объединенный
институт
ядерных
исследований
дубна

879/2-80

25/2-80

Р5 - 12874

Е.Ю.Мазепа, И.Н.Силин, Е.Д.Федюнькин

О ПЕРЕЧИСЛЕНИИ ЧАСТИЧНЫХ ГРАФОВ С ЗАДАННЫМ ЦИКЛОМАТИЧЕСКИМ ЧИСЛОМ

1979

Мазепа Е.Ю., Силин И.Н., Федюнькин Е.Д. P5 - 12874

О перечислении частичных графов с заданным цикломатическим числом

На основе анализа топологических свойств графа предложен быстрый алгоритм поиска всех ν-графов. ν-графом заданного симметрического мультиграфа $G=(X,U)$ с множеством вершин X и множеством ребер U назван частичный граф $G^{\nu}=(X,u)$, $u \subset U$, с цикломатическим числом ν, имеющий столько же компонент связности, сколько граф $G=(X,U)$. В частности, G° является деревом /каркасом/ графа G. При реализации на ЭВМ алгоритм требует минимальной памяти.

Работа выполнена в Лаборатории вычислительной техники и автоматизации ОИЯИ.

Препринт Объединенного института ядерных исследований. Дубна 1979

Mazepa E.Yu., Silin I.N., Fedyun'kin E.D. P5 - 12874

On the Generation of All Fixed Cyclomatic Number Spanning Subgraphs

Fast ν-graph generation algorithm is proposed on the base of graph topology analysis. Spanning subgraph G^{ν} is defined as ν-graph of the undirected multigraph $G=(X,U)$ (set of vertices X and set of edges U), if graph $G^{\nu}=(X,u)$, $u \subset U$, has cyclomatic number ν and the same number of components as graph $G=(X,U)$. In particular G° is spanning subtree (skeleton) of the graph G. Computer realization of the algorithm needs minimal memory.

The investigation has been performed at the Laboratory of Computing Techniques and Automation, JINR.

Preprint of the Joint Institute for Nuclear Research. Dubna 1979

1. ОСНОВНЫЕ ПРИНЦИПЫ

Что касается теории графов, будем в основном придерживаться терминологии Бержа /1/. Симметрический граф $G=(X,U)$ задан, если задано множество вершин X и множество ребер U. Ребро $r \in U$ задается неупорядоченной парой его граничных вершин: $r=(x_i,x_j)$. В случае мультиграфа множество U содержит кратные элементы. Цикломатическим числом графа G с N вершинами, M ребрами и p компонентами связности называется число

$$\nu(G)=M-N+p.$$

Частичным графом графа $G=(X,U)$ называется граф $g=(X,u)$, $u \subset U$, т.е. он получается исключением из заданного графа некоторых ребер. Будем называть ν-графом симметрического мультиграфа $G=(X,U)$ частичный граф $G^{\nu}=(X,u)$ с цикломатическим числом ν, имеющий такой же индекс связности, как граф G, т.е. столько же компонент связности. Мы хотим сконструировать алгоритм, отыскивающий все ν-графы заданного графа.

Алгоритмы перечисления дискретных объектов из некоторого класса обладают рядом общих свойств. Существует две основных разновидности таких алгоритмов.

1. Алгоритмы параллельного действия. В процессе работы алгоритма конструируются некоторые промежуточные структуры, из которых в конце работы алгоритма одновременно получаются искомые объекты. К этой разновидности принадлежит известный японский алгоритм для отыскания всех каркасов заданного графа /2/.

2. Алгоритмы последовательного действия. Искомые объекты производятся последовательно, по одному. Предполагается, что вызывающая процедура умеет последовательно использовать полученные объекты и что для нее несущественен порядок, в котором объекты генерируются. В известных практических задачах это действительно так. Как правило, вычисляется некото-

рая функция от объекта, которая либо интегрируется по всем объектам, либо служит критерием для выбора оптимального объекта.

Мы хотим сконструировать алгоритм, допускающий реализацию в виде программы для ЭВМ, поэтому минимизация объема рабочих массивов приобретает особую важность. Класс параллельных алгоритмов совершенно неприемлем с этой точки зрения. Число промежуточных структур или искомых объектов может достигать 2^N, необходимая память становится фантастической даже для графов со сравнительно малым числом вершин N. Следовательно, искомый алгоритм должен принадлежать классу последовательных алгоритмов.

