ДИСКРЕТНЫЙ АНАЛИЗ И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ
Июль—декабрь 2006. Серия 2. Том 13, № 2. 31–43

УДК 519.6

ОБХОДЫ С УПОРЯДОЧЕННЫМ ОХВАТЫВАНИЕМ В ПЛОСКИХ ГРАФАХ

Т. А. Панюкова

Описан алгоритм построения покрытия плоского связного графа без висячих вершин минимальной по мощности последовательностью цепей с упорядоченным охватыванием и доказана его результативность. Вычислительная сложность алгоритма равна $O\left(|E| \cdot \log_2 |V|\right)$.

Введение

Многие задачи нахождения маршрутов, удовлетворяющих определённым ограничениям, появились из конкретных практических ситуаций. В задачах раскроя листового материала плоский граф является моделью раскройного плана, а маршрут, покрывающий все рёбра, определяет траекторию режущего инструмента. При этом требуется, чтобы отрезанная от листа часть не требовала дополнительных разрезаний.

Если H_S — топологическое представление абстрактного планарного графа H на плоскости S, то H_S в соответствии с [1] называют *плоским графом*, компоненты связности множества $S \backslash H_S$ называют *гранями* плоского графа H_S, а множество граничных точек грани — её *краем*; теоретико-множественное объединение краёв всех граней равно H_S. Поэтому каждое ребро и каждая вершина принадлежат краю хотя бы одной грани, и ясно также, что никакое ребро (в отличие от вершины) не может принадлежать краям более чем двух граней. Ровно одна из граней плоского графа H_S является *внешней* (бесконечной). Остальные грани, в отличие от внешней, называют *внутренними*.

Пусть на плоскости S задан плоский эйлеров граф $G = (V, E)$, и пусть f_0 — внешняя (бесконечная) грань графа G. Для любой части графа $J \subseteq G$ обозначим через $\mathsf{Int}\,(J)$ теоретико-множественное объединение его внутренних граней. Множества вершин, рёбер и граней графа J будем обозначать через $V(J)$, $E(J)$ и $F(J)$ соответственно, а через $|M|$ – число элементов множества M.

© 2006 Панюкова Т. А.

Любой маршрут в графе G будем рассматривать как часть графа, содержащую все вершины и рёбра, принадлежащие маршруту. Тогда с формальной точки зрения сформулированное выше ограничение требует отсутствия пересечения внутренних граней любой начальной части маршрута в заданном плоском графе G с рёбрами в его оставшейся части [7].

Представление плоского графа $G = (V, E)$ с точностью до гомеоморфизма однозначно определяется заданием для каждого ребра $e \in E$ следующих функций [7]:

$v_1(e)$, $v_2(e)$ — вершины, инцидентные ребру e;

$f_k(e)$ — грань, находящаяся слева при движении по ребру e от вершины $v_k(e)$ к вершине $v_{3-k}(e)$, $k = 1, 2$;

$l_k(e)$ — ребро, принадлежащее грани $f_k(e)$ и инцидентное вершине $v_k(e)$, $k = 1, 2$.

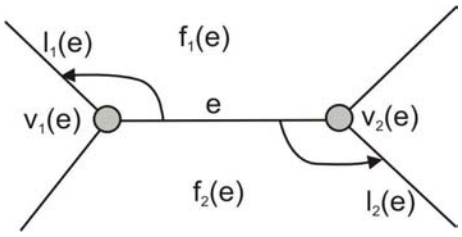


Рис. 1. Представление плоского графа

Введённые функции изображены на рис. 1. Их построение не составляет проблем. Сложность такого представления равна $O\left(|E| \cdot \log_2 |V|\right)$.

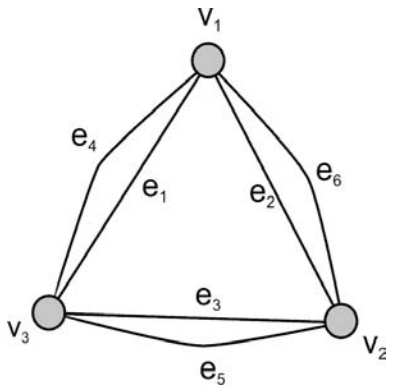


Рис. 2. Пример эйлерова графа

В соответствии с [7] будем говорить, что цепь $C = v_1 e_1 v_2 e_2 \dots e_k v_{k+1}$ имеет *упорядоченное охватывание*, если для любой её начальной части $C_l = v_1 e_1 v_2 \dots e_l$, $l \leqslant k$, выполнено условие $\mathsf{Int}(C_l) \bigcap E = \varnothing$.

