ИССЛЕДОВАНИЕ ЧИСЛЕННЫХ АЛГОРИТМОВ ОПРЕДЕЛЕНИЯ ДЕРЕВА МИНИМАЛЬНОЙ ДЛИНЫ

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

Харьков

В настоящее время придается большое значение решению на цифровых вычислительных машинах (ЦВМ) задач анализа и синтеза сложных сетевых систем. Возможности универсальных ЦВМ, ввиду неприспособленности их структуры для решения данного класса задач, используются неполностью, поэтому возникает необходимость в разработке специализированной вычислительной системы, структура которой максимально приспособлена для расчета сложных сетевых систем. Такая вычисли-

тельная система может быть построена на базе малых ЦВМ, что позволит при небольшой стоимости комплекса рассчитывать относительно сложные сети (сотни ветвей) [1]. При этом возникает необходимость в математическом обеспечении специализированной системы, т. е. в разработке алгоритмов преобразования топологической информации о сети и итерационного расчета потокораспределения, требующих для своей реализации наименьшее количество машинных операций и ячеек оперативного запоминающего устройства (ОЗУ). Особое значение имеют вопросы, связанные с улучшением сходимости итерационного расчета потокораспределения в сетевой системе.

При использовании для расчета сети метода контурных потоков сходимость итерационного процесса может быть существенно повышена за счет выбора системы фундаментальных циклов, опирающихся на дерево минимальной длины графа сети.

Задача выбора дерева минимальной длины заключается в следующем: среди множества деревьев связного конечного графа $G = G(V, E)$ определить дерево $G_1(V, E_{v-1})$, на котором достигается $\min \sum l_{ij}$, где V, E — непустые множества вершин и дуг соответственно; каждому элементу $e_{ij} \in E$ поставлена в соответствие упорядоченная узловая пара i, j и вес (длина) ребра l_{ij}, причем i — начало, j — конец дуги e_{ij}.

Указанная задача возникает также в следующих практически важных случаях расчета сетевых систем:

1. Определении кратчайшей сети, связывающей конечное множество пунктов. Задача характерна для начального этапа проектирования энергетических, информационных, транспортных, коммунальных и других сетей.

2. Выборе базисных последовательных и параллельных переменных в инженерной сети с целью получения минимальной погрешности при измерении их величин.

Известен ряд работ, посвященных нахождению кратчайшей связывающей сети. Изложенные в данных работах методы можно разделить на графические и численные. В работах [2, 3] приведено несколько процедур графического метода, суть которых сводится к следующим двум случаям:

а) последовательный выбор $(v - 1)$ дуг, где $v = \operatorname{card} V$, по возрастанию их весов и одновременным отбрасыванием тех дуг, которые образуют циклы с уже выбранными дугами;

б) выбор последовательности дуг в соответствии с убыванием их весов и отбрасыванием тех из них, которые нарушают связность графа; причем число таких удаленных дуг равно цикломатическому числу графа $\mu = e - v + 1$, где $e = \operatorname{card} E$.

Процедуры построения дерева минимальной длины [2] основаны на двух принципах:

а) всякая вершина, еще не вошедшая в процессе решения задачи в какое-либо кратчайшее поддерево, соединяется с другой вершиной кратчайшей дугой;

б) всякое кратчайшее поддерево соединяется с ближайшей вершиной, инцидентной вершинам, принадлежащим этому поддереву.

Р. К. Прим [2] также описывает численный алгоритм, основанный на последовательном преобразовании матрицы расстояний между узлами графа размерности $v \times v$, результатом которого является набор $(v - 1)$ узловых пар, соответствующий дереву минимальной длины. Другие известные численные методы оперируют, как правило, с исходной информацией о сети в виде матрицы инциденций узлов и дуг размерности $(v \times e)$.

Таким образом, известные алгоритмы требуют для хранения информации, представленной в матричной форме, большие объемы запоминающих устройств ЦВМ. Уменьшение объема ЗУ ЦВМ может быть достигнуто в случае представления исходной информации о сети в виде множества узловых подмножеств [4], так как при этом исключается необходимость в хранении и преобразовании нулевых элементов топологических матриц, число которых резко увеличивается с ростом размеров и сложности сети.

