К ВЫБОРУ АЛГОРИТМОВ НАХОЖДЕНИЯ МНОЖЕСТВА ФУНДАМЕНТАЛЬНЫХ ЦИКЛОВ И ГЛАВНЫХ СЕЧЕНИЙ ДЛЯ СПЕЦИАЛИЗИРОВАННОЙ ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ

А. Г. Евдокимов, Ю. Ю. Завизиступ, Г. Ф. Кривуля, Е. В. Трунов
Харьков

В предлагаемой работе рассмотрены алгоритмы нахождения множества фундаментальных циклов B^* и главных сечений Q^*, эквивалентных топологическим матрицам фундаментальных циклов B_* и глав-

ных сечений Q_α. Алгоритмы предназначены для реализации их в вычислительной системе решения задач потокораспределения на базе малой цифровой машины и специализированного устройства ввода, хранения и отображения топологической информации [1]. Особенностью системы является наличие встроенных подпрограмм для наиболее характерных этапов решения задачи и, в частности, этапа автоматизированного составления уравнений сети по ее топологии. Поэтому к алгоритмам предъявлены дополнительные требования: а) минимальный объем оперативной и программной памяти; б) возможность работы с разрядной сеткой ограниченной длины. Так как для конкретной задачи определение множеств B^* и Q^* производится однократно, то требование к быстродействию алгоритмов менее важно, чем к объему занимаемой памяти. Существующие алгоритмы, в зависимости от вида исходной информации о графе сети, могут быть классифицированы следующим образом.

1. Матричные алгоритмы

Исходной информацией является матрица инциденций A. Путем ее последовательных преобразований могут быть получены подматрицы B_α и Q_α, используя промежуточную матрицу путей [2].

Необходимо также упомянуть алгоритм, получивший название метода Жордана-Гаусса и основанный на выделении в матрице инциденций треугольной матрицы, соответствующей некоторому дереву графа с последующим упорядочением и приведением этой матрицы к матрице главных сечений, содержащих единичную матрицу ветвей дерева [3, 4].

Процедуры [2] требуют хранения результатов промежуточных преобразований, а также операций транспонирования и обращения матриц. Эти операции весьма трудоемки для матриц высоких порядков, что характерно для инженерных сетей, насыщенность топологических матриц которых редко достигает 15—20%. Таким образом, в случае матричных методов [2, 3, 4] память ЦВМ используется крайне неэффективно. При ограничениях объема ОЗУ и разрядной сетки машины эти методы не могут быть реализованы для сетей больших и средних объемов.

2. Алгоритмы, основанные на преобразовании множества узловых подмножеств

Информация о графе сети может быть представлена в виде множества узловых подмножеств $A^*_i \sim A$, каждое из которых $A^*_i \subset A^*$ представляет собой совокупность номеров ветвей, инцидентных i-му узлу, знак элементов подмножеств определяется ориентацией ветвей по отношению к i-му узлу. При данном способе представления информации исключается хранение и преобразование нулевых элементов массива. В работе [5] описаны процедуры, эквивалентные методу Жордана-Гаусса, позволяющие получить множество главных сечений, опирающееся на выбранное дерево графа. Недостатком метода является переменная длина подмножеств, изменяемая в процессе реализации алгоритма и зависящая от выбранного дерева.

3. Алгоритмы, основанные на топологическом преобразовании информации о графе сети

Данная группа алгоритмов позволяет находить множества фундаментальных циклов и главных сечений путем преобразования информации, эквивалентного операциям над графом сети.

В работе [7] описан алгоритм, позволяющий получить множество фундаментальных циклов путем анализа и устранения тупиковых путей графа сети. Анализ ведется из произвольной вершины в произвольном направлении. Если в процессе поиска будет найдена тупиковая ветвь, производится возврат на один шаг и т. д., пока не будет найдена последовательность ветвей, образующих контур (цикл).

В работе [7] приведен алгоритм, позволяющий получить перечисление вершин, входящих в цикл. Анализ рассмотренных алгоритмов показывает, что последние не пригодны для реализации на ЦВМ при ограничениях оперативной и программной памяти, ввиду либо больших затрат памяти машины и неэффективного ее использования [1—5], либо ввиду сложности вычислительных процедур при реализации алгоритма [6—7].

