

平面グラフの2連結化アルゴリズム

東北大工学部 小野口一則 (Kazunori Onoguchi)
千葉 則茂 (Norishige Chiba)
西関 隆夫 (Takao Nishizeki)

1 まえがき

与えられたグラフGに枝を何本か付加することにより2連結グラフにすることを2連結化といい、通信網の信頼性の向上などに応用される。例えば、通信網の中継局を点に伝送路を枝に対応させたグラフを2連結化することは、中継局のうちどの1箇所が故障しても通信が途絶えることのない通信網に拡張することに対応する。グラフを2連結化する場合、付加しなければならない枝の最小本数が何本かということが問題となる。Eswaran と Tarjan⁽¹⁾ は一般グラフを2連結化するのに必要な枝の最小本数に関する必要十分条件を与えておりEswaran と Tarjan⁽¹⁾、Rosenthal とGoldner⁽²⁾ はその枝を具体的に求める線形時間アルゴリズムが得られたと主張している。しかしこの2つの文献のアルゴリズムには少し誤りがあり、そのままでは最小枝数で2連結化するには限らない。本文では、まず実際に最小本数の枝付加で一般グラフを2連結化する線形時間アルゴリズムを与える。この場合、平面グラフが与えられても枝付加により平面性が損われるかもしれない。次に、対象とするグラフを平面グラフに限定し、その平面埋め込みを固定したまま最小本数の枝を付加して2連結平面化する線形時間アルゴリズムを与える。この場合には、グラフの平面性が損われるような枝付加はできず、従来のアルゴリズムは適用できない。

2 準備

ここでは本文で用いる用語を定義する。切断点を含まないGの2連結成分を孤立ブロックと呼ぶ。G自身ではないGの孤立ブロックは q (≥ 0)個あるとする。切断点を丁度1個だけ含むGの2連結成分をペンダントブロックと呼ぶ。Gにはペンダントブロックが p 個あるとする。vをGの任意の点とする。Gの枝は次の同値類 E_i に分割できる。

2本の枝 (x, y) と (w, z) が同じ同値類に含まれるのは、それらを含む道 $(v_1, v_2), (v_2, v_3), \dots, (v_{k-1}, v_k)$ が存在し、しかも $2 \leq i \leq k-1$ なる i に対し $v_i \neq v$ である時であり、かつその時に限る。

この同値類 E_i を用いてGの部分グラフ G_i を次のように定義する。

$$G_i = (V_i, E_i)$$
$$V_i = \{x \in V \mid (x, y) \in E_i\}$$

この部分グラフ G_i をGのv-ブロックと呼び、Gのv-ブロックの個数を $d(v)$ と書く。 d を $d = \max \{d(v) \mid v \in V\}$ と定義する。Gの切断点及び2連結成分を点で表わし、G上で隣接している切断点と2連結成分を枝で結んだ木をblock-cutpoint木 (BC木) と呼び、 $T(G)$ と書く。 $T(G)$ 上で切断点に対応する点をC点、2連結成分に対応する点をB点と呼び、図上ではC点は黒丸で、B点は白丸で表わす。Gのペンダントブロックは $T(G)$ の葉に対応する。 $T(G)$ 上でC点 v から葉 l までの道を $Q = (v, v_1), (v_1, v_2), \dots, (v_m, l)$ としよう。枝 (v_1, v_2) をQの始点の枝、 (v_m, l) を終点の枝と呼ぶ。Gは2連結でないので、面の境界は必ずしも単純な閉路ではないが、面の境界を面閉路と呼ぶことにする。Gが平面グラフの時、ある1つの面 F_i に着目する。 F_i の面閉路上の点及び枝のみからなるGの部分グラフ G_i を F_i の面グラフと呼ぶ。面 F_i に含まれるペンダントブロック P_j とは P_j の外周上のすべての枝が F_i の面閉路上にあるペンダントブロックのことをいう。図1で面 F_1 に含まれるペンダントブロックは P_1 および P_2 であり、面 F_2 に含まれるペンダントブロックは P_3, P_4 および P_5 である。また隣合ったペンダントブロックとは F_i の面閉路を時計回りにたどった時に隣り合って探索されるペンダントブロックのことをいう。図1で P_3 と P_4, P_4 と P_5, P_5 と P_3, P_1 と P_2 はそれぞれ隣り合ったペンダントブロックである。

3 一般グラフの2連結化アルゴリズム

3.1 アルゴリズム

一般グラフ G に枝を付加して 2 連結化するのに必要な枝の本数の最小値を $\mu(G)$ と書くことにする。次の補題が Eswaran と Tarjan⁽¹⁾ によって示されている。

$$[\text{補題 1}] \mu(G) = \max\{d-1, \lceil p/2 \rceil + q\}$$

(補題終)

本章では、一般のグラフ G が与えられた時、 G に $\mu(G)$ 本の枝を付加して 2 連結化する線形時間アルゴリズム BICONNECT を与える。この場合、たとえ G が平面グラフであっても枝付加によりグラフが非平面になることもある。

このアルゴリズムは補題 1 の $\mu(G) \leq \max\{d-1, \lceil p/2 \rceil + q\}$ の証明に基づいているため、この証明を示しておく。

(補題 1 の証明)

$i = \max\{d-1, \lceil p/2 \rceil + q\}$ に関する帰納法で証明する。 G が 2 連結グラフでない場合を考えるので $i \geq 1$ として一般性を失わない。

(1) $i = 1$ 及び 2 の時

$i = 1$ となり得るのは、 $d = p = 2$, $q = 0$ の場合のみである。この場合、2 つのペンダントブロックを切断点以外で 1 本の枝で結ぶと G は 2 連結グラフになる。したがって $\mu(G) = 1$ である。 $i = 2$ となり得るのは、 $d = 2, 3$ 及び $p = 3, 4$ の場合のみである。 $p = 4$ の場合で、ペンダントブロックを丁度 2 つ含む v -ブロックがある時はそのペンダントブロックの 1 つと、他の v -ブロック中のペンダントブロックとを切断点以外で 1 本の枝で結ぶ。それ以外の場合は、かってな 2 つのペンダントブロックを切断点以外で 1 本の枝で結ぶ。するといずれの場合も $d = p = 2$ となり明らかに $\mu(G) = 2$ である。

(2) $i > 2$ の時

$\max\{d-1, \lceil p/2 \rceil + q\} = i$ のとき、この補題が成り立つと仮定する。いま $\max\{d-1, \lceil p/2 \rceil + q\} = i + 1$ とし、いくつかの場合に分けて考える。

