АЛГОРИТМЫ АНАЛИЗА СТРУКТУРНОЙ ИНФОРМАЦИИ
(Вычислительные системы)
1984 год Выпуск 103

УДК 519.1

АЛГОРИТМЫ ДЛЯ ПОСЛЕДОВАТЕЛЬНО-ПАРАЛЛЕЛЬНЫХ ГРАФОВ КАК АЛГОРИТМИЧЕСКАЯ ОСНОВА ИНФОРМАЦИОННОЙ СИСТЕМЫ СТРУКТУРНОЙ ХИМИЧЕСКОЙ ИНФОРМАЦИИ

Б.Ш.Клейман, С.В.Нижний

§1. Основные теоремы о последовательно-параллельных графах

Пусть граф $G = (V,E)$ задается множеством вершин V и множеством ребер E, $v_1, v_2, \ldots, v_n$ – вершины графа, $e_1, e_2, \ldots, e_m$ – ребра, n – число вершин, m – число ребер графа.

Рассмотрим две операции над ребрами графа: подразбиение и расщепление. Пусть v_1, v_2 – вершины, e_1 – ребро графа G, вершина w не является вершиной в G, ребро $e = v_1v_2$ не является ребром в G.

ОПРЕДЕЛЕНИЕ I. Ребро e_1 подразбито, если оно заменено ребрами $q_1 = v_1w$ и $q_2 = wv_2$ (ребра q_1 и q_2 соединены последовательно), ребро e_1 расщеплено, если оно заменено ребрами $e_1 = v_1v_2$, $e = v_1v_2$ (ребра e_1 и e соединены параллельно).

Первую операцию обозначим $S(e_1)$, вторую – $P(e_1)$. Граф, полученный операциями подразбиения и расщепления каких-либо ребер графа G, обозначим $S(G)$ и $P(G)$ соответственно.

ОПРЕДЕЛЕНИЕ 2. Граф с двумя вершинами и одним ребром – последовательно-параллельный граф. Если граф G – последовательно-параллельный граф, то графы $S(G)$ и $P(G)$ последовательно-параллельны.

Сформулируем необходимое и достаточное условие принадлежности графа классу последовательно-параллельных графов [1,2].

ТЕОРЕМА I. Граф G – последовательно-параллельный граф тогда и только тогда, когда он не содержит подграфа $H \subseteq G$, гомеоморфного K_4.

Легко видеть, что граф последовательно-параллельный, если все его двусвязные компоненты последовательно-параллельны.

Определим операцию стягивания вершины степени два как удаление вершины степени два и замену двух ребер, инцидентных данной вершине, ребром, соединяющим две вершины, смежные стягиваемой. Если вершины, смежные стягиваемой, были смежны между собой, то заменим получившиеся параллельные ребра одним ребром.

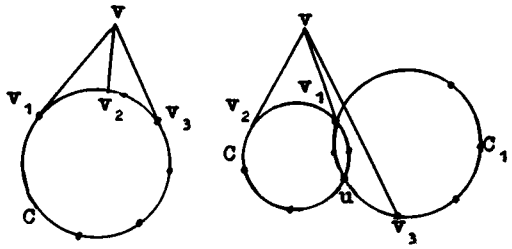
Получим необходимое условие того, что граф принадлежит классу последовательно-параллельных графов, являющееся усилением аналогичных условий, полученных Даффином [1] и Дираком [3].

ТЕОРЕМА 2. Пусть G_0 - двусвязный последовательно-параллельный граф с числом вершин, бо́льшим трех и пусть G - граф, полученный из G_0 заменой параллельных ребер графа G_0 одним из них. Тогда граф G содержит не менее двух несмежных вершин степени два.

ДОКАЗАТЕЛЬСТВО. Докажем, что последовательно-параллельный граф содержит не менее одной вершины степени два. Предположим противное: пусть степени всех вершин последовательно-параллельного графа G больше двух. Из двусвязности графа G следует, что он содержит вершину v такую, что граф $G_1=G-v$ также двусвязен [4]. Так как степень вершины v больше двух, то существуют три вершины $v_1, v_2, v_3 \in G_1$, смежные вершине v. Поскольку G_1 - двусвязный граф, то существует простой цикл C, проходящий через вершины v_1 и v_2. Рассмотрим два возможных расположения вершины v_3: I) v_3 - вершина цикла C, 2) v_3 не является вершиной цикла C.