Представление исходной информации о графе сети в виде семейства подмножеств дуг элементарных узлов графа [4] имеет существенный недостаток, ввиду переменной длины узловых подмножеств в зависимости от степени вершины графа, что усложняет процесс ввода и обработки данного массива информации на ЦВМ с постоянной длиной слова.

Наиболее удобным для реализации на ЦВМ является задание исходных данных о сетевой системе в виде множества e упорядоченных узловых пар и поставленных им в соответствие весов

$$A = \{1\,(i^{(1)},\ j^{(1)}),\ l_{ij}^{(1)};\ 2\,(i^{(2)},\ j^{(2)}),\ l_{ij}^{(2)};\ \ldots e\,(i^{(e)},\ j^{(e)}),\ l_{ij}^{(e)}\}.$$

На рис. 1 приведен ориентированный связный граф для одной из простейших сетевых систем. Соответствующий каждой дуге вес приведен в круглых скобках после номера дуги. Исходная информация о данной сетевой системе имеет следующий вид:

$A = \{1\,(1,\ 2),\ 4$
$2\,(1,\ 6),\ 8$
$3\,(3,\ 6),\ 17$
$4\,(2,\ 3),\ 10$
$5\,(6,\ 5),\ 1$
$6\,(2,\ 5),\ 7$
$7\,(5,\ 4),\ 13$
$8\,(3,\ 4),\ 12$
$9\,(4,\ 1),\ 15\}$

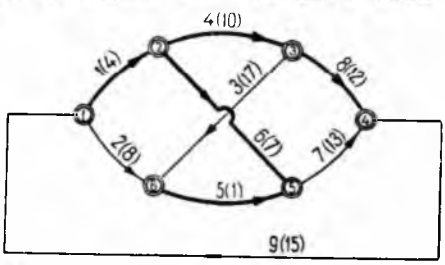


Рис. 1.

Рассмотрим алгоритмы выбора дерева минимальной длины при таком представлении исходной информации о сети. Алгоритмы отличаются объемом требуемой машинной памяти, количеством элементарных операций ЦВМ и скоростью выполнения.

Алгоритм 1. Реализуется первая процедура Краскала [3]. Узловые пары множества A располагаются в порядке возрастания длин соответствующих им дуг, т. е. $l_{ij}^{(1)} \leqslant l_{ij}^{(2)} \leqslant \ldots \leqslant l_{ij}^{(3)}$. Определяется и фиксируется ведущий элемент первой пары $i^{(1)}$, а каждая из остальных узловых пар, содержащая этот элемент, преобразуется путем нахождения операции «ориентированная симметрическая разность» между данной парой и первой:

$$(i^{(1)},\ j^{(1)}) \dot{+} (i^{(k)},\ j^{(k)}) = [(i^{(1)},\ j^{(1)}) \cup (i^{(k)},\ j^{(k)})] \setminus [(i^{(1)},\ j^{(1)}) \cap i^{(k)},\ j^{(k)})].$$

Затем процесс повторяется для следующего непустого подмножества: определяется его ведущий элемент и подмножества, содержащие этот элемент, преобразуются в результате операции «ориентированная симметрическая разность» между этими подмножествами и вторым непустым. После окончания $(v-1)$-го шага процесса множество A будет преобразовано к множеству, содержащему $(v-1)$-непустое, а дуги, соответствующие этим подмножествам, образуют дерево минимальной длины.

Подмножества, соответствующие дугам, которые не могут быть ветвями дерева минимальной длины, автоматически в процессе реализации

алгоритма преобразуются в пустые множества, а число элементов любого преобразованного множества всегда равно 2.

На примере графа, приведенного на рис. 1, продемонстрируем процесс нахождения дерева минимальной длины на основе алгоритма 1.

