МАШИННЫЕ МЕТОДЫ ОБНАРУЖЕНИЯ ЗАКОНОМЕРНОСТЕЙ, АНАЛИЗА СТРУКТУР И ПРОЕКТИРОВАНИЯ
(Вычислительные системы)
1982 год Выпуск 92

УДК 519.1

ОБ ОДНОМ СЕМЕЙСТВЕ СХЕМ РЕКУРСИВНОГО РАЗБОРА ГРАФОВ

Ю.Е.Бессонов, В.А.Скоробогатов

Введение

Под рекурсивным разбором графа понимается следующая совокупность операций: исходному графу G по некоторому правилу ставится в соответствие определенный набор графов $G_1, G_2, \ldots, G_n$, каждый из которых проще исходного. Каждому из полученных графов по этому же правилу ставится в соответствие новый набор графов и т.д. Процесс продолжается до тех пор, пока не будут получены графы, к которым правило разбора окажется неприменимым. Отношение "G_i проще G" определяется в каждом конкретном случае специальным образом.

Понятие разбора графов в таком общем виде впервые ввел А.А.Зыков [1]. Известны работы [2-6], в которых различные конкретные операции разбора применяются для определения характеристик графа, либо его "особых" подмножеств, и работа [7], в которой исследуются особенности операций рекурсивного разбора в соответствии с различными характеристиками графов.

В работе [8] отношение "G_i проще G" уточняется как "G_i есть собственный подграф в G", на основе чего определяется схема рекурсивного разбора, которая обобщает конструкции, рассматриваемые в [3,4,6,7]. Для этой схемы определены в ряде конкретных случаев и исследованы характеристики графов, выражающие их сложность по отношению к разбору.

В настоящей работе отношение "G_i проще G" определено на основе операций относительного разбиения графов [9]. Впервые разбор графа на основе относительных разбиений был определен в работе [6]. Там же была предложена методика оценки трудоемкости разбора, развитая в [8] и применяемая в настоящей работе.

§I. Основные определения и теоремы

I.I. Относительные разбиения. Пояса.

Пусть $G = (V,E)$ - обыкновенный граф с множествами вершин V и ребер E. Число вершин и число ребер графа G будем обозначать соответственно через $p(G)$ и $q(G)$. Через $\Gamma_G(v)$ будем обозначать множество всех вершин графа G, смежных с вершиной v. В дальнейшем будем использовать терминологию из [10].

ОПРЕДЕЛЕНИЕ I. Разбиением связного графа G относительно вершины v (или относительным разбиением G по v [9]) называется следующая упорядоченная совокупность подмножеств $\hat{G}(v) = (\{V_0\}, V_1, V_2, \ldots, V_n)$, где $V_0 = \{v\}$, $V_i = \{u \mid d(u,v) = i\}$, $1 \le i \le n$, $d(u,v)$-расстояние между вершинами u и v в метрике графа G.

Подмножество $V_i = V_i(v)$ будем называть i-м с л о е м. Два слоя V' и V'' назовем смежными, если существуют $v' \in V'$ и $v'' \in V''$ такие, что $\{v', v''\} \in E$.

ОПРЕДЕЛЕНИЕ 2. Пусть $V_i(v), V_{i+1}(v), \ldots, V_{i+k-1}(v)$ - слои графа G, причем для каждого $j \in [i, i+k-2]$ слои $V_j(v)$ и $V_{j+1}(v)$ смежны. Подграф в G, порожденный множеством $\bigcup_{j=0}^{k-1} V_{i+j}$, назовем i-м k-слойным поясом (k-поясом) и обозначим через $W_{i,k}(G,v)$; 2-слойный пояс назовем просто поясом и обозначим через $W_i(G,v)$.

Отметим, что пояса определяются только для компоненты графа, содержащей вершину v, относительно которой строится разбиение. Пусть G состоит из компонент $G_1, G_2, \ldots, G_i$ и его разбиение на k-пояса есть $\{W_{0,k}(G_1,v), W_{1,k}(G_1,v), \ldots, W_{n_1,k}(G_1,v)\}$, $n_1 = n(G_1,v) - k + 1$. Условимся, что разбиение несвязного графа G относительно вершины v есть $\{W_{0,k}(G_1,v), \ldots, W_{n_1,k}(G_1,v), \bigcup_{j=2}^{i} G_j\}$,

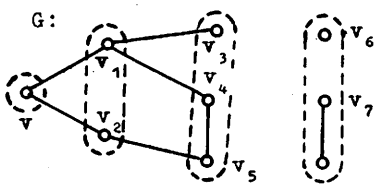

Рис. I

т.е. последним поясом в относительном разбиении несвязного графа относительно вершины v будем считать подграф в G, порожденный всеми вершинами, не связанными с вершиной v.

Поясами в разбиении, приведенном на рис.I, являются порожденные подграфы*) графа G:

*) $\langle X \rangle$ означает подграф в G, порожденный множеством вершин X.

$W_0(v) = \langle\{v, v_1, v_2\}\rangle$, $W_1(v) = \langle\{v_1, v_2, v_3, v_4, v_5\}\rangle$, $W_2(v) = \langle\{v_6, v_7, v_8\}\rangle$, $W_{0,3}(v) = \langle\{v, v_1, v_2, v_3, v_4, v_5\}\rangle$.

ОПРЕДЕЛЕНИЕ 3. Длиной относительного разбиения назовем величину $n = n(G, v)$, равную числу слоев в этом разбиении без единицы.

Очевидно, что $\max_{v \in V(G)} n(G, v) = d(G)$, где $d(G)$ есть диаметр графа G.

ОПРЕДЕЛЕНИЕ 4. Пусть $l = l(G, v)$ – некоторая целочисленная функция (возможно, константа), значение которой не превосходит $n(G, v)$. Неполным относительным разбиением назовем разбиение $\hat{G}(v, l) = (V_0, V_1, \ldots, V_l)$, где $V_i = \{u \mid d(u, v) = i\}$, $1 \le i \le l(G, v) - 1$, а $V_l = \{u \mid d(u, v) \ge l(G, v)\}$.

При $l = n(G, v)$, очевидно, имеет место $\hat{G}(v, n(G, v)) = \hat{G}(v)$ – относительное разбиение в смысле определения I.

2.2. Рекурсивный разбор.

Под рекурсивным разбором графа будем понимать совокупность правил, по которым исходный граф разбивается на k-пояса, затем каждый полученный k-пояс, в свою очередь, разбивается на k-пояса и т.д.

Для пометки графов, получающихся в результате разбора, удобно использовать многокомпонентные индексы вида $s = i_1 i_2 \ldots i_t$, где $i_j \ge 0$ – целые числа, $1 \le j \le t$. Исходный граф G пометим индексом $s = 0$. Пояса $W_{j,k}(G_s, v)$ графа G_s (где $s = i_1 i_2 \ldots i_t$) обозначим через G_{sj}, где $sj \equiv i_1 i_2 \ldots i_t j$. С учетом введенных обозначений процесс разбора можно представить в виде дерева, изображенного на рис.2.

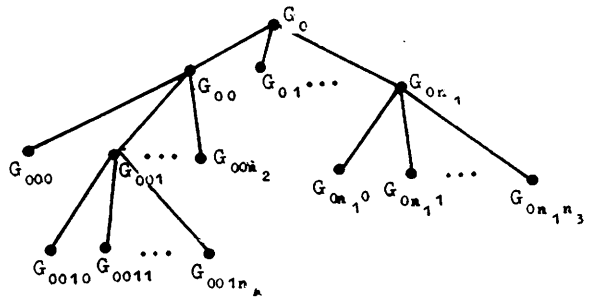

Рис. 2

При $t > r$ индекс, полученный из $s = i_1 i_2 \ldots i_t$ удалением последних r компонент, обозначим через s_{-r}. Тогда, очевидно, граф G_s является поясом графа $G_{s_{-1}}$, который, в свою очередь, является поясом графа $G_{s_{-2}}$ и т.д.

Определим семейство алгоритмов, имеющих сходную структуру. Каждый алгоритм из семейства определяется конкретным заданием правил:

1) остановки О, которое определяет допустимость дальнейшего разбора графа G_s;

2) выбора В, определяющего вершину v_s в графе G_s, относительно которой будет строиться разбиение;

3) накопления Н, которое производит анализ результатов разбора и накопление их в некотором носителе информации BUF;

4) заданием функции $l = l(G, v)$ (возможно, константы), определяющей число слоев в относительном разбиении, и числа $k \geq 2$, определяющего величину поясов, на которые разбивается граф G_s.

Каждый алгоритм из семейства будем называть с х е м о й р е к у р с и в н о г о р а з б о р а графа и обозначать через РАЗБОР (k,l,O,B,H). Исходной информацией для алгоритма является граф G, а результирующей - содержимое накопителя BUF, который при реализации алгоритма на ЭВМ может быть либо областью оперативной памяти, либо внешним устройством хранения данных. Правила О,В и Н будем формулировать как алгоритмы. Для описания алгоритмов будем использовать простой язык, основные конструкции которого приведены ниже.

1) НАЧАЛО АЛГОРИТМА, КОНЕЦ АЛГОРИТМА применяется для обозначения начала и конца описания алгоритма.

2) НАЧАЛО A_1; A_2; ... A_n ; КОНЕЦ применяется для объединения операторов $A_1, \ldots, A_n$ в блок, который считается одним оператором.

3) Оператор присвоения a:=b означает, что выражение b присваивается переменной a; значением b могут быть число, символ, множество, граф и т.д.

4) Любой оператор может быть помечен меткой m, которая обозначает целое число. Метка будет использоваться для передачи управления при помощи оператора ПЕРЕЙТИ К m, а также для ссылок на соответствующие фрагменты алгоритма.

5) ЕСЛИ Y ТО A_1 ИНАЧЕ A_2, где Y - условие, при выполнении которого управление передается оператору A_1, в противном случае оператору A_2. Отметим, что часть ИНАЧЕ A_2 может быть опущена.

6) ПОКА Y ЦИКЛ A КОНЕЦ ЦИКЛА используется для организации цикла. Выполнение оператора A повторяется до тех пор, пока справедливо условие Y .

7) Комментарии заключаются в скобки /* ... */ и могут быть вставлены в любое место описания алгоритма.

А л г о р и т м РАЗБОР (k,l,O,B,H)

НАЧАЛО АЛГОРИТМА

I. Положить s:=0; G_s:=G; BUF:=∅

2. Выполнить алгоритм O; ЕСЛИ правило остановки не выполнено ТО

НАЧАЛО /* здесь при помощи алгоритма B происходит выбор вершины v_s, относительно которой будут построены разбиения графа G_s на k-пояса */.

3. Выполнить алгоритм B и его результат присвоить переменной v_s ; j_s:=0; /*j_s является счетчиком поясов в относительном разбиении графа G_s */

КОНЕЦ

ИНАЧЕ

НАЧАЛО /* здесь правило остановки выполнено и разбор графа G_s прекращается, при этом происходит анализ и накопление результатов разбора в накопителе BUF при помощи алгоритма H */

4. Выполнить алгоритм H

5. ЕСЛИ s = 0 ТО ПЕРЕЙТИ К 8; $s:=s_{-1}$: $j_s:=j_s+1$ ЕСЛИ $j_s > l(G_s, v_s)-k+1$ /* это условие означает, что разбор всех поясов графа G_s закончен */ ТО ПЕРЕЙТИ К 5 ИНАЧЕ ПЕРЕЙТИ К 7

КОНЕЦ

6. Построить относительное разбиение $\hat{G}_s(v_s,l)$

7. Положить $G_{s\,j_s}:=W_{j_s,k}(G_s,v_s)$; $s:=sj_s$ ПЕРЕЙТИ К 2

8. КОНЕЦ АЛГОРИТМА

Результат применения правила выбора B может зависеть от нумерации вершин графа G,и существуют различные варианты разбора графа G при заданных k,l,O,B и H. Эти варианты будем называть р е а л и з а ц и я м и схемы.

I.3. Примеры схем разбора.

Различные варианты правил O,B и H будем отмечать индексами.

Рассмотрим схему РАЗБОР $(2,2, O_1, B_1, H_1)$. Сформулируем правила, управляющие разбором.

А л г о р и т м O_1 (правило остановки)

НАЧАЛО АЛГОРИТМА

ЕСЛИ G_s - полный подграф в G ТО правило остановки выполнено

ИНАЧЕ правило остановки не выполнено

КОНЕЦ АЛГОРИТМА

А л г о р и т м B_1 (правило выбора)

НАЧАЛО АЛГОРИТМА

Выбрать в $V(G_s)$ вершину v_s с наименьшим номером и не смежную хотя бы с одной вершиной из $V(G_s)\setminus\{v_s\}$

КОНЕЦ АЛГОРИТМА

А л г о р и т м H_1 (правило накопления)

НАЧАЛО АЛГОРИТМА

Занести в BUF множество вершин графа G_s

КОНЕЦ АЛГОРИТМА

В соответствии с данной схемой разбор осуществляется следующим образом. Исходный граф $G=G_0$ разбивается на два подграфа: $G_{00} = W_{1,2}(G_0, v_0) = \langle \Gamma_{G_0}(v_0) \cup \{v_0\} \rangle$ и $G_{01} = W_{2,2}(G_0, v_0) = G_0 - v_0$ относительно вершины v_0, не смежной с какой-нибудь другой вершиной графа G_0. Затем каждый из этих графов подвергается аналогичному разбиению и т.д. Полные подграфы G_s не разбиваются. В результате разбора получается некоторое множество полных подграфов графа G. Выбор вершины, не смежной хотя бы с одной из вершин графа G_s, исключает возможность бесконечного продолжения процесса разбора. Действительно, если v_s - такая вершина, то $G_{s0}=G_s$, $G_{s1} = G_s$, в графах G_{s0} и G_{s1} снова может быть выбрана вершина v_s и ситуация повторяется.

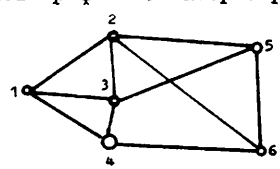

Рис. 3

Протокол работы алгоритма РАЗБОР $(2, 2, O_1, B_1, H_1)$ для графа G, изображенного на рис.3, приводится в табл.I*).

*) Все таблицы приведены в приложении.

Схема РАЗБОР $(2,2,O_1, B_2, H_1)$ отличается от предыдущей только правилом выбора вершины.

