論　文

平面グラフの単純閉路による最小被覆を求めるアルゴリズム*

武野純一** 柿倉正義***
向殿政男** 増田英次郎**

Abstract

We describe an algorithm for the minimal covering problem of a biconnected planar graph with elementary circuits.

The algorithm operates as follows:

(1) generate a set of connected faces of a biconnected planar graph
(2) construct all elementary circuits using this set
(3) express a given graph in a logic equation form using these elementary circuits
(4) calculate the minimal covering set, using the logic equation, and a procedure for the simplification of logic form.

The distinctive features of this algorithm are;

(1) use of the connectivity of the faces of a biconnected planar graph for the generation of elementary circuits
(2) efficient computation of the minimal covering set through an improvement in Slagle's algorithm.

1. まえがき

グラフ理論における最小被覆問題には，頂点に関するもの，辺に関するものと，いくつか紹介されているが[1]，本論文では平面グラフの単純閉路による最小被覆問題について提案し，これについて考察する．ここで，平面グラフの単純閉路による最小被覆問題とは次のように定義される．

「平面グラフGのすべての辺の集合をE, Eの部分集合で単純閉路となる辺の集合をC_iとする．Eの部分集合の族$F=\{C_i\}_{i\in N}$において，Fに属するすべての集合の和集合がEに等しいとき，FをグラフGの単純閉路による被覆といい，Fの要素の数が最小なるものを単純閉路による最小被覆という(Nは添数集合)．」

本論文では，上で述べた平面グラフの単純閉路による最小被覆を求めるアルゴリズム，および例題による説明について述べる．

本稿で扱っている平面グラフの単純閉路による最小被覆問題は，次のような問題を解く場合に典型的に現われる．

「有限個の都市間に，道路網が存在し，その道路網上を販売してあるくセールスマンを考える．各セールスマンは，同一の都市の再通過を認められず，かつ出発した都市へ帰ることが義務づけられている．この時，この道路網に最低幾人のセールスマンを派遣すべきか，またその時の道筋にはいかなるものがあるか．」

2. 用語の定義[2]

以後に使用する，いくつかの用語について定義を与える．

定義 1：2連結な平面グラフとは，そのグラフのどの相異なる2頂点も，その2頂点以外では互いに交わらないような，2個以上の初等的な道 (elementary path)* によって結ばれているグラフである．

定義 2：連結した面とは，面の集合 $\{S_1, S_2, \cdots, S_n\}$

* An Algorithm for Minimal Covering Solution of a Planar Graph with Elementary Circuits by Jun-ichi TAKENO, Masao MUKAIDONO, Eijiro MASUDA (Faculty of Engineering, Meiji University), and Masayoshi KAKIKURA (Automatic Control Division, Electrotechnical Laboratory).

** 明治大学工学部
*** 電子技術総合研究所

であって，任意の2つの面 S_i, S_j に対して面の連鎖 $S_i=S_{k1}, S_{k2}, \cdots, S_{ks}=S_j$ (ただし $S_{kl}(l=1,\cdots,s)\in\{S_1,\cdots,S_n\}$) が存在して，隣り合ったすべての面 S_{kl}, $S_{k(l+1)}$ $(l=1,\cdots,s-1)$ において，共通な境界の辺が存在するものをいう．

定義 3：単純閉路とは，グラフ G の1つの閉じた(始点，終点が一致している)初等的な道 $S=(v_{K1}, v_{K2}, \cdots, v_{Kj}, v_{K1})$ をいう．ここで v_{Ki} は頂点とする．

定義 4：$X-Y$ 行列とは，その要素 a_{ij} を $x_i\in X$ が $y_j\in Y$ を (または $y_j\in Y$ が $x_i\in X$ を) 含めば1，そうでなければ0としたものである．ここで X, Y は各々，閉路，面，辺および頂点のいずれかの集合である．