$A = \{5\ (6,\ 5)$ $A^{(1)} = \{5\ (|\underline{6}|,\ 5)$ $A^{(2)} = \{5\ (|\overline{\underline{6}}|,\ 5)$
$1\ (1,\ 2)$ $1\ (1,\ 2)$ $1\ (|\overline{\underline{1}}|,\ 2)$
$6\ (2,\ 5)$ $6\ (2,\ 5)$ $6\ (2,\ 5)$
$2\ (1,\ 6)$ $2\ (1,\ 5)$ $2\ (5,\ 2)$
$4\ (2,\ 3)$ $4\ (2,\ 3)$ $4\ (2,\ 3)$
$8\ (3,\ 4)$ $8\ (3,\ 4)$ $8\ (3,\ 4)$
$7\ (5,\ 4)$ $7\ (5,\ 4)$ $7\ (5,\ 4)$
$9\ (4,\ 1)$ $9\ (4,\ 1)$ $9\ (4,\ 2)$
$3\ (3,\ 6)\}$ $3\ (3,\ 5)\}$ $3\ (3,\ 5)\}$

$A^{(3)} = \{5\ (|\overline{\underline{6}}|,\ 5)$ $A^{(4)} = \{5\ (|\overline{\underline{6}}|,\ 5)$ $A^{(5)} = \{5\ (|\overline{\underline{6}}|,\ 5)$
$1\ (|\overline{\underline{1}}|,\ 5)$ $1\ (|\overline{\underline{1}}|,\ 5)$ $1\ (|\underline{1}|,\ 5)$
$6\ (|\overline{\underline{2}}|,\ 5)$ $6\ (|\overline{\underline{2}}|,\ 5)$ $6\ (|\overline{\underline{2}}|,\ 5)$
$4\ (3,\ 5)$ $4\ (|\overline{\underline{3}}|,\ 5)$ $4\ (|\overline{\underline{3}}|,\ 5)$
$8\ (3,\ 4)$ $8\ (4,\ 5)$ $8\ (|\overline{\underline{4}}|,\ 5)\}$
$7\ (5,\ 4)$ $7\ (5,\ 4)$
$9\ (4,\ 5)$ $9\ (4,\ 5)\}$
$3\ (3,\ 5)\}$

Алгоритм 1 требует для своей реализации хранения массива узловых пар в ОЗУ, причем исходная информация занимает $2e$ ячеек ОЗУ (e ячеек для узловых пар и e ячеек для весов). Полученный в результате упорядочения узловых пар по возрастанию весов вектор перекодирования также занимает e ячеек ОЗУ.

В специализированной вычислительной системе хранение исходного массива данных о сетевой системе целесообразно осуществлять во внешнем запоминающем устройстве (выполненного, в частности, на базе полупостоянного ЗУ) с возможностью пометки каждого цифрового кода оперативным признаком. При этом количество ячеек ОЗУ, необходимых для реализации алгоритма выбора дерева, должно быть минимальным.

Рассмотрим алгоритмы, разработанные с учетом данных требований. Алгоритмы основаны на следующих предпосылках, аналогичных Р. К. Приму [2], но взятых с определенными ограничениями:

1) образуется поддерево, состоящее из дуги кратчайшей длины;

2) отыскивается кратчайшая дуга, соединяющая любую вершину, смежную вершине выбранного кратчайшего поддерева.

Первый принцип используется только один раз на первом шаге алгоритмов для получения изолированного поддерева T_1, которое затем расширяется путем последовательного применения второго принципа на последующих шагах алгоритмов.

Алгоритм 2. Исходная информация о сети занимает $2e$ ячеек памяти. Информация, полученная в результате реализации алгоритма, занимает $e + v - 1$ ячеек (e — компонентный вектор перекодирования и $v - 1$ узловых пар ветвей дерева).

Сущность алгоритма *2* заключается в следующем. Предварительно производится упорядочение узловых пар по возрастанию весов (анало-

гично алгоритму *1*). Затем образуется поддерево $T_1 = (\{i^{(1)}, j^{(1)}\}, \{e_{ij}^{(1)}\})$, содержащее дугу кратчайшей длины. На последующих шагах алгоритма анализируются узловые пары в соответствии с возрастанием их весов и дуга $e_{ij}^{(k)}$, которой соответствует наименьший вес $\min l_{ij}^{(k)}$, при одном из условий

$$[i^{(k)} \in V(T_{k-1})] \wedge [j^{(k)} \in (V(G) \setminus V(T_{k-1})]; \quad (1)$$