Рассмотрим способ задания информации о графе сети в виде так называемого множества узловых пар $(A^*)'$, где $(A^*)'$ — транспонированное множество узловых подмножеств. Такая форма представления информации для малых цифровых машин имеет следующие преимущества: объем занимаемой памяти для представления исходной информации о графе является близким к минимальному (матрица инциденций для таких графов слабо насыщена); легко осуществляется коррекция исходной информации путем введения или исключения соответствующей узловой пары; в структуре узловой пары содержится информация об ориентации ветвей, если принять соответствующий порядок записи ее узлов; длина машинного слова (количество элементов подмножества в слове) является постоянной, характерно, что указанная форма представления исходных данных о сети соответствует так называемой таблице соединений, составляемой на этапе подготовки информации о сети и является наиболее привычной и удобной для оператора формой кодирования сети.

Ниже будут рассмотрены алгоритмы непосредственного преобразования $(A^*)'$ в множество фундаментальных циклов B^* и множество главных сечений Q^*.

Алгоритм 1. Формирования множества фундаментальных циклов

Излагаемый алгоритм, основанный на преобразовании структуры частичных графов $G'_k \subset G(V, E_{v-1})(G'_1 \subset G\ (V, E)$, где V, E — непустые множества вершин и дуг соответственно, с элементами $i, j \in V$ и $e_{ij} \in E$), которые состоят из выбранного дерева и хорды, образующей фундаментальный цикл G_k, описан в [8].

Алгоритм заключается в целенаправленном устранении ветвей графа G'_k, содержащих висячие вершины, путем анализа степени этих вершин. Ниже будут приведены особенности его реализации.

В каждом шаге алгоритма производится поиск и исключение из множества B'_k узловых пар, $B'_k \sim G'_k$, таких подмножеств $(i, j) \in B'_k$, для которых card $V_i = 1$, либо card $V_i = 1$, $i \in V_i$, $j \in V_j$, где $V_i \subset B'_k$, $V_j \subset B'_k$ — подмножества вершин с одинаковыми индексами.

Легко показать, что за конечное число шагов исключения оставшееся множество будет состоять из подмножеств $(i, j) \in B'_k$, для которых card $V_i =$ card $V_j = 2$, т. е. будет получено подмножество узловых пар, эквивалентное k-му подмножеству множества фундаментальных циклов B^*_k.

Знаки элементов B_k^* могут быть определены с учетом ориентации хорды $(i, j)k$, образующей направление обхода контура, исходя из следующего соотношения:

$$\text{sign}\ (p, q) = \alpha\ \text{sign}\ (i, j), \tag{1}$$

где

$$\alpha = 1 \text{ при } j = p \text{ или } i = q, \tag{2}$$

$$\alpha = -1 \text{ при } j = q \text{ или } i = p, \tag{3}$$

где i, j — индексы элементов подмножества $(i, j) \subset B_k$, знак которых определен по отношению к хорде $(i, j)_k$, $\text{sign}(i, j)_k = 1$; p, q — индексы элементов подмножества $(p, q) \in B_k$, знак которого определяется, причем $(i, j) \cap (p, q) \neq \varnothing$. Совмещая процедуру определения знака подмножества (p, q) с определением его номера в массиве B_k^*, получим множество элементов B_k^*, состоящее из номеров ветвей дерева и хорды, входящих в k-й фундаментальный цикл с соответствующими знаками.

Процесс формирования множества B^* состоит из μ идентичных циклов, где $\mu = e - v + 1$ — цикломатическое число графа.

Описание алгоритма 1

Исходной информацией для реализации алгоритма является массив из e-ячеек. В каждой ячейке хранится начальный (на первом месте) и конечные узлы дуг графа, из которых $(v-1)$-й ячейке, относящимся к ветвям выбранного дерева, присвоен знак (признак) «—», записанный в результате реализации подпрограммы выбора дерева сети. Исходный массив хранится во внешнем запоминающем устройстве, предназначенном для хранения топологической информации о сети, при отсутствии ограничений по объему занимаемой памяти, массив может храниться непосредственно в ОЗУ машины.

Процесс образования подмножества B_k^* состоит из следующих шагов:

Шаг 1. Формирование массива в ОЗУ, состоящего из k-й по порядку узловой пары без пометки и $v-1$ узловой пары со знаками «—», из исходного массива.