3. 定 式 化

本論文で対象とするグラフは，2連結な平面グラフとする．

ここで問題を次のように定式化する．

「2連結な平面グラフ $G=(V, E)$ が与えられている．この時，Gの部分グラフの中で単純閉路となるものの集合をCとすると，$C_i\in C$なる単純閉路の結びによってグラフGを被覆するもの，すなわち

$$G=\bigcup_{i=i_1}^{i_K} C_i$$

を満たすもので，Kが最小となるものの集合，すなわち単純閉路による最小被覆の集合を求めよ．ここでVは頂点の集合，Eは辺の集合を表わす．」

単純閉路を以後単に閉路とよぶことがある．

4. アルゴリズム

前章において定式化した問題を解くアルゴリズムを，次の4段階に分ける．

Step 1. 2連結な平面グラフGにおける，連結した面の集合をすべて求める．

Step 2. 連結した面の集合から閉路を構成する．その際それらの中で複数個の閉路に分離するものを除くことによって，与えられたグラフGのすべての閉路を求める．

Step 3. これらの閉路を用いて，グラフGを表現する条件を論理式に表わす．

Step 4. 論理式の最簡形式を求めることによって，グラフGの閉路による最小被覆の集合を求める．

4.1 Step 1　連結した面の集合のすべてを求めること．

平面グラフ中に存在する任意の閉路が，面および，面のいくつかの連結と1対1に対応している[4]ことに着目する．これによれば，平面グラフ中に存在するすべての閉路を見いだすには，平面グラフGにおける連結した面の集合のすべてを求めればよい．これを次の3段階に分ける．

Step 1.1　平面グラフGの各面に対し，任意に通し番号①, ②, …, ⓝ (nはGの面の数) を付ける．

Step 1.2　平面グラフGの双対グラフ$\tilde{G}$を求める．

Step 1.3　連結した面の集合を順次生成する．

連結したl個の面の集合を V_l** で表わすと，$V_1=\{\langle①\rangle, \langle②\rangle, \cdots, \langle ⓝ\rangle\}$，および $V_n=\{\langle①, ②, ③, \cdots, ⓝ\rangle\}$ は明らか．V_2 は直接連結している面の対からなる集合であり，Step 1.2 の双対グラフ $\tilde{G}$ より容易に得ることができる．ここで V_l は次の(A), (B)に注意しながら，V_2 と V_{l-1} の各要素の組み合わせから構成できる***．

$a_2\in V_2$, $a_{l-1}\in V_{l-1}$ とする．

(A)　a_2 と a_{l-1} は共通な唯一の面を持つこと．

(B)　a_2 と a_{l-1} から V_l の要素 a_l が構成されたとしても，同じ a_l が他の組み合わせ $a_2'\in V_2$, $a'_{l-1}\in V_{l-1}$ からも構成されることがあり得る．

(A)を満足させるには，$a_2\cap a_{l-1}=\phi$, $a_2\subseteq a_{l-1}$ の2つの場合を除外すればよい．(B)の重複を避けるには，新しく $a_2\cup a_{l-1}=a_l\in V_l$ が構成された場合，それが既に V_l の中に存在するか否かを調べればよい．Vで求める集合を表わすと，

$$V=\bigcup_{l=1}^{n} V_l$$

Step 1 を **Fig. 1**(a) (次頁参照) を例として説明する****．

Step 1.1 により，Fig. 1(a) のグラフの各面に通し番号をつける．

Step 1.2 により，双対グラフ $\tilde{G}$ を求める．(Fig. 1 (b))

本論文で開発したプログラムにおいては，面—辺行列を計算機の入力データとして与え，双対グラフ$\tilde{G}$を

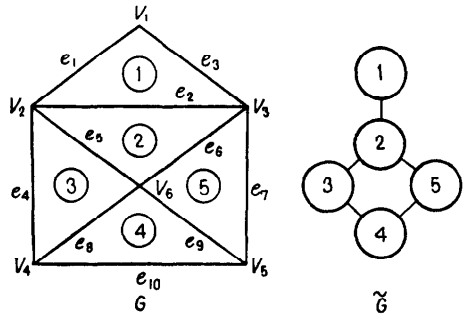
---

* 道とは，グラフGの辺の列 $\{e_1, e_2, \cdots, e_q\}$ で，相異なる2つの辺 e_i, e_{i+1} は必ず共通の頂点を端点として持つものをいう．道の中で，同一の頂点に2度以上出会わないものを初等的な道という．

** 連結した面 $a_q(a_q\in V_q)$ を次のように表現する．$a_q=\langle$①, ②, …, ⓠ$\rangle$ ここでⓠは面を表わす．

*** V_l は $V_K(K\geq 3)$ と V_{l-K+1} からも構成可能であるが，$K=2$ の場合からも必ず V_l のすべてが構成可能であるから，この場合のみを考察すればよい．

**** 以下では，この例題を用いてアルゴリズムの説明を行う．



(a) original graph. (b) dual graph ignoring infinte face.