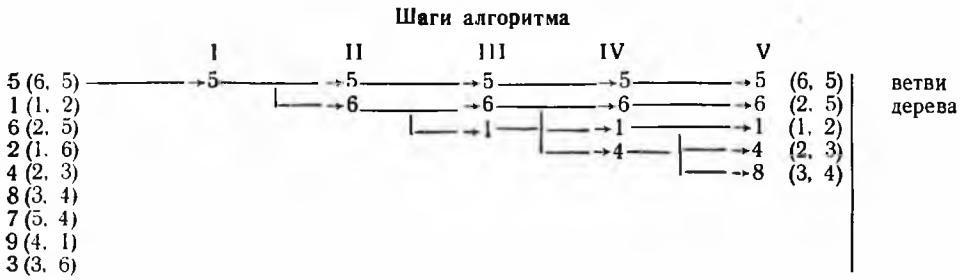
$$[j^{(k)} \in V(T_{k-1})] \wedge [i^{(k)} \in (V(G) \setminus V(T_{k-1})], \quad (2)$$

становится ветвью дерева

$$T_k = T_{k-1}(V(T_{k-1}),\ E(T_{k-1})) \cup G_k(\{i^{(k)}, j^{(k)}\}, \{e_{ij}^{(k)}\}).$$

Дальнейший ход алгоритма аналогичен до получения $(v-1)$ ветвей дерева.

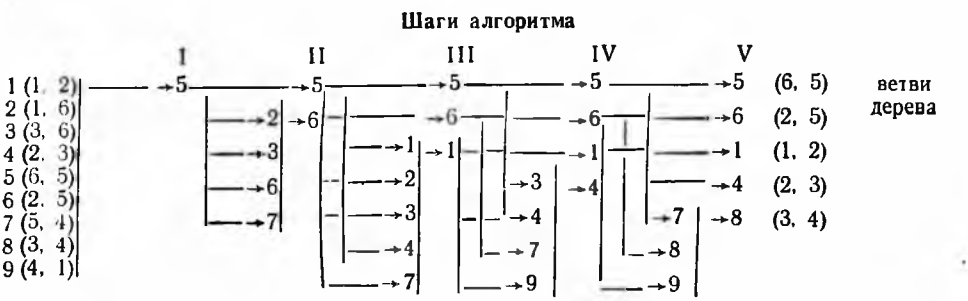
Выполнение алгоритма *2* для графа сети, изображенного на рис. 1, можно представить в следующей последовательности:



Алгоритм 3. При выполнении данного алгоритма не требуется располагать узловые пары в порядке возрастания их весов, т. е. полученный в результате реализации алгоритма *3* массив данных занимает $(v-1)$ ячеек памяти. Для исходной информации требуется $2e$ ячеек.

На первом шаге алгоритма поддерево T_1, содержащее дугу кратчайшей длины, образуется в результате сравнения весов всех дуг. Затем среди множества дуг, удовлетворяющих одному из условий (1) или (2), выбирается дуга с наименьшим весом путем последовательного сравнения весов дуг из данного множества. Последующие шаги аналогичны до образования массива ветвей дерева.

Последовательность выполнения данного алгоритма следующая:



Алгоритм 4. Данный алгоритм подобен предыдущему. Отличие заключается в том, что в процессе реализации алгоритма ветви дерева помечаются в исходном массиве узловых пар оперативными признаками. Количество используемых ячеек ОЗУ в данном случае равно числу рабочих ячеек, необходимых для организации программы.

Число ребер, e		33	60	76	133	213	356
Число вершин, v		22	36	49	79	131	142
Алгоритм 1	Время расположения весов (длин) по возрастанию, *сек*	1,5	6	9	27	65	170
	Время счета, *сек*	1,5	5	7,5	22	43	80
	Число команд программы	323					
	Объем памяти	554	743	855	1254	1814	2815
Алгоритм 2	Время расположения весов (длин) по возрастанию, *сек*	1,5	6	9	27	65	170
	Время счета, *сек*	1,5	5	14	48	195	645
	Число команд программы	307					
	Объем памяти	560	763	888	1317	1929	2941
Алгоритм 3	Время счета, *сек*	—	—	145	790	2450	—
	Число команд программы	254					
	Объем памяти	449	599	702	1020	1496	2101
Алгоритм 4	Время счета, *сек*	—	—	190	1030	2720	—
	Число команд программы	303					
	Объем памяти	457	579	656	914	1286	1869