Например, для плоского эйлерова графа, изображённого на рис. 2, цикл $v_1e_1v_3e_3v_2e_2v_1e_4v_3e_5v_2e_6v_1$ удовлетворяет условию упорядоченного охватывания, а цикл $v_1e_4v_3e_5v_2e_6v_1e_1v_3e_3v_2e_2v_1$ не удовлетворяет, так как $\mathsf{Int}\,(v_1e_4v_3e_5v_2e_6v_1) \supset \{e_1, e_2, e_3\}$.

Существование эйлеровых циклов с упорядоченным охватыванием в плоских эйлеровых графах доказано в [6, 7]. Рекурсивные алгоритмы построения таких циклов представлены в [4, 7]. Алгоритм построения маршрута с упорядоченным охватыванием в плоском графе произвольного вида предложен в [5]. Он имеет вычислительную сложность не более $O\left(|E|^2\right)$. В [6] предложен эффективный алгоритм построения циклов с упорядоченным охватыванием в плоских эйлеровых графах, имеющий вычислительную сложность $O\left(|E| \cdot \log_2 |V|\right)$. Возможность покрытия неэйлеровых плоских графов последовательностью цепей с упорядоченным охватыванием анонсировалась в [2, 3].

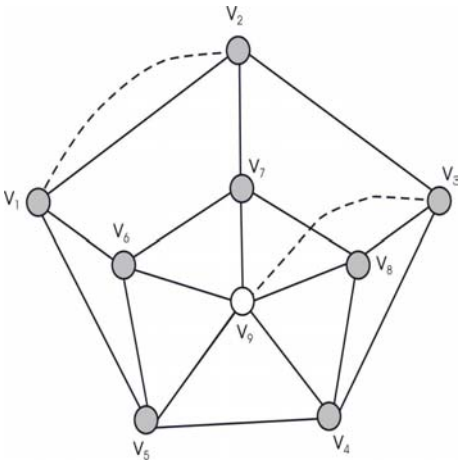


Рис. 3.

Цель настоящей статьи — описание алгоритма построения покрытия произвольного плоского связного графа без висячих вершин последовательностью цепей с упорядоченным охватыванием, а также доказательство его результативности.

1. Построение покрытия плоского графа последовательностью цепей с упорядоченным охватыванием

Легко заметить, что любой граф можно достроить до эйлерова добавлением $n/2$ рёбер, где n — число вершин нечётной степени в исходном графе. При условии сохранения планарности данного представления

графа дополнение $n/2$ рёбер возможно лишь в случае, когда в графе найдётся такое разбиение вершин нечётной степени на пары, что вершины одной пары окажутся инцидентными одной грани. На рис. 3 этому требованию удовлетворяют вершины v_1, v_2 и v_3. Для вершины v_9 такой пары не существует. Следовательно, в общем случае нужно искать другие решения.

В [3] доказано следующее утверждение.

Теорема 1. *Пусть $G = (V, E)$ — плоский связный граф на S без висячих вершин. Существует множество рёбер F, $(F \cap S)\backslash V = \varnothing$, такое, что граф $\hat{G} = (V, E \cup F)$ — эйлеров, и в $\hat{G}$ существует эйлеров цикл $C = v_1e_1v_2e_2...e_nv_1$, $n = |E| + |F|$, для любой начальной части которого $C_l = v_1e_1v_2e_2...v_l$, $l \leqslant |E| + |F|$, выполнено условие $\mathsf{Int}(C_l) \cap E = \varnothing$.*

Доказательство теоремы даёт результативность приведённого ниже алгоритма OrderedEnclosingCover, который строит покрытие плоского графа G последовательностью цепей с упорядоченным охватыванием. Граф G представлен списком рёбер с заданными на них функциями $v_k(e)$, $l_k(e)$, $f_k(e)$, $k = 1, 2$.

АЛГОРИТМ OrderedEnclosingCover

(Input: $G = (V, E)$ — плоский граф; odd V — множество вершин нечётной степени.
Output: first $\in E$, last $\in E$, $\text{mark}_1 \mid E \to E$).
begin
Initiate( );
Order( );
Сортировка списка вершин нечётной степени по убыванию ранга:
SortOdd( );
while (odd $V \neq \varnothing$) **do**
$v^0 = \arg \max_{v \in \text{odd } V} \text{kmark}(v)$;
odd V = odd $V\backslash\{v^0\}$;
v=FormChain(v^0);
odd V = odd $V\backslash\{v\}$;
od
FormChain(v_0);
end.