А л г о р и т м B_2 (правило выбора)

НАЧАЛО АЛГОРИТМА

Выбрать вершину v_s с наименьшим номером среди всех вершин в G_s, имеющих минимальную степень

КОНЕЦ АЛГОРИТМА

Здесь минимальность степени вершины v_s исключает возможность бесконечного продолжения разбора. Действительно, если G_s является нулевым поясом в неполном разбиении графа G_{s-1} относительно v_{s-1}, то v_{s-1} смежна со всеми вершинами в G_s. Если окажется, что v_{s-1} имеет минимальную степень в G_s, то G_s будет полным графом. Но тогда по правилу O_1 прекратится дальнейший разбор. На рис.4 изображен пример реализации данной схемы.

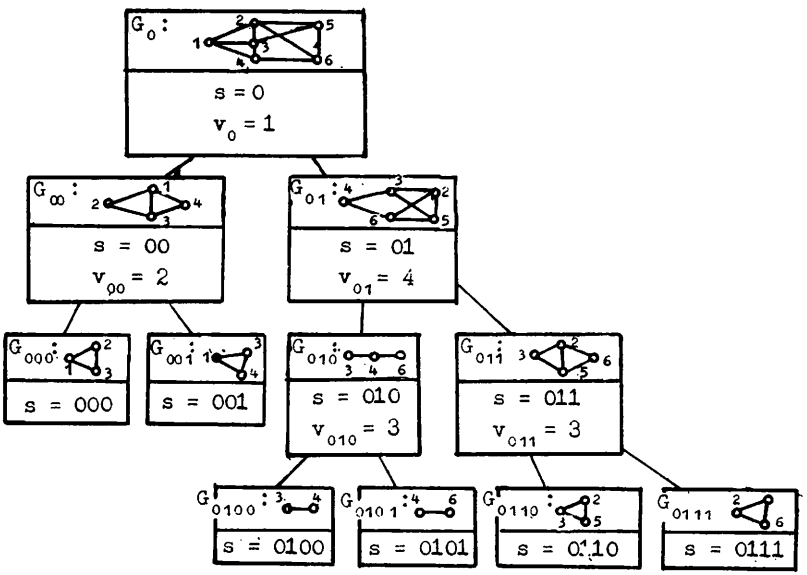

Рис. 4

Схема РАЗБОР $(2,n,O_1,B_2,H_1)$ отличается от предыдущей только тем, что здесь к графам G_s применяется полное относительное разбиение. Данная схема положена в основу алгоритма поиска клик в графах [6].

I.4. Характеристики разбора.

Возьмем некоторую реализацию схемы рекурсивного разбора и определим для нее корневое дерево $T = T(G)$ следующим образом. Исходному графу G_0 поставим в соответствие корневую вершину w_0 дерева $T(G)$. Если правило остановки O не выполняется, то графам $G_{01},G_{02},\ldots,G_{0j_0}$, являющимся k-поясами в разбиении $\hat{G}(v_0,1)$, поставим в соответствие вершины $w_{01},w_{02},\ldots,w_{0j_0}$ дерева $T(G)$, смежные с вершиной w_0. Пусть теперь w_s – некоторая вершина дерева $T(G)$, которой поставлен в соответствие граф G_s. Если правило O выполняется, то w_s – висячая вершина, в противном случае ей смежны вершины $w_{s1},w_{s2},\ldots,w_{sj_s}$, которым соответствуют графы G_{s1}, $G_{s2},\ldots,G_{sj_s}$, являющиеся k-поясами в разбиении $\hat{G}_s(v_s,1)$. Соответствие между графами G_s и вершинами w_s дерева T иногда будем записывать в виде $G_s = h(w_s)$.

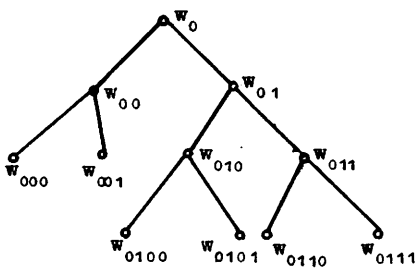

Рис. 5

Дерево T, определенное выше, назовем д е р е в о м р а з б о р а графа G по отношению к схеме РАЗБОР (k,l,O,B,H). Условимся при изображении дерева T располагать корневую вершину вверху, а последователи каждой вершины w_s слева направо в порядке, обусловленном последовательностью слоев в относительном разбиении графа G_s (см. рис.5).

Пусть $V_t(T)$ обозначает множество висячих вершин дерева T. Рассмотрим рекурсивный разбор с точки зрения алгоритмической сложности. Для этого оценим сверху число операций, необходимое для реализации схемы рекурсивного разбора, используя в качестве модели вычислительного процесса дерево разбора. Обозначим через $\theta_s(O)$, $\theta_s(B)$ и $\theta_s(H)$ соответственно числа операций, необходимых для выполнения алгоритмов O,B и H для графа G_s. Тогда, очевидно, в худ-

шем случае число операций, соответствующее вершине w_s дерева разбора, не превосходит величины $c_1 + c_2 \cdot p^2(G) + \theta_s(O) + \theta_s(B) + \theta_s(H)$, где c_i - числовые константы, $i = 0,1,\ldots$; $c_2 \cdot p^2(G)$ - ограничение сверху на число операций, необходимое для построения относительного разбиения. Следовательно, число операций при выполнении всего алгоритма будет ограничено сверху величиной

$$\sum_{w_s \in V(T)} (c_1 + c_2 \cdot p^2(G) + \theta_s(O) + \theta_s(B) + \theta_s(H)).$$

Обозначим через $\mathcal{T} = \mathcal{T}(k,1,O,B,H,G)$ множество всех деревьев разбора графа G по отношению к схеме РАЗБОР $(k,1,O,B,H)$.

ОПРЕДЕЛЕНИЕ 5. Функцией сложности графа G по отношению к схеме РАЗБОР $(k,1,O,B,H)$ называется величина

$$\xi = \xi(k,1,O,B,H,G) =$$

$$= \max_{T \in \mathcal{T}} \sum_{w_s \in V(T)} (c_1 + c_2 \cdot p^2(G) + \theta_s(O) + \theta_s(B) + \theta_s(H)).$$

Наибольшее число вершин дерева разбора среди всех деревьев из $\mathcal{T}$ обозначим через $\eta = \eta(k,1,O,B,H,G) \equiv \max_{T \in \mathcal{T}} p(T)$.

I.5. Свойства схем разбора.

Сформулируем ряд утверждений относительно свойств схем РАЗБОР $(2,2,O_1,B_1,H_1)$ и РАЗБОР $(2,2,O_1,B_2,H_1)$. Утверждение приводится без доказательства, если оно или аналогичное ему доказано в [8].

Обозначим через M(G) множество всех клик графа G.

ТЕОРЕМА I. Для схем РАЗБОР $(2,2,O_1,B_1,H_1)$ и РАЗБОР $(2,2,O_1,B_2,H_1)$ имеет место $h(V_t(T)) \supseteq M(G)$.

На основе этой теоремы в §2 будут построены алгоритмы поиска клик в графах.

ЗАМЕЧАНИЕ I. Пример на рис.4 показывает, что в общем случае $h(V_t(T)) \neq M(G)$, поскольку некоторые висячие вершины дерева разбора соответствуют подграфам клик. Следовательно, при построении алгоритма поиска клик необходимо дополнить правило O_1 для того, чтобы избежать "дробления" клик, приводящего к снижению эффективности алгоритма. Такое дополненное правило будет сформулировано в §2. Можно показать, что равенство $h(V_t(T)) = M(G)$ выполняется, например, для графов $K_{n_1,n_2,\ldots,n_t}$ (при наличии правила O_1).

Приведем теперь теорему, которая связывает функцию сложности графа с наибольшим числом вершин дерева разбора для схем РАЗБОР $(2,2,O_1,B_1,H_1)$ и РАЗБОР $(2,2,O_1,B_2,H_1)$.

ТЕОРЕМА 2. Справедливо соотношение $\xi \le O(p^2(G)\cdot\eta)$.

Эта теорема показывает, что для оценки асимптотики функции сложности графа достаточно знать асимптотику величины η.

ТЕОРЕМА 3. Если $G' \subseteq G$ *), то $\eta(2,2,O_1,B_1,H_1,G') \le \eta(2,2,O_1,B_1,H_1,G)$ и $\eta(2,2,O_1,B_2,H_1,G') \le \eta(2,2,O_1,B_2,H_1,G)$.

СЛЕДСТВИЕ. Пусть $\beta_0(G)$ обозначает вершинное число независимости графа G. Поскольку $\eta(2,2,O_1,B_1,H_1,\vec{K}_n) = 2n-1$, то из теоремы 3 следует $\beta_0(G) \le \frac{1}{2}(\eta(2,2,O_1,B_1,H_1,G)+1)$. Такое же соотношение справедливо и при $B = B_2$.

Пусть $\rho = (\rho_1(G),\rho_2(G),\dots,\rho_t(G))$ - некоторый набор числовых характеристик графа G (например, $p(G)$, $q(G)$, $\beta_0(G)$ и т.п.) и $a = (a_1,a_2,\dots,a_t)$ - некоторый набор натуральных чисел. Обозначим через $\mathcal{G}_\rho(a)$ множество всех графов G таких, что $\rho_i(G) \le a_i$, $i \in [1,t]$.

Обозначим через $V^*(G)$ множество всех вершин графа G, имеющих степень не более $p(G) - 2$.

ОПРЕДЕЛЕНИЕ 6. Функцию $\rho_i(G)$ будем называть монотонной, если $\rho_i(G') \le \rho_i(G)$ для любого $G' \subseteq \langle V^*(G)\rangle$, и строго монотонной, если $\rho_i(G') < \rho_i(G)$ для любого $G' \subset \langle V^*(G)\rangle$.

Например, функция $m(G)$ - плотность графа (т.е. величина наибольшей клики в G) монотонна, и функция $p(G)$ строго монотонна.

Пусть $\rho_i(G)$ - некоторая характеристика графа G. Введем функцию ρ_i^* как сужение функции ρ_i на граф $\langle V^*(G)\rangle$, т.е. $\rho_i^*(G) = \rho_i(\langle V^*(G)\rangle)$.

Пусть $G \in \mathcal{G}_\rho(a)$. Введем функции $a_i' = a_i'(a_i)$, $a_i'' = a''(a_i)$ посредством соотношений

$$a_i' = \max_{G \in \mathcal{G}_{\rho_i}(a_i)} \max_{v \in V^*(G)} \rho_i(\langle \Gamma_G(v) \cup \{v\}\rangle), \qquad (1)$$

*) Запись $G' \subseteq G$ означает, что G' является подграфом в G; если G' - собственный подграф, то это обозначается как $G' \subset G$.

$$a_i'' = \max_{G \in \mathcal{G}_{\rho_i}(a_i)} \max_{v \in V(G)} \rho_i(G-v). \qquad (2)$$

Введем два функционала:

$$N_\rho(a) = N_\rho(k,l,O,B,H,a) = \max_{G \in \mathcal{G}_\rho(a)} \eta(k,l,O,B,H,G),$$

$$X_\rho(a) = X_\rho(k,l,O,B,H,a) = \max_{G \in \mathcal{G}_\rho(a)} \xi(k,l,O,B,H,G).$$

При фиксированном наборе ρ их можно рассматривать как функции от a. Из теоремы 2 следует, что $X_\rho(a) \le c_0 \cdot p^2(G) \cdot N_\rho(a)$ для $O = O_1$, $H = H_1$ и $B = B_1$ или $B = B_2$, $k = 2$ и $l = 2$. Таким образом, для исследования асимптотики максимума функции сложности графа по классу $\mathcal{G}_\rho(a)$ достаточно исследовать асимптотику $N_\rho(a)$.

ЛЕММА I. Пусть $\rho = (\rho_1, \rho_2, \ldots, \rho_t)$ - набор монотонных функций, причем хотя бы одна функция строго монотонна. Тогда справедливо неравенство $N_\rho(2,2,O_1,B,H_1,a) \le \phi(a)$, где $B = B_1$ или $B = B_2$, а $\phi(a) = \phi(a_1, a_2, \ldots, a_t)$ - функция, удовлетворяющая рекуррентному соотношению

$$\phi(a_1,a_2,\ldots,a_t) = \phi(a_1',a_2',\ldots,a_t') + \phi(a_1'',a_2'',\ldots,a_t'') + 1 \qquad (3)$$

с граничными условиями: $\phi(a_1,a_2,\ldots,a_t) = 1$, если хотя бы одна компонента $a_i \le 0$, $i \in [1,t]$.

ДОКАЗАТЕЛЬСТВО. Построим дерево $\tilde{T}$, которое заведомо покрывает любое дерево разбора графа из класса $\mathcal{G}_\rho(a)$, причем число вершин этого дерева будет удовлетворять соотношению (3). Набору $(a_1,a_2,\ldots,a_t)$ поставим в соответствие корневую вершину дерева $\tilde{T}$. Набору $(a_1',a_2',\ldots,a_t')$ поставим в соответствие левый последователь корневой вершины, а набору $(a_1'',a_2'',\ldots,a_t'')$ - правый. Если в каком-либо наборе встретится число, меньшее или равное нулю, то вершину дерева $\tilde{T}$ объявим висячей. Далее, если все числа в a' больше нуля, то соответствующая вершина будет иметь последователи, которым будут приписаны наборы $((a_1')',(a_2')',\ldots,(a_t')')$ и $((a_1')'',(a_2')'',\ldots,(a_t')'')$. Точно так же поступим с вершиной $\tilde{T}$, которой приписан набор a'', и т.д. (см. рис. 6). Из (I) и (2) следует, что $a_i' \le a_i$ и $a_i'' \le a_i$, так как функции ρ_i монотонны. Кроме того,

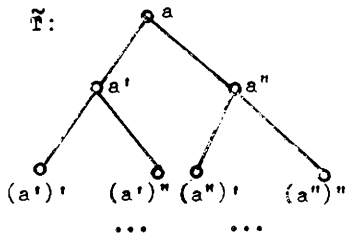

Рис. 6

поскольку в наборе существуют строго монотонные функции, то некоторые неравенства будут строгими. Это означает, что дерево $\tilde{T}$ конечно, причем число вершин его удовлетворяет соотношению (3). Рассмотрим теперь произвольное дерево разбора $T(G)$ графа G из класса $\mathcal{Y}_\rho(a)$. Тогда $G_0 \in \mathcal{Y}_\rho(a)$, а любой граф $\langle\{v\} \cup \Gamma_G(v)\rangle \in \mathcal{Y}_\rho(a')$ и любой граф $G-v \in \mathcal{Y}_\rho(a'')$. Следовательно, любое дерево $T(G)$ изоморфно входит в дерево $\tilde{T}$. Лемма доказана.

Опираясь на лемму I, сформулируем следующую методику получения верхних оценок функции сложности графа, зависящих от параметров графа.