Fig. 1 A graph for explanation.

Table 1 Face-edge matrix for the input data.

F_i	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}
①	1	1	1	0	0	0	0	0	0	0
②	0	1	0	0	1	1	0	0	0	0
③	0	0	0	1	1	0	0	1	0	0
④	0	0	0	0	0	0	0	1	1	1
⑤	0	0	0	0	0	1	1	0	1	0

計算機により求めている．例えば，例題の入力データは **Table 1** のようになる．

Step 1.3 より

$V_1=\{\langle①\rangle,\langle②\rangle,\langle③\rangle,\langle④\rangle,\langle⑤\rangle\}$

$V_2=\{\langle①,②\rangle,\langle②,③\rangle,\langle③,④\rangle,\langle②,⑤\rangle,\langle④,⑤\rangle\}$

V_3 は(A)，(B)に注意し V_2 と V_2 の組み合わせから構成できる．

$V_3=\{\langle①,②,③\rangle,\langle①,②,⑤\rangle,\langle②,③,④\rangle,$
$\langle②,③,⑤\rangle,\langle③,④,⑤\rangle,\langle②,④,⑤\rangle\}$

例えば，$\langle①,②,③\rangle$ は $\langle①,②\rangle,\langle②,③\rangle$ の組み合わせによる．同様に V_4 は V_2 と V_3 の組み合わせから構成する．

$V_4=\{\langle①,②,③,④\rangle,\langle①,②,③,⑤\rangle,$
$\langle①,②,④,⑤\rangle,\langle②,③,④,⑤\rangle\}$

この時，$\langle①,②,④,⑤\rangle$ は $\langle①,②,⑤\rangle,\langle④,⑤\rangle$ および $\langle②,④,⑤\rangle$ と $\langle①,②\rangle$ の組み合わせより構成されるが一方を除去すればよい．

$V_5=\{\langle①,②,③,④,⑤\rangle\}$ は明らか．

$$V=\bigcup_{l=1}^{5} V_l$$

が，グラフ G における連結した面の集合のすべてであり，結果は閉路—面行列として出力される．(**Table 2** (a))

4.2 Step 2 閉路の構成および複数閉路の除去

Table 2 Representation of circuits for the example.
(a) circuit-face matrix. (b) circuit-edge matrix.

	①	②	③	④	⑤
C_1	1	0	0	0	0
C_2	0	1	0	0	0
C_3	0	0	1	0	0
C_4	0	0	0	1	0
C_5	0	0	0	0	1
C_6	1	1	0	0	0
C_7	0	1	1	0	0
C_8	0	0	1	1	0
C_9	0	1	0	0	1
C_{10}	0	0	0	1	1
C_{11}	1	1	1	0	0
C_{12}	1	1	0	0	1
C_{13}	0	1	1	1	0
C_{14}	0	1	1	0	1
C_{15}	0	0	1	1	1
C_{16}	0	1	0	1	1
C_{17}	1	1	1	1	0
C_{18}	1	1	1	0	1
C_{19}	1	1	0	1	1
C_{20}	0	1	1	1	1
C_{21}	1	1	1	1	1

	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}
C_1	1	1	1	0	0	0	0	0	0	0
C_2	0	1	0	0	1	1	0	0	0	0
C_3	0	0	0	1	1	0	0	1	0	0
C_4	0	0	0	0	0	0	0	1	1	1
C_5	0	0	0	0	0	1	1	0	1	0
C_6	1	0	1	0	1	1	0	0	0	0
C_7	0	1	0	1	0	1	0	1	0	0
C_8	0	0	0	1	1	0	0	0	1	1
C_9	0	1	0	0	1	0	1	0	1	0
C_{10}	0	0	0	0	0	1	1	1	0	1
C_{11}	1	0	1	1	0	1	0	1	0	0
C_{12}	1	0	1	0	1	0	1	0	1	0
C_{13}	0	1	0	1	0	1	0	0	1	1
C_{14}	0	1	0	1	0	0	1	1	1	0
C_{15}	0	0	0	1	1	1	1	0	0	1
C_{16}	0	1	0	0	1	0	1	1	0	1
C_{17}	1	0	1	1	0	1	0	0	1	1
C_{18}	1	0	1	1	0	0	1	1	1	0
C_{19}	1	0	1	0	1	0	1	1	0	1
C_{20}	0	1	0	1	0	0	1	0	0	1
C_{21}	1	0	1	1	0	0	1	0	0	1

