An Exhaustive Search Algorithm for Finding Hamiltonian Cycles

Dainis Zeps

1980

In Russian published in:
Elektronische Informationsverarbeitung und Kybernetik
EIK 16 (1980) 1-3, 69-75
(Journal of Information Processing and Cybernetics)
http://www.informatik.uni-trier.de/~ley/db/journals/eik/eik16.html

Abstract (1980)

The paper suggests an exhaustive search algorithm for finding Hamiltonian circuits in an undirected graph based on depth-first search and working successfully on sparse graphs. The method is based on the idea not to search all palms in the graphs but only those which are not branched.

Tutte's 46 vertices graph searching took less than **3** minutes on EC-1022 computer.

Elektronische Informationsverarbeitung und Kybernetik
EIK 16 (1980) 1 – 3, 69 – 75
(Journal of Information Processing and Cybernetics)

Переборный алгоритм поиска гамильтоновых циклов

Д. А. Зепс

Резюме. Предлагается переборный алгоритм для поиска гамильтоновых циклов в неориентированном графе, основанный на поиске „сначала вглубь" и работающий удовлетворительно с графами с малой средней степенью вершин. 46-вершинный граф Татта программа на ЭВМ ЕС-1022 обработала за менее чем 3 минуты.

Важность проблемы поиска гамильтоновых циклов в графе находит отражение в многочисленной научной литературе, посвященной ей. Примеры точных и эффективных с графо-теоретической точки зрения алгоритмов имеются в работах [4, 6]. Однако на практике по известным причинам подобные универсальные алгоритмы не находят широкого применения. Имеющиеся быстродействующие эвристические алгоритмы, например [2], не применимы для графов с достаточно малым числом гамильтоновых циклов или вообще без них. В настоящей работе предложен переборный алгоритм поска гамильтоновых циклов, где объект перебора находится поиском „сначала вглубь" [1]. Алгоритм в сущности является ограниченным перебором Тирнана [7] и для графов с малой средней степенью вершин (например, кубических) оказывается достаточно эффективным для приложений.

В работе автор придерживается терминологии [3] относительно графо-теоретических вопросов. Рассматриваются только неориентированные графы. Подграф графа G, порожденный множеством вершин S, обозначим через $G(S)$. Множество вершин, смежных вершине v, обозначим через $\mathrm{adj}(v)$. Цепь $v_1 - v_2 - \ldots - v_i$ с выделенной первой вершиной v_1 будем называть *путем* с корнем v_1 и записывать выражением $v_1 \overset{*}{-} v_i$. Вершину r на пути p будем называть *отцом* вершины t, если имеем p: $u \overset{*}{-} r - t \overset{*}{-} v$. Будем говорить, что путь q *индуцирован* путем p, если, во-первых, пути имеют общий корень и, во-вторых, произвольная другая вершина из пути q содержится и в пути p и их отцы совпадают. Если путь q не совпадает с путем p, то q называется *собственным* индуцированным путем пути p.

Определение 1. Путь $u \overset{*}{-} v$ называется *максимальным*, если каждая вершина $w \in \mathrm{adj}(v)$ принадлежит этому пути.

Очевидно, любой гамильтонов путь максимальный.

Определение 2. Путь называется *минимальным разделяющим* (МР-путем), если после выбрасывания его вершин из данного связного графа последний становится несвязным (пустой граф будем считать несвязным), однако никакой из собственных индуцированных путей рассматриваемого пути таким свойством не обладает.

В силу определения 2 гамильтонов путь является минимальным разделяющим.

Легко установить справедливость следующей леммы.

Лемма 1. *Любой максимальный путь связного графа индуцирует один и только один МР-путь.*

Множество максимальных путей с корнем u в подграфе $G(S)$ обозначим через $X(u, S)$, причем будем писать $X(u)$, если речь пойдет о всем графе. Аналогично обозначим множества МР-путей — $Y(u, S)$ и гамильтоновых путей — $H(u, S)$. В силу леммы 1 и определения МР-пути имеем $H(u, S) = X(u, S) \cap Y(u, S)$.