I. Выбираем некоторый конкретный набор $\rho = (\rho_1, \rho_2, \ldots, \rho_t)$ параметров графа; в соответствии с (I) и (2) находим функциональные зависимости: $a'_i = a'_i(a_i)$, $a''_i = a''_i(a_i)$.

2. Оцениваем функцию $\phi(a)$, удовлетворяющую соотношению (3).

Приведем некоторые аналитические оценки для функции сложности графа, зависящие от определенных параметров.

ТЕОРЕМА 4. Пусть $\rho(G) = (p^*(G))$. Тогда справедливо неравенство $X_\rho(a_1) \le O(a_1^2 \cdot 1{,}62^{a_1})$.

ЗАМЕЧАНИЕ 2. По теореме I, число висячих вершин дерева $T(G)$ больше либо равно числу клик в G, причем неравенство объясняется наличием подграфов клик, соответствующих некоторым висячим вершинам $T(G)$. Известно [II], что наибольшее число клик в графе из класса $\mathcal{Y}_\rho(a_1)$ имеет порядок $O(I{,}44^{a_1})$. Теорема 4, таким образом, показывает, что доля подграфов клик может быть весьма значительной. Полученная в теореме 4 оценка является точной. Можно доказать, что графами, на которых она достигается, являются дополнения простых цепей.

ТЕОРЕМА 5. Пусть $\rho = (p^*(G), m^*(G))$. Тогда $X_\rho(a) \le O(a_1^{a_2+2})$.

Отметим, что при $a_2 \ll a_1$ оценка, полученная в теореме 5, является достаточно хорошей. Однако если a_1 фиксировано, а a_2 растет, то эта оценка становится все более завышенной.

ЛЕММА 2. Пусть $f(t)$ - функция, определенная на подмножестве натуральных чисел $t \in [k,\infty)$ при помощи рекуррентного соотношения $f(t) = f(t-k)+f(t-1)+1$. Тогда $f(t) = O(x_0^t)$, где x_0 - положительный корень уравнения $x^k - x^{k-1} - 1 = 0$.

ТЕОРЕМА 6. Пусть $\rho = (p^*,m^*)$. Тогда $N_\rho(2, 2,O_1,B_2,H_1,a) \le O(a_1^2 \cdot x_0^n)$, где x_0 - положительный корень уравнения $x^n - x^{n-1} - 1 = 0$ $\left(n = \frac{a_1}{a_2}\right)$.

Данная оценка является более точной, чем оценка из теоремы 5

ТЕОРЕМА 7. Пусть $\rho = (p^*,m^*,q^*, l_0^*)$, где $l_0 = l_0(G)$ - величина наименьшей клики графа G. Тогда $N_\rho(a) \le \phi(a) = \phi\left(\left[\frac{a_2-1}{a_2}a_1\right], a_2 - 1, a_3 - 2(a_4-1), a_4 - 1\right) + \phi(a_1 - 1, a_2, a_3 - a_4 + 1, a_4) + 1$.

Эта теорема может быть использована при численных расчетах функции сложности графа.

§2. Алгоритмы

2.I. Алгоритм поиска всех клик в графах

Рассматриваемый алгоритм осуществляет разбиение исходного графа $G=G_0$ на два подграфа $\langle\{v_0\} \cup \Gamma_G(v_0)\rangle$ и $G_0 - v_0$ относительно вершины v_0 с минимальной степенью в G_0. Каждый из полученных подграфов подвергается аналогичному разбиению и т.д. Этот алгоритм в терминах §I представим схемой РАЗБОР $(2,2,O_2,B_2,H_2)$ с правилами остановки O_2, выбора B_2 и накопления H_2. Правило B_2 состоит в выборе вершины с минимальной степенью.

Рассмотрим правило остановки O_2. Алгоритм O_2 определяет, подлежит ли дальнейшему разбору рассматриваемый в данный момент граф G_s. Граф G_s не подлежит дальнейшему разбору в двух случаях:

I) когда G_s есть подграф некоторого графа $G_{s'}$, разбор которого уже закончен, т.е. когда любая клика в G_s является частью некоторой клики графа $G_{s'}$;

2) когда разность между порядком графа и минимальной степенью*) его вершин меньше четырех: $\delta(G_s) > p(G_s) - 4$. В этом случае граф G_s имеет достаточно простую структуру для того, чтобы можно было перечислить его клики, не прибегая к разбору. Действительно, максимальная степень дополнения графа G_s не превосходит двух, следовательно, G_s есть в общем случае набор компонент, являющихся либо изолированными вершинами, либо простыми цепями, либо простыми циклами.

Алгоритм O_2 (правило остановки)

НАЧАЛО АЛГОРИТМА /* исходной информацией для алгоритма является граф G_s и множество графов $\{G_{s'} | s' = s_{-r}0, r = 2,3, \ldots\}$, разбор которых уже закончен; выходной информацией является утверждение о допустимости или недопустимости разбора графа G_s */.

1. Положить s':=s
2. ПОКА $s' \neq 00$
ЦИКЛ
3. Присвоить переменной j значение последней компоненты s'
4. Положить $s' := s'_{-1}$
5. ЕСЛИ $j \neq 0$ ТО
НАЧАЛО
6. ЕСЛИ $V(G_s) \subseteq V(G_{s'0})$ ТО правило остановки выполнено /* разбор графа $G_{s'0}$ уже закончен, поэтому если G_s является подграфом в $G_{s'0}$, то любая клика графа G_s есть часть клики в $G_{s'0}$ */
ПЕРЕЙТИ К 8
КОНЕЦ
КОНЕЦ ЦИКЛА
7. ЕСЛИ $\delta(G_s) \leq p(G_s) - 4$ ТО правило остановки не выполнено
ИНАЧЕ правило остановки выполнено
8. КОНЕЦ АЛГОРИТМА

ПРИМЕР I. На рис. 7 изображен разбор графа. Пусть процесс разбора дошел до графа G_{010}. Он не является подграфом в G_{00}, разбор которого к настоящему моменту уже закончен. Однако $\delta(G_{010}) > p(G_{010}) - 4$, поэтому выполняется правило остановки. Пусть теперь

*) Здесь $\delta(G_s)$ обозначает минимальную степень вершин графа G_s.

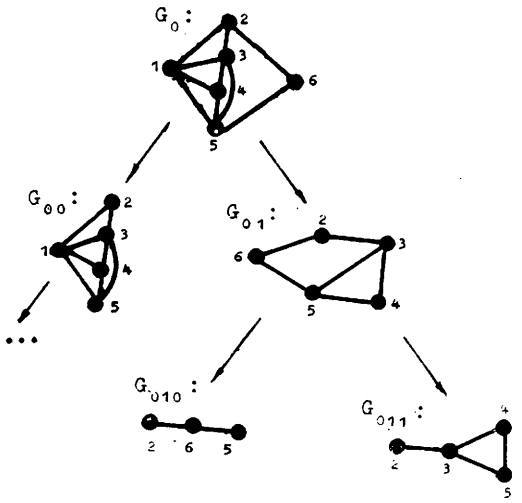

Рис. 7

процесс разбора дошел до графа G_{011}. В табл. 2 приведен протокол выполнения алгоритма O_2 для этого графа.

Рассмотрим правило накопления. Алгоритм H_2 накапливает в лексикографическом порядке клики графа G_s, у которого минимальная степень $\delta(G_s) > p(G_s) - 4$. Каждая клика проверяется на вхождение в графы, разбор которых уже закончен. Если вхождения нет, то клика заносится в BUF. При этом на генерацию каждой клики тратится число операций, пропорциональное ее порядку. Работа алгоритма основана на следующих соображениях.

1) Множество вершин, образующих клику в G_s, является максимальным независимым множеством в $\bar{G}_s$, которое в дальнейшем для краткости будем называть антикликой.

2) Каждая антиклика в $\bar{G}_s$ есть объединение некоторых антиклик, взятых по одной из каждой компоненты $\bar{G}_s$.

3) Пусть некоторая компонента $\bar{G}_s$ есть простая цепь $v_0, v_1, v_2, \ldots, v_t$. Очевидно, что между двумя соседними вершинами цепи, входящими в некоторую антиклику, есть либо одна, либо две вершины цепи. Следовательно, каждую антиклику в простой цепи можно представить упорядоченным набором чисел $k_1, k_2, \ldots, k_r$, каждое из которых равно двум или трем и выражает расстояние между соседними по цепи вершинами, входящими в антиклику. Любая антиклика содержит либо вершину v_0, либо вершину v_1, следовательно, $Q = \{v_0, v_{k_1}, v_{k_1+k_2}, \ldots\}$ либо $Q = \{v_1, v_{k_1+1}, v_{k_1+k_2+1}, \ldots\}$.

4) Упорядоченность антиклик в простой цепи определяется упорядоченностью наборов $k_1, k_2, \ldots, k_r$. Набор, состоящий только из двоек, будем считать наименьшим. Из двух наборов, содержащих разное количество троек, наименьшим будем считать набор, содержащий меньшее количество троек. Для двух наборов, содержащих одинаковое количество троек, очередность определим следующим образом. Любой набор представим в виде: $(r-r_3-r_2-1)$ произвольных (двоек или троек) чисел; r_2 двоек; r_3 троек. Следующим по порядку будем считать набор, имеющий вид: $(r-r_3-r_2-1)$ произвольных (двоек, троек) чисел; тройка; (r_2+1) двоек; (r_3-1) троек. Например, следующим за набором 3 2 3 2 2 3 3 будет набор 3 2 3 3 2 2 2 3.

5) Если компонента $\tilde{G}_s$ является простым циклом $v_0, v_1, v_2, \ldots, v_t$, $\{v_0, v_t\} \in E$, то каждая его антиклика либо содержится в графе $\langle\{v_1, v_2, \ldots v_t\}\rangle$, либо в графе $\langle\{v_0\}\rangle \cup \langle\{v_2, v_3, \ldots, v_{t-1}\}\rangle$. Следовательно, перечисление антиклик в простом цикле сводится к перечислению антиклик в простых цепях.

Алгоритм H_2 (правило накопления)

НАЧАЛО АЛГОРИТМА /* исходной информацией для алгоритма является граф G_s и множество $\{G_{s'} \mid s' = s_{-r}0,\ r = 2,3,\ldots\}$ графов, разбор которых уже закончен, выходной информацией является множество клик графа G_s, которые являются кликами графа G (если такие существуют) */

1. ЕСЛИ $\delta(G_s) \le p(G_s)-4$ ТО ПЕРЕЙТИ К II
2. Выбрать очередную антиклику Q в $\tilde{G}_s$
3. Положить $s' = s$
4. ПОКА $s' \neq 0$ ЦИКЛ
5. Присвоить переменной j значение последней компоненты s'
6. Положить $s' := s'_{-1}$
7. ЕСЛИ $j \neq 0$ ТО
8. ЕСЛИ $Q \subseteq \nabla(G_{s'_0})$ ТО ПЕРЕЙТИ К II

КОНЕЦ ЦИКЛА

9. Занести Q в BUF
10. ЕСЛИ Q не последняя по порядку антиклика в $\tilde{G}_s$ ТО ПЕРЕЙТИ К 2
11. КОНЕЦ АЛГОРИТМА

2.2. Алгоритм нахождения в графе максимальной клики.

Алгоритм поиска максимальной клики определим как схему РАЗ - БОР $(2,2,O_2, B_2, H_3)$. Таким образом, структура у него такая же, как и у алгоритма поиска всех клик, только правило H_3 накопления результатов здесь будет более простым. Это правило основано на следующей теореме.

ТЕОРЕМА 8. Пусть граф G состоит из p_0 изолированных вершин и p_1 компонент G^i, являющихся либо простыми цепями, либо простыми циклами. Тогда для вершинного числа независимости $\beta_0(G)$ справедливо равенство:

$$\beta_0(G) = p_0 + \sum_{i=1}^{p_1} \beta_0(G^i),$$

где

$$\beta_0(G^i) = \begin{cases} \dfrac{p(G^i)+1}{2}, & \text{если } p(G^i) \text{ нечетное;} \\ \dfrac{p(G^i)}{2} & \text{- в противном случае,} \end{cases}$$

и если G^i есть простая цепь; далее

$$\beta_0(G^i) = \begin{cases} \dfrac{p(G^i)-1}{2}, & \text{если } p(G^i) \text{ нечетное;} \\ \dfrac{p(G^i)}{2} & \text{- в противном случае,} \end{cases}$$

и если G^i есть простой цикл.

Алгоритм H_3 работает следующим образом. Если $\delta(G_s) > (G_s)-4$, то вычисляется $\beta_0(\bar{G}_s)$ при помощи теоремы 8, и если антиклика, содержащаяся в BUF, имеет порядок меньше $\beta_0(\bar{G}_s)$, то в $\bar{G}_s$ ищется максимальная антиклика и помещается в BUF. Максимальная антиклика в $\bar{G}_s$ есть объединение максимальных антиклик Q_i каждой его компоненты G^i $(i \in [1,r])$. Если G^i - простая цепь $v_1, v_2, \ldots, v_t$, то $Q_i = \{v_1, v_3, \ldots, v_{t'}\}$, где $t' = t$, если t нечетно, и $t'=t-1$ - в противном случае. Если G^i - простой цикл $v_1, v_2, \ldots, v_t$ (v_1 смеж-

на с v_t), то $Q_i = \{v_1, v_3, \ldots, v_{t''}\}$, где $t'' = t-1$, если t четное, и $t'' = t-2$ – в противном случае.

Алгоритм H_3 (правило накопления)

НАЧАЛО АЛГОРИТМА /* исходной информацией для алгоритма служит граф G_s, а также содержимое накопителя BUF, которое было получено на предыдущих шагах разбора; выходной информацией является содержимое BUF */

I. ЕСЛИ $\delta(G_s) > p(G_s)-4$ ТО

НАЧАЛО

2. Вычислить $\beta_0(G_s)$

3. ЕСЛИ мощность множества, содержащегося в BUF, меньше $\beta_0(G_s)$ ТО

НАЧАЛО

4. Найти максимальную антиклику Q в $\bar{G}_s$

5. BUF:=Q

КОНЕЦ

КОНЕЦ

КОНЕЦ АЛГОРИТМА

2.3. Оценки трудоемкости алгоритмов поиска клик.