Шаг 2. Последовательный выбор ячеек в образованном массиве, выделение начального и конечного узлов, сравнение их с начальными и конечными узлами в остальных ячейках массива. В случае, если какой-либо из узлов не содержится в оставшемся массиве, ячейке, хранящей соответствующую узловую пару присваивается знак «+». Указанная ячейка из дальнейшего анализа исключается. В случае, если найден хотя бы один идентичный узел (начальный или конечный), сравнение по данному узлу прекращается. Процесс заканчивается после анализа $v-1$ ячейки массива.

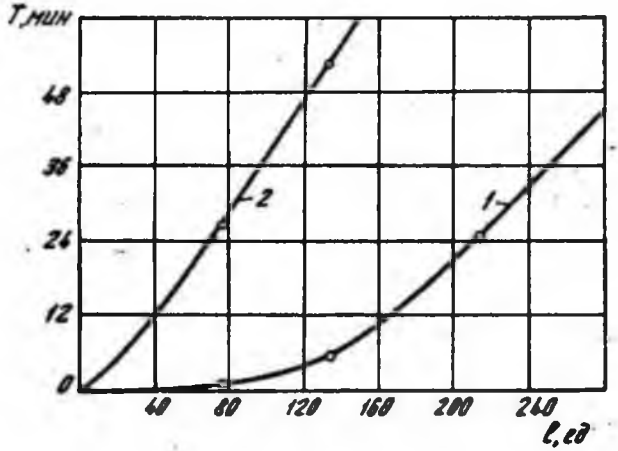
Шаг 3. Формирование знаков элементов B_k^*. Производится путем поочередного сравнения номеров начального узла k-й хорды (первой ячейки в массиве) с начальными и конечными узлами остальных ячеек массива согласно (1÷3), для дальнейшего анализа в качестве опорного принимаем узел ветви (для которой знак определен), противоположный смежному с предыдущей ветвью. Выдача номера узловой пары (номера ячейки) с соответствующим знаком во внешнее запоминающее устройство или на печать.

Алгоритм был реализован на машине «Проминь» — программа содержала 98 команд, и на ЦВМ «Урал 4». На рис. 1 и табл. 1 (кри-

вая 2) приведены результаты нахождения множеств B^* для различных сетей. Объем памяти ОЗУ, необходимый для реализации алгоритма составлял v ячеек для формирования массива B_k и e ячеек для исходной информации.

Таблица 1

Число ребер, ℓ		9	76	133	213
Число вершин, V		6	49	79	131
Программа на языке «Алгол» для машины с плавающей запятой	Число команд программы	256			
	Время счета, сек	2	70	345	1520
Программа на языке «Алгол» для машины с фиксированной запятой	Число команд программы	208			
	Время счета, сек	3	1620	3180	5400



Для оценки производительности алгоритма была получена зависимость, удовлетворительно согласующаяся с экспериментом:

$$T_1 \approx \sum_{k=1}^{\mu} \left\{ c_1 v^2 + 2c_2 \sum_{i=v}^{v-m_k} (i-1) + c_2 \sum_{i=v}^{m_k} i(i-1) + c_1 n + 2c_2 (m_k - 1) \right\}, \quad (4)$$

где T_1 — время счета; μ — цикломатическое число сети; m_k — среднее количество ветвей, входящих в k-й фундаментальный цикл, при расчетах было принято $m_k = \frac{1}{2} v$; c_1 — константа для конкретной машины, характеризующая время чтения и анализа знака ячейки; c_2 — константа, выражающая затраты машинного времени для выделения узлов из узловой пары и сравнения номеров двух узлов.

В выражении (4) $c_1 v^2$ отражает время обращений к ОЗУ с анализом знака ячейки.

$2c_2 \sum_{i=v}^{v-m_k} (i-1)$ — время анализа висячих узлов;

$$c_2 \sum_{i=v}^{m_k} i(i-1)$$

— время анализа узлов, содержащихся в узловых парах, входящих в подмножество B_k; получено в результате приближенного решения уравнения Бернулли относительно числа опытов, необходимых для поиска заданного узла в массиве из v узловых пар, расположенных случайным образом, при известной априорной вероятности нахождения этого узла в массиве и заданной апостериорной вероятности исхода опытов.

$c_1 n + 2c_2 (m_k - 1)$ — время определения знака и номера элемента в упорядоченном массиве узловых пар, где узловые пары, вошедшие в фундаментальный цикл, имеют пометки.