Описанные выше алгоритмы определения дерева минимальной длины реализованы на ЦВМ для сетевых систем различного объема, имеющих

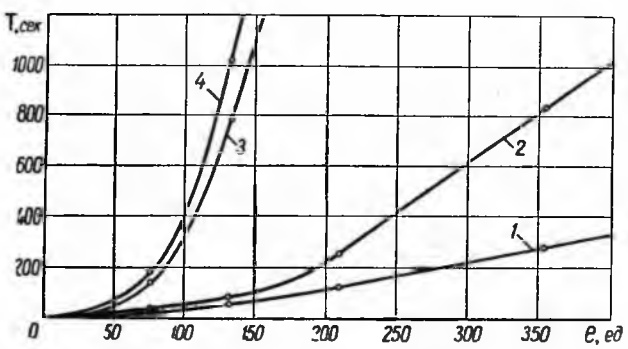


Рис. 2.

примерно равную плотность ребер сети $\rho = \frac{2e}{v(v-1)}$. В таблице приведены результаты экспериментальной проверки данных алгоритмов на

ЦВМ «Урал-4», причем для каждого алгоритма указано необходимое число неполных ячеек ОЗУ. На рис. 2 представлены графические зависимости требуемого машинного времени T от количества дуг e сетевой системы для рассмотренных алгоритмов.

Полученные экспериментальные данные позволяют сделать вывод о целесообразности использования разработанных алгоритмов в каждом конкретном случае расчета сетевых систем. При расчете сложных сетевых систем (сотни ветвей) на универсальных ЦВМ, имеющих достаточный объем памяти, целесообразно использовать алгоритмы 1,2.

При хранении исходной информации о такой сложной сети во внешнем запоминающем устройстве специализированной вычислительной системы целесообразно применять алгоритм 2. Алгоритмы 3, 4 эффективны для расчета сетей сравнительно небольшого объема (до ста ветвей) при хранении информации о сети на полупостоянном ЗУ специализированной вычислительной системы.

ЛИТЕРАТУРА

1. А. Г. Евдокимов, Г. Ф. Кривуля. К вопросу проектирования специализированной ЦВМ для расчета сетевых систем. Тр. семинара «Управляющие машины и системы», вып. 5. Изд. АН УССР, Киев, 1969.

2. Р. К. Прим. Кратчайшие связывающие сети и некоторые обобщения. Кибернетический сборник. Изд-во иностр. лит., № 2, 1961.

3. J. B. Kruskal On the Shortest Spanning Subtree of a graph and the Traveling Salesman Problem. Proc. Am. Math Soc., 7, 1956.

4. А. А. Волков. Линейные операторы сильно связных графов и их определение при помощи ЦВМ. Материалы координационного совещания секции технической кибернетики. Изд. АН УССР, Киев, 1969.

ИССЛЕДОВАНИЕ ДИНАМИКИ ВЕРТИКАЛЬНОГО ЦЕНТРОБЕЖНОГО ЭКСТРАКТОРА

Г. К. Гончаренко, В. Я. Стороженко

Рубежанск

Жидкостная экстракция широко распространена в химической технологии [1, 2, 3]. Автоматизация управления процессами жидкостной экстракции требует определения динамических характеристик аппаратуры.

Одна из основных причин, затрудняющих автоматизацию производственных процессов и отдельных сложных типовых объектов, — недостаточное знание динамических свойств объектов и процессов, что необходимо для построения наиболее эффективных систем управления.

Необходимо изучение динамических свойств типовых промышленных объектов, характера возмущающих воздействий, а также математическое их описание. При этом важнейшая проблема синтеза системы автоматического управления может быть правильно решена только после выяснения динамических характеристик объекта как основного звена системы регулирования. Только на этой основе при проектировании могут быть правильно решены основные вопросы об оптимальных параметрах объекта и регулятора.

Исследовался вертикальный центробежный жидкостный экстрактор, состоящий из конуса и вращающегося ротора корзинного типа. Экспериментальная установка состояла (рис. 1) из напорных баков легкой и тяжелой фазы *1*, *2*, ротаметров *3*, экстрактора *4*, приемных баков легкой и тяжелой фазы *5*, *6*, насоса *7* и ловушки *8*.