В первом случае подграф $H \subseteq G$, вершинами которого являются v и вершины цикла C, а ребрами vv_1, vv_2, vv_3 и ребра цикла C, гомеоморфен K_4 (рис.I,а).

Пусть вершина v_3 не является вершиной цикла C. Тогда существует простой цикл $C_1 \subseteq G_1$, проходящий через вершины v_1 и v_3. Если циклы C и C_1 имеют одну общую вершину u (рис.I,б), отличную от v_1, то подграф $H_1 \subseteq G$, вершинами которого являются v, вершины простой цепи v_3u и вершины цикла C, а ребрами - v_1v, vv_2, vv_3, ребра простой цепи v_3u и ребра цикла C, гомеоморфен K_4. Если циклы C и C_1 имеют только одну общую вершину v_1, то рассмотрим простой цикл C_2, проходящий через вершины v_2 и v_3.



а) б)

Рис. I

Если циклы C, C_1 и C_2 не имеют общих вершин, кроме вершин v_1, v_2 и v_3, то существует простой цикл L, проходящий через вершины v_1, v_2, v_3 и составленный из цепей v_1v_2, v_1v_3, v_2v_3 циклов C, C_1, C_2. При таком расположении вершин v_1, v_2, v_3 мы приходим к случаю I.

Если цикл C_2 имеет хотя бы одну общую вершину с циклом C или C_1, кроме вершин v_2 или v_3, то по доказанному выше существует подграф $H_2 \subseteq G$, гомеоморфный K_4. Следовательно, предположение, что степени всех вершин последовательно-параллельного графа больше двух, неверно.

Докажем, что последовательно-параллельный граф содержит не менее двух несмежных вершин степени два. Доказательство этого утверждения проведем индуктивно по числу вершин графа.

Легко видеть, что не существует двусвязных последовательно-параллельных графов с четырьмя и пятью вершинами, имеющих одну вершину степени два.

Предположим, мы доказали, что не существует двусвязных последовательно-параллельных графов с n вершинами, которые имеют одну вершину степени два. Докажем, что не существует $(n+1)$-вершинных последовательно-параллельных графов, имеющих меньше двух вершин степени два. Доказательство проведем от противного.

Предположим, существует $(n+1)$-вершинный последовательно-параллельный граф, содержащий одну вершину степени два. Проведем операцию стягивания вершины степени два. При этом возможны два случая: а) вершины, смежные стягиваемой вершине, не смежны между собой; б) вершины, смежные стягиваемой вершине, смежны между собой.

В первом случае после стягивания вершины степени смежных ей вершин не изменились и остались больше двух. Полученный n-вершинный граф не является последовательно-параллельным в силу доказанного в первой части теоремы утверждения. Следовательно, и исходный $(n+1)$-вершинный граф не является последовательно-параллельным. По-

лученное противоречие показывает, что (n+1)-вершинный последовательно—параллельный граф содержит более одной вершины степени два.

Во втором случае степень каждой из вершин, смежных стягиваемой, после осуществления операции стягивания уменьшится на единицу. Если степень хотя бы одной из вершин, смежных стягиваемой, осталась больше двух, то по доказанному выше полученный граф, а следовательно, и исходный (n+1)-вершинный не являются последовательно-параллельными, что противоречит предположению индукции.

Пусть степени вершин, смежных стягиваемой, стали равны двум. Осуществим операцию стягивания для одной из этих вершин. Поскольку одна из вершин, смежных стягиваемой, имеет степень два, и число вершин графа больше пяти, то вершины, смежные стягиваемой, не смежны между собой. Поэтому после стягивания вершины степени два степени остальных вершин остались без изменения. В полученном (n-1)-вершинном графе только одна вершина имеет степень два, и по индуктивному предположению (n-1)-вершинный граф не является последовательно-параллельным. Следовательно, предположение о существовании (n+1)-вершинного последовательно-параллельного графа с одной вершиной степени два неверно. Этим заканчивается доказательство теоремы.