Step 2 をさらに2段階に分ける．

Step 2.1 連結した面の集合から閉路を構成する．

これは，連結した面の集合から，面—辺行列(Table 1) を用いて，リング和 (mode 2 の加算) をとることにより構成する．結果は，閉路—辺行列 (Table 2 (b)) として出力される．

Step 2.2 複数閉路の除去．

いくつかの面を連結することにより閉路を求める場合，その結果が1つの閉路とならずに複数個の閉路に分離することがある．例えば，**Fig. 2**(a)における面

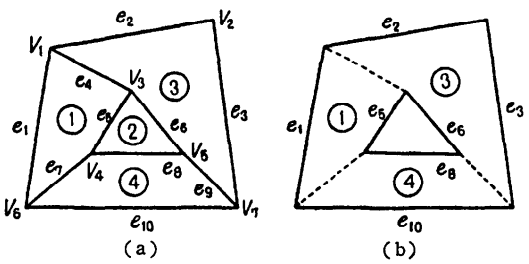


Fig. 2 An exemplification for the case of separated circuits.

の連結〈①,③,④〉のリング和をとると，$e_1e_2e_3e_5e_6e_8e_{10}$ となるが (Fig. 2(b) 参照)，これは $e_1e_2e_3e_{10}$ および $e_5e_6e_8$ なる複数個の閉路に分離してしまう．ところが，$e_1e_2e_3e_{10}$ は〈①,②,③,④〉なる面の連結で，また $e_5e_6e_8$ は〈②〉なる面からも構成される．従って一群の面の連結から複数個の閉路が構成される場合を除去しなければならない．

これには，頂点一辺行列を用いて，辺と辺の接続関係を追い，複数個の閉路に分離するかどうかを調べればよい．(例題においては，複数個に分離するものは存在しない．)

4.3 Step 3　すべての閉路を用いて，グラフ G を表現する条件を論理式に表わす．

グラフ G の辺 e_K を含む閉路 C_i の添字 i のすべてからなる集合を N_K とすると，辺 e_K が存在する条件を表わす論理式 $f(e_K)$ は，便宜上閉路 C_i を論理変数〔$C_i=1$ (C_i 中に e_K が存在する)，$C_i=0$ (C_i 中に e_K が存在しない)〕とみなすと，

$$f(e_K)=\bigvee_{i\in N_K} C_i$$

と表現される．

従って，グラフ G を構成しているすべての辺が表現されるための論理式 $F(G)$ は，

$$F(G)=\bigwedge_{K=1}^{m} f(e_K)=\bigwedge_{K=1}^{m}\bigvee_{i\in N_K} C_i \qquad (1)$$

ただし m はグラフ G の辺の数．

具体的に $f(e_K)$ を求めるには，閉路一辺行列において，e_K 列で1の立っている，すべての閉路の論理和をとればよい．

例題では，Table 2(b) より，

$f(e_1)=C_1\vee C_6\vee C_{11}\vee C_{12}\vee C_{17}\vee C_{18}\vee C_{19}\vee C_{21}$

$f(e_2)=C_1\vee C_2\vee C_7\vee C_9\vee C_{13}\vee C_{14}\vee C_{16}\vee C_{20}$

$f(e_3)=f(e_1)$

$f(e_4)=C_3\vee C_7\vee C_8\vee C_{11}\vee C_{13}\vee C_{14}\vee C_{15}\vee C_{17}$
$\vee C_{18}\vee C_{20}\vee C_{21}$

$f(e_5)=C_2\vee C_3\vee C_6\vee C_8\vee C_9\vee C_{12}\vee C_{15}\vee C_{16}\vee C_{19}$

$f(e_6)=C_2\vee C_5\vee C_6\vee C_7\vee C_{10}\vee C_{11}\vee C_{13}\vee C_{15}\vee C_{17}$

$f(e_7)=C_5\vee C_9\vee C_{10}\vee C_{12}\vee C_{14}\vee C_{15}\vee C_{16}\vee C_{18}$
$\vee C_{19}\vee C_{20}\vee C_{21}$

$f(e_8)=C_3\vee C_4\vee C_7\vee C_{10}\vee C_{11}\vee C_{14}\vee C_{16}\vee C_{18}\vee C_{19}$

