*) См.,напр., Л.В.Канторович "Об одной математической символике,удобной при проведении вычислений на машине," ДАН,1957,113, № 4, 738-741 и Л.Т.Петрова "Некоторые применения схемной символики". Журн. выч.мат. и мат.физ. т.I, № 3, 1961, 513-522.

указанных разделах. Однако, представляется целесообразным разработать специальные алгоритмы, которые бы более полно учитывали специфику рассматриваемой задачи.

Практические задачи порождают графы с числом вершин порядка нескольких тысяч, т.е. являются задачами с большим объемом входной информации. В этих условиях простота алгоритма и удобство его машинной реализации являются решающими обстоятельствами при выборе алгоритма.

Опишем один простой алгоритм решения поставленной задачи. Пусть сетевой график задан списком своих дуг и их длин так, что информация о каждой дуге задается тройкой чисел:

$$(t_{ij}, i, j),$$

где i - номер вершины, являющейся началом дуги, j - номер вершины, являющейся концом дуги, t_{ij} - длина дуги.

Предположим, что для каждой вершины $k=1,2,\ldots,M$ каким-нибудь способом определены числа: T_k' - максимальная длина пути, исходящего из вершины k и T_k'' - максимальная длина пути, заканчивающегося в вершине k. Тогда легко найти критический путь и его длину. Очевидно, длина критического пути

$$T^* = \max_k T_k' = \max_k T_k'' = \max_k (T_k' + T_k''),$$

а вершины k, для которых $T_k' + T_k'' = T^*$, лежат на критическом пути. Для вычисления величин T_k' и T_k'' может использоваться следующий конечный процесс.

Первоначально полагаем $T_k' = T_k'' = 0$, $k=1,2,\ldots,M$

На ℓ-м шаге выполняются следующие вычисления. Последовательно просматриваются все дуги графика, и для каждой дуги с информацией (t_{ij}, i, j) перевычисляются значения T_i' и T_j'' по формулам:

$$T_i' = \max [T_i'^*, T_j'^* + t_{ij}],$$
$$T_j'' = \max [T_j''^*, T_i''^* + t_{ij}],$$

где $T_k'^*$ и $T_k''^*$ - значения T_k' и T_k'', полученные к моменту перевычисления. Если на ℓ-м шаге сделано хотя бы одно исправление, т.е. по крайней мере для одной дуги $T_i'^* < T_j'^* + t_{ij}$ или $T_j''^* < T_i''^* + t_{ij}$, то выполняется следующий $(\ell+1)$-ый шаг (снова просматриваются все дуги графика). В противном случае процесс построения величин T_k' и T_k'' $(k=1,2,\ldots,M)$ заканчивается.

Число шагов в указанном процессе, очевидно, не превосходит $(L+1)$, где L - максимальное число дуг в пути данного сетевого графика.

2. Временная интерпретация сетевого графика

В реальных задачах сетевой график может, в частности, определять последовательность выполнения некоторых событий (вершины графика) при заданных временных интервалах между ними (длина дуги t_{ij} есть нижняя оценка величины временного интервала от события i до события j). В такой интерпретации длина критического пути определяет наименьший временной интервал, в течение которого могут совершиться все события графика. В этой связи практический интерес представляет определение промежутка допустимых сроков t_k выполнения каждого из событий, не повышающих оптимального срока T^* выполнения всего комплекса событий. Эти промежутки, очевидно, определяются неравенствами:

$$T_k'' \leq t_k \leq T^* - T_k', \quad k=1,2,\ldots,M.$$

Таким образом, величина

$$T_k = T^* - T_k'$$

определяет самый далекий допустимый срок выполнения события k, а величина

$$r_k = T^* - T_k' - T_k''$$

- резерв времени, характеризующий размер допустимого сдвига реального срока относительно T_k. Величины T_k и r_k могут быть вычислены при реализации нашего основного алгоритма. Ясно, что для вершин, лежащих на критических путях (и только для них), $r_k = 0$; для начальных вершин (не имеющих входящих дуг) $T_k = r_k$; для конечных вершин (не имеющих исходящих дуг)

$$T_k = T^*.$$

Таким образом, описанный алгоритм позволяет помимо критического пути и его длины получить также другие важные характеристики сетевого графика. Результаты можно представить в виде таблицы:

$$(T_k, k, r_k), \quad k=1,2,\ldots,M.$$

Для удобства обозрения и использования результатов (особенно в случае больших графиков) целесообразно упорядочить указанную таблицу по возрастанию r_k и в качестве окончательного

результата представить лишь часть таблицы, отвечающую небольшим значениям τ_k, так как именно эти данные представляют наибольший практический интерес. Алгоритм такого упорядочения таблицы рассмотрен в 4.