§2. Линейные алгоритмы идентификации и канонизации последовательно-параллельных графов

В этом параграфе рассматриваются двусвязные последовательно-параллельные графы, вершины и ребра которых могут быть помечены. Общее число меток (кодов) равно P.

Будем называть последовательность вершин и ребер графа $v_0, e_1, v_1, \ldots, e_r, v_r$ ($v_i \in V$, $e_j \in E$) цепочкой, если $r > 1$, степени всех вершин за исключением степеней вершин v_0 и v_r равным двум. Вершины v_0 и v_r назовем концами цепочки, а последовательность $e_1, v_2, e_2, \ldots, v_{r-1}, e_r$ – внутренностью цепочки.

Определим код цепочки как конкатенацию кодов вершин и ребер, образующих цепочку.

Введем две операции над последовательно-параллельными графами, используемые в алгоритмах их идентификации и канонизации.

I. Замена K параллельных ребер, соединяющих две вершины, одним ребром. Код полученного в результате проведения этой операции

ребра имеет следующий вид: код начала параллельного соединения || код первого ребра || разделитель кодов параллельных ребер ||...|| код К-го параллельного ребра || код окончания параллельного соединения, где || - знак конкатенации кодов.

2. Замена внутренности цепочки одним ребром, код которого равен коду соответствующей цепочки.

ЛЕММА I. Операции I и 2 оставляют последовательно-параллельный граф последовательно-параллельным, тогда как граф, не относящийся к данному классу, не станет последовательно-параллельным при применении к нему этих же операций.

Опишем схему алгоритмов идентификации и канонизации последовательно-параллельных графов.

Алгоритм идентификации последовательно-параллельных графов заключается в последовательном применении к исходному графу, пока это возможно, операции I, а затем операции 2. Если после очередного применения операции I полученный граф будет содержать меньше двух несмежных вершин степени два, то согласно теореме 2 исходный граф не является последовательно-параллельным. Если в результате последовательного применения операций I и 2 удастся свести исходный граф к K_2 или к простому циклу, то исходный граф - последовательно-параллельный.

Оценка сложности алгоритма идентификации последовательно-параллельных графов будет проведена одновременно с оценкой сложности алгоритма канонизации последовательно-параллельных графов.

Алгоритм канонизации последовательно-параллельных графов, как и алгоритм идентификации, заключается в последовательном применении к исходному графу, пока это возможно, сначала операции I, а затем операции 2. Поскольку канонизируемый граф - последовательно-параллельный, то в результате выполнения операций I и 2 исходный граф будет сведен или к простому циклу, или к K_2. При этом код каждого получившегося ребра будет содержать коды всех вершин и ребер, удаление которых в результате выполнения операций I и 2 привело к созданию этого ребра.

При канонизации графа, сведенного к K_2, надо определить, какая из двух вершин K_2 является начальной вершиной канонического представления графа и какая конечной, и определить порядок обхода всех вершин и ребер исходного графа. Для этого лексикографически

упорядочим коды параллельных ребер при движении от первой вершины ко второй, а затем в обратном направлении. Старший код будет кодом исходного графа, а порядок кодов вершин графа в каноническом коде будет каноническим представлением вершин исходного графа. Для канонизации графа, сведенного к циклу, определяются начальная вершина и направление обхода вершин цикла, которым соответствует максимальная кодовая запись, затем вершины и ребра цикла представляются в виде двух вершин (первая - начальная вершина, вторая - последняя при обходе вершин цикла), соединенных составным ребром, полученным из остальных вершин и ребер цикла. Код двухвершинного графа - канонический код исходного графа.

ТЕОРЕМА 3. Последовательное применение операций I и 2 позволяет получить каноническое представление последовательно-параллельного графа.

Теорема доказывается индуктивно по числу ребер графа.

