ИЗВЕСТИЯ
ТОМСКОГО ОРДЕНА ТРУДОВОГО КРАСНОГО ЗНАМЕНИ ПОЛИТЕХНИЧЕСКОГО ИНСТИТУТА имени С. М. КИРОВА

Том 154 1967

ОБ ОДНОМ АЛГОРИТМЕ АВТОМАТИЗАЦИИ РАСЧЕТОВ ПРОЦЕССОВ УПРАВЛЕНИЯ РАЗРАБОТКАМИ СЛОЖНЫХ СИСТЕМ

В. М. РАЗИН, Ю. Н. ЕФИМОВ, В. И. КИЗЕВ

(Представлена научным семинаром факультета автоматики и вычислительной техники)

В последнее время в литературе [1] все чаще затрагивается вопрос о внедрении принципиально новых методов комплексного оперативного планирования и управления разработками сложных систем. Сущность планирования по этим методам сводится к разработке сетевых моделей процессов, анализу этих моделей с помощью электронных цифровых вычислительных машин (ЭЦВМ) с последующим использованием их как инструмента управления на всех этапах выполнения проекта. Одной из разновидностей указанных выше методов является система, учитывающая только длительность разработки. В такой системе, известной за рубежом под названием „pert-time", обработка информации, представляемой сетевым графиком, сводится к нахождению „критического пути", концентрируя тем самым внимание руководителей на участках потенциальных затруднений в ходе работы. Для решения задач по нахождению „критического пути" предлагается использовать быстродействующие трехадресные ЭЦВМ с большим объемом оперативной памяти [2]. В настоящее время значительное распространение получают двухадресные ЭЦВМ с ограниченным объемом оперативной памяти. В связи с этим представляет интерес рассмотрение вопроса алгоритмизации и программирования задач по нахождению „критического пути" по времени применительно к ЭЦВМ упомянутого класса.

Анализ сетевого графика обычно сводится к следующей задаче. Имеется сетевой график без замкнутых контуров, содержащий M событий и N работ. Каждой работе R_{ij}, ведущей от событий i к событию j, соответствует число $l_{ij} \geqslant 0$, которое назовем длиной работы. Длина работы представляет собой не что иное, как ее продолжительность во времени. Путем в графике назовем последовательность работ от начального события до конечного, причем для этой последовательности конец каждой предыдущей работы совпадает с началом последующей. Под длиной пути будем понимать сумму длин всех работ, принадлежащих данному пути. Для такого графика требуется определить „критический путь", т. е. путь с наибольшей длиной из всех возможных путей, не имеющий резерва времени, и несколько „подкритических" путей, имеющих минимальный резерв времени.

В основу разработанного нами метода решения поставленной задачи положен принцип полного перебора всех возможных путей в сетевом графике.

Сетевой график задается списком работ и их длин, расположенных в любой последовательности, в виде

$$\kappa)\ ijl_{ij_n},$$

где κ — номер ячейки памяти, т. е. $\kappa = m, m+1, m+2, \ldots$

i, j_n — номера событий, определяющих начало и конец работы R_{ij_n},

l_{ij_n} — длина работы R_{ij_n},

$n = 1,2,3\ldots$ индексы работ, выходящих из одного и того же события i.

Обработка информации такого вида осуществляется в два этапа. На первом этапе производится упорядочение всех работ по коду их начал, на втором — обеспечивается прохождение всех возможных путей и выбор из них „критических“ и „подкритических“.

Упорядочение списка работ по коду их начал заключается в размещении информации о работе R_{ij_n} в ячейке памяти машины, адрес которой равен номеру событий начала работы i. Одновременно вырабатываются специальные признаки r и a, так что получается информация вида

$$i)\ r, a\ j_n,\ l_{ij_n},$$

где i — номер ячейки памяти,

j_n — номер конца работы, исходящей из события i,

r — величина, равная числу оставшихся работ, исходящих из события i; если нет разветвлений, $r = 0$,

a — величина, характеризующая местонахождение в памяти информации о другой работе, исходящей из этого же события i. Информация об этой работе помещается в ячейку с адресом $(a + c)$, где $c = \text{const}$. Удобно принять константу c равной наибольшему номеру события сетевого графика.