$f(e_9)=C_4\vee C_5\vee C_8\vee C_9\vee C_{12}\vee C_{13}\vee C_{14}\vee C_{17}\vee C_{18}$

$f(e_{10})=C_4\vee C_8\vee C_{10}\vee C_{13}\vee C_{15}\vee C_{16}\vee C_{17}\vee C_{19}$
$\vee C_{20}\vee C_{21}$

となり，従ってグラフ G の辺を表現する論理式は次のようになる．

$$F(G)=f(e_1)\wedge f(e_2)\wedge f(e_3)\wedge f(e_4)\wedge f(e_5)\wedge f(e_6)\wedge f(e_7)\wedge f(e_8)\wedge f(e_9)\wedge f(e_{10})$$
$$=\bigwedge_{K=1}^{10} f(e_K) \qquad (2)$$

4.4 Step 4　論理式の最簡形式を求めることによりグラフ G の単純閉路による最小被覆集合を求める．

(1)式は乗法標準形 ($f(e_K)$ は和項) で与えられるので，グラフ G の単純閉路による最小被覆の集合は (1)式をブール代数を用いて展開し，加法標準形 $F(G)=g_1\vee g_2\vee\cdots\vee g_K(g_i\not\subseteq g_j|i\neq j)$ を求め (g_i は積項)，積項のうち項数最小のものを求めれば，それが最小被覆の集合である．しかし，このブール代数を用いる展開には，$\alpha\vee\alpha\wedge\beta=\alpha$，$\alpha\vee\alpha=\alpha$，$\alpha\wedge\alpha=\alpha$ なる吸収則を用いる必要があり，一般に項数が多くなると吸収則を多数回使用しなければならなくなるため，手数が極端に多くなり実用的ではない．そこで筆者らは，Slagle のアルゴリズム[3]を改良し，乗法標準形から文字数 (閉路数) 最小の項を求める高速アルゴリズム A-1，A-2[6] を開発した．

グラフ G の単純閉路による最小被覆の集合を求めるには，変数の数 (閉路の数) の最小なものを1つ見いだすアルゴリズム (A-1)，および最小の組み合わせのすべてを見いだすアルゴリズム (A-2) があればよい．この目的のために Slagle のアルゴリズムを次のように改良した．

アルゴリズム A-1　変数の数の最小なものを1つ見いだすアルゴリズム．

Slagle のアルゴリズムを depth-first 法[1]で展開する．以下で D は根からの深さ (depth)，w はそれまで得られている暫定解，L はその文字数を表わす．また f_p は節 p に割り当てられる論理式，$|f_p|$ は f_p における和項の個数，x_{p1} は f_p における変数のうちで出現頻度が一番高いもの，$|x_{pi}|$ は f_p における変数 x_{pi} の出現頻度とする．

① 木の根をP点とし，ここより depth-first による展開を始める．それゆえ $f_p=F(G)$，$D=0$，$w=x_1\wedge x_2\wedge x_3\wedge\cdots\wedge x_n$，$L=n$，となる．②へ．

② f_p における各変数の出現頻度を求め③へ．

③ 出現頻度の一番大きい変数を x_{p1} とする．

$$L\leqq D+\lceil|f_p|/|x_{p1}|\rceil \qquad (3)$$

(ここで記法 $\lceil\ \rceil$ は切り上げ)

であればP点より木枝 x_{p1} を張り，その先にラベル (×) を付けて⑤へ．さもなくば④へ．

④ f_p より x_{p1} を含むすべての和項を消去した式を f_p' とし，f_p より x_{p1} を消し，x_{p1} の出現頻度を0とする．$f_p'=\phi$ ならば⑦へ，さもなくば木枝 x_{p1} を張ってその先を新しいP点とし，f_p' を新しく f_p とし，$D=D+1$ として②へ．

⑤ $D=0$ であれば停止．

$D\neq 0$ であれば，P点より一つ前の点にもどり，その点を新しくP点として⑥へ．

⑥ すべての変数の出現頻度が0または空なる和項がある場合は⑤へ．さもなくば③へ．

⑦ P点より木枝 x_{p1} を張り，その先にラベル(○)を付け，ラベル(○)から根に至る木枝に割り当てられている変数すべての積を新しく暫定解とする．$L=D+1$ として⑤へ．