При оценке сложности алгоритма канонизации последовательно-параллельных графов используются две леммы.

ЛЕММА 2. Длина кода помеченного последовательно-параллельного графа не превышает $4m - 2n + 3$.

Под сложностью алгоритма понимается число действий по порядку величины, необходимое для его завершения. Алгоритм имеет сложность $O(g(n))$, если для его выполнения требуется не более $Cg(n)$ действий, где C - константа. Алгоритм со сложностью $O(n)$ называется линейным.

ЛЕММА 3 (Ахо [5]). Пусть даны K цепочек длины l_i ($i = 1-k$). Элементы, составляющие цепочки, принадлежат алфавиту из P символов. Лексикографическое упорядочение цепочек выполняется за $O(P+l^*)$ действий, где $l^* = \sum_{i=1}^{k} l_i$.

СЛЕДСТВИЕ. Пусть цепочки упорядочены по длине $l_1 \geq l_2 \geq \ldots \geq l_k$. Лексикографическое упорядочение цепочек выполняется за $O(P+l_j)$ действий, где $l_j = \sum_{i=2}^{k} l_i$.

Из следствия вытекает, что длина самой длинной цепочки не влияет на сложность алгоритма.

В ряде приложений степени вершин графа ограничены заданным числом, в частности, степени вершин графов, соответствующих струк-

турным формулам химических соединений, не превышают 8. Будем предполагать, что степени вершин рассматриваемых последовательно-параллельных графов ограничены числом D.

ТЕОРЕМА 4. Сложность алгоритма канонизации последовательно-параллельных графов линейна по числу вершин графа.

ДОКАЗАТЕЛЬСТВО. Первоначальное представление графа - структура смежности: вершины графа пронумерованы и для каждой вершины указаны ее метка, номера смежных вершин и тип ребра, соединяющего данную вершину со смежной.

Легко показать, что для получения начальных списков вершин, соединенных параллельными ребрами, и вершин степени два требуется $O(n)$ действий.

Оценим сложность выполнения всех операций I. При выполнении этой операции осуществляются следующие процедуры:

а) определение очередных двух вершин, соединенных параллельными ребрами из списка вершин, соединенных параллельными ребрами;

б) уменьшение степеней вершин, соединенных параллельными ребрами, на число параллельных ребер минус один;

в) занесение номеров вершин, степень которых стала равной двум, в список вершин степени два;

г) получение кодов параллельных ребер из списка кодов инцидетных ребер одной из вершин, соединенных параллельными ребрами;

д) вычисление кода ребра, полученного в результате операции I.

Выполнение первых трех процедур "а"-"в" требует $O(1)$ действий, четвертая процедура "г" требует $O(d_j) \sim O(1)$ действий в случае ограниченных степеней вершин. Следовательно, общая сложность выполнения первых четырех процедур всех операций I не превосходит $O(1)$.

Оценим сложность выполнения процедуры "д".

Пусть длины кодов t параллельных ребер равны $l_1, l_2, \ldots, l_t$ и упорядочены так, что $l_1 \geq l_2 \geq \ldots \geq l_t$.

Согласно следствию леммы 3, для лексикографического упорядочения кодов t параллельных ребер требуется $O(p + \sum_{i=2}^{t} l_i)$ действий. Этот результат можно интерпретировать следующим образом: сложность добавления к произвольному коду $t-1$ кода меньшей длины и лексикографического упорядочения всех t кодов пропорциональна суммар-

ной длине добавляемых кодов. Поскольку длина кода графа пропорциональна числу вершин графа, то лексикографическое упорядочение всех параллельных ребер, образующихся в результате операций I, требует $O(n)$ действий.

Оценим суммарную сложность операций 2.