При описании и анализе алгоритма будем использовать обозначения $\overline{v}_k(\ )$, $\overline{l}_k(\ )$, $\overline{f}_k(\ )$, $k = 1, 2$, для функций, построенных алгоритмом, в отличие от первоначально заданных функций $v_k(\ )$, $l_k(\ )$, $f_k(\ )$, $k = 1, 2$.

В теле алгоритма также формируются дополнительные функции:

Stack: $V \to E$ — указатель на очередь v-списка M2-помеченных рёбер;

$\text{mark}_k(\ ) : E \to E$ и $\text{prev}_k(\ ) : E \to E$, $k = 1, 2$, для организации двусвязных списков с целью обеспечения операций вставки/удаления за время $O(1)$.

Кроме того, функция $\mathrm{mark}_1(\ )$ используется для представления результата выполнения алгоритма.

Алгоритм OrderedEnclosingCover состоит из последовательного выполнения процедур Initiate, Order, SortOdd( ), $n/2$-кратного выполнения FormChain( ) ($n = |\mathrm{odd}\ V|$) и, наконец, при необходимости ещё одного выполнения процедуры FormChain( ).

В алгоритме также используется следующая процедура

Процедура REPLACE
(Input: e — ребро, для которого нужно изменить нумерацию вершин)
begin
$t_1 = v_2(e)$; $t_2 = l_2(e)$;
$v_2(e) = v_1(e)$; $l_2(e) = l_1(e)$;
$v_1(e) = t_1$; $l_1(e) = t_2$;
end.

Данная процедура переопределяет введённые функции $v_k(e)$, $l_k(e)$, $f_k(e)$, $k = 1, 2$, так, что в построенном алгоритмом цикле движение по ребру $e \in E$ происходит от вершины $v_2(e)$ к вершине $v_1(e)$.

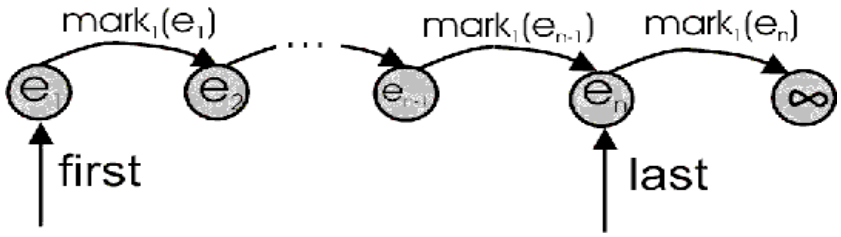


Рис. 4. Организация очереди M1-помеченных рёбер

В теле следующей процедуры

Процедура Initiate
begin
$(\forall v \in V)$ **do** $\mathrm{Stack}(v) = \varnothing$ **od**;
$\forall e \in E$ **do**
$\mathrm{mark}_1(e) = \mathrm{mark}_2(e) = \infty$;
$\mathrm{prev}_1(e) = \mathrm{prev}_2(e) = 0$;
if $((f_1(e) = f_0)$ **or** $(f_2(e) = f_0))$ $e_0 = e$;
od
if $(f_2(e_0) = f_0)$ REPLACE(e_0);
first=last=e_0; $v_0 = v = v_1(e_0)$; ne= $l_1(e_0)$;
$k = 1$; $\mathrm{kmark}(e_0) = 1$;
end.

присваиваются начальные значения $\mathrm{Stack}(v) = \varnothing$, $v \in V$, $\mathrm{mark}_1(e) = \infty$, $e \in E$, а также определяется ребро $e_0 \in E$, принадлежащее границе внешней грани f_0. Функции на ребре e_0 переопределяются так, что

$\overline{f}_1(e_0) = f_0$, т. е. чтобы ребро $\overline{l}_1(e_0)$ принадлежало границе внешней грани. В теле данной функции инициализируется очередь M1-помеченных рёбер (рис. 4), состоящая из единственного ребра e_0, переменные first и last используются для указания соответственно первого и последнего элементов очереди. Переменная ne используется для определения следующего кандидата для включения в список M1-помеченных рёбер. Переменная v_0 используется для запоминания вершины, принадлежащей границе внешней грани, а переменная v — как текущая вершина для задания ориентации ребра, определяемого переменной ne.

```
Процедура Order
begin
  while ( first ≠ ∞) do
    while (mark(ne)= ∞ and last≠ ne) do
      M1: kmark(ne)= k;
      mark1 (last)=ne;
      if (v2(ne) ≠ v) REPLACE(ne);
      v = v1(ne); last=ne; ne= l1(ne);
    od
    e =first; first=mark1(first); v = v2(e); ne= l2(e);
    M2: k =kmark(e)+1; mark1(e) =Stack(v1(e)); mark2(e) =Stack(v);
    if (mark1(e) ≠ 0) do
      if (v1(e) = v1(mark1(e))) prev1(mark1(e)) = e;
        else prev2(mark1(e)) = e;
    od
    if (mark2(e) ≠ 0) do /* Помещение ребра в стеки вершин v1(e) и v2(e) */
      if (v = v1(mark2(e))) prev1(mark2(e)) = e; else prev2(mark2(e)) = e;
    od
    Stack(v) = e; Stack(v1(e)) = e;
  od
end.
```