3. Реализация алгоритма на машине

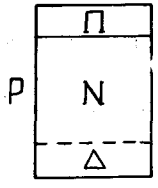
Имеется в виду некоторая трехадресная машина. Таблица исходных данных имеет вид $(t, i, j)_n$, $n=1,2,\ldots,N$. Информация о каждой дуге размещается в одной ячейке. Объем оперативной памяти машины обозначим через P, объем основной программы (включая рабочие ячейки и константы) – Π. Помимо программы и исходных данных в памяти машины необходимо разместить поля для хранения чисел T'_k и T''_k, $k=1,2,\ldots,M$. При этом отметим, что в одной ячейке можно разместить несколько таких чисел. Так, например, если единицей измерения времени считать неделю и на число отвести девять разрядов, то рассматриваемый временной интервал может достигать почти 10 лет ($2^9 = 512$).

Рассмотрим три варианта программы, вычисляющей для данного сетевого графика таблицу

$$(T_k, k, \tau_k), \quad k=1,2,\ldots,M$$

Из них первые два варианта предполагают использование только оперативной памяти машины, а третий включает использование магнитных запоминающих устройств.

1) Здесь предполагается, что число вершин $M \leq \frac{P-\Pi}{2}$, а число дуг $N \leq P-\Pi$. В этом случае исходная информация о каждой дуге графика размещается в адресной части ячейки на поле N, а для хранения чисел T'_k и T''_k, $k=1,2,\ldots,M$ отводятся разряды кода операции на полях N и Δ. После вычисления T'_k и T''_k строится полная таблица результатов



$$(T_k, k, \tau_k), \quad k=1,2,\ldots,M$$

Эта таблица размещается на поле исходных данных. Пусть, например, P = 4096. Тогда по этой программе могут анализироваться графики с 2000 вершин. Число дуг при этом не должно превышать 4000. Такой вариант программы реализован для ЭВМ СО АН СССР. В этом случае Π = 40. По программе выполнены некоторые экспериментальные вычисления.

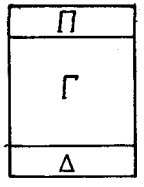
2) В этом случае T'_k и T''_k размещаются на разрядах кода операции на поле N, а также на поле Δ по ν чисел в ячейке. При этом должно выполняться следующее соотношение между объемом памяти и размерами графика:

$$N + \nu\Delta \geq 2M$$

Например, при P = 4096 и ν = 5 могут анализироваться по этой программе сетевые графики, в которых

$$M \leq 2500, \quad N \leq 3750;$$
$$M \leq 2700, \quad N \leq 3650;$$
$$M \leq 3000, \quad N \leq 3500;$$
$$M \leq 3200, \quad N \leq 3400.$$

3) Смыслом задачи обусловливается тот факт, что даже при большом числе вершин графика (порядка нескольких тысяч) максимальное число дуг в пути сравнительно невелико (несколько десятков). Эта особенность задачи определяет удовлетворительную эффективность алгоритма (число просмотров исходной информации порядка максимального числа дуг в пути), а также позволяет сделать некоторые рекомендации относительно распределения памяти. Величины T'_k и T''_k, $k=1,2,\ldots,M$, удобно хранить в оперативной памяти (на поле Γ), а исходную информацию можно разместить во внешней памяти в магнитных запоминающих устройствах. Для обработки исходная информация частями считывается в оперативную память на поле Δ. Если в каждой ячейке на поле Γ хранятся ν чисел T'_k или T''_k, то на число вершин графика накладывается следующее ограничение:



$$2M \leq \nu\Gamma$$

Число дуг в графике практически не ограничено. Упорядоченная по τ_k таблица результатов строится на поле Δ лишь для τ_k, не превосходящих некоторой наперед заданной величины δ.

4. Упорядочение таблицы результатов

Как уже отмечалось, для больших графиков целесообразно выдавать не полную таблицу результатов, а лишь ее часть, наи-

более важную для практического использования. В окончательную таблицу результатов входят вершины, лежащие на критическом пути ($z_к = 0$), а также вершины с небольшим резервом времени $z_к \leq \delta$. Эта таблица упорядочивается по возрастанию $z_к$, а для $z_к = 0$ по $T_к$.

Построение окончательной таблицы результатов и ее упорядочение по $z_к$ осуществляется следующим алгоритмом, требующим всего двух просмотров полной таблицы результатов.

Первый просмотр. В ячейках $(a+s)$, $s = 0,1,\ldots,\delta$, подсчитывается число n_s вершин с резервом времени $z_к = s$. После этого определяются начальные адреса P_s размещения s-й группы с резервом времени $z_к = s$: P_0 задается, $P_s = P_{s-1} + n_{s-1}$, $s = 1,2,\ldots,\delta$. Адреса P_s размещаются в ячейках $(a+s)$, $s = 0,1,\ldots,\delta$.

Второй просмотр. Последовательно просматриваются все строки полной таблицы результатов

$$(T_к, k, z_к), \quad k = 1,2,\ldots,M.$$

Строки с $z_к > \delta$ пропускаются, а строки с $z_к = s \leq \delta$ заносятся в ячейки P_s. После каждого такого занесения адрес P_s, хранящийся в ячейке $(a+s)$, увеличивается на I.

Упорядочение по $T_к$ вершин, лежащих на критическом пути, связано с обработкой небольшой информации и выполняется обычным образом.