Лемма 2. *Пусть граф G содержит гамильтонов цикл. Тогда для любого u оба множества, $X(u)$ или $Y(u)$, содержат гамильтонов путь $u \xrightarrow{*} v$ со свойством $u \in \mathrm{adj}(v)$.*

Следуя лемме 2, поиск гамильтоновых циклов можно свести к перебору множеств $X(u)$ и $Y(u)$. Ранний алгоритм просмотра графа, соответствующий перебору множества $X(u)$, предложил Тирнан [7]. Однако в [7] нерекурсивное выполнение алгоритма не дает желаемой наглядности. Джонсон [5] пользовался одной версией ограниченного просмотра Тирнана для нахождения всех элементарных циклов в графе. Приведем алгоритм перебора множества $X(u)$ в виде, как он нам понадобится дальше.

Пусть V — множество вершин графа G, $S \subset V$ и $s \notin S$. Множество максимальных путей подграфа $G(S \cup \{s\})$ с корнем s и с фиксированным первым ребром $s - t$ ($t \in S$) обозначим через $s - X(t, S)$.

Лемма 3. *Если $S \subset V$, $v \in S$ и $\mathrm{adj}(v) \cap S \neq \varnothing$, то*

$$X(v, S) = \bigcup_{w \in \mathrm{adj}(v) \cap S} v - X(w, S \setminus \{v\}) .$$

Доказательство непосредственное.

Лемма 3 позволяет прямо по рекурсивному свойству множества максимальных путей написать соответствующую процедуру:

Программа 1

```
begin
  procedure XSET (v, S);
    comment вершина v — текущая;
    begin
1.        for w ∈ adj (v) ∩ S do
2.            XSET (w, S \ {v});
    end XSET;
3.  V:= множество вершин данного графа;
4.  XSET (u, V);
    comment вершина u — корень;
end
```

Текущая вершина вместе с содержимым её стека рекурсии образуют путь с корнем u. Будем говорить, что этот путь просмотрен. По лемме 3 программа 1 осуществляет просмотр всех максимальных путей (с корнем u) данного графа без повторений.

Легко видеть, что для большинства графов $|Y(u)| / |X(u)| < 1$, причем при уменьшении средней степени вершин это отношение уменьшается. Поэтому мы считаем целесообразным перебирать множество $Y(u)$. Для этого, т.е. для просмотра МР-путей, нам необходимо прекратить описанный в программе 1 просмотр вершин текущего пути, как только граф $G(S \setminus \{v\})$ становится несвязным. С учетом этого получаем программу:

Программа 2

```
   begin
     procedure YSET (v, S);
       begin
1.       for w ∈ adj(v) ∩ S do
2.         if граф G(S \ {v}) связный then
3.           YSET(w, S \ {v});
       end YSET;
4.   V := множество вершин данного графа;
5.   YSET(u, V);
   end
```

Чтобы стеки рекурсии не содержали подмножества множества V, программу 2 можно естественным образом модифицировать:

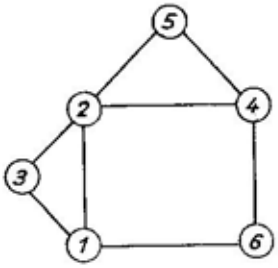
Программа 3

```
   begin
     procedure YSET(v);
       comment вершина v — текущая;
       begin
1.       I := I + 1; NUM(v) := I;
2.       for w ∈ adj(v) do
3.        if NUM(w) = 0 then
4.          if граф G({X | NUM(x) = 0}) связный then
5.            YSET(w);
6.            NUM(v) := 0; I := I − 1;
       end YSET;
7.   I := 0;
8.   for k := 1 until N do NUM(k) := 0;
     comment N — число вершин графа;
9.   YSET(u);
   end
```

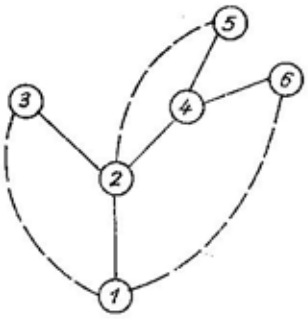
Массив NUM содержит новую нумерацию вершин, генерируемую просмотром, причем, если вершина x не принадлежит текущему пути (т.е. не просмотрена), то NUM(x) равно нулю.