Функциональное назначение процедуры Order состоит

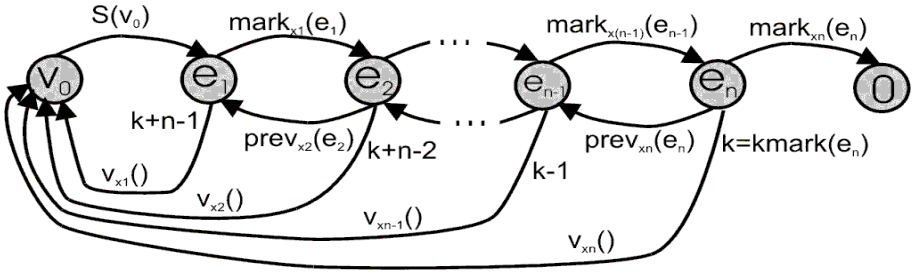
в определении на каждом ребре $e \in E$ значения kmark(e);

в формировании для каждой вершины списка инцидентных рёбер (рис. 5), упорядоченных в порядке убывания значения kmark( ).

В процедуре Order переменная k — число шагов, а функция kmark( ) : $E \to N$ равна номеру шага, на котором ребро ставится в очередь M1-помеченных рёбер.

Данная процедура выполняется следующим образом. На первом шаге в очередь M1-помеченных рёбер вводятся все рёбра $e \in E$, инцидентные внешней грани f_0, а их ориентация задаётся так, чтобы выполнялось равенство $\overline{f}_1(e) = f_0$. На $(k+1)$-м шаге каждое ребро $e \in E$, попавшее в очередь M1-помеченных рёбер на шаге k, переводится в состояние M2-помеченного и помещается в списки вершин $\overline{v}_l(e)$, $l = 1, 2$, (см. рис. 5),

а в очередь M1-помеченных включаются все непомеченные рёбра, инцидентные грани, общей с ребром e.



$$x_k = \begin{cases} 1, \text{ если } v_1(e_k) = v_0; \\ 2, \text{ если } v_2(e_k) = v_0. \end{cases}$$

Рис. 5. Организация v_0-списка M2-помеченных рёбер

Для анализа результативности алгоритма положим

$$E_k = \{e \in E \mid \mathrm{kmark}(e) = k\}\,,\ E_k^* = \{e \in E : \mathrm{kmark}(e) \leqslant k\}\,,\ \overline{E}_k = E \backslash E_k^*.$$

Через $G(E')$ будем обозначать плоский граф, порождённый множеством рёбер $E' \subset E$.

Лемма 1. *Для любого* $k = 1, 2, 3, \ldots, M$, *где* $M = \max\limits_{e \in E} \mathrm{kmark}(e)$,

$$\mathrm{Int}(G(E_k)) \supset G(\overline{E}_k); \quad S \backslash \mathrm{Int}(G(E_k)) \supset G(E_k^*).$$

Доказательство проведём методом математической индукции по k. Из описания алгоритма следует, что $k = 1$ при выполнении процедуры Initiate, а значения kmark( ) $= 1$ устанавливаются только на рёбрах, вводимых в очередь M1-помеченных при первом выполнении тела внешнего цикла в процедуре Order. В соответствии с описанием процедуры Initiate рёбра e_0 и $e_1 = \overline{l}_1(e_0)$ инцидентны внешней грани f_0 графа G, значение переменной ne указывает на ребро e_1, а значение переменной $v = v_1(e_0)$ — на вершину v_1, инцидентную рёбрам e_0 и e_1. Поэтому при первом выполнении тела цикла Order в очередь M1-помеченных рёбур будут последовательно внесены все рёбра цепи $v_1 e_1 v_2 e_2 \ldots e_m v_{m+1}$, удовлетворяющей условиям:

$$v_2 = e_1 = v_1,\ \overline{v}_2(e_{i+1}) = v_{i+1} = \overline{v}_1(e_i),\ 1 \leqslant i \leqslant m,$$
$$e_{i+1} = l_1(e_i),\ \overline{f}_1(e_i) = \overline{f}_1(e_1),\ 1 \leqslant i \leqslant m-1,\ \overline{l}_1(e_m) = e_0.$$

Таким образом, значение kmark( ) = 1 будет определено на всех рёбрах, инцидентных внешней грани f_0 графа G. В этом случае справедливость утверждений леммы очевидна.