評価式(3)による刈り込みの根拠を次に述べる．

節 p より深い所でラベル(○)が付けられて得られる解の文字数は，最小の場合で $D+\lceil|f_p|/|x_{p1}|\rceil$ となる．従って，暫定解 w の文字数 L に関して次式

$$L\leqq D+\lceil|f_p|/|x_{p1}|\rceil$$

が満足されれば，L より小さな文字数の解を得る可能性がなくなることになる．これにより木枝の先にラベル(×)を付けて，それより深い所での探索を停止する．

例題 ((2)式) にこのアルゴリズム A-1 を適用すると，**Fig. 3** のようになる．

なお，Fig. 3 では，

$f_p=f(e_i)\wedge f(e_{i+1})\wedge\cdots\wedge f(e_{i+m})=f_{i,i+1,i+2,\cdots,i+m}$

なる略記法を用いている．

また，$F(G)=f_{1,2,4,5,6,7,8,9,10}$ には f_3 が含まれていないが，これは $f_1=f_3$ より，吸収則を用いて f_3 を消去したものを根の論理式としたからである．

Fig. 3 を用いて，刈り込み原理を説明する．論理式 $F(G)$ の変数の中で出現頻度が一番高いものはいくつかあるが，その1つ C_{17} を取り，$F(G)$ の和項の中で C_{17} を含むものをすべて除去し木枝 C_{17} を張り，残った論理式 $f_{2,5,7,8}$ を枝の先にわりあて新しく f_p とする．ここで，もとの $F(G)$ から変数 C_{17} を消去しておく．

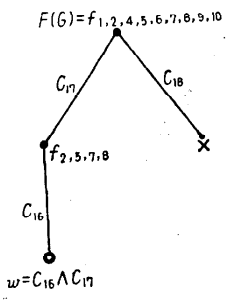


Fig. 3 An example of an expanded tree by algorithm A-1.

次にこの f_p の変数の中で出現頻度の高いものは C_{16} であり，f_p より C_{16} を含む和項のすべてを除去すると f_p は空になるので木枝 C_{16} を張り，その先にラベル(○)を付ける．これが暫定解 $w_1=C_{16}\wedge C_{17}(L=2)$ となる．

次はP点を1つもどし，$f_p=f_{1,2,4,5,6,7,8,9,10}$ で考える．出現頻度が一番高い C_{16} を選ぶ (C_{17} はすでに消去されている)．この時，評価式(3)について考えてみると，$L=2$，$|f_p|=9$，$|C_{16}|=6$，$D=0$ であるので，$2\leqq 0+\lceil 9/6\rceil=2$ となり(3)式が満足され，ラベル(×)が木枝 C_{16} の先につけられ刈り込みが行われる．

この結果は Fig. 3 のようになり，暫定解 $w_1=C_{16}\wedge C_{17}(L=2)$ が文字数最小の解となる．

アルゴリズムA-2　文字数最小の解すべてを見いだすアルゴリズム (文字数最小の解における最小文字数 l はアルゴリズム A-1 よりすでに求められているとする)．これは，アルゴリズムA-1を次のように変形することにより実現できる．ここで W は文字数最小の解すべての集合とし，w をその要素とする．

① 根をP点とする．$f_p=F(G)$，$D=0$，$W=\phi$，$L=l$ (l は最小文字数) とし②へ．

② A-1 と同じ．

③ 出現頻度の一番大きな変数を x_{p1} とする．

$$L<D+\lceil|f_p|/|x_{p1}|\rceil$$

であれば，P点より木枝 x_{p1} を張り，その先にラベル(×)を付け⑤へ，さもなくば④へ．

④，⑤，⑥は A-1 と同じ．

⑦ f_p より x_{p1} を消し，x_{p1} の出現頻度を0とする．P点より木枝 x_{p1} を張り，その先にラベル(○)を付け，ラベル(○)から根に至る木枝に割り当てられている変数すべての積 w を求め，$W=W\cup w$ として⑥へ．

例題にアルゴリズムA-2を適用してみると **Fig. 4** (次頁参照)のようになり，解 $W=C_{16}\wedge C_{17}\vee C_{13}\wedge C_{19}$ を得る．

これにより，グラフ G (Fig. 1(a)) の最小被覆解は2種類あり，最小文字数は2であることが分る．すなわち，グラフ G は高々2つの閉路 (C_{16}, C_{17} または C_{13}, C_{19}) により被覆される．これを図示すると **Fig. 5** (次頁参照) のようになる．