В терминах §I число операций при выполнении алгоритма поиска клик РАЗБОР $(2,2,O_2,B_2,H_2)$ ограничено сверху функцией сложности

$$\xi = \max_{T \in \mathcal{T}} \sum_{w_s \in V(T)} (c_1 + c_2 \cdot p^2(G) + \theta_s(O_2) + \theta_s(B_2) + \theta_s(H_2)).$$

Выбор вершины с минимальной степенью требует в худшем случае $O(p^2(G))$ операций. Оценим $\theta_s(O_2)$ и $\theta_s(H_2)$. Из описания алгоритма O_2 следует, что $\theta_s(O_2) = \sum_{i=1}^{8} \tau_i b_i$, где τ_i обозначает число операций при выполнении оператора, имеющего метку i, а b_i – число выполнений этого оператора. Очевидно, $b_1 = 1$; $b_j = O(p(G))$, $j = \overline{2,6}$, так как число компонент индекса s' не более $p(G)$; $b_7 = b_8 = 1$; $\tau_1 = O(p(G))$; $\tau_r = O(1)$, $r=\overline{3,5}$; $\tau_6 = O(p(G))$; $\tau_7 = O(p^2(G))$.

Таким образом, $\theta_s(O_2) = O(p^2(G))$. Более наглядно можно получить эту оценку, используя понятие дерева разбора. Рассмотрим поддерево, состоящее из цепи, соединяющей вершину w_s с корневой

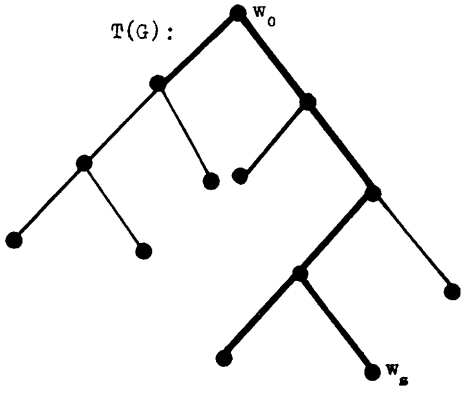

Рис. 8

вершиной w_0 и всех левых отростков, т.е. вершин, смежных с вершинами из цепи и лежащих слева от нее (на рис.8 это поддерево выделено жирными линиями). Граф G_s проверяется на вхождение в графы G_{s_i}, соответствующие отросткам. Поскольку каждая такая проверка требует $O(p(G))$ операций и величина поддерева есть также $O(p(G))$, то трудоемкость алгоритма есть $O(p^2(G))$

Оценим теперь $\theta_s(H_2)$. Представим эту величину в виде $\sum_{i=1}^{11} \tau_i b_i$, где τ_i и b_i имеют тот же смысл, что и выше в выражении для $\theta_s(O_2)$. Имеем: $b_1 = b_{11} = 1$; $b_2 = b_3 = b_9 = b_{10} = |M(G_s)|$; $b_j = O(p(G)\cdot|M(G)|)$ для $j = \overline{4,8}$; $\tau_1 = O(1)$ (так как $\delta(G_s)$ уже вычислено в алгоритме O_2); $\tau_2 = O(p(G))$, $\tau_3 = O(p(G))$, $\tau_r = O(1)$ для $r = \overline{4,7}$; $\tau_8 = \tau_9 = \tau_{10} = O(p(G))$, $\tau_{11} = O(1)$. Окончательно имеем $\theta_s(H_2) = O(p^2(G)\cdot|M(G)|)$.

Поскольку алгоритм H_2 выполняется только в том случае, когда w_s - висячая вершина дерева T, то

$$\xi = \max_{T\in\mathcal{T}} \sum_{w_s\in V(T)} (c_1 + c_2\cdot p^2(G) + \theta_s(O_2) + \theta_s(B_2) + \theta_s(H_2)) \le$$

$$\le \max_{T\in\mathcal{T}} [c_0 p^2(G)(|V(T)\setminus V_t(T)| + \sum_{w_s\in V_t(T)} |M(G_s)|)].$$

Здесь через $V_t(T)$ обозначено множество висячих вершин дерева T.

ТЕОРЕМА 9. Справедливо соотношение

$$\xi = \xi\left(2,2,O_2,B_2,H_2,G\right) \le O\left(p^2(G)\cdot 3^{\frac{p(G)}{3}}\right).$$

ДОКАЗАТЕЛЬСТВО. Учитывая предыдущие рассуждения, достаточно показать, что

$$|V(T)\setminus V_t(T)| + \sum_{w_s \in V_t(T)} |M(G_s)| \le c_1 \cdot 3^{\frac{p(G)}{3}}$$

для каждого дерева T. Известно [II], что

$$|M(G_s)| \le c_2 \cdot 3^{\frac{p(G_s)}{3}} .$$

Пусть

$$f(T) \equiv |V(T)\setminus V_t(T)| + c_2 \sum_{w_s \in V_t(T)} \cdot 3^{\frac{p(G_s)}{3}} .$$

Если дерево T состоит из одной вершины, то

$$f(T) = O(3^{\frac{p(G)}{3}}) .$$

Предположим, что $p(T) > 1$. Определим последовательность $T_1, T_2, \ldots, T_r$ деревьев следующим образом: $T_1 = T$; T_{i+1} получается из T_i по правилу: если все последователи вершины w_s дерева T_i висячие, то они удаляются (рис.9); дерево T_r состоит из одной вершины w_0. Обозначим через $V_t'(T_i)$ множество висячих вершин дерева T_i, после удаления которых получается дерево T_{i+1}. Докажем, что $f(T_i) \le f(T_{i+1})$, откуда будет следовать искомая оценка. Введем для краткости обозначения: $X_i \equiv V(T_i)\setminus V_t(T_i)$; $X_{i+1} \equiv V(T_{i+1}) \setminus V_t(T_{i+1})$; $Y \equiv V_t(T_{i+1})\setminus(V_t(T_i)\setminus V_t'(T_i))$. Тогда $X_i = (X_{i+1} \cup V_t(T_{i+1})) \setminus (V_t(T_i)\setminus V_t'(T_i))$. Поскольку $X_{i+1} \cap V_t(T_{i+1}) = \emptyset$, $X_{i+1} \cap (V_t(T_i) \setminus V_t'(T_i)) = \emptyset$, то это дает нам право записать: $X_i = X_{i+1} \cup (V_t(T_{i+1})\setminus(V_t(T_i)\setminus V_t'(T_i))) = X_{i+1} \cup Y$ и $|X_i| = |X_{i+1}| + |Y|$. Следовательно,

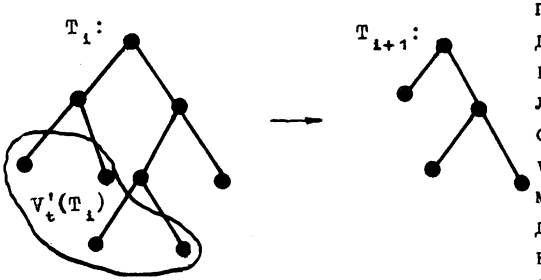

Рис. 9

$$f(T_i) = |X_i| + c_2 \cdot \sum_{w_s \in V_t(T_i) \setminus V'_t(T_i)} 3^{\frac{p(G_s)}{3}} + c_2 \cdot \sum_{w_s \in V'_t(T_i)} 3^{\frac{p(G_s)}{3}} =$$

$$= |X_{i+1}| + |Y| + \sum_{w_s \in Y} c_2 \cdot \left(3^{\frac{p(G_{s0})}{3}} + 3^{\frac{p(G_{s1})}{3}}\right) + \sum_{w_s \in V_t(T_i) \setminus V'_t(T_i)} c_2 \cdot 3^{\frac{p(G_s)}{3}} =$$

$$= |X_{i+1}| + \sum_{w_s \in Y} \left(1 + c_2 \cdot 3^{\frac{p(G_{s0})}{3}} + c_2 \cdot 3^{\frac{p(G_{s1})}{3}}\right) + \sum_{w_s \in V_t(T_i) \setminus V'_t(T_i)} c_2 \cdot 3^{\frac{p(G_s)}{3}};$$

$$f(T_{i+1}) = |X_{i+1}| + \sum_{w_s \in Y} c_2 \cdot 3^{\frac{p(G_s)}{3}} + \sum_{w_s \in V_t(T_i) \setminus V'_t(T_i)} c_2 \cdot 3^{\frac{p(G_s)}{3}}.$$

Осталось доказать, что для любого s имеет место неравенство:

$$1 + c_2 \cdot 3^{\frac{p(G_{s0})}{3}} + c_2 \, 3^{\frac{p(G_{s1})}{3}} \leq c_2 \cdot 3^{\frac{p(G_s)}{3}}$$

Из построения алгоритма O_2 следует, что $p(G_{s0}) \leq p(G_s) - 4$ и $p(G_{s1}) \leq p(G_s) - 1$. Тогда

$$1 + c_2\left(3^{\frac{p(G_{s0})}{3}} + 3^{\frac{p(G_{s1})}{3}}\right) \leq 1 + c_2\left(3^{\frac{p(G_s)-4}{3}} + 3^{\frac{p(G_s)-1}{3}}\right).$$

Представим константу c_2 в виде $c_2 = c_3 \cdot 3^{\frac{7}{3}}$ и докажем неравенство:

$$1 + c_3 \cdot 3^{\frac{p(G_s)+3}{3}} + c_3 \cdot 3^{\frac{p(G_s)+6}{3}} \leq c_3 \cdot 3^{\frac{p(G_s)+7}{3}}.$$

Эквивалентным ему будет следующее неравенство:

$$3^{-\frac{4}{3}} + 3^{-\frac{1}{3}} + \frac{1}{c_3} \cdot 3^{\frac{p(G_s)+7}{3}} \leq 1.$$

Левая часть этого неравенства - монотонно убывающая функция от $p(G_s)$, и ее максимальное значение составляет 0,9779 с точностью до четвертого знака после запятой при $p(G_s)=1$ и $c_3 = 1$. Теорема доказана.

ТЕОРЕМА 10. Пусть $\rho = (\rho_1,\rho_2,\ldots,\rho_t)$ - набор монотонных функций, заданных на графах, причем хотя бы одна функция из них строго монотонна, и $a = (a_1,a_2,\ldots,a_t)$ - некоторый набор целых чисел. Тогда число операций при выполнении алгоритма РАЗБОР $(2,2,O_2,B_2,H_2)$ поиска всех клик, а также при выполнении алгоритма РАЗБОР $(2,2,O_2,B_2,H_3)$ поиска максимальной клики не превосходит величины $c_0 \cdot p^2(G) N_\rho(2,2,O_1,B_2,H_1,a)$, где $N_\rho(2,2,O_1,B_2,H_1,a)$ есть наибольшее число вершин дерева разбора по всем графам из класса $\mathcal{G}_\rho(a)$.

ДОКАЗАТЕЛЬСТВО. Рассмотрим дерево разбора T_1 графа G при выполнении алгоритма РАЗБОР $(2,2,O_2,B_2,H_2)$. Из построения алгоритма H_2 следует, что подграф G_s, соответствующий каждой висячей вершине дерева T_1, либо игнорируется, либо в нем перечисляются клики в лексикографическом порядке. Множество висячих вершин w_s, которым соответствует перечисление клик в G_s обозначим через $\tilde{V}_t(T_1)$. Произведем разбор каждого графа G_s, соответствующего висячим вершинам из $\tilde{V}_t(T_1)$, при помощи алгоритма РАЗБОР $(2,2,O_1,B_2,H_1)$. Тогда дерево разбора T_1 вместе с такими деревьями, добавленными к его вершинам из $\tilde{V}_t(T_1)$, образует дерево разбора графа G в соответствии со схемой РАЗБОР $(2,2,O_1,B_2,H_1)$.

Обозначим через $\mathcal{T}_1 \equiv \{T_1(G)\}$ и $\mathcal{T}_0 \equiv \{T_0(G)\}$ множества деревьев разбора соответственно схемам РАЗБОР $(2,2,O_1,B_2,H_2)$ и РАЗБОР $(2,2,O_1,B_2,H_1)$. Каждой вершине из $V(T_1)\setminus\tilde{V}_t(T_1)$ соответствует $O(p^2(G))$ операций, а вершине из $\tilde{V}_t(T_1)$ соответствует $O(p^2(G)|M(G)|)$ операций. Поэтому для наибольшего числа операций при поиске всех клик в графе из класса $\mathcal{G}_\rho(a)$ можно записать:

$$\max_{G\in\mathcal{G}_\rho(a)} \xi(2,2,O_2,B_2,H_2,G) \le c_0 p^2(G) \max_{G\in\mathcal{G}_\rho(a)} \max_{T_1\in\mathcal{T}_1} (|V(T_1)| +$$

$$+ \sum_{w_s \in \mathcal{V}_t(T_1)} (|M(G_s)| - 1)) \le c_0\, p^2(G) \max_{G \in \mathcal{G}_\rho(a)} \max_{T_1 \in \mathcal{T}_1} (|V(T_1)| +$$

$$+ \sum_{w_s \in \mathcal{V}_t(T_1)} p(T_0(G_s))) \le c_0 \cdot p^2(G) \max_{G \in \mathcal{G}_\rho(a)} \max_{T_0 \in \mathcal{T}_0} p(T_0(G)) =$$

$$= c_0 \cdot p^2(G) N_\rho(2, 2, O_1, B_2, H_1, a).$$

Поскольку алгоритму нахождения максимальной клики соответствует дерево разбора графа, являющееся просто поддеревом в $T_0(G)$, то отсюда непосредственно и вытекает искомая оценка. Теорема доказана.

Теорема 10 показывает, что для получения априорных оценок числа операций при выполнении алгоритма перечисления всех клик графа и алгоритма нахождения максимальной клики можно использовать методику, разработанную в §2. В частности, из теоремы 10 и теоремы 6 получаем

СЛЕДСТВИЕ. Число операций при выполнении алгоритмов поиска всех клик и поиска максимальной клики не превосходит $c_0 \cdot p^2(G) x_0^{p(G)}$, где x_0 - корень уравнения $x^\nu - x^{\nu-1} - 1 = 0$ $\left(\nu = \frac{p(G)}{m(G)}\right)$.

ПРИМЕР 2. Оценим число операций при выполнении алгоритма поиска всех клик для полных k-дольных графов $K_{\underbrace{3,3,\ldots,3}_{k}}$ и $K_{\underbrace{2,2,\ldots,2}_{k}}$ Эти графы являются дополнениями k несвязанных треугольников и k несвязанных ребер соответственно, поэтому $p(K_{3,3,\ldots,3}) = 3k$, $p(K_{2,2,\ldots,2}) = 2k$, $m(K_{3,3,\ldots,3}) = m(K_{2,2,\ldots,2}) = k$, $|M(K_{3,3,\ldots,3})| = 3^k$, $|M(K_{2,2,\ldots,2})| = 2^k$. Корень уравнения $x^3 - x^2 - 1 = 0$ приблизительно равен $\sqrt[3]{3}$, откуда, по теореме 6, имеем оценку для числа операций $O\left(p^2\, 3^{\frac{p(G)}{3}}\right)$ на графах $K_{3,3,\ldots,3}$. Это говорит о том, что верхняя оценка, полученная по теореме 9, достигается на графах $K_{3,3,\ldots,3}$, известных как кликово-экстремальные графы [11], и, следовательно, оценка, полученная в теореме 9, не может быть суще -

ственно улучшена. Далее, положительным корнем уравнения $x^2-x-1=0$ является $x_0 = \frac{1+\sqrt{5}}{2} \approx 1,62$. Оценка $O(p^2(G)\cdot 1,62^{p(G)})$ уже является завышенной, поскольку в действительности число операций при поиске всех клик в $K_{2,2,\ldots,2}$ будет иметь порядок $O(p^2(G)\, 2^{\frac{p(G)}{2}})$.

