

ОБ ОДНОМ АЛГОРИТМЕ ПРЕОБРАЗОВАНИЯ ФОРМЫ ПРЕДСТАВЛЕНИЯ ГРАФА

Ю. Н. ЕФИМОВ, П. А. СЕДЕЛЬНИКОВ

(Представлена научным семинаром ОВТ НИИ АЭМ)

Благодаря успехам вычислительной техники существенно расширился круг прикладных задач, использующих сетевые модели и сформулированных в терминах теории графов. Известно, что при этом могут быть применены три равнозначных формы представления топологии графа [1]:

1. $G = (I, \Gamma)$ — перечень вершин и их прямых отображений.
2. $G = (I, \Gamma^{-1})$ — перечень вершин и их обратных отображений.
3. $G = (I, U)$ — перечень вершин и дуг.

Обычно алгоритмы анализа сетевых моделей требуют определенной формы задания топологии, которая часто может не совпадать с исходной формой представления, определяемой сложившимися формами документов, традициями или просто удобством формирования. Например, в АСУП-Томск [2] при расчете планов запуска сетевая модель должна быть представлена в форме 1, а при формировании ведомости комплектации — в форме 2. Причем исходная топология в свою очередь может быть представлена в форме 1 или 2 в зависимости от используемых для ее составления форм конструкторской документации (сводные или угловые спецификации изделия).

В связи с этим возникает задача алгоритмического преобразования формы представления графа. Переход от 1 и 2 форм задания к 3 и от 3 к 1 и 2 тривиален и в дальнейшем рассматриваться не будет. Более сложным случаем является преобразование формы 1 в форму 2 и наоборот.

Опишем вторую задачу преобразования формы представления графа, возникающую в АСУП, использующих сетевые модели, составленные на основе конструкторской документации на изделия [2]. Для упрощения и ускорения вычислений при формировании планов запуска желательно иметь только сводную сетевую модель на несколько товарных изделий, имеющих много общих продуктов (приборов, узлов, деталей и т. д.). Однако трудности составления сводных сетевых моделей в силу существующих на предприятии форм конструкторской документации вызывает необходимость разработки алгоритма для машинного «сшивания» (объединения) частных сетей товарных изделий в сводную сеть заказа. Наличие такого алгоритма обеспечивает возможность формирования отдельных массивов топологии разными исполнителями независимо друг от друга, что повышает достоверность входной информации. Поскольку описываемая задача является более общей и включает в себя первую задачу как частный случай, дадим более подробное описание ее решения.

В терминах теории графов [1] задачу «сшивания» сетевых моделей можно сформулировать следующим образом:

Дано n графов $G_1 = (I_1, U_1), \dots, G_k = (I_k, U_k), \dots, G_n = (I_n, U_n)$,
 где:

I_k — множество вершин графа G_k ,
 U_k — множество дуг графа G_k , $\bigcup U_k \neq \emptyset$.

Каждой дуге $(i_k, j_k) \in U_k$ поставлена в соответствие некоторая совокупность параметров M_{i_k, j_k} . Требуется найти граф $G = (I, U)$, где

$$I = \bigcup_{k=1}^n I_k, U = \bigcup_{k=1}^n U_k, \text{ причем каждая дуга } (i, j) \in U \text{ должна сохранить со-}$$

ответствующие параметры $M_{i,j}$.

Для решения поставленной задачи можно применить операцию объединения множеств дуг $U_k (k=1, \dots, n)$. Для этого, рассматривая дуги всех исходных графов как числа, составленные из кодов начальной и конечной вершин каждой дуги, необходимо их упорядочить по возрастанию (или убыванию), после чего из полного множества дуг исключить повторяющиеся. Однако реализация такого алгоритма на ЭВМ приведет к большим затратам машинного времени в силу значительных объемов исходной информации и неэффективности существующих [3] алгоритмов внешней сортировки.