Две основные проблемы, которые должны быть решены - как обеспечить генерацию всех объектов и как обеспечить перебор без повторений. Очевидное решение последней проблемы - держать в памяти список уже полученных объектов и сравнивать со списком очередной найденный объект. Однако такой метод приводит к алгоритмам, которые по своим параметрам мало чем отличаются от класса параллельных алгоритмов, - требуется гигантская память и прогрессивно ухудшается быстродействие. Удовлетворительный принцип состоит в следующем: алгоритм должен быть устроен так, чтобы осуществлялся жестко упорядоченный перебор искомых объектов. Тогда информация, которую нужно хранить, чтобы избежать повторений, - текущее состояние процесса перебора. Этот принцип автоматически гарантирует и генерацию всех искомых объектов.

По-видимому, существует единственная процедура, удовлетворяющая указанному принципу и универсально применимая ко всем алгоритмам перебора объектов из заданного класса. Алгоритмы, сконструированные на основе этой процедуры, мы будем называть стековыми алгоритмами. То же самое Кристофидес /3/ называет деревом решений. Последнее название следует признать неудачным, поскольку смысл процедуры чисто комбинаторный. Установленный выше принцип, вообще говоря, требует в нашем случае упорядоченного перебора $C_M^{M_\nu}$ сочетаний ребер ($M_\nu = N+\nu-p$) с проверкой сохранения индекса связности в каждом частичном графе. Мы увидим ниже, что стековая процедура позволяет избежать перебора всех сочетаний.

2. ОСНОВНОЙ АЛГОРИТМ

Перенумеруем в каком-либо порядке ребра исходного графа. Нам нужно исключить $M-M_\nu$ ребер. Если при исключении ребра 1 индекс связности сохраняется, то исключаем это ребро и загружаем его в стек. Если нет - переходим к ребру 2.

В конце концов какое-то ребро будет исключено. Попробуем исключить еще одно ребро, начиная с номера, на единицу большего номера последнего ребра, помещенного в стек. Действуя так $M-M_\nu$ раз, мы получим первый ν-граф. В стеке при этом окажутся ребра, исключенные из исходного графа. Для того, чтобы найти очередной ν-граф, исключим из стека последнее загруженное ребро и вернем это ребро в граф. Попробуем исключить из графа ребро с номером, на единицу большим номера ребра, только что присоединенного к графу. Если это возможно, мы получим очередной ν-граф, если нет - перейдем к следующему ребру и т.д. Если мы достигнем ребра с максимальным номером и ни одно из ребер нельзя исключить, достанем из стека последнее загруженное в него ребро и вернем это ребро в граф. Попробуем исключить из графа ребро с номером, на единицу большим номера ребра, только что присоединенного к графу, и т.д. Каждый раз, когда сформирован очередной искомый ν-граф, в стеке оказывается $M-M_\nu$ исключенных ребер.

Действуя таким образом, мы переберем все ν-графы. Процесс окончится, когда мы попытаемся достать ребро из пустого стека. Алгоритм устроен так, что если частичная конфигурация исключенных ребер изменяет индекс связности, то автоматически игнорируются все "висящие" на ней полные конфигурации. Заметим, что состояние процесса перебора однозначно определяется содержимым стека в момент выдачи очередного ν-графа.

На рис. 1 представлена логическая схема основного алгоритма. Исходный граф задается матрицей G размерности (2,M), M - число ребер исходного графа. G(1,i) содержит номер первой граничной вершины ребра i, G(2,i) содержит номер второй граничной вершины. При первом обращении к алгоритму должно быть установлено m:=0. Результат работы алгоритма засылается в флаговый массив W размерности M. Элементы этого массива принимают значения 0 или 1. W(i)=0 означает, что ребро i исходного графа отсутствует в данном ν-графе. Переменная L -счетчик числа ν-графов. Мы считаем для простоты заданным число вершин N исходного графа / N можно легко найти из матрицы G /. Чтобы не усложнять логическую схему, мы полагаем также, что исходный граф не имеет изолированных вершин и петель - ребер типа (x_i, x_i).

Функция CT(G,W,m) принимает значения 0 или 1, CT(G,W,m)=1, если при исключении ребра m из частичного графа, заданного матрицей G и флаговым массивом W, сохраняется индекс связности. Функция PT(G) определяет индекс связности исходного графа G. Переменные m и k являются рабочими переменными.

