数理解析研究所講究録
第 353 巻 1979 年 68-77

ある種のグラフの極大マッチングを与える線形時間アルゴリズム

東海大 理 夜久竹夫

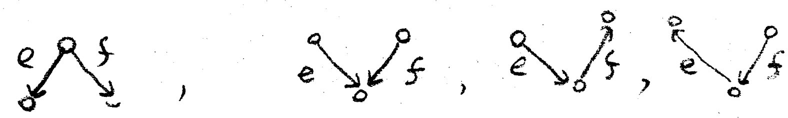
要約 有向グラフを node disjoint な path (chain) の集まりにより覆う問題を考える.(一点からなるグラフを長さ 0 の chain と考える). そのような chain の集まりを chain matching, path cover あるいは単に covering (被覆)などという. chain matching に属する chain の個数が最小のとき、その chain matching を極大という.

我々は与えられた dag に対して極大な chain matching を与えるアルゴリズムを扱う. この問題が交互部分グラフ [5] に対する問題に帰着する事を示し、更に、出次数が高々2であるような dag に対して頂点一線形時間アルゴリズムを示す.

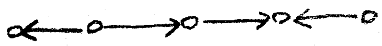
更に、このアルゴリズムはある種の二組グラフに対する極大(辺) matching 問題に適用できる事を示す.

1．準備

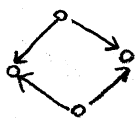
有向グラフをG＝(N, E)で示す．ここでNは頂点の集合，Eは辺の集合である．pathは相異なる辺の列(v_1, v_2) $(v_2, v_3)\cdots(v_{n-1}, v_n)$である．各頂点が相異なるときpathはchainといわれる．二つの辺eとfは(i)それらが相異なり；(ii)それらが一つの頂点を共有するときadjacentという：



semipathは次のような相異なる辺の列$e_1 e_2 \cdots e_n$である：(i)頂点の列$v_1, v_2, \cdots, v_{n+1}$が存在して　(ii) $e_i = (v_i, v_{i+1})$であるか又は、$e_i = (v_{i+1}, v_i)$かである$(1 \leqq i \leqq n)$.



semipathは$e_2 e_3 \cdots e_n e_1$がsemipathのときsemi-cycleという．



有向グラフGにおける頂点vの入次数と出次数，indegree(v, G)とoutdegree(v, G)とそれぞれ示す，をつぎのように定める：

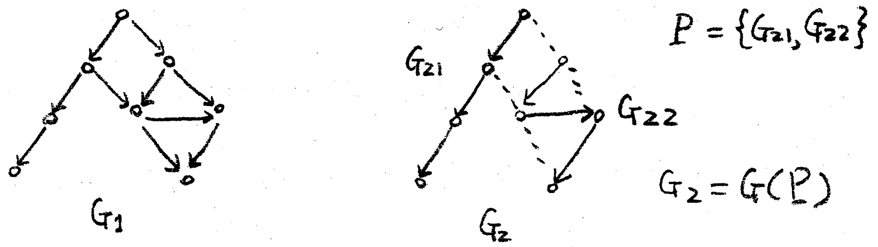
$$\mathrm{indegree}(v, G) = \#\{u \in G ; (u, v) \in E\}$$
$$\mathrm{outdegree}(v, G) = \#\{u \in G ; (v, u) \in E\}.$$

次に chain matching を定める.

$G_1 = (N_1, E_1)$, $G_2 = (N_2, E_2)$, $\cdots$, $G_n = (N_n, E_n)$ を有向グラフ $G = (N, E)$ の部分グラフとする. もし $N_1 \cup N_2 \cup \cdots \cup N_n = N$ かつ $N_i \cap N_j = \phi\ (i \neq j)$ ならば $P = \{G_1, G_2, \cdots, G_n\}$ を G の分割という. 各部分グラフ $G_i = (N_i, E_i)$ を component という. 分割 P は各 G_i がつぎの条件 (i) 又は (ii) を満たすとき chain matching といわれる:

(i) G_i は一つの chain からなる.

(ii) G_i は一点 $(\{v\}, \phi)$ である.



上の P によるグラフ, $G(P) = (N, E(P))$ と書く, は $G(P) = G_1 \cup G_2 \cup \cdots \cup G_n$ のことである.