При выполнении этой операции осуществляются следующие процедуры:

а) определение по списку вершин степени два и структуре смежности номера первой вершины создаваемой внутренности цепочки;

б) добавление кода вершины степени два и кода инцидентного ей ребра к коду формируемой внутренности цепочки (добавляемая вершина смежна одной из текущих концевых вершин цепочки); если добавляемая вершина - первая вершина цепочки, то к коду цепочки кроме кода вершины добавляются два кода инцидентных ей ребер;

в) отметка в строке структуры смежности соответствующей номеру вершины, добавленной к внутренности цепочки, о ее удалении из списка вершин степени два;

г) проверка степеней вершин, смежных вершине, добавленной к внутренности цепочки (если степень хотя бы одной из них равна двум, то осуществляется продолжение формирования цепочки, иначе внутренность цепочки сформирована);

д) изменение строк структуры смежности, соответствующих концевым вершинам цепочки, степень которых больше двух (если ранее эти вершины не были смежными вершине, добавленной к внутренности цепочки, то теперь же они стали смежными вершинами; если ранее эти вершины были смежными, то теперь соединены параллельными ребрами).

Процедуры "а"-"г" требуют $O(I)$ действий. Процедура "д" требует $O(d_j) \sim O(1)$ действий. Для получения кода цепочки, состоящей из K вершин процедуры "б"-"г" выполняются K раз, а процедура "д" 2 раза. Поэтому сложность создания цепочки, состоящей из K вершин оценивается $O(K)$ действиями. А так как общее число вершин в графе равно n, то сложность создания цепочек составляет $O(n)$.

Однако если не учитывать ограниченности степеней вершин, процедура "д" может потребовать $O(n)$ действий, следовательно, общая сложность операций 2 может превысить $O(n)$.

В силу изложенного выше, общая сложность получения канонического кода графа оценивается $O(P+n)$ действиями, если исходный граф последовательным применением к нему операций I и 2 сведен к двум вершинам, соединенным параллельными ребрами.

Пусть операциями I и 2 граф сведен к простому циклу, содержащему R вершин. В работе Сисло [8] показано, что определение вершины цикла и направления обхода вершин и ребер цикла максимизирующие кодовую запись, требует O(R) действий. Следовательно, алгоритм канонизации двусвязного последовательно-параллельного графа имеет линейную сложность.

Рассмотрим произвольный последовательно-параллельный граф с N вершинами и M ребрами. Выделение двусвязных компонент, построение дерева блоков и точек сочленения и выделение корня дерева требует O(n) действий. Построим каноническое представление этого дерева. Затем канонизируем двусвязные компоненты, начиная с компонент, являющихся терминальными вершинами дерева блоков и точек сочленения. Для каждой двусвязной компоненты одна вершина, к которой будет стянута двусвязная компонента, - точка сочленения, отделяющая компоненту от корня дерева. Возможность такого сжатия вытекает из теоремы 2.

Двусвязная компонента, содержащая одно ребро, будет стянута к точке сочленения, отделяющей компоненту от корня. Двусвязная компонента, содержащая более одного ребра и не являющаяся корнем дерева блоков и точек сочленения, операциями I и 2 сводится к графу K_2 или простому циклу. Вторая вершина K_2 определяется в результате выполнения операций I и 2. Простой цикл сжимается к K_2. Для этого определяется направление обхода вершин и ребер цикла, максимизирующее кодовую запись цикла (начальной вершиной считается точка сочленения, отделяющая цикл от корня дерева блоков и точек сочленения). Затем по описанным выше правилам происходит сжатие простого цикла к K_2. Описанным способом осуществляется сжатие всех двусвязных компонент.

Коды двусвязных компонент, находящихся ближе к корню дерева, содержат коды компонент, находящихся дальше от корня и подчиненных им.

Легко видеть, что алгоритм канонизации произвольного последовательно-параллельного графа линеен по числу вершин.

Этим заканчивается доказательство теоремы.

Рассмотрим кодирование последовательно-параллельного графа, приведенного на рис.2.