Приведенный пример позволяет сделать предположение, что чем больше отношение $\frac{p(G)}{m(G)}$, тем точнее оценка $O(p^2(G)\cdot x_0^{p(G)})$.

ТЕОРЕМА II. Объем оперативной памяти, необходимой для работы алгоритмов поиска всех клик и максимальной клики, имеет порядок $O(p^2(G))$.

ДОКАЗАТЕЛЬСТВО. Будем считать, что при перечислении всех клик каждая вновь полученная клика сразу же поступает на некоторый внешний носитель информации. Поэтому необходимая память будет состоять из памяти, в которой хранятся начальные данные, и памяти, в которую по мере выполнения алгоритма помещаются промежуточные результаты. Объем начальных данных определяется информацией, которая представляет исходный граф и имеет в худшем случае порядок $O(p^2(G))$. Для оценки памяти, необходимой для хранения промежуточных результатов, предположим, что на некотором шаге алгоритма исследуется граф G_s.

Это исследование состоит в выделении множества $V_л$ вершин, смежных с вершинами цепи $w_s, w_{s-1}, \ldots, w_0$, и лежащих слева от нее, и в проверке вхождения графа G_s в графы G_s, соответствующие вершинам из множества $V_л$ (см. выше оценку трудоемкости алгоритма O_2). Поэтому в качестве промежуточной информации необходимо хранить множества $V(G_{s_i})$ вершин графов G_{s_i}, соответствующих висячим вершинам поддерева, образованного вышеуказанной цепью и ее левыми отростками (см. рис.8). Поскольку число вершин в этом поддереве имеет порядок $O(p(G))$ и $|V(G_s)| = O(p(G))$, то память, требуемая для поиска всех клик, имеет объем порядка $O(p^2(G))$. Для алгоритма поиска максимальной клики доказательство аналогично. Теорема доказана.

Теперь оценим результаты экспериментального исследования алгоритмов поиска клик. Были поставлены следующие цели:

1) сравнить по эффективности алгоритм РАЗБОР $(2,2,O_2,B_2,H_2)$ с другими алгоритмами поиска клик, основанными на разборе графа;

2) получить экспериментальные данные о трудоемкости алгоритмов.

Для сравнения были выбраны алгоритм Брона-Кербоша [3] и его модификация, которую в терминах §1 можно записать в виде РАЗБОР$(2,2,O_3,B_2,H_1)$, где правило остановки O_3 отличается от правила O_2 только оператором

7. ЕСЛИ G_s не полный граф ТО правило остановки не выполнено ИНАЧЕ правило остановки выполнено.

Работу алгоритма Брона-Кербоша можно схематично представить следующим образом. В графе G выбирается вершина с максимальной степенью, и относительно нее выделяются два подграфа: $\langle \Gamma_G(v) \rangle$ и $G - v$. Вершина v заносится в стек, который на каждом шаге работы алгоритма содержит множество вершин, образующее полный подграф. Если на каком-то шаге оказалось, что $\langle \Gamma_{G'}(v) \rangle = \emptyset$, то стек содержит клику. Если получен граф G', входящий в окружение вершины, относительно которой выделялись подграфы на предыдущих шагах, то G' игнорируется. Таким образом, алгоритм Брона-Кербоша представим в виде рекурсивного разбора*), которому естественным образом можно поставить в соответствие дерево разбора.

В алгоритме РАЗБОР $(2,2,O_3,B_2,H_1)$ в графе G выбирается вершина v с минимальной степенью и G разбивается на два подграфа: $\langle \{v\} \cup \Gamma_G(v) \rangle$ и $G-v$ (если G - неполный граф). Аналогичная операция применяется ко всем полученным таким путем графам, если только выполняются два условия: граф окажется либо неполным, либо не входящим в какой-либо другой граф, исследование которого уже закончено.

Почему для сравнения эффективности были выбраны именно алгоритм Брона-Кербоша и его модификация?

1) Анализ литературы по известным алгоритмам поиска клик показал, что алгоритм Брона-Кербоша является одним из наиболее эффективных.

2) Эти алгоритмы представимы в виде рекурсивного разбора графов, что позволяет для сравнения их эффективности использовать такие характеристики разбора, как число вершин дерева разбора, которые являются инвариантными по отношению к программной реализации алгоритма.

---

*) Схема рекурсивного разбора для алгоритма Брона-Кербоша подробно исследована в [8].

3) Сравнение с алгоритмами, в которых для поиска клик используется не разбор графа, а другие принципы (см., например, [I7,I8]), возможно только по времени реализации на одних и тех же графах и на одной и той же ЭВМ. Такой способ сравнения в значительной мере зависит от техники программирования и поэтому не всегда объективен.

4) Сравнение алгоритма Брона-Кербоша с алгоритмом РАЗБОР (2, 2, O_3, B_2, H_1) позволило выяснить, в каких случаях выбор вершины с минимальной степенью предпочтительнее выбора вершины с максимальной степенью.

Алгоритмы были запрограммированы на языке ПЛ/I и исследовались на ЭВМ ЕС-I033 и ЕС-I050.

В качестве тестов были выбраны графы из двух классов.

I. Графы соответствий пары полных графов - $L(K_i, K_j)$ (см. определение 7 в п.2.4). Выбор этого класса обусловлен тем, что, во-первых, графы $L(K_i, K_j)$ в значительной мере "насыщены" кликами (их число равно $\frac{i!}{(i-j)!}$, $i \geq j$), во-вторых, для этих графов легко подсчитываются различные характеристики (плотность, число ребер и т.п.) и, в-третьих, одним из приложений алгоритмов являются задачи, сводящиеся к поиску клик в графах соответствий (см. п. 2.4).

2) Графы Муна-Мозера, которые определяются как дополнения набора несвязанных треугольников и являются традиционными тестами для испытания алгоритмов поиска клик, поскольку содержат наибольшее число клик, приходящихся на одну вершину [II].

Для сравнения алгоритмов была выбрана инвариантная величина $\tilde{\eta}$, которая для алгоритмов Брона-Кербоша и РАЗБОР (2,2,O_3, B_2, H_1) определялась как число вершин соответствующего дерева разбора, а в алгоритме РАЗБОР (2,2,O_2,B_2,H_2) как число вершин дерева разбора плюс число клик во всех графах G_s, в которых происходило выделение клик при помощи алгоритма H_2. Поскольку деревья разбора для схем РАЗБОР (2,2,O_2,B_2,H_2) и РАЗБОР (2,2,O_2,B_2,H_3) совпадают, то попутно был исследован алгоритм поиска максимальной клики, сформулированный в п.2.2. На рис.I0 изображена зависимость величины $\ln \tilde{\eta}$ от числа ребер графов $L(K_i, K_j)$. Графики I-4 соответствуют алгоритмам РАЗБОР (2,2, O_2,B_2,H_3), РАЗБОР (2,2, O_2,B_2,H_2), РАЗБОР (2,2, O_3,B_2,H_1) и Брона-Кербоша. Для этих же графов на рис.II показаны зависимости величины γ - числа вершин дерева разбора, приходящихся на одну клику, от порядка графов $G = L(K_i, K_j)$. Каждой кривой из семейства соответствует постоянное значение плотности графа G.

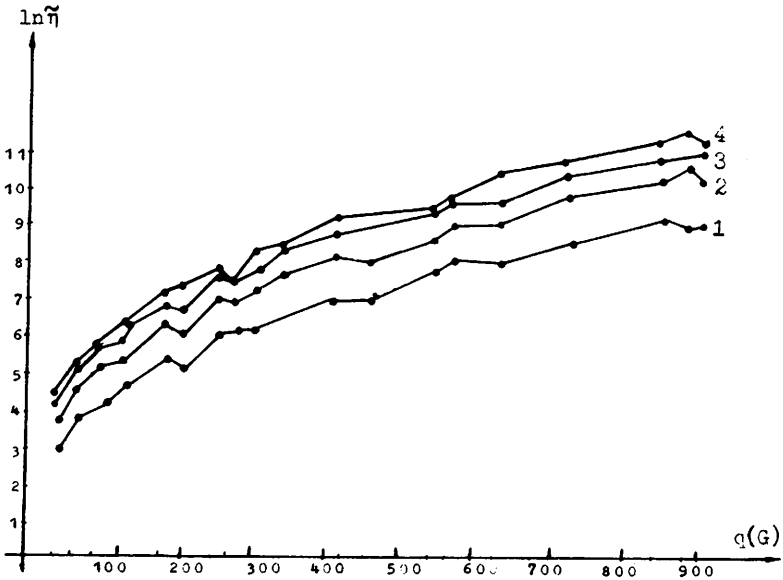

Рис. I0. Экспериментальные зависимости $\ln \tilde{\eta}$ от числа ребер графов.

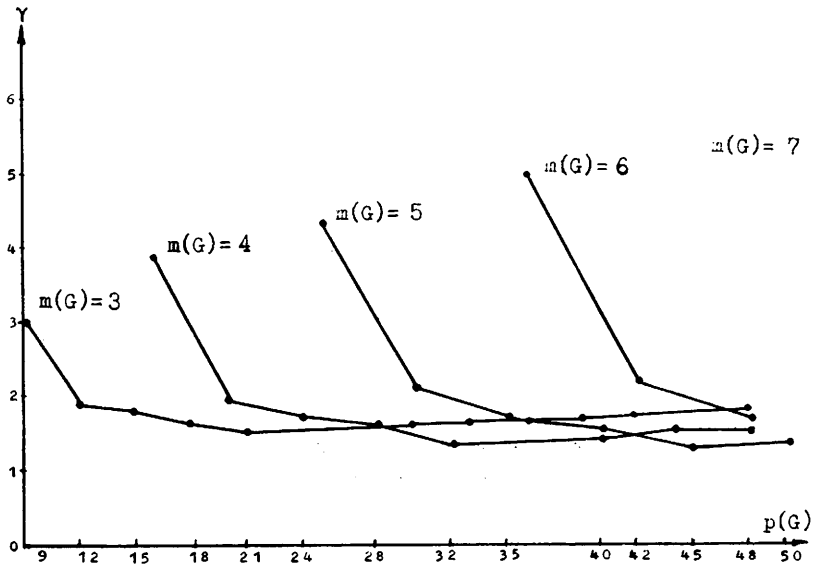

Рис. II. Экспериментальные зависимости отношения $\tilde{\eta}$ к числу клик графа от его порядка.

Основываясь на результатах, представленных на рис.II, можно предполагать, что на графах $G = L(K_i, K_j)$ при $i \neq j$ число операций, приходящихся на одну клику, растет, как $p^2(G)$, а на графах $L(K_i, K_i)$ быстрее, чем $p^2(G)$.

Анализ времени работы алгоритмов показал, что время, затраченное на одну вершину дерева разбора на графах $L(K_i, K_j)$, для всех алгоритмов приблизительно одинаково и составляет в среднем 0,I сек на ЭВМ ЕС-I033. Для одних и тех же графов время выполнения алгоритма Брона-Кербоша в среднем на 30% больше времени выполнения алгоритма РАЗБОР $(2,2,O_3,B_2,H_1)$, которое, в среднем, на 50% больше времени выполнения алгоритма РАЗБОР $(2,2,O_2,B_2,H_2)$.

Результаты исследования алгоритмов на графах Муна-Мозера приведены в табл. 3. Из табл.3 видно, что на графах Муна-Мозера величина $\tilde{\eta}$ для алгоритмов Брона-Кербоша и РАЗБОР $(2,2,O_3,B_2,H_1)$ одинакова, однако время выполнения алгоритма Брона-Кербоша приблизительно в два раза меньше. Отсюда можно заключить, что структура графа в значительной мере определяет не только число вершин дерева разбора, но и число операций, приходящихся на каждую его вершину.

Из табл.3 также видно преимущество алгоритма РАЗБОР $(2,2,O_2, B_2,H_2)$ на данном классе графов, которое объясняется следующим образом. Алгоритмы Брона-Кербоша и РАЗБОР $(2,2,O_3,B_2,H_1)$ выполняют разбор графа до появления клик, а алгоритм РАЗБОР $(2,2,O_2,B_2,H_2)$ - до появления подграфов G_* таких, что $\delta(G_*) > p(G_*)-4$. Поскольку граф Муна-Мозера удовлетворяет условию $\delta(K_{3,3,\ldots,3}) = p(K_{3,3,\ldots,3})- 3$, то сразу же выполняется алгоритм H_2, который перечисляет все клики.

В табл.4 показано, на сколько отличаются значения числа вершин дерева разбора, полученные экспериментально, от теоретической верхней оценки, вычисляемой при помощи рекуррентных соотношений из леммы I. В каждой клетке таблицы первые три числа есть значения $\tilde{\eta}$ для алгоритмов РАЗБОР $(2,2,O_2,B_2,H_2)$, РАЗБОР $(2,2,O_3,B_2,H_1)$ и Брона-Кербоша соответственно. Четвертое число есть решение рекуррентного соотношения $\phi(a_1,a_2)=\phi\left(\left[\frac{a_2-1}{a_2}a_1\right],a_2-1\right)+\phi(a_1-1,a_2)+1$, определяющее верхнюю оценку числа вершин дерева разбора для графов с плотностью не более a_2 и числом вершин не более a_1.

2.4. Алгоритмы определения общих частей в графах.

Пусть $G_1, G_2, \ldots, G_t$ – некоторое множество графов. Задача отыскания общих частей в данном множестве формулируется следующим образом: найти набор графов $G_{i_1}, G_{i_2}, \ldots, G_{i_r}$ таких, что $G_{i_j} \subseteq G_k$ для всех $k \in [1,t]$ и $j \in [1,r]$.

Распознавание изоморфизма двух графов и изоморфного вхождения одного графа в другой есть частные случаи этой задачи.

При $t=2$ известны [I2-I4] методы решения задачи определения общих частей, основанные на поиске клик или антиклик в графах соответствий G_1 и G_2. Обобщим эти методы для $t > 2$.