(2-1) G が非連結の時

2 つの連結成分 G_1, G_2 に着目する。 G_1, G_2 は孤立ブロックであるか、さもなければペンダントブロックを含んでいる。 G_1 と G_2 を 1 本の枝で結ぶ。ただし G_1 が孤立ブロックでないならば、その枝の 1 つの端点は G_1 のペンダントブロック内の切断点でない点とする。その付加する枝のもう 1 つの端点を G_2 内に同様に選ぶとする。得られるグラフにおける p, d, q は次のようになり $\max\{d-1, \lceil p/2 \rceil + q\} = i$ を満たすことが解る。

(I) G_1 及び G_2 がともに孤立ブロックの場合

$$p := p + 2, q := q - 2, d := d - 1$$

(II) G_1 あるいは G_2 の片方だけが孤立ブロックの場合

$$p \text{ 変化なし}, q := q - 1, d := d - 1$$

(III) G_1, G_2 とも孤立ブロックでない場合

$$p := p - 2, q \text{ 変化なし}, d := d - 1$$

仮定より、このグラフは i 本の枝付加で 2 連結化できるので、結局 G は $i + 1$ 本の枝付加で 2 連結化できる。

(2-2) G が連結の時

このとき $q = 0$ である。次の 3 つの場合がある。

(2-2-1) $d - 1 > \lceil p/2 \rceil$ の時

このとき $d(v) = d$ である切断点 v は 1 個しかない (2 つ以上あるとすると $d - 1 \leq \lceil p/2 \rceil$ となり仮定に反する)。この切断点 v の v -ブロックのうち、異なる 2 つの v -ブロックからそれぞれ 1 つずつペンダントブロックを選び、切断点以外で 1 本の枝で結ぶ。すると 2 つの v -ブロックが 1 つになるため d が 1 つだけ減少し、新しいグラフ G' は $\max\{d' - 1, \lceil p'/2 \rceil\} = i$ を満たす。ここで、 d', p' はグラフ G' における d, p である。仮定より、 G' は i 本の枝付加で 2 連結化できるので G は $i + 1$ 本の枝付加で 2 連結化できる。

(2-2-2) $d - 1 = \lceil p/2 \rceil$ の時

このとき $d(v)=d$ である切断点 v は高々2つしかない。このような切断点 v の v -ブロックの中で、2つ以上ペンダントブロックを持つ v -ブロック (それを G_1 とする) を選ぶ。もし $d(w)=d$ を満たす v 以外の切断点 w が存在するなら、その点 w は G_1 に含まれており、点 v の G_1 以外の各 v -ブロックにはペンダントブロックは1個しか含まれていない。 G_1 に含まれているペンダントブロックと他の v -ブロックに含まれているペンダントブロックを1つずつ選び、切断点以外で1本の枝で結ぶ。すると p が少なくとも2減少し、かつ2つの v -ブロックが1つになるため d も1つ減少する。 $d(w)=d$ なる切断点がもし存在するならばそれは G_1 中に存在するため、 w -ブロックの数も1つだけ減少する。よって、得られたグラフ G' は $\max\{d'-1, \lceil p'/2 \rceil\} = i$ を満たす。仮定より、 G' は i 本の枝付加で2連結化できるので G は $i+1$ 本の枝付加で2連結化できる。

(2-2-3) $d-1 < \lceil p/2 \rceil$ の時

$d(v)=d$ である任意の切断点 v に注目する。ただし、 $d=2$ かつ $p>3$ の場合次のように v を選び直す。(この部分が文献(1) および(2) で欠けている。下の注意参照) v 以外の切断点 v_0 があって、その v_0 -ブロック B でちょうど2つだけペンダントブロックを含むものに v が含まれるなら v_0 を新たに v として選ぶ。例えば図2(a) において最初に注目した切断点 v が点 e または f であるなら点 a を v として選び直す。

いま v -ブロックの中には2つ以上ペンダントブロックを含むものが必ず存在するため、そのような v -ブロックに含まれているペンダントブロックとそれ以外の v -ブロックに含まれているペンダントブロックを1つずつ選び、それらを切断点以外で1本の枝で結ぶ。すると $\lceil p/2 \rceil$ が1つ減少するため得られたグラフ G' は $\max\{d'-1, \lceil p'/2 \rceil\}$ を満たす。仮定より、 G' は i 本の枝付加で2連結化できるので G は $i+1$ 本の枝付加で2連結化できる。

(証明終)

[注意]

$d=2$ かつ $p>3$ の場合、すべての切断点 v が $d(v)=d=2$ を満たすが、上述のように切断点 v の選び方に制約を加えている。これは1本枝を付加することにより新しいペンダントブロックが出現する場合があるからである。例えば、図2(a) のグラフの場合、 $d=2$ 、 $p=6$ であるが切断点 e ($d(e)=2$) に注目すると、ペンダントブロック1及び5が異なる e -ブロック (一方は2つ以上ペンダントブロックを持つ) に含まれているため、この2つが枝で結ばれる可能性がある。このとき、図2(b) のように a を切断点とするペンダントブロックが新たに1つ出現するため、全体としてペンダントブロックの個数は1つしか減らず、帰納法がうまく働かない。このように新しいペンダントブロックができるのは、ちょうど2つのペンダントブロックを含んでいる v -ブロック (図2の a -ブロック) が存在し、その v -ブロック内の2つのペンダントブロックを枝で結んだ時に限る。このような場合を避けるため、この2つのペンダントブロックが同じ v -ブロックに含まれるような切断点を v として選べばよい (例えば、図2の切断点 a)。なお、図3の切断点 a のように、その a -ブロック内に3つ以上ペンダントブロックが存在する場合には、その a -ブロック内の2つのペンダントブロックを枝で結んでも、 a -ブロック内にペンダントブロックが1つ以上残るため a を切断点とする新しいペンダントブロックは出現しない。

以上の証明に基づくアルゴリズム BICONNECT(G) を次に示す。

Procedure BICONNECT(G);

begin

while G が非連結 do

begin

G の異なる2つの連結成分を選び G_1 、 G_2 とする;

if G_1 が孤立ブロック

then $v_1 := G_1$ の任意の点

else $v_1 := G_1$ に含まれるペンダントブロック中の切断点でない点;

if G_2 が孤立ブロック

then $v_2 := G_2$ の任意の点

else $v_2 := G_2$ に含まれるペンダントブロック中の切断点でない点;