Положим коды вершин: "C", "S", "N", коды связей: "-", "=", коды начала и конца параллельного соединения соответственно: "<",

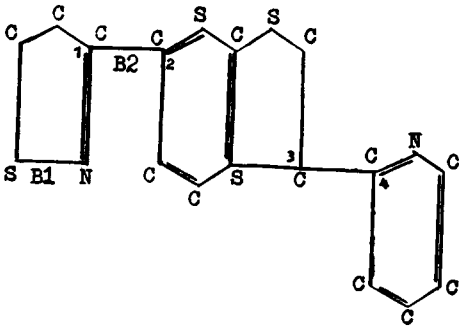


Рис. 2

" >", коды начала и конца двусвязной компоненты, входящей в состав старшей компоненты: "(","(", код разделителя параллельных ребер и двусвязных компонент, инцидентных одной точке сочленения: "!".

Расположим коды в порядке убывания старшинства: " <", "(","C", " N", "S", "=", "-", "!", ")", " >".

Для удобства обращения с элементами графа пронумеруем все вершины графа. Рассматриваемый в данном примере граф содержит 5 блоков и 4 точки сочленения - вершины с номерами I,2,3,4.

Код блока BI без учета кода вершины I, являющейся точкой сочленения, имеет вид: (=N-S-C!-)C.

Вхождение блока BI в блок B2 выглядит следующим образом: -C((=N-S-C!-)C).

Код всего графа, рассмотренного в данном примере, имеет вид:
C(=!-S=C((=N-S-C-!-)C))-C=C-!-S-C-C(-C((=N-C=C-C=!-)C))-)S.
B1
B2

§3. Алгоритм построения базисного набора циклов наименьшей длины последовательно-параллельного графа

При составлении номенклатурного названия молекулы и в задачах поиска соединений особую важность имеет определение базисного набора циклов наименьшей длины. Разработан ряд алгоритмов построения базисного набора циклов наименьшей длины: из полного набора циклов графа [7], эвристический алгоритм [8]. Указанные алгоритмы имеют нелинейную оценку сложности.

Опишем линейный алгоритм получения базисного набора циклов наименьшей длины последовательно-параллельного графа из его канонического кода. Очевидно, что базисный набор циклов графа - объединение базисных наборов циклов его двусвязных компонент. Поэтому рассмотрим построение базисного набора циклов наименьшей длины двусвязного последовательно-параллельного графа.

При построении циклов изменяется порядок следования параллельных ребер в коде графа: вначале идут параллельные ребра с меньшей длиной, а затем с большей. Такое переупорядочение выполняется со всеми параллельными ребрами, образующимися при построении канонического кода.

Опишем вычисление длины одного из параллельных ребер, соединяющих вершины, являющимися терминальными в коде графа. Для вычисления длины ребра проанализируем его код.

Выделим в коде участок, находящийся между символами "начало параллельного соединения" и "конец параллельного соединения", не содержащий внутри себя других таких символов. Этот участок соответствует коду ребра, полученного применением операции I к параллельным внутренностям цепочек, коды которых находятся между символами "начало параллельного соединения" и "конец параллельного соединения".

Длина ребра (длина участка кода) равна длине кратчайшей внутренности цепочки. Длина внутренности цепочки равна числу входящих в нее ребер. Таким образом вычисляются длины ребер, полученных объединением цепочек. Одновременно с вычислением длины ребра параллельные цепочки упорядочиваются по их длинам. Если цепочки имеют одинаковую длину, то старшей принимается лексикографически старшая цепочка.

Длина ребра, полученного применением операции I к параллельным цепочкам, равна длине кратчайшей параллельной цепочки.

Повторяя процедуру вычисления длины ребра, заменяющего параллельные цепочки, найдем длину параллельного ребра, соединяющего вершины, являющиеся терминальными в коде графа. Аналогично найдем длины всех параллельных ребер, соединяющих эти вершины.

Процедура вычисления по коду графа длин всех параллельных ребер и переупорядочивания ребер по длинам требует $O(n)$ действий.

Приведем процедуру построения независимых циклов.

Обозначим l_{min} длину кратчайшего параллельного ребра, соединяющего терминальные вершины в коде графа.

Если терминальные вершины кода соединены K параллельными ребрами, то построим $K-1$ независимых циклов минимальной длины, соединяющих эти вершины. Эти циклы получаются объединением ребра, длина которого l_{min}, с остальными $K-1$ ребрами. Очевидно, что полученные $K-1$ циклов образуют независимое множество циклов. Будем называть эти операции процедурой получения независимых циклов.