Предположим, что доказываемые утверждения имеют место для K, где $k < K \leqslant M$. Рассмотрим множество

$$F_{K-1} = \left\{ e \in E_{K-1} |\ l_2(e) \notin E^*_{K-1} \right\}.$$

Из связности графа G и включения $F_{K-1} \subset \mathsf{Int}(G(E_K))$ следует, что $F_{k-1} \neq \varnothing$. Поэтому в соответствии с описанием алгоритма значение функции kmark( ) = K будет определено на рёбрах, максимальных по включению цепей $C(e) = v_1 e_1 v_2 e_2 \dots e_m v_{m+1}$, $e \in F_{K-1}$, удовлетворяющих условиям

$$v_1 = \overline{v}_2(e),\ e_1 = \overline{l}_2(e);\ \overline{v}_2(e_{i+1}) = v_{i+1} = \overline{v}_1(e_i),\ 1 \leqslant i \leqslant m;$$
$$e_{i+1} = \overline{l}_1(e_i),\ \mathrm{mark}(e_{i+1}) = \infty,\ 1 \leqslant i \leqslant m-1; \overline{f}_1(e_i) = \overline{f}_2(e),\ 1 \leqslant i \leqslant m.$$

Таким образом, $E_K = \bigcup\limits_{e \in F_{K-1}} E\left(C(e)\right)$ и все цепи $C(e)$, $e \in F$, являются непересекающимися по рёбрам. Из связности графа G и отсутствия в нём висячих вершин следует, что последняя вершина v_{m+1} в цепи $C(e)$, $e \in F_{K-1}$, принадлежит $V(E_{K-1})$, где $V(E_{K-1})$ — множество вершин, инцидентных рёбрам из E_{K-1}.

Если цепь $C(e)$, $e \in F_{K-1}$, такая, что $v_1 = v_{m+1}$, то $C(e)$ — цикл. Если $v_1 \neq v_{m+1}$, то в цепи, содержащей ребро $\overline{l}_1(e_m)$, существует единственное ребро $e' \in F_{K-1}$ такое, что $\overline{v}_2(e') = v_{m+1}$, поскольку $G^*(E_{K-1})$ — объединение непересекающихся по рёбрам цепей.

Кроме того, по построению имеем

$$\mathsf{Int}(G(E_K)) = \mathsf{Int}(G(E_{K-1})) \backslash \left(\bigcup_{e \in E_K} \overline{f}_1(e) \right), \tag{1}$$

$$S \backslash \mathsf{Int}(G(E_K)) = \left(\bigcup_{e \in E_K} \overline{f}_1(e) \right) \bigcup (S \backslash \mathsf{Int}(G(E_{K-1}))). \tag{2}$$

Поскольку $\overline{E}_K \subset \overline{E}_{K-1} \subseteq \mathsf{Int}(G(E_{K-1}))$, $\overline{E}_K \not\subset \bigcup\limits_{e \in E_K} \overline{f}_1(e)$, то из (1) следует, что $S \backslash \mathsf{Int}(G(E_K)) \supseteq \overline{E}_K$.

Так как

$$E^*_K \subseteq E^*_{K-1} \bigcup E_K, E_K \subseteq \bigcup_{e \in E_K} \overline{f}_1(e), E^*_{K-1} \subseteq S \backslash \mathrm{Int}(G(E_K))$$

то из (2) следует, что $S\backslash\mathsf{Int}(G(E_K)) \supseteq E_K^*$. Лемма 1 доказана.

Лемма 2. *Если* $M = \max\limits_{e\in E} \mathrm{kmark}(e)$, *то* $\overline{E}_M = \varnothing$.

Доказательство. Предположим противное, т. е. $\overline{E}_M \neq \varnothing$. Из леммы 1 следует, что $\overline{E}_M \subseteq \mathsf{Int}(E_M)$.

Рассмотрим множество $F_M = \left\{ e \in E_M \,\middle|\, \overline{l}_2(e) \in \overline{E}_M \right\}$. Из связности графа G следует, что $F_M \neq \varnothing$. В соответствии с описанием алгоритма

$$(\forall e \in F_M)\,(\mathrm{kmark}(e) = M + 1)\,,$$

т. е. имеем противоречие с условием леммы. Лемма 2 доказана.