ОПРЕДЕЛЕНИЕ 7. Графом соответствий $L(G_1, G_2, \ldots, G_t)$ графов $G_1, G_2, \ldots, G_t$ называется граф G, множеством вершин которого является совокупность всех упорядоченных наборов $u = (v_1, v_2, \ldots, v_t)$, $v_i \in V(G_i)$, а множеством ребер – такие пары $\{u', u''\} = \{(v'_1, \ldots, v'_t), (v''_1, \ldots, v''_t)\}$ этих наборов, что $(\forall i \in [1,t])(v'_i \neq v''_i) \& \{v'_i, v''_i\} \in E(G_i)) \vee (\forall i \in [1,t])(v'_i \neq v''_i) \& \{v'_i, v''_i\} \notin E(G_i))$.

Метод выделения подграфов, общих для данного набора графов, дает следующая

ТЕОРЕМА I2. Каждой клике графа $G = L(G_1, G_2, \ldots, G_t)$ соответствует граф, который является наибольшим (по включению) общим подграфом для всех G_i, $i \in [1,t]$, и наоборот.

ДОКАЗАТЕЛЬСТВО. Пусть множество $\{z_1, z_2, \ldots, z_r\}$ образует полный подграф в G, причем $z_i = (v_1^i, v_2^i, \ldots, v_t^i)$. Для некоторого $k \in [1,t]$ рассмотрим множество $V_k = \{v_k^1, v_k^2, \ldots, v_k^r\}$. Докажем,*) что $\langle V_k \rangle_{G_k} \simeq \langle V_j \rangle_{G_j}$ для всех $j \in [1,t]$. Из определения 9 вытекает, что если $\{v_k^{i_1}, v_k^{i_2}\} \in E(G_k)$, то $\{v_j^{i_1}, v_j^{i_2}\} \in E(G_j)$, и если $\{v_k^{i_1}, v_k^{i_2}\} \notin E(G_k)$, то $\{v_j^{i_1}, v_j^{i_2}\} \notin E(G_j)$. Следовательно, $\langle V_k \rangle_{G_k} \simeq \langle V_j \rangle_{G_j}$. Поскольку $\langle \{z_1, z_2, \ldots, z_r\} \rangle$ – клика, то не существует подграфов H_j в G_j и H_k в G_k таких, что $\langle V_k \rangle_{G_k} \subset H_k$,

*) $\langle X \rangle_G$ обозначает подграф в G, порожденный множеством вершин X.

$\langle V_j \rangle_{G_j} \subset H_j$ и $H_k \simeq H_j$. Доказательство обратного утверждения проводится аналогично с использованием определения 7.

Решение задачи нахождения общих подграфов состоит из трех этапов:

1) построения графа соответствий $L(G_1, G_2, \ldots, G_t)$;

2) нахождения всех клик графа соответствий;

3) восстановления по каждой клике $\{z_1, z_2, \ldots, z_r\}$ графа $\langle\{v_k^1, v_k^2, \ldots, v_k^r\}\rangle_{G_k}$, общего для данного набора графов $G_1, G_2, \ldots, G_t$.

Отметим, что каждый граф соответствий содержит избыточную информацию, т.е. одному и тому же графу, являющемуся общим подграфом для $G_1, G_2, \ldots, G_t$, могут соответствовать различные клики в $L(G_1, G_2, \ldots, G_t)$. Поэтому в процессе поиска клик в графе соответствий необходимо исключать клики, несущие избыточную информацию. Поскольку процесс нахождения клик можно представить как рекурсивный разбор, то нахождение общих частей в заданном множестве графов мы будем интерпретировать как рекурсивный разбор графа соответствий данного множества графов. Поэтому для разработки алгоритмов нахождения общих частей в графах и получения оценок их трудоемкости могут быть применены результаты, полученные в §1. Проиллюстрируем применение последних к решению задачи распознавания изоморфизма двух графов. Рассмотрим три алгоритма решения этой задачи.

Первый алгоритм назовем РАЗБОР $(2,2,O_2,B_2,H_4)$. Он реализует метод, предложенный в [13], и основан на следующем утверждении: графы G_1 и G_2 изоморфны тогда и только тогда, когда в $L(G_1, G_2)$ существует клика порядка $p(G_1) = p(G_2)$.

Во втором алгоритме, который назовем РАЗБОР $(2,2,O_3,B_3,H_4)$, существенно используются особенности структуры графа $L(G_1, G_2)$.

Третий алгоритм основан на схеме РАЗБОР $(2,2,O_3,B_3,H_4)$ и позволяет дополнительно учитывать структурные особенности графов G_1 и G_2.

Схему РАЗБОР $(2,2,O_2,B_2,H_4)$ мы определим как алгоритм, который выясняет, существует ли в графе $G = L(G_1, G_2)$ клика, имеющая порядок $p(G_1)$ (предполагается, что $p(G_1) = p(G_2)$), и, следовательно, изоморфны ли графы G_1 и G_2. Структура данной схемы почти такая же, как и у алгоритма нахождения максимальной клики и отличается только правилом накопления результатов H_4.

А л г о р и т м H_4 (правило накопления)

НАЧАЛО АЛГОРИТМА /* исходной информацией для алгоритма служит граф G_s, а результатом - высказывание об изоморфизме либо неизоморфизме графов G_1 и G_2, которое помещается в BUF */

1. BUF:= ГРАФЫ НЕИЗОМОРФНЫ
2. ЕСЛИ $\delta(G_s) \le p(G_s)-4$ ТО ПЕРЕЙТИ К 5
3. ЕСЛИ $\beta_0(\bar{G}_s) \neq p(G_1)$ /*$\beta_0(\bar{G}_s)$ вычисляется при помощи теоремы 8 */ ТО ПЕРЕЙТИ К 5
4. s:=0; BUF:= ГРАФЫ ИЗОМОРФНЫ
5. КОНЕЦ АЛГОРИТМА

ЗАМЕЧАНИЕ.3. Схема РАЗБОР $(2,2,O_2,B_2,H_4)$ может быть применена для исследования изоморфного вхождения графа G_1 в G_2, если $p(G_1) < p(G_2)$. Действительно, необходимым и достаточным условием изоморфного вхождения G_1 в G_2 является существование клики в $L(G_1,G_2)$, имеющей $p(G_1)$ вершин. Оценим трудоемкость алгоритма, которая в терминах §I есть функция сложности графа G по отношению к схеме РАЗБОР $(2,2,O_2,B_2,H_4)$. Оценим $N_{p^*}(a)$ наибольшее число вершин дерева разбора графа G из $\mathcal{Y}_{p^*}(a)$. Рассмотрим некоторую невисячую вершину дерева разбора, и пусть ей соответствует граф G_s. Из определения правила O_2 следует, что $p^*(G_{s0}) \le p^*(G_s)-4$ и $p^*(G_{s1}) \le p^*(G_s) - 1$. Поэтому из (I) и (2) имеем: $a' = a - 4$ и $a''= a-1$. По лемме I, справедливо неравенство $N_{p^*}(a) \le \phi(a)$, где $\phi(a) = \phi(a-4) + \phi(a-1) + 1$. По лемме 2, $\phi(a) = O(x_0^a)$, где x_0 - корень уравнения $x^4-x^3-1 = 0$. Решая уравнение, находим $x_0 \approx 1{,}38$. Следовательно, $N_{p^*}(a) = O(1{,}38^a)$, и окончательно для функции сложности графа $G \in \mathcal{Y}_{p^*}(a)$ имеем $X_{p^*}(a) = O(a^2 \cdot 1{,}38^a)$. Далее, если $G \in \mathcal{Y}_{p^*}(a)$, то $G_1, G_2 \in \mathcal{Y}_{p^*}(p_0)$, где $p_0 = \sqrt{a}$. Поэтому число операций при распознавании изоморфизма графов G_1 и G_2 из класса $\mathcal{Y}_{p^*}(p_0)$ имеет порядок $O(p_0^4 \cdot 1{,}38^{p_0^2})$.

Рассмотрим схему РАЗБОР $(2,2,O_4,B_3,H_4)$. Представим множество вершин графа соответствий $G = L(G_1,G_2)$ в виде квадратной матрицы (u_{ij}): $V(G)=\{v \mid v=u_{ij};\ i,j \in [1,p(G_1)]\}$ таким образом, что $u_{ij} = (u_i', u_j'')$, где $u_i' \in V(G_1)$ и $u_i'' \in V(G_2)$. Через $R_1(v)$ и $R_2(v)$ соответственно обозначим номера строки и столбца матрицы, на пересечении которых находится вершина v. Далее, пусть V_{i1} и V_{i2}

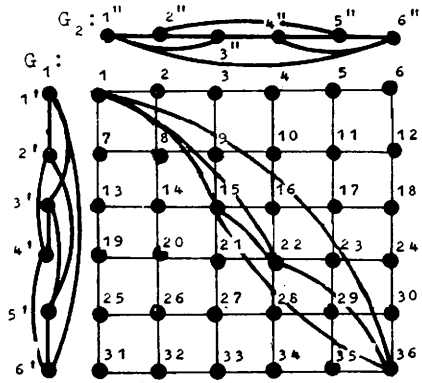

Рис. I2

обозначают соответственно множества вершин из V, образующих i-ю строку и j-й столбец.

ПРИМЕР 3. На рис.I2 множество вершин графа соответствий представлено в виде матрицы. Поскольку $L(G_1,G_2)$ имеет большое число ребер, то на рисунке изображены только те ребра, которые образуют клику, соответствующую одному из общих подграфов в G_1 и G_2. Из рисунка видно, что $R_1(9)=2$, $R_2(9)=3$, $V_{11}=\{1,2,3,4,5,6\}$.

ТЕОРЕМА I3. Пусть $V_s \subseteq V(G)$. Если существуют такие $i \in [1,p(G_1)]$, что $V_s \cap V_{ij} = \emptyset$, то в графе $\langle V_s \rangle$ нет полных подграфов порядка $p(G_1)$.

ДОКАЗАТЕЛЬСТВО. Из определения 7 следует, что каждый столбец и каждая строка матрицы (u_{ij}) образуют независимые множества, т.е. никакая пара вершин из этих множеств не смежна в графе G. Следовательно, никакие две вершины в полном подграфе графа G не могут лежать в одном столбце и одной строке, т.е. в полных подграфах вершины располагаются по одной в каждой строке и каждом столбце. Пусть пересечение V_s с какой-либо строкой или каким-либо столбцом пусто, и в то же время в V_s содержатся $p(G_1)$ вершин, образующих полный подграф. Поскольку числа строк и столбцов равны $p(G_1)$, то в этом полном подграфе найдутся две вершины, лежащие в одном столбце или одной строке. Полученное противоречие доказывает теорему.

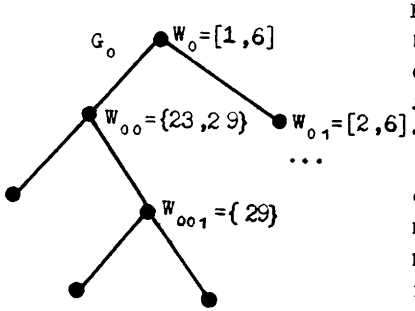

Рис. I3

Свойство графа соответствий, сформулированное в теореме I3, используется для сокращения перебора при нахождении в $G = L(G_1, G_2)$ клики порядка $p(G_1)$, если в правило остановки включить проверку условия теоремы I3 для множества $V(G_s)$: если условие теоремы I3 вы-

полняется, то выполняется правило остановки (см. рис.I3). Дейст - вительно, если $V(G_s) \cap V_{ij} = \emptyset$ для некоторых i и j, то дальнейший разбор графа G_s никакой новой информации не дает.

А л г о р и т м O_4 (правило остановки)

НАЧАЛО АЛГОРИТМА/* исходной информацией для алгоритма служит граф G_s, множество графов $\{G_{s'} \mid s' = s_{-r}0,\ r = 2,3,\dots\}$, разбор которых уже закончен, и матрица (u_{ij}), в виде которой представ - лено множество вершин графа $G = L(G_1, G_2)$; результатом является заключение о выполнении, либо невыполнении правила остановки */

I. Положить $s' := s$

2. ПОКА $s' \neq 00$

ЦИКЛ

3. Переменной j присвоить значение последней компоненты ин - декса s'

4. Положить $s' := s'_{-1}$

5. ЕСЛИ $j \neq 0$ ТО

6. ЕСЛИ $V(G_s) \subseteq V(G_{s',0})$ ТО ПЕРЕЙТИ К 8

КОНЕЦ ЦИКЛА

7. ЕСЛИ пересечение множества $V(G_s)$ с любой строкой и с лю - бым столбцом матрицы (u_{ij}) непусто и $\delta(G_s) \leq p(G_s) - 4$ ТО правило остановки не выполняется

ИНАЧЕ

8. Правило остановки выполняется

КОНЕЦ АЛГОРИТМА

Для увеличения эффективности целесообразно проводить процесс разбора таким образом, чтобы условие теоремы I3 выполнялось по возможности как можно раньше.

Для этого правило выбора B_3 мы определим таким образом, что- бы вершина v_s выбиралась из такого подмножества U_s в $V(G_s)$, которое является пересечением $V(G_s)$ с некоторой строкой или не- которым столбцом и имеет наименьшую мощность.

Работа алгоритма B_3 состоит в следующем. Если $s = 0$, то мно - жество U_0 полагается равным первой строке матрицы (u_{ij}). Пусть $s \neq 0$. Если граф G_s есть первый пояс в разбиении графа $G_{s_{-1}}$ от- носительно вершины $v_{s_{-1}}$ (т.е. если индекс s оканчивается на еди- ницу), то новое множество U_s образуется из старого U_s удалением

вершины $v_{s_{-1}}$. Это гарантирует, что через $|U_s|$ шагов вправо от w_s по дереву разбора будет выполнено условие теоремы I3. Если же граф G_s есть нулевой пояс в указанном выше относительном разбиении, то U_s остается прежним. Из множества U_s выбирается вершина v^*, у которой пересечение окружения со строками имеет минимальную мощность. Это пересечение образует множество U_{s0}, из которого для графа G_{s0} при следующем выполнении B_3 будет выбираться вершина v^*. Тогда через $|U_{s0}|$ шагов вправо от w_s по дереву разбора будет выполнено условие теоремы I3. Во избежание зацикливания при выборе вершины v^* просматриваются не все строки и столбцы, а только те, которые не содержат вершин $v \in V(G)$ таких, что $V(G_s)$ входит в нулевые пояса разбиений относительно этих вершин на предыдущих шагах разбора. Действительно, пусть, например, $V(G_s)$ есть нулевой пояс в разбиении $G_{s_{-1}}$ относительно вершины $v_{s_{-1}}$. Тогда вершина $v_{s_{-1}}$ является смежной со всеми вершинами графа G_s. Поэтому пересечение $V(G_s)$ со строкой и столбцом, содержащим $v_{s_{-1}}$, будет состоять из одной вершины $v_{s_{-1}}$, и, значит, на следующем шаге снова будет выбрана $v_{s_{-1}}$, и ситуация повторится.