Упорядочение работ производится за один просмотр исходной информации. При просмотре из каждой строки исходного списка работ выделяется i, и информация об этой работе записывается по адресу i. Если же в эту ячейку уже была произведена запись, то в информации по адресу i величина r увеличивается на 1, формируется a, а запись прозводится по адресу $a + c$ и т. д.

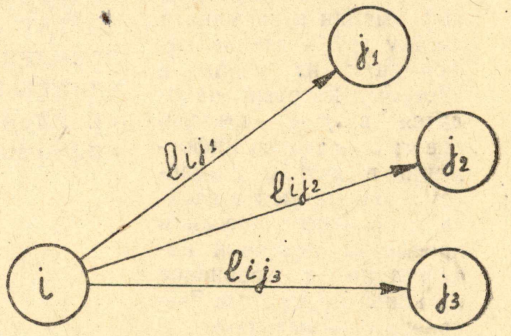

Рис. 1. Пример упорядочения информации.

Ниже приводится пример упорядочения исходной информации и блок-схема программы, реализующей алгоритм упорядочения (рис. 1 и 2).

В ячейке i размещается одна из работ, которая при просмотре исходной информации встречается первой. Пусть это будет работа R_{ij_1}, тогда упорядоченное расположение информации (рис. 1) будет следующим:

$$i)\ r_1,\ a_1,\ l_{ij_1}\ (r_1 = 3),$$

. . . . . .

$$c + a_1)\ r_2,\ a_2,\ l_{ij_2}\ (r_2 = 2),$$

$$c + a_2)\ r_3,\ a_3,\ l_{ij_3}\ (r_3 = 1,\ a_3 = 0).$$

Таким образом, признак r представляет собой число еще не просмотренных работ, выходящих из одного и того же события, увеличенное на единицу.

Признак a образуется следующим образом. Каждый раз, когда при упорядочении информации встречаются события, из которых выходит более одной работы из числа неупорядоченных, в специальный счетчик добавляется единица. За величину параметра a рассматриваемой работы принимается число, накопленное в счетчике к моменту просмотра работы, выходящей из события, являющегося началом непросмотренных работ.

Параметр a принимается равным нулю в двух случаях:

1) когда просматриваемая работа является единственной, выходящей из данного события;
2) когда просматривается работа, являющаяся последней из всех работ, выходящих из данного события.

Формирование признаков a и r иллюстрируется блок-схемой программы упорядочения исходной информации на рис. 2.

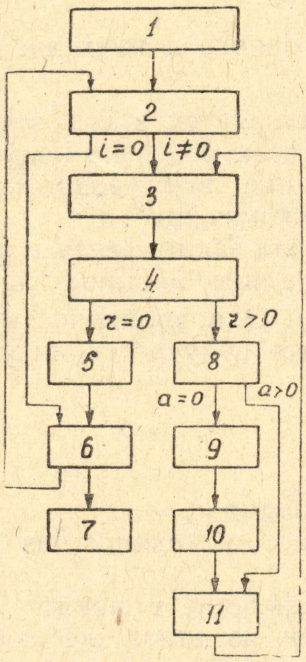

Рис. 2. Блок-схема упорядочения исходной информации. 1 — восстановление; 2 — выделение i из строки исходной информации и засылка в ячейку f; 3 — чтение информации из ячейки с адресом, который находится в f; 4 — выделение r; 5 — запись информации в ячейку с адресом, хранящимся в ячейке f; 6 — переадресания просмотра исходной информации и сравнение на конец просмотра; 7 — конец; 8 — выделение и анализ a; 9 — добавление единицы в счетчик; 10 — формирование величины a, равной содержимому счетчика; 11 — формирование адреса $(a+c)$ и засылка его в ячейку f.

После упорядочения списка работ легко осуществить выбор всех возможных путей на сетевом графике. Анализ сетевого графика производится по программе, блок-схема которой представлена на рис. 3.

Задавшись начальным событием, нужно прочесть информацию из ячейки памяти, имеющей номер начального события, выделить событие j, принять его за начальное событие и т. д., до тех пор, пока процесс не дойдет до конечного события. Выбранные пройденные работы последовательно записываются в z ячейках. Таким образом, получаем пройденный путь в виде следующей таблицы информации

$$r_n,\ a_n,\ j_n,\ l_n,$$
$$\cdots\cdots$$
$$r_к,\ a_к,\ j_к,\ l_к,$$
$$0,\ 0,\ 0,\ L_к,$$

где $n = 1, 2, 3 \ldots . к$.