$G := G + \{v_1, v_2\}$; (G に枝 (v_1, v_2) を付加)

end;

```

Gのd, pを求める;
while max{d-1, ⌈p/2 ⌉} > 2 do
  begin (Gが連結)
    v:=d(v)=dなる切断点;
    if d-1 > ⌈p/2 ⌉ then
      begin
        異なる2つのv-ブロックG1, G2を選ぶ;
        v1:=G1に含まれるペンダントブロック中の切断点でない点;
        v2:=G2に含まれるペンダントブロック中の切断点でない点;
      end else
      begin (d-1 ≤ ⌈p/2 ⌉)
        if (d=2) and (p > 3)
          then if v ≠ v0なる切断点v0があり, しかも2つだけペンダントブロックを含むv0-ブロックにv
              が含まれる
                then v:=v0;
                ペンダントブロックを2つ以上含むv-ブロックを1つ選びG1とする;
                G1以外のv-ブロックを1つ選びG2とする;
                v1:=G1に含まれるペンダントブロックの切断点でない点;
                v2:=G2に含まれるペンダントブロックの切断点でない点;
              end;
          G:=G+{v1, v2}; (Gに枝(v1, v2)を付加)
          d, pを更新; (枝付加後のG上でのd, p, qを求める)
        end;
      if 1 ≤ i ≤ 2 then
        begin (この場合d=2, 3及びp=3, 4である.)
          p=4でペンダントブロックを丁度2つ含むv-ブロックがあるときは, まず, そのペンダントブロックの
            1つと, 他のv-ブロック中のペンダントブロックとを切断点以外で1本の枝で結ぶ. ペンダントブロック
            (高々4個)がなくなるまで, かつてな2つのペンダントブロックを選び切断点以外で1本の枝で結ぶ
          end;
        end;
      end;

```

アルゴリズムBICONNECT に対して次の補題が成立つ.

[補題2] アルゴリズムBICONNECT はGを $\mu = \max\{d-1, \lceil p/2 \rceil + q\}$ 本の枝付加で2連結化する.

(証明) 補題1の証明より明らか.

(証明終)

3.2 計算時間

ここではアルゴリズムBICONNECT の計算時間が $O(|V|+|E|)$ であることを示す. グラフGが非連結な間(連結になるまで)にかかる計算時間と連結な間(連結になってから2連結になるまで)にかかる計算時間に分けて解析する.

(a) Gが非連結な間にかかる計算時間

Depth First Searchを用いると入力グラフGの各連結成分, ペンダントブロック, 孤立ブロックは $O(|V|+|E|)$ の計算時間で求まる. また, 異なる2つの連結成分 G_1, G_2 に枝を付加して1つの連結成分にする時, G_1, G_2 とも孤立ブロックであるなら枝付加後 G_1, G_2 はペンダントブロックになる(図4(a)). G_1 あるいは G_2 の片方例えば G_1 だけが孤立ブロックであるなら枝付加後 G_1 はペンダントブロックとなり, 枝を付加された G_2 のペンダントブロックはペンダントブロックでなくなる(図4(b)). また G_1, G_2 とも孤立ブロックでないなら, 枝で結ばれる G_1, G_2 の2つのペンダントブロックはペンダントブロックでなくなる(図4(c)). このように枝付加後のGの連結成分, 及びそれに含まれるペンダントブロック, 孤立ブロックは枝付加前のGから

直接求まる。さらに、枝を1本付加したことによる p , d , q の値の変更も次のように定数時間でできる。

(I) G_1 及び G_2 がともに孤立ブロックの場合

$$p := p + 2, \quad q := q - 2, \quad d := d - 1$$

(II) G_1 あるいは G_2 の片方だけが孤立ブロックの場合

$$p \text{ 変化なし}, \quad q := q - 1, \quad d := d - 1$$

(III) G_1 , G_2 とも孤立ブロックでない場合

$$p := p - 2, \quad q \text{ 変化なし}, \quad d := d - 1$$

連結成分が n 個あったとすると $n-1$ 本の枝付加で連結グラフになり、連結成分やペンダントブロックを見つける前処理の時間を除けば1本当たり定数時間で付加できる。よって G が非連結な間の操作にかかる計算時間は全体でも $O(|V|+|E|)$ である。

(b) G が連結な間の計算時間

G 上で直接 v -ブロックやペンダントブロックを探索すると、線形時間で G を2連結化することは難しい。このため、 G のBC木 $T(G)$ を作成し、 $T(G)$ 上で v -ブロックやペンダントブロックを探索する。 $T(G)$ の枝数、点数は明らかに $O(|V|)$ であり、 $T(G)$ の隣接リスト AL は $O(|V|+|E|)$ の計算時間で作成できる。 $T(G)$ の任意の C 点を v とし、 v に対応する G の切断点を v' としよう。また、 v から任意2つの葉 l_1 , l_2 への2本の枝素な道を Q_1 , Q_2 としよう。このとき l_1 , l_2 に対応する G の2つのペンダントブロック P_1 , P_2 は2つの v' -ブロック B_1 , B_2 にそれぞれ含まれている(図5)。 $T(G)$ は木なので、 v に隣接している異なる2つの B 点 v_1 , v_2 を選び枝 (v, v_1) , (v, v_2) から始めて $T(G)$ 上の枝を任意の葉 l_1 , l_2 に到達するまで探索していけば、 v から l_1 までの道 Q_1 と v から l_2 までの道 Q_2 が定まる。もちろん Q_1 と Q_2 は枝素な道である。 AL は図6のような双方向リストであり、枝 (v_i, v_j) に対応する要素がリスト $AL(v_i)$ 及び $AL(v_j)$ の両方に含まれ、互いにポインタで結ばれており、片方から他方が直接アクセスできる。図6の太線で示すようなリストのたどり方をすると、道 (v_i, v_j) , (v_j, v_k) を見つけることができる。この隣接リスト AL を用いると Q_1 と Q_2 が含む枝の本数の枝探索で Q_1 , Q_2 が求まり、異なる v -ブロック B_1 と B_2 に含まれるペンダントブロック P_1 と P_2 を見つけたことになる。

$d-1 > \lceil p/2 \rceil$ または $p=d \leq 3$ の場合には上のようにして見つけたペンダントブロックを枝で結べばよい。従って、 Q_1 , Q_2 の長さ(枝の本数)をそれぞれ m_1 , m_2 とすれば、明らかに $O(m_1+m_2)$ 時間で実行できる。