А л г о р и т м B_3 (правило выбора)

НАЧАЛО АЛГОРИТМА /* исходной информацией для алгоритма является граф G_s, матрица (u_{ij}) и множество U_s $(s \neq 0)$, которое было получено раньше при выполнении B_3; выходной информацией является вершина $v_s \in U_s$ и множество U_{s0}, которое послужит исходной информацией при последующем выполнении B_3 */

I. Положить $s' := s$; $I_1 = \{1,2,\ldots,p(G_1)\}$; $I_2 := I_1$ /* I_1 и I_2 - соответственно множества номеров строк и столбцов (u_{ij}) */

2. ЕСЛИ $s' = 0$ ТО положить U_s, равным первой строке матрицы (u_{ij})

ИНАЧЕ

НАЧАЛО

3. ЕСЛИ последняя компонента s' равна I ТО

4. $U_{s'} := U_{s'} \setminus \{v_{s'_{-1}}\}$

ИНАЧЕ

НАЧАЛО

5. Положить $I_1 := I_1 \setminus \{ R_1(v_{s'_{-1}}) \}$; $I_2 := I_2 \setminus \{R_2(v_{s'_{-1}})\}$ / * из I_1 и I_2 исключаются соответственно номера строк и столбцов, содержащие вершины $v \in V(G_s)$ такие, что $V(G_s)$ входит в нулевые пояса разбиений относительно этих вершин на предыдущих шагах разбора */

КОНЕЦ

6. ПОКА последняя компонента s' не равна 0

7. ЦИКЛ положить $s' := s'_{-1}$ КОНЕЦ ЦИКЛА

8. ЕСЛИ $s' \neq 0$ ТО ПЕРЕЙТИ К 5

КОНЕЦ

9. /* выбор v_s из $U_{s'}$ и формирование U_{s0} */выбрать вершину v_s из $U_{s'}$ и строку или столбец $V_{i^*j^*}$ такие, что

$$|\Gamma_{G_s}(v_s) \cap V_{i^*j^*}| = \min_{v \in U_{s'}} \min_{j \in [1,2]} \min_{i \in I_j \setminus \{R_j(v)\}} |\Gamma_{G_s}(v) \cap V_{ij}|$$

10. Положить $U_{s0} := \Gamma_{G_s}(v_s) \cap V_{i^*j^*}$

11. КОНЕЦ АЛГОРИТМА

ПРИМЕР 4. Рассмотрим разбор графа $L(G_1, G_2)$, где G_1 и G_2 взяты из рис.12 в соответствии со схемой РАЗБОР $(2,2,O_4,B_3,H_4)$. На рис.13 изображен фрагмент дерева разбора данного графа, а в табл.5 представлен протокол вычислений по данному алгоритму для указанного фрагмента дерева разбора. Можно проверить, что полное дерево разбора состоит из 19 вершин и результатом разбора является утверждение: ГРАФЫ НЕИЗОМОРФНЫ.

ЗАМЕЧАНИЕ 4. Алгоритм РАЗБОР $(2,2,O_4,B_3,H_4)$ может быть легко преобразован для распознавания изоморфного вхождения графа G_1 в G_2 $(p(G_1) = p_1 < p(G_2) = p_2)$. Для этого необходимо представить множество вершин графа $L(G_1,G_2)$ в виде прямоугольной матрицы (u_{ij}) с p_1 строками p_2 столбцами. Тогда для выполнения правила возвращения будет достаточно, если пересечение $V(G_s)$ хотя бы с одной строкой (u_{ij}) пусто, либо по меньшей мере с $p_2 - p_1 + 1$ столбцами.

Пусть заданы два относительных разбиения $\hat{G}_1(u_0^1) = (\{ u_0^1 \}, V_1^1, \ldots, V_n^1)$ и $\hat{G}_2(u_0^2) = (\{u_0^2\}, V_1^2, \ldots, V_n^2)$, удовлетворяющие условию: $|V_i^1| = |V_i^2|$ для всех $i \in [1,n]$. Такие разбиения мы будем называть с р а в н и м ы м и. Обозначим через V_0^1 и V_0^2 множества $\{u_0^1\}$ и $\{u_0^2\}$ соответственно.

ОПРЕДЕЛЕНИЕ 8. Графом соответствий $L(\hat{G}_1, \hat{G}_2)$ двух сравнимых относительных разбиений [16] графов G_1 и G_2 назовем подграф в

$L(G_1,G_2)$, порожденный множеством вершин $\bigcup_{i=0}^{n} \{(u_i^1,u_i^2) | u_i^1 \in V_i^1, u_i^2 \in V_i^2\}$.

ПРИМЕР 5. На рис. I4 изображено множество вершин графа $L(\hat{G}_1(1'),\hat{G}_2(1''))$, состоящее из подмножества, каждое из которых есть декартово произведение V_i^1 и V_i^2.

Очевидно, что множество вершин графа $L(\hat{G}_1,\hat{G}_2)$ можно представить в виде объединения строк или столбцов, только эти строки и столбцы будут подмножествами соответствующих строк и столбцов матрицы (u_{ij}) вершин графа $L(G_1,G_2)$. Поэтому к графу $L(\hat{G}_1,\hat{G}_2)$ применима схема РАЗБОР $(2,2,O_4,B_3,H_4)$. Ее результат ГРАФЫ ИЗОМОРФНЫ будет означать изоморфизм графов G_1 и G_2. Однако результат ГРАФЫ НЕИЗОМОРФНЫ еще не будет означать, что $G_1 \not\cong G_2$, потому что сравнимость двух разбиений $\hat{G}(v_1)$ и $\hat{G}(v_2)$ графа G еще не означает, что вершины v_1 и v_2 подобны (см. рис.I5). Поэтому для одного разбиения графа G_1 необходимо просмотреть все разбиения графа G_2, сравнимые с данным разбиением, и к каждому графу соответствий этих разбиений применить алгоритм РАЗБОР $(2,2,O_4,B_3,H_4)$.

Алгоритм, в котором реализован данный принцип, строит все относительные разбиения двух графов G_1 и G_2 и формирует матрицы мощностей слоев (λ_{ij}^1) и (λ_{ij}^2), где λ_{ij}^r – мощность j-го слоя в разбиении графа G_r относительно вершины с номером i $(1 \le i \le p(G_r)$, $r \in \{1,2\})$. Если матрицы не совпадают с точностью до перестановки строк, то графы G_1 и G_2 неизоморфны и алгоритм заканчивает работу. Если матрицы совпадают с точностью до перестановки строк, то еще неизвестно, изоморфны графы G_1 и G_2 или нет. Такой предварительный анализ двух графов впервые использован в работе [I5]. Поэтому среди разбиений графа G_1 выбирается разбиение $\hat{G}_1 = (V_0^1 V_1^1,\ldots,V_n^1)$, в котором величина $y \equiv \max_{0 \le i \le n} |V_i^1|$ минимальна, и выделяются все разбиения $\hat{G}_2^1,\ldots,\hat{G}_2^r$ графа G_2, сравнимые с $\hat{G}_1$. Затем к каждому графу $L(\hat{G}_1,\hat{G}_2^i)$ применяется алгоритм РАЗБОР $(2,2,O_4,B_3,H_4)$.

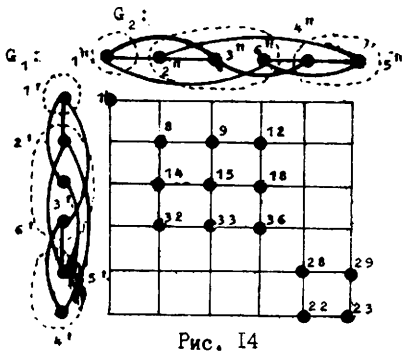

Рис. I4

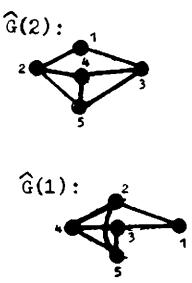

Рис.I5

Алгоритм распознавания изоморфизма двух графов путем разбора графа соответствий их относительных разбиений)

НАЧАЛО АЛГОРИТМА/* исходной информацией являются графы G_1 и G_2 с одинаковым числом вершин; результатом является сообщение А об их изморфизме или неизоморфизме */

I. А:=ГРАФЫ НЕИЗОМОРФНЫ

2. Построить разбиения графов G_1 и G_2 относительно всех вершин и сформировать матрицы (λ^1_{ij}) и (λ^2_{ij})

3. ЕСЛИ (λ^1_{ij}) совпадает с (λ^2_{ij}) с точностью до перестановки строк ТО

НАЧАЛО

4. Выбрать разбиение $\hat{G}_1 = (V^1_0, V^1_1, \ldots, V^1_n)$, в котором величина $\max\limits_{0\le i\le n} |V^1_i|$ минимальна

5. Выбрать все разбиения $\hat{G}^1_2, \hat{G}^2_2, \ldots, \hat{G}^r_2$, сравнимые с $\hat{G}_1$

6. j:=1

7. ПОКА $j \le r$

ЦИКЛ

8. Выполнить разбор графа $L(\hat{G}_1, \hat{G}^j_2)$ в соответствии со схемой РАЗБОР $(2,2,O_4,B_3,H_4)$

9. А:=ВИТ

10. ЕСЛИ А = ГРАФЫ ИЗОМОРФНЫ ТО ПЕРЕЙТИ К I2

11. j:=j+1

КОНЕЦ ЦИКЛА

КОНЕЦ

I2. Напечатать А

I3. КОНЕЦ АЛГОРИТМА

Дадим обоснование выбора разбиения $\hat{G}_1$. Для этого оценим число вершин дерева разбора графа $L(\hat{G}_1, \hat{G}_2)$ в зависимости от величины r. Пусть w_s - некоторая невисячая вершина дерева $T(L(\hat{G}_1, \hat{G}_2))$ и ей соответствует множество U_s, полученное в результате работы алгоритма B_3. Тогда вершина $w_{s\underbrace{11\ldots1}_{1+|U_s|\text{ раз}}}$ не будет обладать правым последователем, так как соответствующий ей граф будет удовлетворять условию теоремы I3. С другой стороны, вершина $w_{s\underbrace{00\ldots0}_{p(\hat{G}_1)-1\text{ раз}}}$ будет висячей, так как число нулей в индексе s' равно числу вершин степени $p(G_{s'})-1$ в графе $G_{s'}$, без единицы. Поскольку U_s

определяется как пересечение окружения некоторой вершины со строкой или столбцом, то, очевидно, $|U_s| \le y$.Так же, как при доказательстве леммы I, мы определим дерево $\tilde{T}$, которое содержит заведомо больше вершин, чем любое дерево разбора,и дадим оценку для $p(\tilde{T})$.Корневой вершине дерева $\tilde{T}$ припишем пару чисел $(y, p(G_1)-1)$. Правой вершине, смежной с корневой,припишем $(y, p(G_1) - 2)$, а левой - $(y-1, p(G_1) - 1)$. Пусть $\tilde{w}$ - некоторая вершина дерева $\tilde{T}$ и ей приписана пара (r,t); если $t = 0$,то $\tilde{w}$ - висячая вершина, в противном случае она имеет левый последователь, которому приписана пара $(y, t-1)$. Если $r = 0$, то вершина $\tilde{w}$ не имеет правого последователя, в противном случае правому последователю $\tilde{w}$ приписана пара $(r-1, t)$ (см. рис.I6). Обозначим через $\phi(y, p(G_1)-1)$ число вершин дерева $\tilde{T}$. Очевидно следующее рекуррентное соотношение: $\phi(y, p(G_1)-1) = y \cdot \phi(y, p(G_1) - 2)$, откуда $\phi(y, p(G_1)-1) = y^{p(G_1)-1}$. Полученная оценка дает обоснование минимизации величины y при выборе относительного разбиения графа G_1. Пусть

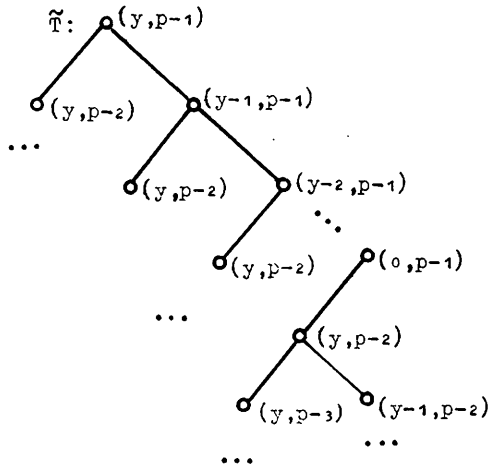

Рис. I6

$$y_{min} = \min_{1 \le i \le p(G_1)} \max_{1 \le j \le n(G_1, v_i)} \lambda_{ij} .$$

Тогда число вершин графа $L(\hat{G}_1, \hat{G}_2^i)$ имеет порядок $O(y_{min}^2)$, число разбиений $\hat{G}_2^1, \ldots, \hat{G}_2^r$, сравнимых с $\hat{G}_1$, в худшем случае имеет порядок $O(p(G_1))$. Следовательно, число операций при разборе r графов $L(\hat{G}_1, \hat{G}_2^i)$ имеет порядок $O(p(G_1) \cdot y_{min}^{p(G_1)})$ и определяет порядок числа операций при выполнении всего алгоритма, поскольку предварительный анализ требует $O(p^3(G_1))$ операций.