Учитывая вышеизложенное, а также представление исходных графов в виде множества вершин J_k и их прямых отображений Γ_{ik} (форма 1), нами предлагается следующий алгоритм сшивания, состоящий из 4 основных этапов.

На первом этапе просматривается информация о топологиях всех «сшиваемых» сетей и формируется множество $I = \bigcup_{k=1}^n I_k$, которому ставит-

ся в соответствие множество W . Каждый элемент $\omega_k \in W$ определяет локальную степень [4] ρ_{i_k} вершины $i_k \in J$ некоторого графа $G^* = (J, \Gamma)$, где

$$I = \bigcup_{k=1}^n I_k, \Gamma = \{\Gamma_1, \Gamma_2, \dots, \Gamma_n\}.$$

Процедура построения множеств I и W выполняется за один просмотр исходной информации.

На втором этапе производится упорядочение множества J по возрастанию величин его элементов, что позволяет ускорить процесс исключения повторяющихся шифров вершин в исходном множестве. При сортировке J необходимы преобразования и в W с тем, чтобы не нарушилось однозначное соответствие между этими двумя множествами. Поскольку величина $|J|$ много меньше по сравнению с объемом исходных данных, то сортируемая информация обычно вмещается в оперативную память машины и можно применить эффективные алгоритмы внутренней сортировки (например, алгоритм поразрядной сортировки). После сортировки J произведем перекодировку вершин $i \in J$ в исходной информации. Перекодировка заключается в замене десятичных шифров в первых строках записей (i_{10}) на некоторые их восьмеричные эквиваленты (i_8). За восьмеричный шифр i_8 вершины $i_{10} \in J$ удобно принять ее порядковый номер в отсортированном массиве J . В результате получим сквозную нумерацию вершин (от единицы подряд) в восьмеричной системе счисления. Перекодировка осуществляется также за один просмотр исходной информации.

На третьем этапе работы алгоритма формируется граф G^* . Это основной и наиболее трудоемкий (в смысле расходования машинного времени) этап вычислений, предлагаемый вместо обычной процедуры внешней сортировки. Поясним его несколько подробнее.

Разобьем упорядоченное множество J на подмножества.

$$I^1 = \{i_1, i_2, \dots, i_\alpha\}, I^2 = \{i_{\alpha+1}, \dots, i_\beta\}, \dots, I^k = \{i_{\mu+1}, \dots, i_\nu\}, \dots, I^m = \{i_{\nu+1}, \dots, i_n\}$$

такие, что $\bigcup_{k=1}^m I^k = I, I^k \cap I^l = \emptyset$ при $k \neq l$.

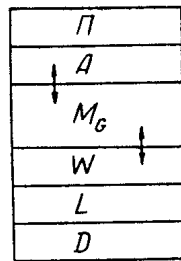


Рис. 1. Распределение оперативной памяти на третьем этапе работы алгоритма «сшивания». Π — рабочая программа; A — массив относительных адресов (переменный); M_G — массив для размещения информации о подграфах; $G^{*k} W = \{w_i/i \in I\}$; L — поле для считывания информации с МЛ; D — программа-диспетчер.

Пусть на каждом шаге работы алгоритма (3-го этапа) объем M_G оперативной памяти (рис. 1) позволяет разместить все записи для вершин $i \in J^k$, то есть сформировать подграф $G_{*k} \subseteq G^*$. Тогда процесс формирования подграфа G^{*k} можно осуществить за один просмотр исходной информации следующим образом.

Каждой вершине $i \in J_k$ поставим в соответствие элемент $a_i \in A$, являющийся относительным адресом размещения в оперативной памяти записи для этой вершины. Элементы a_i нетрудно вычислить, используя рекуррентное соотношение:

$$a_{i_{\mu+k+1}} = a_{i_{\mu+k}} + w_{i_{\mu+k}} + 1.$$

Для первого по порядку элемента $i_{\mu} \in J^k$ имеем $a_{i_{\mu}} = 0$.