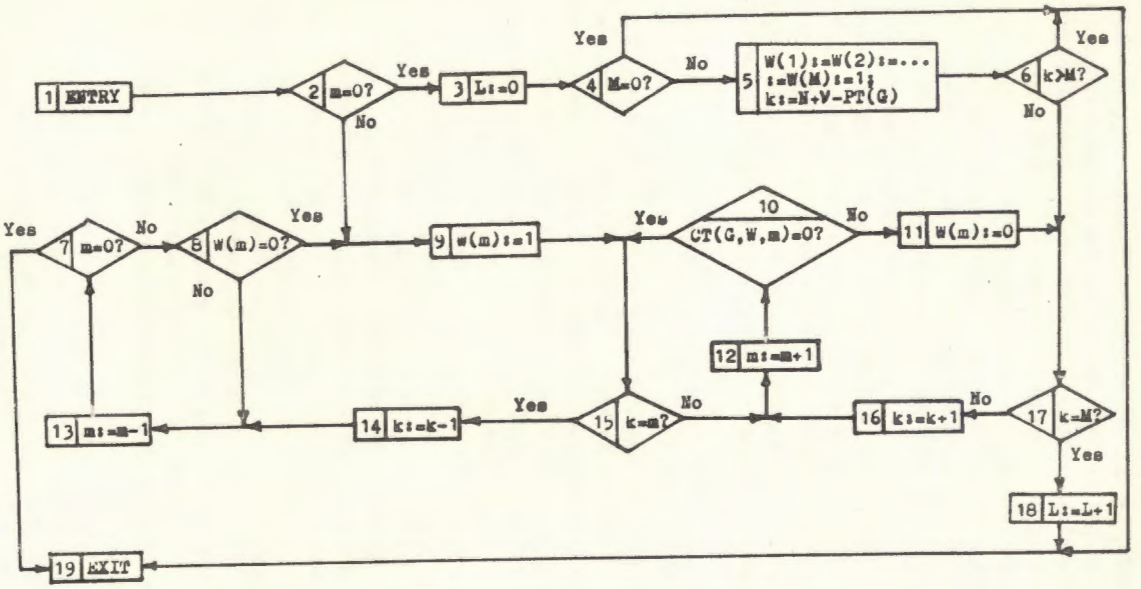

Рис. 1. Основной алгоритм.

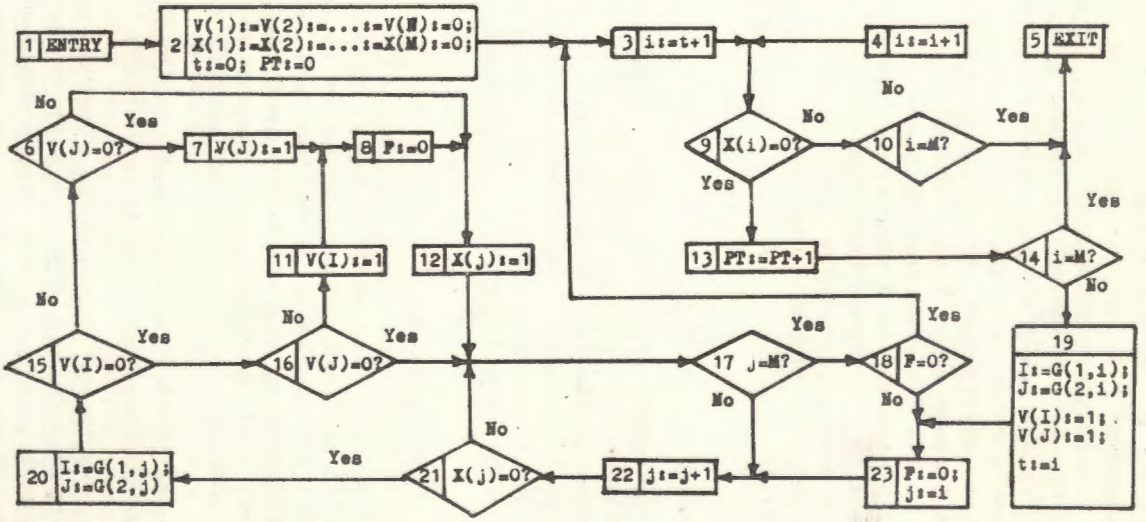

Рис. 2. Функция PT(G).