$d-1 > \lceil p/2 \rceil$ または $p=d \leq 3$ のどちらでもない場合にはBICONNECTは2つ以上のペンダントブロックを持っている v -ブロックを1つ選ぶ必要がある。つまり、 B_1 , B_2 の少なくともどちらか一方は2つ以上ペンダントブロックを含んでいるかどうか判定しなければならないが、これは Q_1 (Q_2)上に次数が3以上の v 以外の点があるかどうか判定すればよい。なぜなら次数3以上の点は Q_1 (Q_2)上の点以外の点と隣接しているため、 B_1 (B_2)に対応する $T(G)$ の部分木 T_1 (T_2)には l_1 (l_2)以外の葉が存在することを意味するからである(図5)。このようにして Q_1 (Q_2)を見つける時に Q_1 (Q_2)上の各点 v_i が3次以上かどうかを調べていけばよい。よって、高々 Q_1 と Q_2 が含む枝の本数の3倍の枝探索で異なる2つの v -ブロックに含まれるペンダントブロックを定めることができ、かつそれらの v -ブロックが2つ以上のペンダントブロックを持つかどうか判定できる。

B_1 , B_2 の少なくともどちらか一方が2つ以上のペンダントブロックを持つ場合には、道 Q_1 , Q_2 を見つけることにより得られた2つのペンダントブロック P_1 , P_2 を選ぶ。もし両方とも1つずつしかペンダントブロックを含んでいない場合、つまり Q_1 , Q_2 に含まれる v 以外の点がすべて次数2以下の場合には、 P_1 , P_2 を選ぶことはできない。この時道 Q_1 , Q_2 をそれぞれ1本の枝 (v, l_1) , (v, l_2) に置き換えてから新たに2つの道を探す(図7)。従って最終的に選ばれた道 Q_1 , Q_2 の長さをそれぞれ m_1 , m_2 とし、 Q_1 , Q_2 以外のたどられた枝の本数を m_3 とすると、枝付加できる2つのペンダントブロックが $O(m_1+m_2+m_3)$ の計算時間で見つかる。

さらに、 $d=2$ かつ $p>3$ の場合には、 $v_0 \neq v$ なる切断点 v_0 があり、ちょうど2つだけペンダントブロックを含む v_0 -ブロック B に v が含まれているかどうかを確かめる必要があり、もしそうならば v_0 を新たに v として選び直してペンダントブロックの探索を行なっていく。 $T(G)$ を用いるとこの操作は次のようになる。

まず、最初に選択した v (次数は2であることに注意)から適当に $T(G)$ 上をたどって2つの葉 l_1 , l_2 まで

の道 Q_1, Q_2 を求める。 Q_1, Q_2 が次のような形をしているかどうか調べる (図8(b))。

$$\begin{aligned} Q_1 &= (v, v_1), \dots, (v_n, l_1) \\ Q_2 &= (v, u_1), \dots, (u_i, u_{i+1}), \dots, (u_m, l_2) \end{aligned} \quad (1)$$

ただし、 l_1, l_2 は葉、 u_i は次数3のB点、 u_i 以外の Q_1, Q_2 上のB点の次数はすべて2であるとする。 Q_1 と Q_2 が(1)の形をしている時には u_i に隣接する切断点 $v_0 (\neq v)$ があり、 l_1, l_2 に対応する丁度2つだけのペンダントブロックを含む v_0 -ブロック B があり v を含む。このとき l_1, l_2 に対応する G の2つのペンダントブロックに枝付加すると切断点 v_0 のまわりに新しいペンダントブロックが出現するため、 v_0 に隣接している u_i 以外の点 w_1 を選び、道 Q_2 を

$$Q_2 = (v, u_1), \dots, (u_i, v_0), (v_0, w_1), \dots, (w_s, l_3) \quad (2)$$

のように選び直す (図8(c))。また $(u_{i+1}, u_{i+2}), \dots, (u_m, l_2)$ を1本の枝 (u_{i+1}, l_2) に置き換える。この時選び直した Q_1, Q_2 から得られる葉 l_1, l_3 に対応する G のペンダントブロックは v_0 の異なる2つの v_0 -ブロックにそれぞれ含まれている。つまり、 v_0 を切断点として、 v_0 から2本の枝素な道を探索して2つの葉を定めたのと同じことになる。

Q_1, Q_2 が(1)のような道であるかどうかの判定は Q_1 と Q_2 が含む枝数に比例した計算時間でできる。もし、 Q_1, Q_2 が(1)のような道である時は(2)のように選び直さなければならない。このとき(1)の道に含まれる枝で(2)の道に含まれていない枝 $(u_{i+1}, u_{i+2}), \dots, (u_m, l_2)$ を1本の枝 (u_{i+1}, l_2) で置換えなければならないがこの置き換えは直ちにできる。最終的に選ばれた道を Q_1, Q_2 とし、その長さをそれぞれ m_1, m_2 とする。またそれら以外のたどられた枝数を m_3 とすると、この場合にも枝付加できる2つのペンダントブロックが $O(m_1 + m_2 + m_3)$ 時間で見つかる。

G の2つのペンダントブロック P_1, P_2 からそれぞれ切断点以外の点を1つずつ選び枝を付加すると、いくつかの2連結成分が1つの2連結成分になり、いくつかの切断点が切断点でなくなる (図9(a)(b))。枝付加後のグラフを G' とした時、 $T(G)$ を $T(G')$ に変更しなければならない。次のアルゴリズム UPDATE により効率良くこの変更ができる。ここで、 G のBC木上のB点とC点を結ぶ2次の点のみからなる任意の道を1本の枝で置き換えてできる木もBC木と再度定義し直し $T(G)$ で表わすことにする。このことは、その道上の全てのB点に対応する2連結成分を1つの2連結成分として考えることを意味する。

Algorithm UPDATE

ステップ1. G の切断点で G' において切断点でなくなるのは $T(G)$ 上の Q_1, Q_2 に含まれる次数2のC点に対応する切断点である (図10(a))。このため、 $T(G)$ からこのような切断点をまず除去する (図10(b))。

ステップ2. Q_1, Q_2 のB点全体に対応する G の2連結成分は G' では1つの2連結成分となるため、 Q_1, Q_2 に含まれるB点の中から1点 v_B を適当に選び、 Q_1, Q_2 上の他のB点すべてを v_B に短絡する (図10(c))。

ここで次の補題が成立つ。

[補題3] アルゴリズム UPDATE は $T(G)$ を $T(G')$ に変更する。

(証明) UPDATE を実行した後のBC木を T' とする。枝付加により G' では Q_1, Q_2 上のB点に対応する G の2連結成分が1つの2連結成分となるが、これは T' の v_B に対応する。これ以外の G' の2連結成分は G の2連結成分と同じであり、これに対応する T のB点は T' において変更を受けていない。ステップ1において、 G において切断点でなくなるC点は T' から除去され、他のC点はそのまま T' 上に残る。よって T' は G' のBC木 $T(G')$ である。