Алгоритм 2. Определения множества главных сечений

В [8] описан алгоритм нахождения Q^*, основанный на поиске множества хорд, узлы которых инцидентны частичным подграфам G^+ и G^-, образованных в результате удаления из дерева графа $G_1(V_1, E_{v-1})$, ветви $(e_{ij})_k$, образующей k-е сечение. Предлагаемый алгоритм приводит к этой же цели, но обладает меньшей избыточностью вычислительного процесса.

Алгоритм заключается в нахождении подмножества хорд, инцидентных частичному дереву графа G^+, содержащему начальный узел удаляемой ветви дерева $(e_{ij})\xi$, где ξ — индекс соответствующего сечения ($\xi = 1 \div v-1$). Указанное построение справедливо также для случая поддерева, содержащего конечный узел ветви $(e_{ij})\xi$.

Рассмотрим частичный граф $G_1^+(V_1, E_1)$, образованный из условия:

$$G_1^+(V_1, E_1) = G_1(V_1, E_{v-1}) \setminus \{G_1^-(V_2, E_2) \cup G_\xi \setminus [(i,j)\xi\ (e_{ij})\xi]\},$$

здесь же

$$i \in V_1,\ j \in V_2;\ V_1 \cup V_2 = V,\ V_1 \cap V_2 = \varnothing\ E_1,\ E_2 \subset E_{v-1}.$$

Образуем подмножество дуг $Q_\xi^* = (e_{ij})\xi \cup \left\{ \bigcup_{k=1}^{k} (e_{k,l})_k \right\}$; $Q_\xi^* \subset Q^*$.

при условии

$$(k \in V_1) \wedge (l \in V_1), \qquad (5)$$

$$(l \in V_1) \wedge (k \in V_1). \qquad (6)$$

Знаки элементов подмножества Q_ξ^* определим следующим образом:

$$\operatorname{sign}(e_{ij})\xi = +; \quad \operatorname{sign} l_{kl} = \text{„+“ при условии (5)}$$

$$\operatorname{sign} l_{kl} = \text{„−“ при условии (6)}$$

Полученное подмножество Q_ξ^* представляет собой ξ-е подмножество множества главных сечений Q^*, так как элементами этого подмножества являются хорды, связывающие компоненты графа G_1^+ и G_1^-, образованные в результате удаления ξ — ветви дерева, включая эту ветвь дерева.

Описанное выше преобразование графа $G(V, E_{v-1})$ может быть заменено эквивалентным преобразованием множества узловых пар $(A^*)'$. Преобразование состоит из следующих основных этапов:

1. Выделение в множестве $(A^*)'$ подмножества $(i, j)\xi$, (индекс ξ имеет узловые подмножества, относящиеся к ветвям выбранного дерева).

2. Построение множества: $H_\xi^+ \sim V_1$, $i \in (i,j)_\xi^+$, $i \in H_\xi^+$, $j \in H_\xi$.

3. Построение множества $Q_\xi^* = \bigcup_{k=1}^{p} (i,j)_k$

При условии

$$(i \in H_\xi^*) \wedge (j \in H_\xi) \tag{7}$$

$$(j \in H_\xi) \wedge (i \in H_\xi) \tag{8}$$

$i, j \in (i, j)_k$ k — индекс узловых подмножеств, относящихся к хордам.

При этом элементы $(i, j)_k$ подмножества Q_ξ^* имеют знак «+» в случае (7) и «—» в случае (8). Индекс k подмножества совпадает с номером этого элемента в множестве $(A^*)'$.

Описание алгоритма 2

Исходной информацией является массив узловых пар с пометками для ветвей дерева и хорд, как и в предыдущем алгоритме. Процесс преобразования массива узловых пар в массив, содержащий ξ_e подмножество множества главных сечений Q^* состоит из следующих шагов.

Шаг 1. Выбор из исходного массива ξ — узловой пары. Выделение начального узла.

Шаг 2. Образование массива узловых пар, имеющих пометки «—» в исходном массиве и имеющих общие узлы с начальным узлом, найденным в предыдущем шаге, и образуемым массивом, включающим в себя и этот узел. Осуществляется путем выделения начального и конечного узла текущей ветви из исходного массива и последовательного сравнения их с начальными и конечными узлами каждой ячейки массива; если обнаружен хотя бы один идентичный узел, сравнения для данной ячейки прекращается и последняя записывается в образуемый массив.