Из леммы 2 следует, что на каждом ребре $e \in E$ алгоритм определит значение функции kmark( ), т. е. каждое ребро $e \in E$ будет включено в очередь M1-помеченных рёбер. Так как после включения ребра $e \in E$ в эту очередь $\mathrm{mark}_1(e) \neq \infty$, то такое включение возможно единственный раз. Каждое ребро, попавшее в очередь M1-помеченных рёбер, переводится в состояние M2-помеченного включением его в стеки вершин $\overline{v}_k(e)$, $k = 1, 2$, в порядке, определяемом очередью, т. е. в порядке возрастания величины $\mathrm{kmark}(e)$. Поэтому после завершения процедуры Order для каждой вершины $v \in V$ будем иметь $\mathrm{Stack}(v) = \arg\max\limits_{e\in E(v)} \mathrm{kmark}(e)$;

$$\left(e' \in E(v),\ \mathrm{kmark}(e') > \min_{e\in E(v)} \mathrm{kmark}(e)\right) \\ \Rightarrow \left(\exists e'' = \mathrm{mark}_x(e') \in E(v),\ \mathrm{kmark}(e'') = \mathrm{kmark}(e') - 1\right),$$

где $E(v) = \{e \in E \mid\ (v = \overline{v}_2(e)) \vee (v = \overline{v}_1(e))\}$,

$$x = \begin{cases} 1, & \text{если } v_1(e') = v; \\ 2, & \text{если } v_2(e') = v. \end{cases}$$

Таким образом, результативность процедуры Order доказана.

После выполнения процедуры Order выполняется упорядочение вершин нечётной степени $v \in \mathrm{odd}V$ в порядке возрастания их ранга с помощью процедуры SortOdd. За ранг вершины v принимается значение функции $\mathrm{kmark}(\mathrm{Stack}(v))$. Далее выполняется цикл while...do с использованием процедуры FormChain, в которой строится последовательность из $|\mathrm{odd}V|/2$ простых цепей между парами вершин нечётной степени. Эта процедура строит цепь с упорядоченным охватыванием, начинающуюся в заданной вершине $w \in \mathrm{odd}V$ и заканчивающуюся в некоторой вершине $v \in \mathrm{odd}V$, $v \neq w$.

Процедура FormChain
(Input: w — вершина нечётной степени, из которой будет построена цепь
Output: v — вершина нечётной степени, завершающая построенную цепь)
begin
$v = w$; $e = \mathrm{Stack}(v)$; first = last = e;
do
if $(v_1(e) = v)$ REPLACE(e);
$\mathrm{Stack}(v) = \mathrm{mark}_2(e)$;
if$(v = v_1(\mathrm{mark}_2(e)))$ $\mathrm{prev}_1(\mathrm{mark}_2(e)) = 0$; **else** $\mathrm{prev}_2(\mathrm{mark}_2(e)) = 0$;
$v = v_1(e)$;
if $(\mathrm{prev}_1(e) \neq 0)$
if $(e = \mathrm{mark}_1(\mathrm{prev}_1(e)))$ $\mathrm{mark}_1(\mathrm{prev}_1(e)) = \mathrm{mark}_1(e)$;
else $\mathrm{mark}_2(\mathrm{prev}_1(e)) = \mathrm{mark}_1(e)$;
else do
$\mathrm{Stack}(v) = \mathrm{mark}_1(e)$;
if $(v = v_1(\mathrm{mark}_1(e)))$ $\mathrm{prev}_1(\mathrm{mark}_1(e)) = 0$; **else** $\mathrm{prev}_2(\mathrm{mark}_1(e)) = 0$;
od
if $(v \in \mathrm{odd}(V))$ **do**
$\mathrm{mark}_1(\mathrm{last}) = 0$;
return v;
od
$e = \mathrm{Stack}(v)$;
M3: $\mathrm{mark}_1(\mathrm{last}) = e$; last = e;
od while $(\mathrm{last} \neq 0)$;
return v**;**
end.

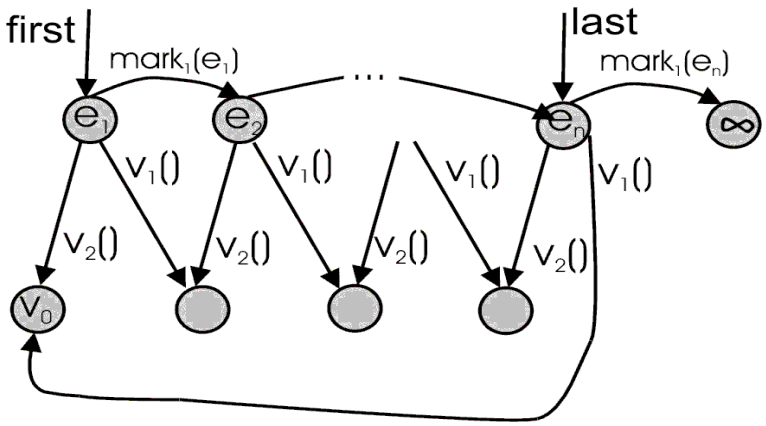


Рис. 6. Организация списка М3-помеченных рёбер

Сначала производится инициализация списка М3-помеченных рёбер (рис. 6), который будет являться представлением построенной цепи. Первоначально этот список состоит из ребра e, находящегося в начале w-