(証明終)

データ構造として図6の隣接リストを使い、 Q_1, Q_2 を見つける時に次数2のC点を記憶しておけば、アルゴリズム UPDATE のステップ1の点除去は1個当たり定数時間で実行できる。また、ステップ2で短絡する時に多重枝が生じるので、AL上でこの多重枝を1本の枝に置換えなければならないが、 Q_1, Q_2 に含まれるC点を記憶しておけばこれも1本当たり定数時間でできる。ステップ2で Q_1, Q_2 上にあるB点 v' を v_B に短絡するには、 $AL(v')$ につながっている要素を $AL(v_B)$ に追加するようにつなぎ換えればよい。この追加は Q_1, Q_2 を定めるときにたどった $AL(v')$ と $AL(v_B)$ の要素間のポイントのつなぎ換えにより実行でき、B点の番号 v', v_B を知る必要はない。図6のデータ構造を用いるとこのつなぎ換えは定数時間でできる。ただし、点 v'

の隣接点は短絡により v_B に隣接するようになるが、 $AL(v_B)$ の各要素の点番号を v' から v_B にいちいち書き換える必要はない。図6のデータ構造を用いると枝の一方の端点から他方の端点へ直接ポインタをたどっていけるため各要素で点番号を参照することなく葉 l_1, l_2 までの道 Q_1, Q_2 を見つけることができる。しかも、 Q_1, Q_2 上の各点の次数が3以上かどうかは、 AL 上で Q_1, Q_2 を見つける時に探索した各要素の left と right のポインタが同一の要素を示しているかどうかで判断できる。よって、点番号を書換えなくとも AL を探索して異なる2つの v -ブロックに含まれるペンダントブロックを見つかることができ、かつそれらの v -ブロックが2つ以上ペンダントブロックをもつかどうか判定できる。以上からステップ1, ステップ2の処理は1つの点当たり定数時間ですむため、アルゴリズムUPDATEは $O(m_1 + m_2 + m_3)$ の計算時間で $T(G)$ を $T(G')$ に変更する。

以上から $T(G)$ をデータ構造として用いると枝付加するペンダントブロックはどの場合も $O(m_1 + m_2 + m_3)$ の計算時間で見つかかり、 $T(G')$ も $T(G)$ からその計算時間で求まる。ここで、 G' は G に枝を1本付加した後のグラフである。

次に d, p の更新が $O(m_1 + m_2 + m_3)$ の計算時間でできることを以下に述べる。

(1) p の変更

枝付加する2つのペンダントブロックを見つかるのに探索した2本の道 Q_1, Q_2 が両方とも次数2以下の点しか含まない場合、枝付加により新しいペンダントブロックが1つできるため、 $p := p - 1$ とし、それ以外は $p := p - 2$ とする。

(2) d の変更

枝付加によりいくつかの C 点の次数が変化するが、これに伴う d の変更を $O(m_1 + m_2 + m_3)$ 時間で実現するため、各 C 点の次数を図11で示すようなデータ構造でおさえておく。つまり、リスト $DA[i]$ に次数 i の C 点 v を入れた要素をすべてつないでおく。また C 点の個数だけのポインタの配列 CA を用意し、 $CA[v]$ で直接 $DA[i]$ 中の要素 v をアクセスできるようにしておく。その要素に次数 i も収めておく。このデータ構造は $T(G)$ から $O(|V| + |E|)$ で作成できる。枝付加により次数が変わる C 点は Q_1, Q_2 上の C 点だけであり、それらの C 点は丁度1だけ減少する。点 v が Q_1 または Q_2 上の C 点である時、リスト $DA[i]$ につながっている要素 v を $CA[v]$ から直接アクセスし、その要素をリスト $DA[i]$ からリスト $DA[i-1]$ につなぎ換え、 v の次数として $i-1$ をその要素に代入すれば、点 v の次数変更に伴うリスト DA の変更は1個当たり定数時間でできる。よって、図11のデータ構造を用いると $O(m_1 + m_2)$ の計算時間で C 点の次数変化に伴うリスト DA の変更が実行できる。このような変更をした後、リスト $DA[d]$ が NIL となったならば $d := d - 1$ 、そうでなければ d は変更しない。このようにして $O(m_1 + m_2 + m_3)$ の計算時間で枝付加後の d の値が求まる。また、 $DA[d]$ のリストを参照することにより直ちに $d(v) = d$ である点 v も見つかる。

以上から次の補題が成立つ。

[補題4] アルゴリズムBICONNECT は $O(|V| + |E|)$ の計算時間で G を2連結化する。

(証明) i 本目の枝を G に付加するのに、 $T(G)$ 上で探索された枝の本数を $m_{1i} + m_{2i} + m_{3i}$ とする。 m_{1i}, m_{2i}, m_{3i} は i 本目の枝を付加した時の m_1, m_2, m_3 である。このとき、 G を2連結化するのに探索した枝の本数は

$$\sum_{i=1}^{\mu(G)} (m_{1i} + m_{2i} + m_{3i}) = \sum_{i=1}^{\mu(G)} (m_{1i} + m_{2i}) + \sum_{i=1}^{\mu(G)} m_{3i}$$

となる。ここで $\mu(G) = \max\{d-1, \lceil p/2 \rceil\}$ であり、 i 本目の枝付加が道 Q_{1i} と Q_{2i} の葉 l_{1i} と l_{2i} に対応するペンダントブロック P_{1i} と P_{2i} の間になされたとしている。アルゴリズムUPDATEで $T(G)$ を $T(G')$ に変更するとき点除去や点短絡により、 Q_{1i} と Q_{2i} に含まれる枝の少なくとも半分、すなわち $(m_{1i} + m_{2i}) / 2$ 本の枝が $T(G)$ から除去される。従って、 G が2連結グラフになるまでには最初にあった $T(G)$ の枝だけが除去されるので、 $T(G)$ の高々2倍の本数の枝を Q_1, Q_2 として走査することになる。よって、