При каждом обращении к алгоритму конструируется один ν-граф. Чтобы получить очередной ν-граф, нужно обратиться к алгоритму еще раз и т.д. Признак конца перебора - выход из алгоритма с переменной m=0, это означает, что последний ν-граф был выдан в предыдущем обращении. Особый случай возникает, когда m=0 после первого обращения к алгоритму. Если при этом L=1, то исходный граф совпадает с искомым; если L=0, то искомого графа не существует.

При выбранной нами конструкции алгоритма нет нужды реализовывать стек в явном виде. Вместо классического стека мы имеем флаговый массив W и указатель последнего исключенного ребра m. Флаговый массив W выполняет, таким образом, две функции. На рис. 1 хорошо видны два конкурирующих процесса. Первый из них соответствует заполнению стека /блоки 10,11,12,15,16,17/, второй - опорожнению стека /блоки 7,8,9,13,14,15/.

3. КОНТРОЛЬ СВЯЗНОСТИ

На рис. 2,3 представлены логические схемы функций PT и CT. Массив X - рабочий массив размерности M, массив V - рабочий массив размерности N. G, W, M, m имеют тот же смысл, что и на рис. 1, остальные переменные являются рабочими. С целью минимизации размеров массива V мы здесь считаем, что вершины исходного графа пронумерованы без разрывов.

Роль, которую играет процедура CT, можно было бы реализовать посредством простой модификации процедуры PT. Внимательный анализ основного алгоритма показывает, однако, что CT(G, W, m) является наиболее трудоемкой частью алгоритма, потребляющей большую часть времени. Чтобы выяснить, изменит ли исключение ребра m индекс связности, нет необходимости вычислять сам индекс - можно использовать упрощенную процедуру, которая в среднем дает существенный выигрыш во времени.

Реализуя процедуру CT(G,W,m), мы основывались на следующей простой теореме, которую приводим без доказательств.

Теорема 1. Пусть частичный граф $g=(X,u)$ получен удалением из графа $G=(X,U)$ ребра (x_i, x_j). Графы G и g имеют одинаковое число компонент связности и задают одинаковые разбиения множества вершин в том и только том случае, если в графе g существует цепь, соединяющая вершины x_i и x_j.

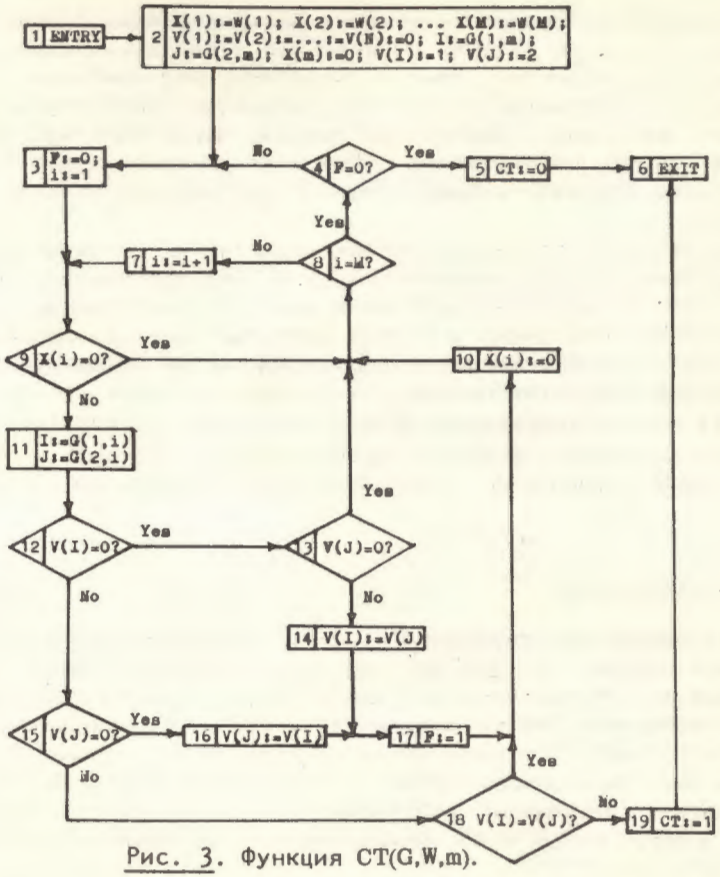

Рис. 3. Функция CT(G,W,m).