По мнению автора переборный алгоритм можно считать практически эффективным только в случае, если объект перебора находится в линейное (или близкое к линейному) время взамен обычного требования полиномиальности. В программе же 3 при просмотре каждой вершины пути необходимо проверить связность некоторого графа, т.е. при каждой вершине требуется $O(M)$ времени [1] (M — число ребер данного графа).

Покажем, что программу 3 можно модифицировать таким образом, что для фиксирования всего МР-пути потребуется $O(M)$ времени.



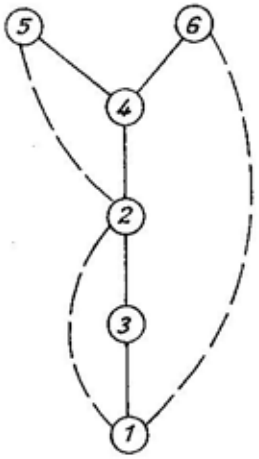
а) Исследуемый граф



в) Пальма, генерируемая первым ПСГ. Низшей вершиной ветвления является вершина с номером 2

$X(1)$	$Y(1)$
1 2 3	1 2
1 2 4 5	
1 2 4 6	
1 2 5 4 6	
1 3 2 4 5	1 3 2 4
1 3 2 4 6	
1 3 2 5 4 6	1 3 2 5 4 6
1 6 4 2 3	1 6 4 2
1 6 4 2 5	
1 6 4 5 2 3	1 6 4 5 2 3

б) Множества $X(u)$ и $Y(u)$, где корнем выбрана вершина с номером 1



г) Пальма, генерируемая вторым ПСГ. Низшей вершиной ветвления является вершина с номером 4

Рис. 1. Пример множеств максимальных и МР-путей и пальм, генерируемых первыми двумя просмотрами „сначала вглубь" данного графа

Для этой цели, начиная поиск очередного МР-пути, произведем обход всех непросмотренных вершин графа просмотром „сначала вглубь" (ПСГ). Каждый такой просмотр порождает ориентированный граф, который навзывается *пальмой* [1] (Рис. 1). Вершины, которые являются отцами (в дереве пальмы на рис. 1 отмечены непрерывными линиями) для более чем одной вершины, будем называть вершинами ветвления соответствующего ПСГ (например, вершины с номерами 2 и 4 на рис. 1в). Выделим среди этих вершин ближайшую по дереву от корня пальмы, которую будем называть чизшей вершиной ветвления (НВВ). С учетом определения 2 и свойств пальмы [1] легко убедиться, что путь по дереву от корня до НВВ является МР-путем. Таким образом, наша задача сводится к отысканию НВВ очередного ПСГ.

Отметим среди вершин графа те, которые становятся текущими после рекурсивных возвратов во время ПСГ, пока все вершины ещё не исчерпаны. Легко убедиться, что среди них вершина с минимальным новым номером является НВВ. Этот номер находим, модифицируя после каждого рекурсивного возврата специальную переменную, начальное значение которой достаточно большое. Тогда НВВ станет определенной при достижении просмотром последней вершины графа и последующий возврат необходимо прервать, как только НВВ станет текущей.

Далее, аналогично программе 3, начинаем поиск нового МР-пути (после того как НВВ и, если необходимо, ещё некоторые вершины рекурсивно удалены из текущего пути). Заметим, однако, что при возвратах нельзя сразу бывшие текущие вершины отмечать непросмотренными. Это можно делать только после достижения НВВ, когда отмечаемые вершины (не принадлежащие текущему пути) собраны в специально отведенном стеке.

Программа перебора МР-путей в указанном смысле имеет следующий вид:

Программа 4

```
    begin
      Boolean FLAG;
      procedure YSET(v, f);
          comment v — текущая вершин, f — её отец;
          begin
1.        I := I + 1; NUM(v) := I;
2.        for w ∈ adj(v) do if NUM(w) = 0 then
            begin
3.            if FLAG then
                comment начинается поиск нового МР-пути;
                begin
4.                FLAG := false; I := SAVE; SAVE := N - 1;
                end;
5.            YSET(w, v);
6.            if I ¬= N then
7.              SAVE := min(SAVE, NUM(f));
              else
8.            begin
                if NUM(v) = SAVE then
                  begin
9.                  while(VSTACK не пустой) do
                    begin
                      для вершины x из VSTACK:
10.                   NUM(x) := 0;
11.                   x удалять из VSTACK;
                    end
12.                 FLAG := true;
                  end
              end
            end
```

```
13.        записать вешину v в стек VSTACK;
           comment если I = N и SAVE = N − 1 то
           вершина v вместе со своим стеком
           рекурсии образуют гамильтонов путь;
14.        if FLAG then SAVE := SAVE − 1;
         end YSET;
15.    опустошить стек VSTACK;
16.    массив NUM заполнить нулями;
17.    I := 0; SAVE := N − 1; FLAG := false;
18.    YSTET (u, 0);
     end
```

В программе 4 при помощи переменной SAVE находится новый номер отца НВВ, что равносильно нахождению самого НВВ. В стеке VSTACK собираются вершины, не принадлежащие текущему пути. Логическая переменная FLAG служит для организации перехода к новому ПСГ после достижения отца НВВ: в те моменты, когда FLAG имеет значение „истинно", ищется вершина, которая могла бы служить продолжением нового МР-пути (здесь также возможны рекурсивные отступления, и переменная SAVE играет при этом роль нового номера текущей вершины).

Фиксирование МР-пути, т. е. нахождение НВВ, требует полного ПСГ всех непросмотренных вершин графа. Это требует $O(M)$ времени [1]. Заполнение вершинами и освобождение от них стека VSTACK не ухудшает временную сложность ПСГ, так как количество этих вершин не превышает N.

Таким образом, программа 4 корректно перебирает все МР-пути без повторений за общее время $O(|Y(u)| \times M)$.

Вариант программы 4 на ФОРТРАНе обрабатывал 46-вершинный граф Тамма ([3], стр. 87) менее чем за 3 минуты (ЭВМ ЕС-1022).

Выражаю благодарность *Кикусту П. Б.*, оказавшему большую помощь при подготовке этой статьи.

Литература

[1] *Aho, A. V., J. E. Hopcroft, J. D. Ullman*, The Design and Analysis of Computer Algorithms. Addison-Wesley Publishing Company, 1974.

[2] *Angluin, D., L. G. Valiant*, Fast Probabilistic Algorithms for Hamiltonian Circuits and Matchings. J. Comp. Syst. Sci. **18** (1979), 155—193.

[3] *Харари Ф.*, Теория графов. Изд. Мир, Москва 1973.

[4] *Held, M., R. M. Karp*, A dynamic programming approach to sequencing problems. J. Soc. Industr. Appl. Math. **10** (1962) 1, 196—210.

[5] *Johnson, D. B.*, Finding All the Elementary Circuits of a Directed Graph. SIAM J. Comput. **4** (1975) 1, 77—84.

[6] *Коробков В. К., Р. Е. Крицевский*, Некоторые алгоритмы для решения задачи коммивояжера. В: Математические модели и методы оптимального планирования; Новосибирск; Изд. Наука, 1966; стр. 106—108.

[7] *Tiernan, J. C.*, An Efficient Search Algorithm to Find the Elementary Circuits of a Graph. Comm. ACM **13** (1970) 12, 722—726.

Kurzfassung

Es wird ein umfassender Suchalgorithmus zum Auffinden der Hamiltonschen Kreise in ungerichteten Graphen angegeben, der für Graphen mit geringer mittlerer Valenz effektiv arbeitet. Der Algorithmus ist in FORTRAN programmiert; für den Tutte-Graphen mit 46 Knoten werden die Hamiltonschen Kreise auf der Anlage ES 1022 in weniger als 3 Minuten bestimmt.

Abstract

The paper suggests an exhaustive search algorithm for finding Hamiltonian circuits in an undirected graph based on depth-first search and working successfully on sparse graphs. Tutte's 46 vertices graph searching took less than 3 minutes on EC-1022 computer.

(Поступило: первая редакция 18/VI/1979,
настоящая редакция 3/I/1980)

Адрес автора:

Зепс Д. А.
Бульв. Райня 29
Вычислительный центр ЛГУ
г. Рига
Латвийская ССР
СССР