$$\sum_{i=1}^{\mu(G)} (m_{1i} + m_{2i}) \leq O(|E|)$$

である。従って、

$$\mu(G) \sum_{i=1} m_{3i} \leq 0(|E|)$$

となることを証明すればよい。

そこで、 C 点 v が道探索の始点として選ばれ、探索された道が次数2以下の点しか含まず Q_1 、 Q_2 として採用されなかった場合を考える。今、この道を v, w, \dots, ℓ (葉)とすると、この道は1本の枝 (v, ℓ) に置換えられるため、これ以降この道上の枝は再度探索されることはない。ここで、置換えた枝 (v, ℓ) は、 v に隣接する他のどの枝 (v, w) よりも後に探索されるようにしておく。そのうえで枝 (v, ℓ) が探索されたときのことを考える。点 v がこの道探索の始点である場合には C 点 v から出る全ての道は既に各々1本の枝で置換えられているはずであり枝 (v, ℓ) は必ず Q_1 か Q_2 として採用され、アルゴリズムUPDATEにより除去される。一方 v 以外の点 u が道探索の始点で、かつ枝 (v, ℓ) が Q_1 または Q_2 に含まれたならアルゴリズムUPDATEによりこの枝は除去される。 Q_1 と Q_2 どちらにも含まれないときは、この時探索された道 u, \dots, v, ℓ 上の枝は1本の枝に置換わるため、 (v, ℓ) は再度探索されることはなくなる。以上から Q_1 、 Q_2 以外として探索された枝は必ず除去されるのでどの枝も高々1回しか探索されないことが解る。さらに、置換えにより追加される枝の総数は明らかに n 本以下であるので、

$$\mu(G) \sum_{i=1} m_{3i} \leq 0(|E|)$$

である。

(証明終)

本章では、文献(1)、(2)のアルゴリズムの不備な点を正した線形時間2連結化アルゴリズムBICONNECTを与えた。アルゴリズムBICONNECTはBC木を表現するデータ構造として一般的なデータ構造である隣接リストを用いているため、特殊なデータ構造を用いている文献(2)のアルゴリズムよりリスト処理等インプレメンテーションも直観的にわかり易くなっている。

4 平面グラフ2連結化アルゴリズム

4.1 アルゴリズム

ここでは、平面グラフ G に最小本数の枝を付加して2連結平面グラフにする線形時間アルゴリズムPBICONNECT(G)を与える。ただし、 G の平面埋め込みは固定して変更しないとする。平面グラフの2連結化の場合、次の点一般グラフのアルゴリズムと異なる。

(a) 枝付加後も平面性を保持しなければならないため、異なる面に含まれるペンダントブロックや孤立ブロックを枝で結ぶことはできない。図12に示す平面グラフの場合、ペンダントブロック A と B は同じ面に含まれているため枝付加しても平面性は保持されるが、 C と D はそれぞれ異なる面に含まれているため枝付加すると必ず点以外で枝が交差し平面性が損われる。このため、各面ごとにその面に含まれているペンダントブロックや孤立ブロックを処理し、2連結化していかなければならない。

(b) ペンダントブロックや孤立ブロックに含まれる点の中から枝を付加する2点を選ぶ時、必ずその外周上の点(例えば図12の点 u)を選択しなければならない。ペンダントブロックまたは孤立ブロックの外周以外の内部の点(例えば図12の点 v)を選択し、他のペンダントブロックまたは孤立ブロックに含まれる点との間に枝を1本付加すると、この枝は少なくとも前者のペンダントブロックまたは孤立ブロックの外周の枝と交差するため平面性が損われる。

(a)、(b)から次の補題が成立つ。

[補題5] G の各面の面グラフをそれぞれ最小本数の枝付加で平面性を保持したまま2連結化すれば、 G に最小本数の枝を付加した平面2連結グラフが得られる。

(証明) (a)より各面に含まれるペンダントブロック及び孤立ブロックはそれらが含まれている面内で処理

されなければならない。よって、各面ごとに平面性を保って最小本数の枝付加で2連結化すればG全体を平面グラフのまま最小本数の枝付加で2連結化したことになる。また、(b)より枝付加する点は必ず各面の面閉路上になければならないため面閉路に含まれる点や枝のみを考えれば良い。よって面グラフを2連結化すればよい。

(証明終)

アルゴリズムPBICONNECTの概略を以下に示す。

Algorithm PBICONNECT(G)

ステップ1. Gのすべての面 F_i についてその面グラフ G_i を求める。

ステップ2. 各 G_i を最小本数の枝付加でかつ平面性を保持したまま2連結化する。

ステップ2で各 G_i を2連結化するが、この操作に一般グラフの2連結化アルゴリズムBICONNECTを用いる。ただし、平面性を保持するように枝付加しなければならないため G_i が連結の場合には枝付加するペンダントブロックの選び方に少し制約を加える。 G_i が連結の場合2つのペンダントブロック P_1, P_2 を任意に選択し枝付加すると、 P_1, P_2 が含まれていた面 F_1 が2つの面 F_2, F_3 に分割される。このとき、同じ面 F_1 に含まれていた P_1, P_2 以外のペンダントブロックが異なる2つの面 F_2, F_3 にそれぞれ分離して含まれることがある。図13(a)のグラフの面 F_i に着目する。ペンダントブロックAとBの間に枝付加すると図13(b)の点線で示した場合のようにペンダントブロックBとDが異なった面に含まれてしまい、BとDの間には枝を付加できなくなる。アルゴリズムBICONNECTをそのまま G_i に適用すると、これら枝付加できない組合せが生じ、かつそれらを枝で結ぶ場合がある。BICONNECTを用いて平面性を保ったまま G_i を2連結化するときは P_1, P_2 の選び方に制約を加えて枝付加した後も他のペンダントブロックがすべて同一の面内に含まれるようにしながら行なう。このためアルゴリズムBICONNECTに以下のような変更(1)(2)を加える。

(1) BICONNECTでは、v-ブロック G_2 として G_1 以外のv-ブロックならどれを選んでもよかったが、ここでは切断点vのまわりでv-ブロック G_1 の隣りに埋め込まれているv-ブロックを G_2 として選ぶ(図14)。

(2) G_1, G_2 からそれぞれ枝付加するペンダントブロックを選ぶ時、BICONNECTでは適当に1個ずつ選択してよかったが、ここでは隣り合ったペンダントブロック同士を選ぶようにする(図14の斜線で示されたペンダントブロック)。

このようにすれば残りのペンダントブロックが2つの面に分離されることはない。ここで次の補題が成立つ。

[補題6] 変更(1)(2)を加えたアルゴリズムBICONNECTは連結グラフ G_i を平面性を保ったまま最小本数の枝付加で2連結化する。

(証明) 変更(1)(2)により枝付加後も G_i の残りのペンダントブロックは1つの面に含まれている。よって、平面性を保ったまま枝付加が行なわれる。また、アルゴリズムBICONNECTでは、v-ブロックやペンダントブロックはかってに選択してよかったが、変更(1)(2)はこの選択に制約を加えることになる。この制約のもとでも枝付加できるペンダントブロックが必ず見つけられるので、BICONNECTは $\max\{d_i - 1, \lceil p_i / 2 \rceil\}$ 本の枝付加で G_i を2連結化するといえる。ここで d_i, p_i は G_i における p, d である。(証明終)