списка M2-помеченных рёбер. Вершина w определяется как текущая вершина v. В цикле do...while с помощью процедуры REPLACE устанавливается $\overline{v}_2(e) = v$ и ребро e исключается из $v_1(e)$- и $v_2(e)$-списков M2-помеченных рёбер, а текущей становится вершина $v_1(e)$. Если $v \notin \mathrm{odd}V$, очередь M3-помеченных рёбер пополняется ребром e, в противном случае процедура возвращает текущую вершину $v \in \mathrm{odd}V$.

Процедура FormChain строит простую цепь $C = v^0 e_1 v_1 e_2 ... e_k v_k$ такую, что $v_1, v_2, ...v_{k-1} \notin \mathrm{odd}V,\ \ v^0, v_k \in \mathrm{odd}V$,

$$e_i = \arg \max_{e \in E(v_i) \setminus \{e_l | l<i\}} \mathrm{kmark}(e),\ v_{i+1} = \overline{v}_1(e_i),\ 1 \leqslant i \leqslant k.$$

Кроме того, для любой начальной части $C_l = v^0 e_1 v_1 e_2 v_2 \dots e_l$, $l \leqslant k$, и для любой вершины $v \in V$ имеет место неравенство

$$\min_{e \in E(v) \bigcap E(C_l)} \mathrm{kmark}(e) > \max_{e \in E(v) \setminus E(C_l)} \mathrm{kmark}(e). \tag{3}$$

Лемма 3. *Для любых* $j = 1, 2, \dots, l$ *и* $m = 1, 2, \dots, M$, *где* $M = \max\limits_{e \in E} \mathrm{kmark}(e)$, *множество* $\mathrm{Int}(C_j) \bigcap E_m$ *пусто.*

Доказательство. Воспользуемся методом математической индукции по переменной m.

Поскольку рёбра множества E_1 образуют цикл, ограничивающий внешнюю грань f_0 графа G, то $(\forall H \subseteq G)\,(E_1 \cap \mathsf{Int}(H) = \varnothing)$, что доказывает справедливость леммы при $m = 1$.

Покажем, что из $\mathsf{Int}(C_j) \cap E_m = \varnothing,\ 1 \leqslant j \leqslant l,\ 1 \leqslant m \leqslant K-1$, где $K \leqslant M$ следует, что

$$\mathsf{Int}(C_j) \cap E_m = \varnothing,\ 1 \leqslant j \leqslant l. \tag{4}$$

Предположим противное, что найдётся l' такое, что

$$\mathsf{Int}(C_{l'}) \cap E_m \neq \varnothing.$$

В силу леммы 1 $\mathsf{Int}(G(E_m)) \supseteq \overline{E_m} = \{e \mid \mathrm{kmark}(e) > m\}$, $1 \leqslant m \leqslant M$. Из связности графа G и леммы 1 следует, что множество $\mathsf{Int}(C_{l'}) \cap E_m$ содержит ребро $e' \in E$ такое, что $v_2(e') = v \in V(C_{l'})$. Поэтому

$$\min_{e \in E(v) \bigcap EC_{l'}} \mathrm{kmark}(e) < \mathrm{kmark}(e') \leqslant \max_{e \in E(v) \setminus EC_{l'}} \mathrm{kmark}(e),$$

что противоречит (3). Таким образом, справедливо равенство (4). Применяя принцип математической индукции, получаем утверждение леммы

$$\mathsf{Int}(C_j) \cap E_m = \varnothing,\ 1 \leqslant j \leqslant l,\ 1 \leqslant m \leqslant M.$$

Лемма 3 доказана.

Доказательство леммы 3 завершает доказательство результативности процедуры FormChain.