$к$ — число работ, составляющих выбранный путь,

$L_к$ — длина выбранного пути.

При этом длина выбранного пути $L_к$ получается в результате суммирования всех длин работ данного пути. Информацию об одной работе, входящей в выбранный путь и записанной в одной ячейке памяти, назовем строкой пути. После выбора одного из путей в строках таблицы информации, начиная с последней, производится анализ признака r. При наличии в строке признака $r > 1$ (эту строку назовем граничной) формируется обращение к ячейке памяти с номером

$a+c$. Приняв событие j в этой ячейке за начальное, выбирается путь согласно алгоритму, описанному выше. Информация о пути заносится в ячейки z, начиная с граничной строки. Получается новый путь, таблица информации которого отличается от предыдущего только с граничной строки. Процесс повторяется до тех пор, пока во всех строках пути не окажется $r = 1$. При этом будут выбраны все возможные пути сетевого графика.

Благодаря предварительному упорядочению процесс выбора всех возможных путей протекает сравнительно быстро. Выбор всех возможных путей заканчивается запоминанием m длин „критических“ и „подкритических“ путей (m — любое целое число, меньшее p, где p — число всех возможных путей). При этом выбор и запоминание m наиболее длинных путей организуется посредством программирования следующим образом. На начальном этапе запоминается m разных длин путей, вычисленных первыми. Все следующие вычисленные длины путей сравниваются с минимальной из запомненных длин. Если сравниваемая длина будет больше и если среди запомненных нет величины ей равной, то она записывается в ячейку памяти вместо ранее запомненной минимальной длины. При вторичном просчете выводятся на печать таблицы информации для путей, длины которых совпадают с ранее запомненными.

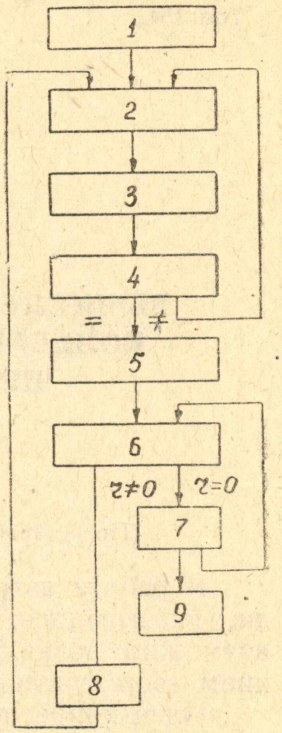

Рис. 3. Блок-схема основной программы. 1 — засылка начального события в ячейку f; 2 — запись ячейки с адресом в f в ячейку z; 3 — выделение j в f; 4 — сравнение на конец пути; 5 — нахождение L и сравнение с min; 6 — анализ r; 7 — сравнение на конец; 8— выделение a и формирование новых адресов; 9 — останов.

Отметим, что рассмотренный метод не заключает в себе возможности автоматического отыскания контуров.

При необходимости получить только один критический путь рекомендуется несколько изменить описанный алгоритм с целью повышения быстродействия. Выбор путей происходит, как и в первом случае, но при анализе признаков r в таблице информации для каждого события, имеющего несколько исходящих работ, оставляется только одна работа, по которой проходил бы „критический путь“, если это событие принять за начальное. Процесс заканчивается нахождением „критического пути“, причем вторичного просчета не требуется.

По описанному алгоритму была составлена программа и опробована на ЭЦВМ „Минск-1“. При объеме оперативной памяти машины в 1024 слова по программе может быть произведен анализ сетевого графика, имеющего до 800 работ и не более 512 событий ввиду ограничений за счет малой разрядной сетки машины. Принципиально можно производить анализ сетевого графика любого объема, предварительно разбив его соответствующим образом на участки.

ЛИТЕРАТУРА

1. Г. С. Поспелов, А. И. Тейман. Автоматизация процессов управления разработками больших систем или сложных комплексов. Известия АН СССР, Техническая кибернетика, № 4, 1963.

2. Вычислительные системы, изд. СО АН СССР, Новосибирск, 1964, вып. № 11.