G_i が非連結の場合には、アルゴリズムBICONNECTをそのまま用いても平面性は保たれるため、アルゴリズムPBICONNECTはステップ2で G_i が非連結の場合には、アルゴリズムBICONNECTをそのまま適用し、 G_i が連結な場合のみ変更(1)(2)を加えたBICONNECTを適用する。このため、アルゴリズムPBICONNECT(G)は各 $G_i \in G$ を $\max\{d_i - 1, \lceil p_i / 2 \rceil + q_i\}$ 本の枝付加で2連結化する。 d_i, p_i, q_i は G_i の d, p, q である。よって、アルゴリズムPBICONNECT(G)は

$$\sigma(G) = \sum_{G_i \in G} \max\{d_i - 1, \lceil p_i / 2 \rceil + q_i\}$$

本の枝付加でG全体を平面2連結化する。Gを平面2連結化するには少なくとも $\sigma(G)$ 本の枝付加が必要であることは補題5,6より明らかであるため以下の定理が成立つ。

[定理1] アルゴリズムPBICONNECTは平面グラフ G を平面性を保ったまま最小本数 $\sigma(G)$ の枝付加で2連結化する。(定理終)

4.2 計算時間

ここではアルゴリズムPBICONNECTが線形時間アルゴリズムであることを示す。 G の各面 F_i の面グラフ G_i は F_i の面閉路を時計回りにたどることによって求まる。図4.9のようなデータ構造をもつ G の平面埋め込みリストを用いると、 $O(|E|)$ の計算時間ですべての G_i が求まる。以下で変更(1)(2)を加えたアルゴリズムBICONNECTが各 G_i を $O(|V_i| + |E_i|)$ の計算時間で2連結化することを示す。ここで、 G_i の枝集合を E_i 、点集合を V_i とする。

(a) G_i が非連結な場合

アルゴリズムBICONNECTと同じ動作をするので $O(|V_i| + |E_i|)$ の計算時間で G_i を2連結化する。

(b) G_i が連結な場合

G_i の平面埋め込みどおりに埋め込まれたBC木 $T(G_i)$ を作成する必要があるが、これは F_i の面閉路を時計回りに一巡することにより $O(|V_i| + |E_i|)$ の計算時間でつくれる。この $T(G_i)$ を用いると、BICONNECTに付加した変更(1)(2)は次のような操作に置き換わる。ここで、枝付加する2つのペンダントブロックを見つけるために、 $T(G_i)$ 上で求める2本の道を

$Q_1 = (v, v_1), (v_1, v_2), \dots, (v_n, l_1)$ $Q_2 = (v, u_1), (u_1, u_2), \dots, (u_m, l_2)$ とする。ただし、点 v は $T(G_i)$ 上のC点、及び l_1, l_2 は葉である。

(1) 点 v の回りで枝 (v, v_1) の反時計回りに隣の枝を Q_2 の最初の枝 (v, u_1) として選ぶ。

(2) Q_1 を探索する時、枝 (v_{i-1}, v_i) の次の枝 (v_i, v_{i+1}) は点 v_i の回り、枝 (v_{i-1}, v_i) の時計回りに隣の枝を常に選ぶ(図15)。 Q_2 を探索する時、枝 (u_{i-1}, u_i) の次の枝 (u_i, u_{i+1}) は点 u_i の回り、枝 (u_{i-1}, u_i) の反時計回りに隣の枝を常に選ぶ(図15)。

変更(1)(2)を付加したアルゴリズムBICONNECTが見つけた P_1, P_2 の葉 l_1, l_2 は明らかに隣合ったペンダントブロック同士に対応している。 $T(G_i)$ の埋め込み隣接リストとして図6のデータ構造を用いると、時計回りに隣の枝、反時計回りに隣の枝は即座にアクセスできるため、変更(1)(2)をアルゴリズムBICONNECTに付加しても枝探索に要する時間は変化しない。一般グラフに適用するアルゴリズムBICONNECTと各 G_i に適用する平面性を保持したアルゴリズムBICONNECTとの間の相異点は道 Q_1, Q_2 を見つけるための枝探索に、以上の変更(1)(2)を付加するかしないかという点だけなので、(1)(2)を付加した平面グラフ用のアルゴリズムBICONNECTは $O(|V_i| + |E_i|)$ の計算時間で各 G_i を平面2連結化する。以上から次の補題が成立つ。

[補題7] アルゴリズムPBICONNECT(G)は $O(|V|+|E|)$ の計算時間で平面グラフ G を平面性を保ったまま2連結化する。

(証明) G の各面グラフ G_i は $O(|E_i|)$ で求まり、各 G_i は $O(|V_i| + |E_i|)$ で平面2連結化される。よって、 G 全体は $O(|V|+|E|)$ で平面2連結化される。(証明終)

4.3 むすび

本章では、与えられたグラフ G の平面埋め込みを固定したままで枝を付加して平面2連結化する線形時間アルゴリズムを与えた。 G の平面埋め込みを固定しない場合には、ペンダントブロックをどの面に埋め込むかで2連結化するのに必要な付加枝の本数、即ち $\sigma(G)$ が変化する。 $\sigma(G)$ が最小となるような G の平面埋め込みを求める問題は未解決である。しかし、実用上の問題では、 G の平面埋め込みは固定されており、変更できない場合が多いと思われる。

(参考文献)

[1] K. P. Eswaran and R. E. Tarjan, Augmentation Problems, SIAM J. Comput, 5, 4, pp653-665, 1976

[2] A. Rosenthal and A. Goldner, Smallest augmentations to biconnect a graph, SIAM J. Comput, 6, 1, pp55-66, 1977