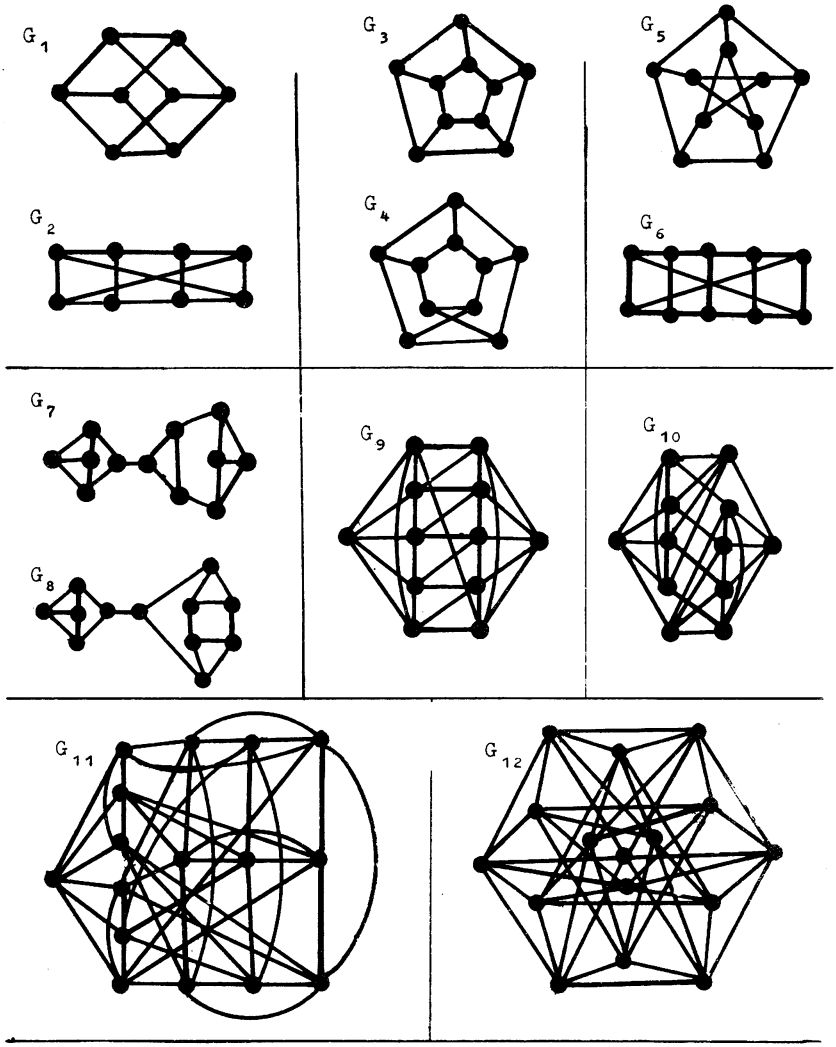

Рис. I7. Графы, на которых испытывались программы PROBA и PROBI .

Приведем результаты эксперимента с алгоритмами распознавания изоморфизма. Алгоритм РАЗБОР $(2,2,O_4,B_3,H_4)$ и алгоритм, основанный на разборе графа соответствий относительных разбиений, были реализованы в программах, названных соответственно PROBA и PROB1, и испытаны на ряде примеров. В качестве примеров были выбраны пары графов, обладающих достаточно высокой симметрией. Неизоморфные пары графов выбирались таким образом,чтобы их общая часть была достаточно велика. Алгоритмы оценивались по времени работы и по числу вершин дерева разбора, при этом в программе PROB1 значения числа вершин дерева разбора суммировались по всем деревьям разбора графов соответствий относительных разбиений. На рис.I5 приведены примеры графов, на которых испытывались программы, а в табл.6 - соответствующие характеристики трудоемкости. Ноль в четвертом столбце означает, что неизоморфизм графов был установлен еще при сравнении их относительных разбиений. Был произведен подсчет числа вершин дерева разбора при поиске всех клик в графах соответствий некоторых пар графов из рис.I7. В частности,для $L(G_1, G_2)$ и $L(G_3,G_6)$ получены соответственно значения 3045 и 60905,которые определяют трудоемкость непосредственного применения алгоритма поиска клик для распознавания изоморфизма без использования каких-либо способов сокращения перебора. Из табл. 6 видно,что время работы алгоритмов определяется в значительной степени числом операций, приходящихся на каждую вершину дерева разбора.Это подтверждает тот факт, что каждой вершине дерева разбора графа $L(G', G'')$ соответствует число операций, имеющее порядок $O(p^4(G'))$. (Действительно, легко показать, что каждой вершине дерева разбора соответствует $O(p^2(L(G' G")))$операций, а $p(L(G' G")) = p^2(G')$.)

Полученные алгоритмы распознавания изоморфизма не сравнивались с известными алгоритмами, поскольку основная цель статьи - показать возможность применения методики, основанной на рекурсивном разборе графа.

Авторы благодарят Л.И.Макарова за внимательное прочтение рукописи и ряд ценных замечаний.

Литература

I. ЗЫКОВ А.А. Функции от графов, определяемые линейными уравнениями (сообщ. I,2,3) - Изв. Сиб. отд. АН СССР, I959, № 5, с. 3-I9; I960, № 9, с. I7-33; I960, № I2, с. I4-27.

2. ПЛЕСНЕВИЧ Г.С., САПАРОВ М.С. Алгоритмы теории графов.-Ашхабад: Ылым. I98I. - 3I3 с.

3. BRON C., KERBOSCH J. Finding all cliques of an undirected graph. - Commun.ACM,1973,v.16, N 9,p.575-577.

4. DAS S.R., SHENG C., CHEN Z. An algorithm for finding all maximal complete subgraphs and an estimate of the order of computational complexity.-Comput.Elect.Engng,1978,v.5,N 4,p.365-368.

5. TARJAN R.,TROJANOVSKY A. Finding a maximum independent set.- SIAM J.Computing,1977,v.6,N 3,p.537-546.

6. БЕССОНОВ Ю.Е., СКОРОБОГАТОВ В.А. Применение относительных разбиений для поиска клик. -В кн.: Автоматизация проектирования в микроэлектронике. Теория. Методы. Алгоритмы (Вычислительные системы, вып. 77). Новосибирск, 1978, с. 25-33.

7. VOLDŘICH J. Gradual partition of a graph into complete graphs.- Cas.pěstov.mat.,1978,v.103, N 1,p.8-16.

8. БЕССОНОВ Ю.Е., СКОРОБОГАТОВ В.А. О рекурсивном разборе графов. -В кн.: Алгоритмические основы обработки структурной информации (Вычислительные системы, вып. 85). Новосибирск, 1981, с. 3-20.

9. СКОРОБОГАТОВ В.А. Относительные разбиения и слои графов. -В кн.: Вопросы обработки информации при проектировании систем (Вычислительные системы, вып. 69). Новосибирск, 1977, с. 3-10.

10. ХАРАРИ Ф. Теория графов. -М.: Мир, 1973. - 300 с.

11. MOON J.,MOSER L. On cliques in graphs. - Israel J.Math. , 1965,v.3, N 1, p.23-28.

12. GHAHRAMAN D.E., WONG A.K.C.,AU T. Graph optimal monomorphism algorithms. - IEEE Transactions on Systems, Man and Cybernetics, 1980, v.10, N 4,p.181-188.

13. BARROW H.G., BURSTALL R.M. Subgraph isomorphism, matching relational structures and maximal cliques. - Inform.Process.Lett., 1976,v.4, N 4,p.83-84.

14. ВИЗИНГ В.Г. Сведение проблемы изоморфизма и изоморфного вхождения к задаче нахождения неплотности графа. -В кн.: Тезисы докл. III Всесоюзн. конф. по пробл. теоретической кибернетики. Новосибирск, 1974, с.124-125.

15. СКОРОБОГАТОВ В.А. О распознавании изоморфизма неориентированных графов. -В кн.: Вычислительные системы, вып. 33. Новосибирск, 1969, с. 34-36.

16. СКОРОБОГАТОВ В.А. Нахождение общих частей в семействах графов. -В сб.: Прикладные задачи на графах и сетях. Материалы Всесоюзного совещания. Новосибирск, 1981, с. 117-132.

17. Сен ГУПТА А., ПАЛИТ А. Об образовании клик с помощью булевых уравнений. -ТИИЭР, 1979, т. 67, № 1, с. 197-198.

18. A new algorithm for generating all the maximal independent sets/ Tsukiyama S., Ide M., Ariyoshi H., Shirakawa I. - SIAM J.Comput.,1977,v.6, N 3,p.505-517.

Поступила в ред.-изд.отд.
23 января 1982 года

Таблица I

Протокол работы алгоритма РАЗБОР $(2,2,O_1,B_1,H_1)$

№ шага	Метка	s	v_s	j_s	G_s	Правило остановки выполнено?	BUF
I	2	3	4	5	6	7	8
1	1	0			$\langle\{1,2,3,4,5,6\}\rangle$		$\emptyset$
2	2					нет	
3	3		1	0			
4	6	00			$\langle\{1,2,3,4\}\rangle$		
5	2					нет	
6	3		2	0			
7	6	000			$\langle\{1,2,3\}\rangle$		
8	2					да	
9	4						$\{1,2,3\}$
10	5	00		1			
11	6	001			$\langle\{1,3,4\}\rangle$		
12	2					да	
13	4						$\{1,2,3\},\{1,3,4\}$
14	5	00		2			
15	5	0		1			
16	6	01			$\langle\{2,3,4,5,6\}\rangle$		
17	2					нет	
18	3		2	0			
19	6	010			$\langle\{2,3,5,6\}\rangle$		
20	2					нет	
21	3		3	0			
22	6	0100			$\langle\{3,4,5\}\rangle$		
23	2					да	
24	4						$\{1,2,3\},\{1,3,4\}$ $\{3,4,5\}$
25	5	010		1			
26	6				$\langle\{3,5,6\}\rangle$		
27	2					да	
28	4						$\{1,2,3\},\{1,3,4\}$ $\{3,4,5\},\{3,5,6\}$
29	5	010		2			
30	5	01		1			
31	6	011			$\langle\{3,4,5,6\}\rangle$		
32	2					нет	
33	3		3	0			
34	6	0110			$\langle\{3,4,5\}\rangle$		
35	2					нет	
36	3		4	0			
37	6	01100			$\langle\{3,4\}\rangle$		
38	2					да	
39	4						$\{1,2,3\},\{3,4\},$ $\{1,3,4\},\{3,4,5\},$ $\{3,5,6\}$

Продолжение таблицы I

I	2	3	4	5	6	7	8
40	5	0 1 1 0		1			
41	6	0 1 1 0 1			⟨{3,5}⟩		
42	2	0 1 1 1 0				да	{1,2,3},{1,3,4},{3,4,5},{3,5,6},{3,4},{3,5}
43	4						
44	5	0 1 1 0		2			
45	5	0 1 1		1			
46	6	0 1 1 1			⟨{4,5,6}⟩		
47	2					нет	
48	3		4	0			
49	6	0 1 1 1 0			⟨{4,6}⟩		
50	2					да	
51	4						{1,2,3},{1,3,4},{3,4,5},{3,5,6},{3,4},{4,6},{3,5}
52	5	0 1 1 1		1			
53	6	0 1 1 1 1			⟨{5,6}⟩		
54	2					да	
55	4						{1,2,3},{1,3,4},{3,4,5},{3,5,6},{3,4},{4,6},{5,6},{3,5}
56	5	0 1 1 1		2			
57	5	0 1 1		2			
58	5	0 1		2			
59	5	0					
60	5						
61	7						

Таблица 2

Протокол работы алгоритма O_2

№ шага	Метка	s'	j	Правило остансвки выполнено?
1	1	0 1 1		
2	2			
3	3		1	
4	4	0 1		
5	5			
6	6			
7	2			
8	3		1	
9	4	0		
10	5			
11	6			да
12	8			

Таблица 3

Оценки трудоемкости алгоритмов поиска клик на графах Муна-Мозера

Алгоритм	Характеристика трудоемкости	Число вершин в графе Муна-Мозера				
		15	18	21	24	27
Брона-Кербоша	$\tilde{\eta}$	485	1457	4373	13121	39365
	время	59,56 сек	2 мин 55,84 сек	8 мин 49,38 сек	26 мин 15,64 сек	58 мин 10,24 сек*)
РАЗБОР (2,2, O_3,B_2,H_2)	$\tilde{\eta}$	485	1457	4373	13121	39365
	время	1 мин 35,08 сек	4 мин 35,54 сек	14 мин 48,76 сек	47 мин 33,8 сек	106 мин 47,4 сек*)
РАЗБОР (2,2, O_2,B_2,H_2)	$\tilde{\eta}$	244	730	2188	6562	19684
	время	29,12 сек	1 мин 19,4 сек	3 мин 46,7 сек	11 мин 2,6 сек	32 мин 59,8 сек

Примечание: Время счета, помеченное *), получено на ЕС-1050; остальное время счета получено на ЕС-1033.

Таблица 4

Оценки числа вершин дерева разбора

Порядок графа p	Плотность графа			
	m = 3	m = 4	m = 5	m = 6
I	2	3	4	5
9	18,31,39,61			
12	46,69,99,141			
15	110,173,207, 269		244,727, 727, 1025	
16		86,159,243, 645		
18	193,351,375, 457			730,2185,2185, 4135
20		235,391,643, 1451		
21	318,591,615, 717			
24		637,1067, 1475,2885		
25			511,963,1763, 8169	
28		1324,2475, 2979,5111		
30	1153,1867, 1881,2049			
32		2472,4849, 5443,8641		
35			4378,7723, 12363,38413	
36				-,3687,6825, 127719
40		7376,13967, 14643,21795	10439,19907, 27263,72225	
42				11274,19661, 44667,290465
45			21770,-,-, 127525	
48				34957,63973, 118851, 612575
50			41893,87501, 100503,215927	

Таблица 5

Протокол работы алгоритма РАЗБОР $(2,2,O_4,B_3,H_4)$

№ шага	s	$V(G_s)$	BUF	O_4 выполнено?	j_s	U_s	v_s
1	0	[1,36]	∅				
2				нет			
3					0	[1,6]	1
4	00	{1,8,9,12,14,15,18,22, 23,28,29,32,33,36}					
5				нет			
6					0	{23,29}	23
7	000	{1,8,12,14,18,23,32,36}					
8				да			
9			графы неизоморфны				
10	00				1		
11	001	{1,8,9,12,14,15,18,22, 28,29,32,33,36}					
12				нет			
13					0	{29}	29
14	0010	{1,8,12,14,18,29,32,36}					
15				да			
16			графы неизоморфны				
17	001				1		
18	0011	{1,8,12,14,18,32,36}					
19				да			
20			графы неизоморфны				
21	001				2		
22	00				2		
23	0				1		
24	01	[2,36]					
25				нет			
26					0	[2,6]	2

Таблица 6

Характеристики трудоемкости алгоритмов распознавания изоморфизма

Графы	PROBA		PROB1	
	Число вершин дерева разбора	Время	Суммарное число вершин дерева разбора	Время
G_1, G_2	162	4 мин 12,8 сек	0	5,62 сек
G_3, G_4	282	15 мин 5,14 сек	0	12,7 сек
G_5, G_6	282	16 мин 32,8 сек	0	11,94 сек
G_3, G_6	282	12 мин 13,1 сек	22	7 мин 36,4 сек
G_7, G_8	146	14 мин 44,56 сек	0	32,3 сек
G_9, G_{10}	626	60 мин 5,84 сек	50	11 мин 57,08 сек
G_{11}, G_{12}	610	2 часа 20 мин 21,08 сек	86	1 час 3 мин 30,24 сек

Примечание: Результаты для графов G_{11}, G_{12} получены на машине ЕС-1050; для всех остальных графов на ЕС-1033.