Рассмотрим код каждого из параллельных ребер, соединяющих терминальные вершины. Если часть какого-то ребра, полученного применением операции 1 к кодам параллельных ребер, - объединение параллельных цепочек, то применением процедуры получения независимых циклов к этим цепочкам расширим множество независимых циклов.

ЛЕММА 4. Процедура построения независимых циклов позволяет получить базисный набор независимых циклов.

Доказательство леммы индуктивное по числу применений процедуры.

Полученный базис циклов может не быть базисным набором циклов минимальной длины. Опишем алгоритм построения базисных циклов минимальной длины из полученного базисного набора циклов.

Рассмотрим параллельные ребра, соединяющие терминальные вершины в коде графа. Обозначим: R_0 - ребро, длина пути которого минимальна и равна l_{min}, R_1 - ребро, длина пути которого равна l_1. Цикл, образованный путями длины l_{min} и l_1, проходящими соответственно по ребрам R_0 и R_1, входит в базисный набор циклов минимальной длины.

Легко видеть, что все построенные базисные циклы, содержащие ребра пути длины l_{min}, входят в базисный набор циклов минимальной длины.

Рассмотрим Q независимых циклов, образованных Q+1 параллельными цепочками, объединение кодов которых входит в код ребра R_1. Пусть длина кратчайшей из Q+1 параллельных цепочек равна l_Q. Тогда $l_1 = l_Q + l_{1Q} + l_{2Q}$, где l_{1Q} - длина пути от первой терминальной вершины до цепочки, а l_{2Q} - длина пути от второй терминальной вершины.

Если $l_Q \le l_{min} + l_{1Q} + l_{2Q}$, то Q циклов входят в базис циклов минимальной длины. Если $l_Q > l_{min} + l_{1Q} + l_{2Q}$, то, заменив цепочку длиной l_Q цепочкой длины $l_{min} + l_{1Q} + l_{2Q}$, проходящей через терминальные вершины, получим Q циклов, входящих в базис циклов минимальной длины.

Проделав такую процедуру со всеми параллельными цепочками, построим базис циклов минимальной длины.

§4. Алгоритмы визуализации последовательно-параллельных графов по их каноническому коду

Задача визуализации структурных формул химических соединений может быть алгоритмически решена для тех соединений, структурные формулы которых являются последовательно-параллельными графами. Исходной информацией для построения изображения является канонический код графа. Изображение структурной формулы, полученное в результате визуализации, является каноническим, поэтому отпадает необходимость хранения в информационной системе изображений структур.

Алгоритм визуализации состоит из ряда процедур, которые по характеру выполняемых операций можно разделить на следующие блоки:

- анализа структуры и выделения компонент,
- укладки циклических компонент,
- изображения отдельных двусвязных компонент,
- обнаружения и устранения наложений,
- расположения графа на плоскости.

В блоке анализа и выделения компонент осуществляется определение участков канонического кода, соответствующих заданным подграфам исходного графа: циклическим, связывающим и деревоподобным компонентам. Циклическая компонента - двусвязная компонента, содержащая более одного ребра. Связывающая компонента - простая цепочка, соединяющая две циклические компоненты, циклическую и связывающую компоненты, две связывающие компоненты. Деревоподобная компонента - корневое дерево, корень которого - вершина циклической или связывающей компонент.

В блоке укладки циклических компонент определяется взаимное расположение их ребер и вершин. При этом происходит переупорядочение параллельных цепочек в порядке возрастания их длин.

После проведения укладки циклических компонент осуществляется каноническая визуализация графа. Визуализация выполняется последовательно, начиная с центральной компоненты - корня дерева блоков и точек сочленения.

Циклы длиной 3-7 имеют канонические изображения, соответствующие общепринятым в химии. Циклы большей длины строятся алгоритмически с учетом ранее изображенных смежных циклов. При изображении смежных циклов используются таблицы соответствия общих ребер в смежных циклах, позволяющие построить смежный цикл по ранее изображенному общему ребру.