В результате выполнения цикла while...do алгоритм OrderedEnclosingCover строит последовательность цепей с упорядоченным охватыванием

$$C^0 = v^0 e_1^0 v_1^0 e_2^0 \dots e_{k_0}^0 v_{k_0}^0,\ C^1 = v^1 e_1^1 v_1^1 e_2^1 \dots e_{k_1}^1 v_{k_1}^1, \dots,$$
$$C^{n-1} = v^{n-1} e_1^{n-1} v_1^{n-1} e_2^{n-1} ... e_{k_{n-1}}^{n-1} v_{k_{n-1}}^{n-1},$$

где $n = |\mathrm{odd}V|/2$, такую, что odd $V = \left\{v^0, v_{k_0}, v^1, v_{k_1}, \dots, v^{n-1}, v_{k_{n-1}}\right\}$ и $\mathsf{Int}(\bigcup_{i=0}^{k} C^i) \cap C^l = \varnothing$ при любых $k < (|\mathrm{odd}V|/2) - 1$ и $l > k$.

После этого выполняется процедура FormChain для вершины v_0, инцидентной с внешней гранью. Результатом выполнения данной процедуры будет цикл с упорядоченным охватыванием C^n, состоящий из рёбер, не содержащихся в цепях C^i, $0 \leqslant i \leqslant n-1$, где $n = |\mathrm{odd}V|/2$.

Легко заметить, что число шагов алгоритма B не превосходит $O\left(|E| \cdot \log_2 |V|\right)$.

Для доказательства теоремы 1 осталось заметить, что в качестве множества рёбер F, существование которого утверждается в теореме, можно взять следующее: $F = \left\{\{v_{k_0}, v^1\},\ \{v_{k_1}, v^2\}, \dots \{v_{k_{n-2}}, v^{n-1}\}\right\}$, если существует вершина нечётной степени, инцидентная с внешней гранью, либо $F = \left\{\{v_{k_0}, v^1\},\ \{v_{k_1}, v^2\}, \dots \{v_{k_{n-1}}, v_0\}\right\}$ в противном случае. Вершины v_{k_i} и v^i, $i = 0, ..., n-1$, принадлежат приведённому выше множеству odd$V$, а вершина v_0 — любая вершина, инцидентная с внешней гранью. Искомым циклом будет

$$C = C^0\{v_{k_0}, v^1\}C^1\{v_{k_1}, v^2\} \dots C^{n-1}\{v_{k_{n-2}}, v^{n-1}\}C^n.$$

Теорема 1 доказана.

2. Заключение

Алгоритм OrderedEnclosingCover строит покрытие плоского связного графа без висячих вершин минимальной по мощности последовательностью цепей с упорядоченным охватыванием за число шагов, не превосходящее $O\left(|E| \cdot \log_2 |V|\right)$.

Автор благодарен А. В. Косточке и Л. С. Мельникову за полезные замечания, сделанные на конференции DAOR'2002.

ЛИТЕРАТУРА

1. **Зыков А. А.** Основы теории графов. М.: Вузовская книга, 2004.

2. **Панюков А. В., Панюкова Т. А.** Эйлеровы циклы с упорядоченным охватыванием // Проблемы теоретической кибернетики. Тезисы докладов XII Международной конференции (Нижний Новгород, 17–22 мая 1999 г.). Часть II. М.: Изд-во механико-математического факультета МГУ, 1999. С. 148.

3. **Панюкова Т. А.** Алгоритм построения эйлеровых циклов специального вида. // Проблемы теоретической кибернетики. Тезисы докладов XIII Международной конференции (Казань, 27–31 мая 2002 г.). Часть II. М.: Изд-во механико-математического факультета МГУ, 2002. С. 142.

4. **Панюкова Т. А.** Построение эйлеровых циклов специального вида в планарном графе // Материалы VII Международного семинара «Дискретная математика и её приложения»; Ч. II М.: Изд-во центра прикладных исследований при механико-математическом факультете МГУ, 2001. С. 149.

5. **Панюкова Т. А.** Построение маршрутов с упорядоченным охватыванием в плоских графах // Труды 36-й Региональной молодежной конференции «Проблемы теоретической и прикладной математики». Екатеринбург: УрО РАН, 2005. С. 61–66.

6. **Panyukov A. V., Panioukova T. A.** The algorithm for tracing of flat Euler cycles with ordered enclosing // Proceedings of Chelyabinsk Scientific Center #4(9), 2000. P. 18–22. http://csc.ac.ru/news/2000_4/2000_4_1_4.zip

7. **Panioukova T. A., Panyukov A. V.** Algorithms for construction of ordered enclosing traces in planar Eulerian graphs // The International Workshop on Computer Science and Information Technologies' 2003 (Ufa, September 16–18, 2003). Proc. Vol. 1. Ufa: Ufa State Technical University, 2003. P. 134–138.

Адрес автора:

Южно-Уральский гос. университет
пр. Ленина, 76,
454080 Челябинск,
Россия.
E-mail: kwark@mail.ru

Статья поступила
14 февраля 2005 г.

Переработанный вариант —
24 сентября 2006 г.