Шаг 3. Образование массива узловых пар, удовлетворяющих условиям (7), (8), путем последовательного сравнения начального и конечного узлов узловых пар из массива, образованного в предыдущем шаге, с начальными и конечными узлами в ячейках исходного массива, имеющих пометки «+». В случае если встречается идентичный узел, сравнение по нему прекращается и переходим к анализу следующего узла. Формирование знаков и номеров элементов подмножеств $(i, j)_k$, выдача их номера и номера ветви, выбранной в шаге 1 во внешнее ЗУ или на печать. Последняя операция совмещается с анализом узловых пар с пометкой «—» в принадлежности к массиву, образованному во 2-м шаге, при соблюдении условий (7), (8).

Программа для указанного алгоритма реализована на ЦВМ «Урал-4». На рис. 1 и табл. 1 показаны результаты просчета конкретных инженерных сетей (кривая 1). При этом затраты оперативной памяти составляют v — ячеек, для образования массива во 2-ом шаге алгоритма и массива e — ячеек для исходной информации.

Затраты машинного времени для этого алгоритма могут быть приближенно оценены следующим образом:

$$T_2 \leqslant 2c_2 \sum_{\xi=1}^{v-1} \left| \sum_{i=1}^{m_\xi} (2i-1) + \mu m_\xi + c_1 (v-2)^2 \right| \qquad (9)$$

T_2 — время счета.

$m\xi$ — среднее количество узловых пар в массиве для 2-го шага алгоритма: $m_\xi \approx \left(\frac{1}{2} \div \frac{1}{3}\right) v$.

Остальные величины аналогичны предыдущему случаю.

ВЫВОДЫ

В работе проанализирован ряд алгоритмов поиска фундаментальных циклов и главных сечений для различных способов представления информации о графе сети. В результате анализа показано, что предложенные алгоритмы 1, 2, оперирующие с исходной информацией в виде множества узловых пар, являются наиболее экономичными по объему занимаемой памяти. Особенностью предложенных алгоритмов является то, что процесс нахождения множеств B^* и Q^* может быть разбит на ряд независимых циклов, требующих небольших затрат памяти ЦВМ. Анализ результатов расчета сетей, приведенных в таблице 1, и соответствующих зависимостей (4, 9) показывает, что алгоритм 1 применим для поиска множества фундаментальных циклов в сетях, содержащих до 100 ветвей. Алгоритм 2 поиска множества главных сечений является более быстродействующим ввиду большей степени упорядоченности для него исходного массива, а также совмещения этапов нахождения элементов множества и анализа их знака. Алгоритм может быть эффективно использован для сетей, содержащих порядка нескольких сот ветвей. Алгоритмы требуют для своей реализации небольшого объема программной памяти (не более 20 ячеек). Проведенные исследования показывают возможность эффективного использования предложенных алгоритмов в вычислительной системе.

ЛИТЕРАТУРА

1. А. Г. Евдокимов, Ю. Ю. Завизиступ, Г. Ф. Кривуля. Вычислительная система для решения задач потокораспределения. (В настоящем сборнике).

2. М. Л. Брэнин (F. H. Branin). Методы анализа цепей с помощью вычислительной машины. 1967, № 11.

3. А. А. Волков, А. Г. Евдокимов, Р. Т. Волколупова. Топологический анализ шахтных вентиляционных сетей. «Известия вузов, Горный журнал», 1967, № 1.

4. A. G. Yevdokimov. A theory of the Solution of steady state network problems with special reference to mine ventilation networks. International Jornal of numerical methods in engineering, vol. 1, 279—299, (1969).

5. А. А. Волков. Линейные операторы сильно связанных графов и их определение и с помощью ЦВМ. Материалы координационного совещания секции технической кибернетики, 1966.

6. Є. Цой и С. Цхай. Электронно-вычислительная техника в вентиляционной службе шахт. Алма-Ата, 1966.

7. И. Б. Задыхайло. Алгоритм выделения всех вершин ориентированного графа, принадлежащих контурам. «Цифровая вычислительная техника и программирование», вып. 2, «Сов. радио», 1967.

8. А. Г. Евдокимов, Ю. Ю. Завизиступ, Г. Ф. Кривуля. Определение фундаментальных циклов и главных сечений в сложной сети. «Материалы 4-ой республ. конференции молодых исследователей по системотехнике», Киев, 1969.