4. ОБ ОПТИМАЛЬНОЙ РЕАЛИЗАЦИИ СТЕКОВОГО АЛГОРИТМА

Единственная известная авторам реализация стекового алгоритма принадлежит Хусаинову /4/. Действуя в духе изложенных выше принципов, Хусаинов создал алгоритм для перечисления всех деревьев /т.е. 0-графов/ заданного графа. Подход Хусаинова в некотором смысле обратный нашему: искомый граф строится не методом исключения ребер из заданного графа, а методом последовательного присоединения ребер к первоначально пустому множеству. Такой подход приводит к проигрышу по быстродействию, когда цикломатическое число исходного графа сравнительно невелико, и поэтому не может быть рекомендован в указанном случае.

Способ существенного улучшения быстродействия следует, однако, искать не на пути оптимальной реализации технических деталей алгоритма. Выше мы отметили, что наиболее трудоемкая часть алгоритма - контроль связности. Анализируя применение алгоритма к конкретным графам, можно заметить, что не для каждого исключаемого ребра этот контроль нужно осуществлять. Соответствующие методы будут развиты в следующем разделе.

Что касается основного алгоритма, отметим следующий факт: все рабочие массивы являются по сути дела флаговыми и поэтому могут быть плотно упакованы вместе с массивом G. Этим самым будет обеспечена и полная реентерабельность программы. При практической реализации реентерабельного алгоритма на ЭВМ БЭСМ-6 нам потребовалось $M+3$ слова для того, чтобы держать полную информацию об исходном графе, результирующем графе и состоянии процесса перебора. Число ребер M исходного графа ограничено только доступной оперативной памятью ЭВМ.

5. КВАЗИЦИКЛЫ

Из требования сохранения индекса связности следует, что ν-граф содержит в себе все перешейки исходного графа. Используя это обстоятельство, можно сразу улучшить быстродействие алгоритма. Чтобы продвинуться дальше, нам предстоит углубиться в топологические аспекты структуры графа.

Назовем квазициклом графа $G=(X,U)$ множество ребер $K \subset U$, обладающее следующими свойствами:

1. Каждое ребро $r_i \in K$ принадлежит по крайней мере одному циклу;
2. Для любой пары ребер $r_i, r_j \in K$ и любого цикла $C \subset U$ справедливо: если $r_i \in C$, то $r_j \in C$;
3. Не существует ребра $r \in U \setminus K$ такого, что множество $\{r,K\}$ обладает свойствами 1 и 2 /т.е. множество K не может быть расширено/.

Квазицикл может состоять из единственного ребра. В этом случае свойство 2 теряет смысл. Перечислим непосредственные следствия указанных выше свойств.

1. Квазицикл не содержит перешейков.
2. Квазицикл принадлежит одной и только одной компоненте связности.
3. Для любой пары ребер $r_i, r_j \in K$ и любого цикла $C \subset U$ справедливо: если $r_i \notin C$, то $r_j \notin C$.
4. Если ребро $r \in K$ принадлежит единственному циклу $C \subset U$, то $K \equiv C$.

Теорема 2. Если K_1 и K_2 - два различных квазицикла графа G, то $K_1 \cap K_2 = \phi$. Предположим обратное. Пусть r - общее ребро K_1 и K_2, r_1 принадлежит только K_1. Применяя свойство 2 к $r, r_1 \in K_1$, найдем, что все циклы, содержащие r, содержат r_1, и поэтому множество K_2 может быть расширено, что противоречит свойству 3.

Следствие теоремы 2: если два квазицикла имеют непустое пересечение, они совпадают.

Теорема 3. Если K - квазицикл графа G, то исключение из графа G любой пары ребер $r_i, r_j \in K$ изменяет индекс связности. Предположим обратное. Пусть g - частичный граф, полученный исключением из графа G ребер r_i, r_j и пусть x_1, x_2 - граничные вершины ребра r_j. Если индексы связности графов g и G совпадают, то, на основании теоремы 1, в графе g существует цепь, соединяющая вершины x_1 и x_2. Но тогда в графе G существует цикл, составленный из этой цепи и ребра r_j и не содержащий ребра r_i, что противоречит свойству 2.

Следствие теоремы 3: если K - квазицикл графа G, то в частичном графе G_1, полученном исключением ребра $r \in K$ из графа G, ребра $\{K \setminus r\}$ являются перешейками.