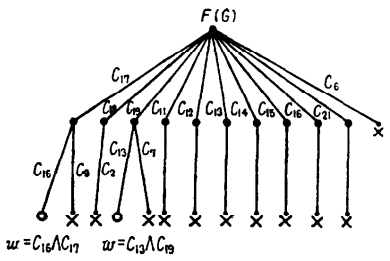
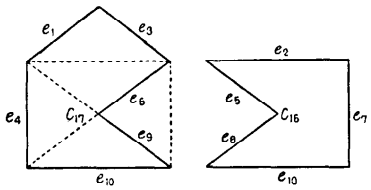
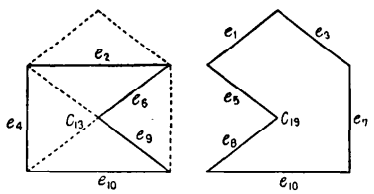


Fig. 4 An example of an expanded tree by algorithm A-2.



(a) minimal covering solution 1.



(b) another solution 2.

Fig. 5 Schematic representation of solutions.

5. 計算の手間に関する考察

本アルゴリズムは，全閉路の数え上げを基本としている．全閉路の数は，面の数 n に対して，一般に，指数関数的に増大する(付録参照)．このために，記憶容量の点から本アルゴリズムの適用可能範囲は制限を受けることになる．

今までに知られている全閉路の数えあげのアルゴリズムは，基本閉路（面の数 n に等しい）を基底としたベクトル空間を構成し，その中から閉路を見いだすものである．この場合，閉路の候補となるものの数は，グラフの構造に無関係に 2^n-1，すなわちオーダーは $O(2^n)$ である[5]．一方，閉路の候補として本アルゴリズムで用いた連結した面を利用する場合，連結した面の数はグラフの構造に依存し，最悪の場合でもオーダーは $O(1.85^n)$ で済むと考えられる（付録参照)．

よって，全閉路を見いだすために，連結な面を利用する方法は有効な手法と考えられる．ただし，本論文で述べた連結した面を求める Step 1.3 のアルゴリズムは便宜上非常に簡単なものを用いたために，厳密な手数の評価は困難であるが，多くの手数を要しており，実行可能な範囲内での両者の時間的な差は，顕著ではない．この意味で，連結した面のすべてを求める効率の良いアルゴリズムの開発が望まれる．

次に，最小被覆問題のためのアルゴリズム A-1, A-2 の計算量について簡単に考察する．

変数の数 l，和項の数 m の部分問題が $l\times m$ の行列表現で与えられている場合，出現頻度の一番大きい変数の木枝を伸ばして，次の部分問題を構成するに要する計算量，すなわち一本の木枝に対応する計算量を求めてみる．

まず，点Pにおいて各変数の出現頻度が計算されているとする．ここから一本の木枝を伸ばして次の点 P′ を得るには出現頻度の一番大きい変数を選ぶのに l 回の比較が，P′ に論理式を割り当てるのに，すなわち部分問題を構成するのに行列をコピーしなければならないから $l\times m$ 回の転送が，P′ での各変数の出現頻度を計算するのに $l\times m$ 回の加算が，それぞれ必要である．その他の操作の計算量も同程度かこれ以下であるので（○印または×印で終る木枝に対しても同様）一つの木枝に対して，計算量のオーダーは $O(l\times m)$ であることが分る．

次に，木枝の本数のオーダーの上限を求めてみる．第1段目の木枝の本数は高々 ${}_lC_1=l$(本)，第2段目の木枝の本数は高々 ${}_lC_2=l(l-1)/2$(本)，以下同様に第 S 段目の木枝の本数は高々 ${}_lC_S$(本) である．よって S 段で終結したとすると木枝の総数は高々

$$\sum_{i=1}^{S} {}_lC_i \qquad (4)$$

となる．よって木枝の本数の上限は

$$\sum_{i=1}^{l} {}_lC_i=2^l-1$$

となり，オーダーは $O(2^l)$ である．

さて，グラフの単純閉路による最小被覆の場合，m は辺の数，l は閉路の数となる．グラフの面の数を n とすると $O(m)=O(n)$，$O(l)=O(\alpha^n)$ $(1<\alpha\leqq 2)$ となるから $l\gg m$ としてよい．また(4)式において，S は和項の数 m よりは大きくなり得ないので，(4)式のオーダーは $O(l^m)$ 以下である．