有向グラフ $G = (N, E)$ の chain matching $M = \{G_1, G_2, \cdots, G_n\}$ が極大であるとは, G の任意の chain matching N に対して次が成り立つことである:

$$\#(E-E(M)) \leqq \#(E-E(N)) \quad \text{即ち}$$
$$\#E(M) \geqq \#E(N).$$

注1. 有向グラフGの極大chain matching $M=\{G_1, G_2, \cdots, G_n\}$ は最小のblock数を持つ. 即ち、Gの任意のchain matching $N=\{H_1, \cdots, H_m\}$ に対して $n \leqq m$ である.

有向グラフからchain matchingを求める問題は有向グラフ状のデータ構造を"線形化"[1]して連続状の記憶構造により表現する問題に関係する. 例えばプログラム・フローチャートをcodingにより線形化する事はそのフローチャートのchain matchingを求めることである. その際に、最小のblock数を伴なうchain matchingは分枝(GOTO文)最小のcodingに対応する[4].

サイクルのない有向グラフ, dagと略す, に対して極大なchain matchingを求めるアルゴリズムに関しては $k(\#N)^{5/2}$-time のものが知られている[2, 1977]. 入次数, 出次数とも高々2のdagに関して $k\cdot\#N$-time のものが知られている[4, 1978] (1975). 我々は出次数だけ高々2と制限して $k\cdot\#N$-time ($1 \leqq k \leqq 2$) のものを示す. なお一般の有向グラフに関してはNP-complete

である[2].

2. 交互グラフ

この節では

問題1. 極大 chain matching を求める

か

問題2. 分枝最小の生成出グラフを求める [5, 1975]

に帰着できる事を示す. 有向グラフの辺の間には交互同値という関係が知られている[5]. 更に問題2はつぎの問題3に帰着できる[5].

問題3. 上の同値関係により分割された、各部分グラフについて分枝数最小の生成出グラフを求める.

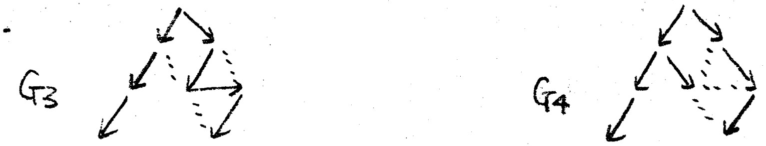
定義. 有向グラフ $G=(N,E)$ における頂点 v の<u>分枝数</u> $\mathrm{branch}(v,G)$ は

$$\mathrm{branch}(v,G) = \mathrm{outdegree}(v,G) \dot{-} 1$$

と定められる. Gの<u>分枝数</u> $\mathrm{branch}(G)$ は

$$\sum_{v \in N} \mathrm{branch}(v,G)$$

である.



G_3 G_4

$\mathrm{branch}(G_3)=1$ 図2 $\mathrm{branch}(G_4)=2$.

定義．有向グラフ $G=(N,E)$ の部分グラフ $H=(N,F)$ は各頂点 v に対して唯一つの辺 $(u,v)\in F$ $(u\in N)$ が存在するとき、Gの<u>生成出グラフ</u>という，ただし $\mathrm{indegree}(v,G)\geq 1$．

有向グラフの辺の間に同値関係を定める．その同値関係によって有向グラフを分割する．このとき、分枝数最小の生成出グラフは、その分割により得られる各部分グラフ（極大交互グラフという）に対する分枝数最小の生成出・グラフを求める事により得られる．この節の残りの部分は[5]に従がう．

定義．有向グラフ $G=(N,E)$ の辺 (u,v) と (w,x) が <u>alternately adjacent</u> である，$(u,v)\sim(w,x)$ とかく，とは (i) $(u,v)\neq(w,x)$ であり (ii) $u=v$ 又は $w=x$ である：

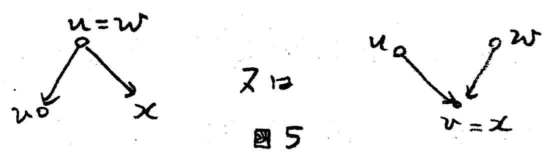
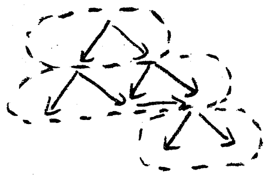