Визуализация графа осуществляется на цифровом поле. Попытка изобразить два символа в одной позиции или в двух соседних клетках, если символы не являются изображением элемента таблицы Менделеева или частью изображения одной цепи, называется наложением. Для устранения наложения используется ряд процедур, позволяющих изменить изображение текущей компоненты или ранее изображенных компонент.

Если устранить наложение текущей компоненты не удается, то вместо нее ставится символ логической связи, а компонента изображается отдельно.

Подготовленное изображение структуры может быть выведено специальной процедурой на любое цифровое поле.

§5. Информационная система химической структурной информации

Описанные алгоритмы для последовательно-параллельных графов используются в информационной системе химической структурной информации. В системе используются базы данных, создаваемые системой управления базами данных ИНЭС.

Выделим основные функции системы:

- ввод и контроль исходной информации,
- каноническое представление структурных формул химических соединений,
- построение базисного набора циклов наименьшей длины,
- поиск структур химических соединений, обладающих заданными особенностями, в базе данных системы,
- визуализация структур химических соединений по их каноническому коду.

Исходная структурная информация представляется в виде структуры смежности. При вводе структур осуществляется жесткий синтаксический и семантический контроль. Структуры, прошедшие контроль, обрабатываются программами идентификации и канонизации последовательно-параллельных графов. Канонический код структуры вместе с присвоенным структуре номером государственной регистрации заносятся в базу данных. Применение алгоритма идентификации последовательно-параллельных графов к графу, не являющемуся последовательно-параллельным, позволяет свести его к графу меньшей размерности, степени всех вершин которого больше двух. Канонизация полученного графа осуществляется другими методами.

Программа построения базисного набора циклов наименьшей длины последовательно-параллельного графа позволяет, исходя из канонического кода графа, определить циклическую структуру соединения. Знание циклической структуры позволяет построить циклический указатель и осуществлять поисковые функции.

Программная реализация алгоритма визуализации позволяет получить на любом алфавитно-цифровом поле каноническое изображение структурных формул химических соединений, которым соответствуют последовательно-параллельные графы.

Для визуализации структур, не являющихся последовательно-параллельными графами, следует создать массив шаблонов-изображений структур, а при визуализации использовать объединение шаблонного и алгоритмического методов.

Авторы выражают искреннюю признательность И.А.Фараджеву за ценные замечания и плодотворное обсуждение.

Литература

1. DUFFIN R.J. Topology of series-parallel networks. - J. of Mathematical Analysis and Applications,1965,v.10,N 2,p.303-318.

2. ТРАХТЕНБРОТ Б.А. К теории бесповторных контактных схем.- Труды МИАН СССР, 1958, т.51, с. 226-269.

3. DIRAC G.A. A property of 4 chromatic graphs and some remarks on critical graphs.- J.London Math.Soc.,1952,v.27,p.85-92.

4. CHARTRAND G., KAUGARS A.,LICH D.K. Critically n-connected gpaphs.-Proc.of Amer.Math.Soc.,1972,v.32,N 1,p.63-68.

5. АХО А., ХОПКРОФТ Дж., УЛЬМАН Дж. Построение и анализ вычислительных алгоритмов. -М.: Мир, 1979. - 536 с.

6. SYSLO M. Linear time algorithm for coding outerplanar graphs. Beitrage zur Graphentheoric und deren Anwendungen Vorgetragen Auf dem Internationalen Kolloquium in Oberhof (GDR) 10-16. 04.1977, Ilmenau 1978,p.259-269.

7. Plotkin M. Mathematical Basis of ring-finding algorithms in CIDS. - J. of Chem.Docum.,1971,v.11,N 1,p.60-63.

8. GASTEIGER J., JOCHUM C. An algorithm for the perception of syntetically important rings. - J.Chem.Inf.and Comp.Sci.,1979, v.19,N 1,p.43-48.

Поступила ред.-изд.отд.
4 мая 1984 года