よって，この場合には木枝の本数の上限のオーダーは $O(\alpha'^n)$ $(1<\alpha<2)$ となる．

6. むすび

2連結平面グラフの単純閉路による最小被覆問題を提出し，それを解決するための1つのアルゴリズムを提案した．本アルゴリズムは，グラフ上の全閉路の数え上げを基本としており，全閉路の数は面の数に対して，指数関数的に増加するために，面数 20 程度のグラフまでしか実用的には取り扱うことができない．しかし，全閉路を数え上げるに当り，面の連結関係を利用し，指数関数的爆発を若干延ばしている点と，最小被覆集合を求めるために，高速アルゴリズムを開発し，最適解を求めている点とに特徴がある．なお，大規模グラフに対しては，現在ヒューリスティックス法を開発中である．

終りにあたり，日ごろ御指導をたまわる明治大学講師後藤以紀先生，同教授小川康男先生，並びに電子技術総合研究所佐藤孝平制御部長，長田正情報制御研究室長の各氏に深甚なる謝意を表する．

参考文献

1) 尾崎他：グラフ理論，コロナ社，東京 (1975).
2) 野口他：グラフ理論，筑摩書房，東京 (1974).
3) J. R. Slagle, C. L. Chage, and R. C. T. Lee: A New Algorithm for Generating Prime Implicants, IEEE Vol. C-19, No. 4 (1970).
4) C. ベルジュ：グラフの理論 I，サイエンス社，東京 (1976).
5) Narsingh Deo: Graph Theory with Applications to Engineering and Computer Science, Prentice-hall, London (1974).
6) 増田，向殿：最小被覆問題の為のある計算機アルゴリズムについて，昭和52年度，電子通信学会総合全国大会，1331.
7) R. G. バサッカー他：グラフ理論とネットワーク，基礎と応用，培風館，東京 (1970).

付　録：単純閉路の数について

面の数 n のグラフの単純閉路の数は，一般に n の増加に対して指数関数的に増大することを示すことにする．それには，一例として単純閉路の数が連結した面の数に等しい，**Fig. A**(a) のような簡単なグラフを考え，グラフ中に番号づけられている面を，その番号順に1つずつ増加させた時，連結した面の数がどのように増加するかを計算すればよい．

今，面 i を付加したとき，面 i を含むすべての連結した面の数を N_i とすると漸化式

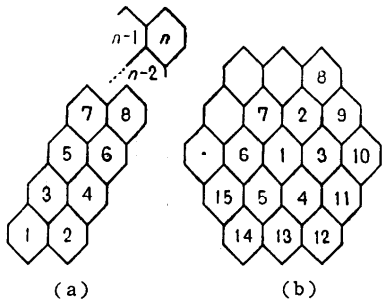


Fig. A

$$N_i = N_{i-1} + N_{i-2} + 1 \quad (i \geqq 3)$$

が成立している．ここで $N_1=1$，$N_2=2$ である．$n \gg 1$ とすると，漸近解として

$$N_n \fallingdotseq C\left(\frac{1+\sqrt{5}}{2}\right)^n \fallingdotseq C(1.618)^n$$

を得る．

単純閉路の総数を l_n とすると

$$l_n \fallingdotseq \sum_{i=1}^{n} N_i = \frac{C \times 1.618 \times (1.618^n - 1)}{1.618 - 1} \fallingdotseq C'(1.618)^n$$

(ここで C, C' は定数)

定数 C' を実測により求めてみると

$$C' \fallingdotseq 3.065$$

となる．以上により

$$l_n \fallingdotseq 3 \times (1.618)^n$$

すなわち　$O(l_n) = O(1.618^n)$

となり，指数関数的に増大する．

一般に，単純閉路の数 l は，面の数 n に対して

$$O(l) = O(\alpha^n) \quad (1 < \alpha \leq 2)$$

であることが分ったが，最悪の場合 α はどのくらいになるか興味のある問題である．

「一般の平面グラフにおいて，任意の領域面と隣接する領域の平均数は6より小さい[7]」ということが知られているので，連結する面の総数が最大となるグラフは，すべての面が6角形をなすモザイク模様のグラフと考えればよい (**Fig. A**(b))．

面番号を図のように付け，その順序により面数を増加させた時の連結した面の総数が，どのように増大するかの実験を本プログラムを用いて行った場合，

$$\alpha \fallingdotseq 1.85$$

という結果が得られている．

(昭和52年6月25日受付)
(昭和53年2月20日再受付)