図5

記号 $\overset{*}{\sim}$ で $\sim$ の反射推移閉包をあらわす．$G/\overset{*}{\sim}$ の各元をGの<u>極大交互部分グラフ</u>という．



G_1

図6

定理1． $G_1, G_2, \cdots, G_n$ を dag Gの極大交互部分グラフの全てとする。 $i \leq n$ に対して H_i を G_i の分枝数最小の生成出グラフとすると $\bigcup_{1 \leq i \leq n} H_i$ は Gの分枝数最小の生成出グラフである．（dag はサイクルのない有向グラフ）．

3． アルゴリズム

次の二つのアルゴリズムを合わせたものが求めるアルゴリズムである． これらのアルゴリズムでは、有向グラフはリスト表現（リンク表現）により表現されている、と仮定する。更に、INLIST(v) は頂点vに入ってくる頂点の並び、OUTLIST(v) はvから出て行く頂点の並びをそれぞれ表わす。

```
        PROCEDURE ALTSEARCH(v, G, F)
/*  G (input/OUTPUT)  an alternate dag; vertices in G are            */
/*                    marked "UNCOVERED", "COVERED", "UNVISITED"     */
/*                    and/or "VISITED".                              */
/*  v (input)         a vertex in G; searching start from v.         */
/*  F (output)        the edges of a dag; a spanning out-forest      */
/*                    for G of minimal branches                      */
```

```
begin
    mark v "VISITED";
    while vertex u remaining in OUTLIST(v)
          marked "UNCOVERED" do
          begin
              add (v,u) to F ; mark u "COVERED";
              while vertex x remaining in INLIST(u)
                    marked "UNVISITED" do
                    ALTSEARCH(v, G, F)
          end
end
```

PROCEDURE ALTMATCH (G, F)

```
/*   G (input)   an alternate dag [5] with outdegree   */
/*               atmost two.                           */
/*   F (output)  the edges of a spanning out-forest for */
/*               G of minimal branches                  */

begin
    F := φ;
    mark all vertices v with outdegree(v) ≥ 1
    "UNVISTIED"; mark all vertices v with indegree(v)
    ≥ 1 "UNCOVERED";
    if there is a vertex v with outdegree(v) = 1
        then ALTSEARCH(v, G, F)
        else
        begin
            if there is an alternate semicycle [5] C
            in G
            then
```

```
        begin
          choose any vertex v in C with outdegree(v)
          =2;
          ALTSEARCH(v, G, F)
        end
        else
        begin
          choose any vertex v with outdegree(v)≧1;
          ALTSEARCH(v, G, F)
        end
      end
end
```

与えられた出次数高々2の dag $G=(N, E)$ の極大交互部分グラフの各々について ALTMATCH は depth first に動いて $k\#E$-時間 $(1 \leqq k \leqq 2)$ で分枝数最小の生成出 forest を与える．得られた生成出 forest から極大 chain matching が得られる．

Gから極大交互部分グラフの全てを得る #E-時間アルゴリズムが構成できる．更に，これらのアルゴリズムとΣの ALTMATCH を組み合わせて全体として $k \cdot \#E$・時間 $(1 \leqq k \leqq 2)$ で動くような depth first なアルゴリズムを構成できる．仮定より $\#E \leqq 2 \cdot \#N$ であるから得られたアルゴリズムは $k \cdot \#N$ $(1 \leqq k \leqq 4)$ 時間で動く．

交互グラフは二組グラフと同一視できる． 従って得られたアルゴリズムは一方の頂点の集合に関して次数2以下の二組グラフに適用できる．

文献

1. T.B. Boffey, The linearization of flowcharts, BIT 15 (1975), 341-350.
2. F.T. Boesch and J.F. Gimpel, Covering the points of a digraph with point-disjoint paths and its application to code optimization, JACM 24 (1977), 192-198.
3. 岩田茂樹, GOTO文最少のプログラム, 京大数研講究録 259 (1976).
4. S. Iwata, Programs with minimal goto statements, Inform. Control 37 (1978), 105-114.
5. 夜久竹夫, フローグラフの分枝数について, 京大数研講究録 259 (1976), 158-164.
6. T. Yaku, Algorithms on matchings of digraphs, A seminar in ICM 78, 1978, Helsinki.