Теорема 4. Если ребра $r_i, r_j \in U$ не являются перешейками и при исключении пары r_i, r_j из графа $G=(X,U)$ изменяется индекс связности, то существует квазицикл K, такой, что $r_i, r_j \in K$. Свойство 1 выполняется по определению; нам осталось доказать, что выполняется свойство 2. Предположим обратное. Пусть существует цикл C, такой, что $r_i \in C$, $r_j \notin C$. Обозначим через G_1 частичный граф, полученный исключением из графа G ребра r_j. Поскольку r_j не является перешейком, индексы связности графов G_1 и G совпадают. Поскольку $r_j \notin C$, цикл C существует и в графе G_1. Обозначим через G_2 частичный граф, полученный исключением из графа G_1 ребра r_i, и пусть x_1, x_2 - граничные вершины этого ребра. В графе G_2 существует цепь, соединяющая вершины x_1, x_2 и полученная удалением из цикла C ребра r_i. Тогда на основании теоремы 1 совпадают индексы связности графов G_1 и G_2 и, следовательно, совпадают индексы связности графов G и G_2, что противоречит условиям теоремы 4.

Теорема 5. Если $S_0 \subset U$ - множество перешейков графа $G=(X,U)$ и частичный граф G_1 получен исключением ребра $r \in U \setminus S_0$ из графа G, S_1 - множество перешейков графа G_1, то множество ребер $K = \{r, S_1 \setminus S_0\}$ является квазициклом графа G. На осно-

вании теоремы 4 каждая пара r, r_i ($r_i \in S_1 \setminus S_0$) принадлежит некоторому квазициклу K_i. На основании следствия теоремы 2 все квазициклы K_i совпадают. На основании следствия теоремы 3 множество K не может быть расширено.

Пусть S_0 - множество перешейков графа $G=(X, U)$, $u = U \setminus S_0$. Из теоремы 2 следует, что существует единственное разбиение множества u на систему квазициклов. Из теоремы 3 следует, что при построении ν-графа нельзя исключать более одного ребра из кваждого квазицикла. Пусть некоторый ν-граф задан множеством исключенных ребер $R \subset U$ и ребро $r \in R$ принадлежит квазициклу K. Если вместо r исключить любое другое ребро, принадлежащее квазициклу K, то получится тоже ν-граф. Для доказательства восстановим ребро r, тогда в ν-графе появится лишний цикл, целиком содержащий в себе квазицикл K. Чтобы разоравать лишний цикл, мы можем исключить любое принадлежащее ему ребро и, в частности, любое ребро, принадлежащее квазициклу K.

Теперь у нас есть метод существенного ускорения основного алгоритма. Будем называть квазицикл исключенным, если он содержит исключенное ребро. Контроль на связность нужно делать только при переходе к новой конфигурации исключенных квазициклов. При переборе ν-графов, ассоциированных с одной и той же конфигурацией исключенных квазициклов, контроль на связность делать не нужно.

Концепция квазициклов делает особенно прозрачным топологический смысл контроля на связность. Используя квазициклы, мы как бы сворачиваем исходный граф к минимальному топологическому эквиваленту.

6. МОДИФИЦИРОВАННЫЙ АЛГОРИТМ

Теорема 5 дает конструктивный метод разбиения графа на систему квазициклов. На рис. 4 дан пример такого разбиения, ребра помечены номером соответствующего квазицикла, номером 0 помечены перешейки.

На рис. 5 изображена логическая схема процедуры KT(G,W,K), производящей разбиение исходного графа на систему квазициклов. G, W, M имеют тот же смысл, что и в основном алгоритме. Массив K размерности M - флаговый, K(i) принимает значения 0 или 1. A↔B означает перестановку объектов A и B. Переменные m, t, τ - рабочие.

Процедура KT переупорядочивает массив G таким образом, что в начале массива оказываются все перешейки, после них расположены все ребра, принадлежащие первому квазициклу, затем все ребра, принадлежащие второму квазициклу, и т.д. В массив K засылается следующая информация: K(i) = 1 означа-

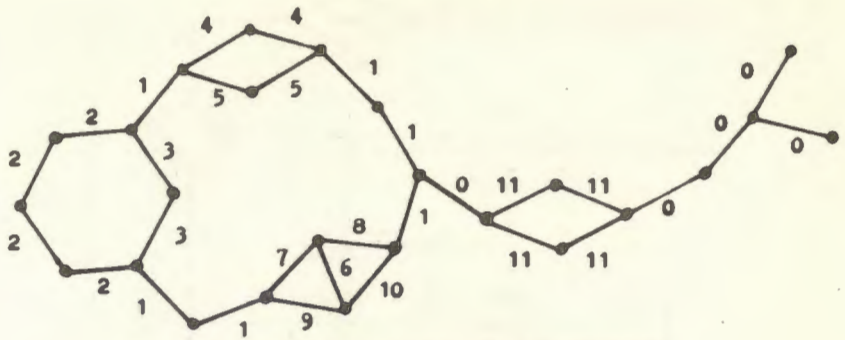

Рис. 4. Разбиение графа на квазициклы.