Теперь, просматривая исходную информацию, для вершин $i \in J^k$ записи переписываем в поле в соответствии с адресами $a_i \in A$. В результате за один просмотр исходной информации подграф G^{*k} будет сформирован и, отсортировав элементы внутри каждой записи, его можно переписать на МЛ. Очевидно, что после выполнения подобной процедуры с каждым из подмножеств $J^k (k=1, \dots, m)$ будет сформирован граф G^* .

Так как после перекодировки мы получаем сквозную нумерацию вершин, то при формировании подмножеств J_k достаточно запоминать только их начальные $i_n^{k_n}$ и конечные $i_k^{k_n}$ элементы.

В целом работа третьего этапа алгоритма сводится к следующему. Принимается $A = \emptyset$, $M_G = \text{const}$. Формируется $i_n^1 = 1$ для J^1 . Для каждого элемента i , вводимого в J^1 , должно выполняться условие:

$$w_i + 1 \leq M_G.$$

Если для некоторого элемента i это условие не выполняется, то $i_n^1 = i_{n-1}$, а $i_n^2 = i$. При выполнении условия элемент относится к J^1 , в A заносится соответствующий относительный адрес и изменяется значение M_G .

$$M_G := M_G - (w_i + 1) - 1.$$

После получения J^1 формируется соответствующий подграф, затем подмножество J^2 , для которого начальный элемент уже получен, и т. д. до формирования последнего подграфа G^{*m} . Нетрудно видеть, что при выполнении третьего этапа требуется m просмотров исходной информации. Отметим также, что количество элементов в J^k может быть различным. Однако если все записи имеют постоянную длину, то за счет переменных по величине полей A и M_G предлагаемый алгоритм обеспечивает $|J^k| \leq |J^{k+1}|$, что повышает быстродействие.

На четвертом этапе просматривается G^* и из него исключаются повторяющиеся дуги. Причем исключаются только те дуги, у которых равны и соответствующие параметры $M_{i,j}$. В противном случае необходимо отпечатать информацию о допущенной ошибке в одной из «сшиваемых» топологий. Предлагаемый алгоритм инвариантен к 1 и 2 формам представления топологии.

Для решения первой задачи (преобразования 1-й формы во вторую, и наоборот) предлагается алгоритм, использующий идеи общей задачи и состоящий из двух этапов. Пусть исходная топология задана в форме 1, тогда на первом этапе формируется множество W , каждый элемент которого $w_j \in W$ равен количеству дуг, входящих в вершину $j \in J$ (т. е. количеству элементов $i \in \Gamma_j^{-1}$). Формирование множества W заключается в просмотре всех записей исходной информации и выполнении $w_j = w_j + 1$ для каждого j , входящего в i -ю запись. На втором этапе формируется граф $G = (J, \Gamma^{-1})$. Для этого множество J также разбивается на пересекающиеся подмножества $J^k (k=1, \dots, m)$. За каждый просмотр исходной информации формируется подграф $G^k = (J^k, \Gamma^{-1k})$, включающий записи только для вершин $j \in J^k$, где k — порядковый номер просмотра.

Следует отметить, что предлагаемый алгоритм также инвариантен к первым двум формам представления топологии графа.

Вышеизложенные алгоритмы были реализованы на ЭВМ «Урал-11Б» и показали достаточно высокую эффективность. Предложенная методика может быть также применена для внешней сортировки записей переменной длины.

ЛИТЕРАТУРА

1. К. Берж. Теория графов и ее применения. М., изд-во ИЛ, 1962.
2. Ю. Н. Ефимов, В. И. Кизев, В. И. Невраев, Ф. И. Перегудов, П. А. Седелников. Основные принципы построения и функционирования АСУП-Томск. (Настоящий сборник).
3. С. Б. Погребинский, Л. С. Лозинский. К вопросу о сортировке информации с помощью магнитных лент. «Кибернетика», № 2, 1965.
4. О. Оре. Теория графов. М., «Наука», 1968.