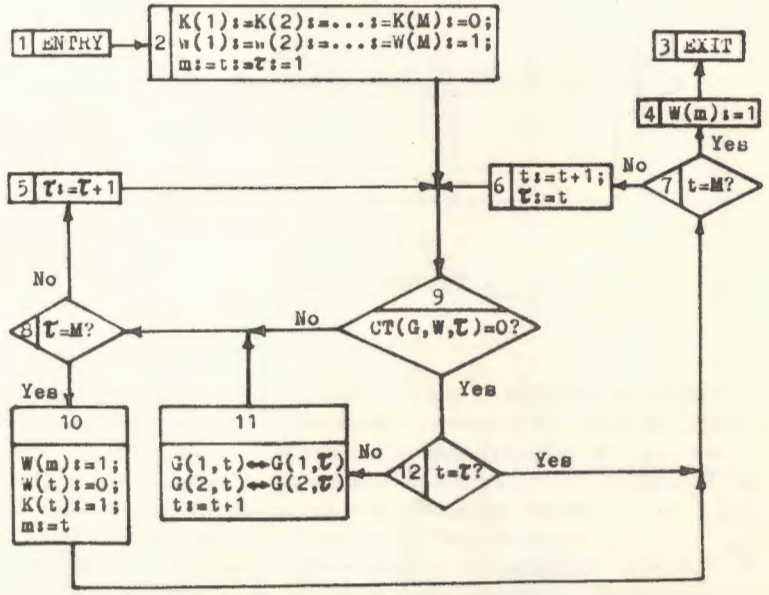

Рис. 5. Процедура KT(G,W,K).

ет, что ребро i в массиве G является первым ребром нового квазицикла. После выхода из процедуры все W(i) = 1. При перестановках теряется информация об оригинальном номере ребра. Если эта информация существенна, номер можно держать вместе

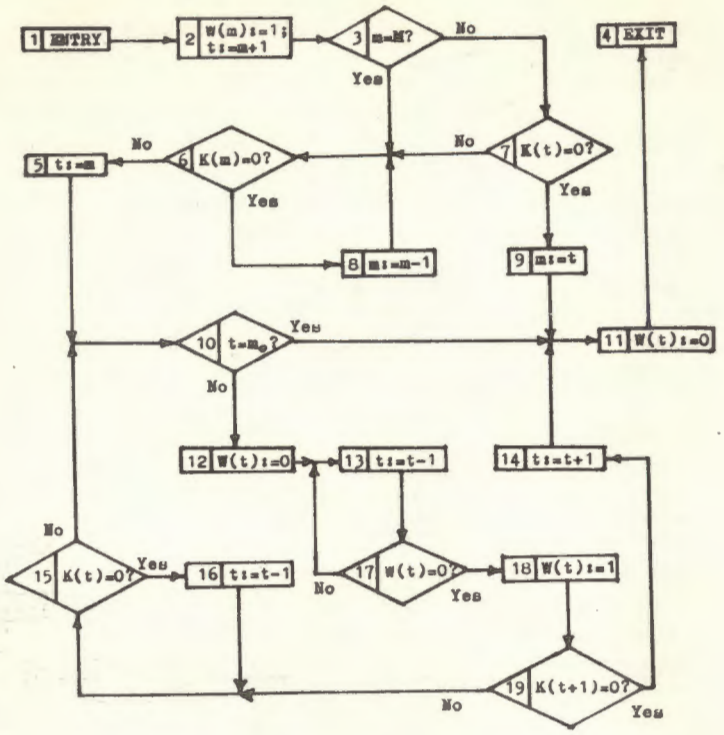

Рис. 6. Процедура $NT(W,K,m_0,m,t)$.

с ребром в массиве G и переставлять номера при перестановке ребер. В этом случае массив G будет иметь размерность (3,M).

На рис. 6 представлена логическая схема процедуры $NT(W,K,m_0,m,t)$, которая генерирует ν-графы, ассоциированные с одной конфигурацией исключенных квазициклов. Здесь m_0 - номер первого ребра первого исключенного квазицикла, m - номер последнего исключенного ребра, t - номер нового по сравнению с предыдущим ν-графом исключенного ребра. Если после выхода из процедуры $t = m_0$, то последний ν-граф, ассоциированный с данной конфигурацией, был выдан в предыдущем обращении. Заметим, что процедура NT сама по себе является стековым алгоритмом.

На рис. 7 представлена логическая схема модифицированного алгоритма. Логика работы алгоритма, связанная со сменой конфигураций исключенных квазициклов, практически

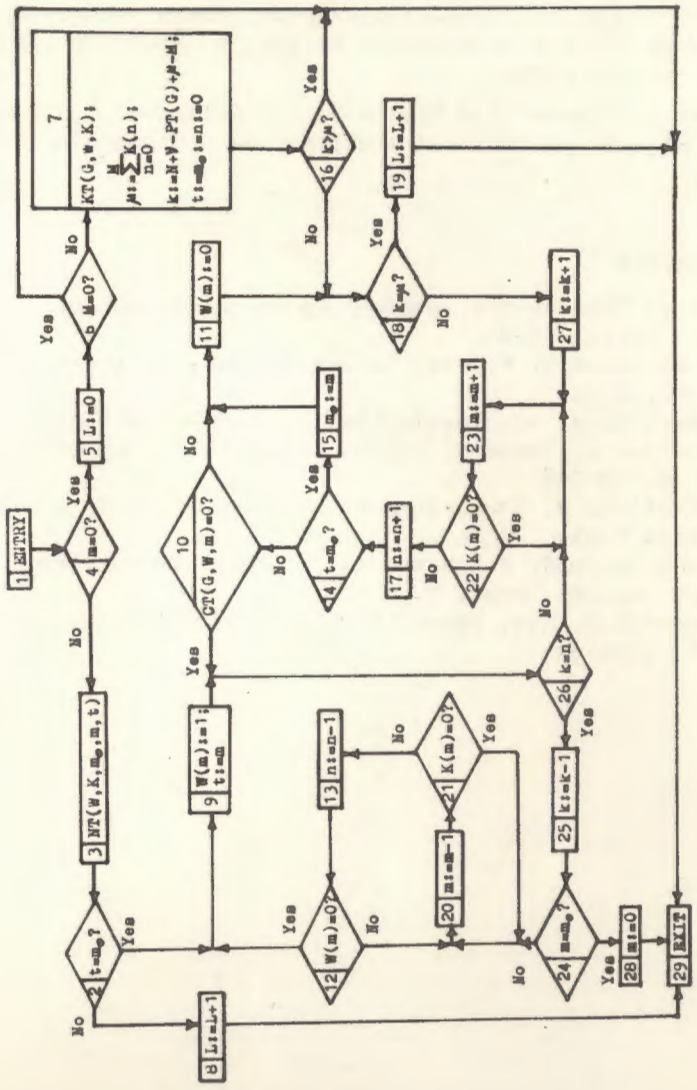

Рис. 7. Модифицированный алгоритм.

эквивалентна логике основного алгоритма. n - номер последнего исключенного квазицикла. Вход-выход организован так же, как в основном алгоритме.

Анализ процедуры NT показывает, что она будет работать быстрее, если квазициклы расположены в порядке возрастания их мощности. Соответствующую сортировку можно реализовать в процедуре NT. Это не отражено на рис. 5, чтобы не усложнять логическую схему.

Авторы благодарят М.И.Гуревича, С.С.Лебедева, А.П.Сапожникова, которые участвовали в обсуждении настоящей работы.

ЛИТЕРАТУРА

1. Berge C. Théorie des graphes et ses applicationss. Dunod, Paris, 1958.
   Русский перевод: К.Берж. Теория графов и ее применения. ИИЛ, М., 1962.
2. Kasahara Y. et al. Topological Evaluation of System Determinants. Technol. Repts., Osaka Univ., 12, oct., 1962, p.239-248.
3. Christofides N. Graph Theory, an Algorithmic Approach. Academic Press, 1975.
   Русский перевод: Н.Кристофидес. Теория графов, алгоритмический подход. "Мир", М., 1978.
4. Хусаинов Ш.Н. Изв. вузов СССР /радиоэлектроника/, 1977, 20, 6, с.96-99.

Рукопись поступила в издательский отдел
19 октября 1979 года.