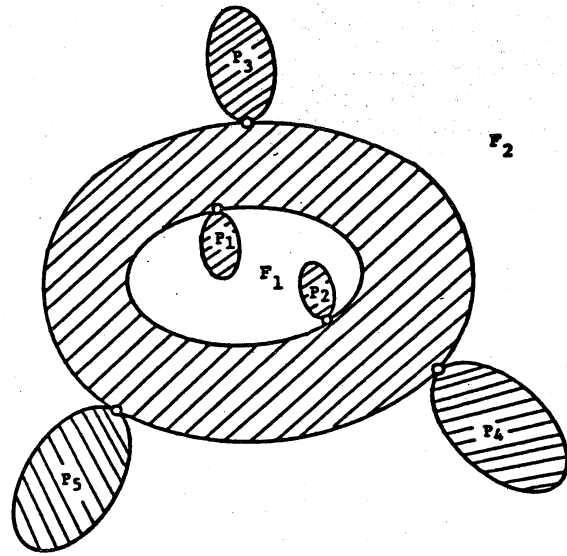


図1 面とペンダントブロックの関係

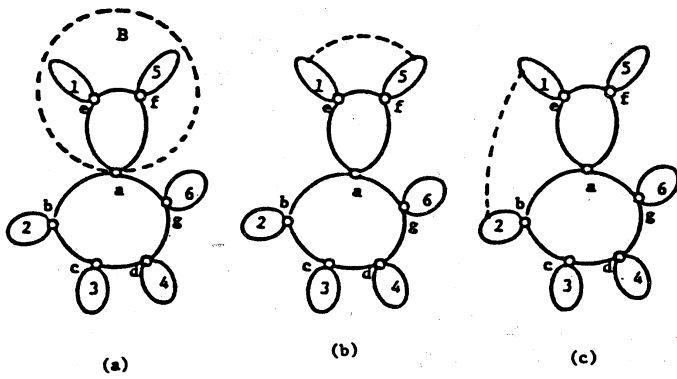


図2 枝付加の特殊例

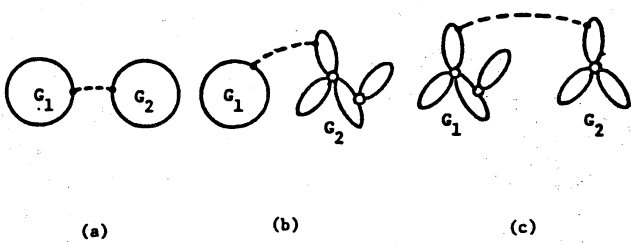


図4 各連結成分間の枝付加

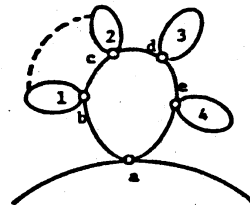


図3 ペンダントブロックができない例

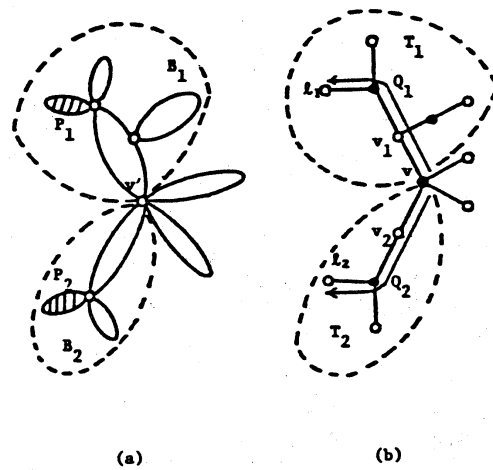


図5 GとT(G)の関係
(a) G (b) T(G)

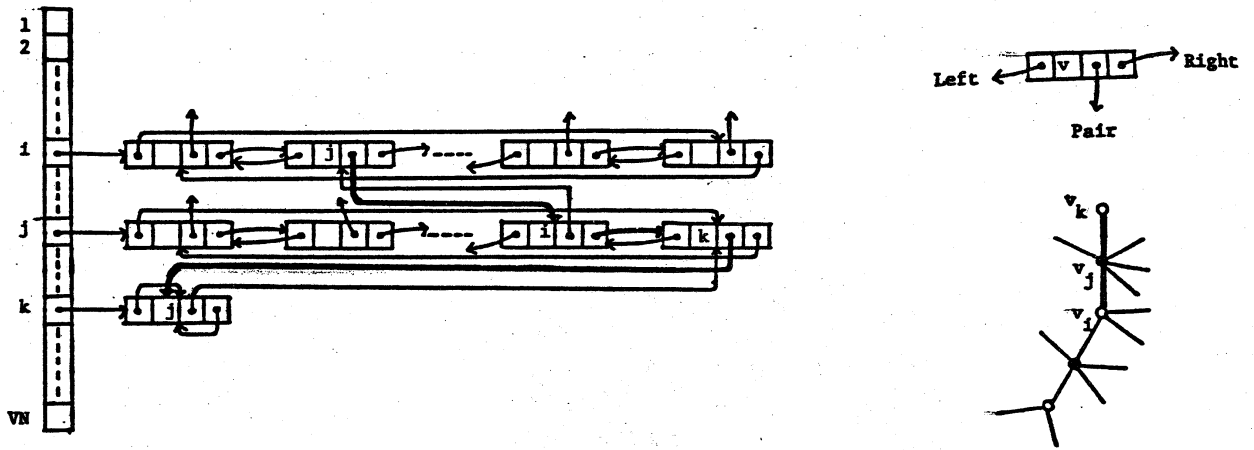


図6 ALのデータ構造

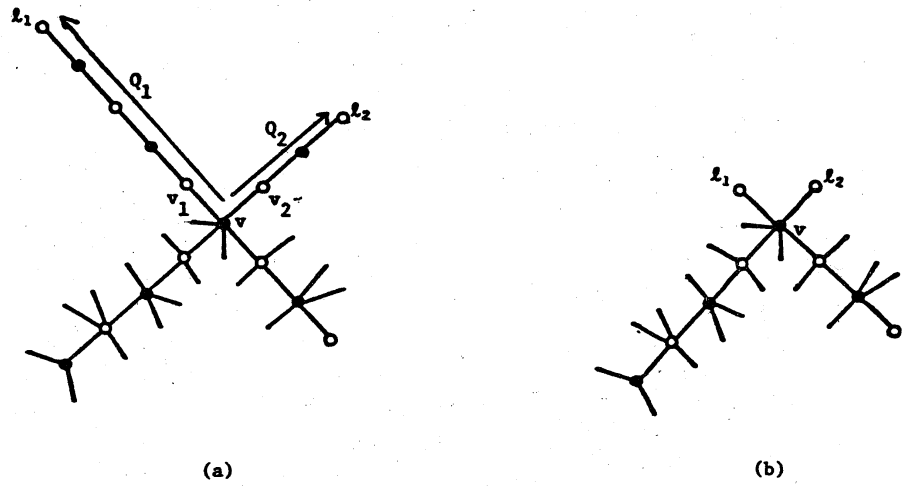


図7 Q_1, Q_2 の短絡
(a) 短絡前 (b) 短絡後

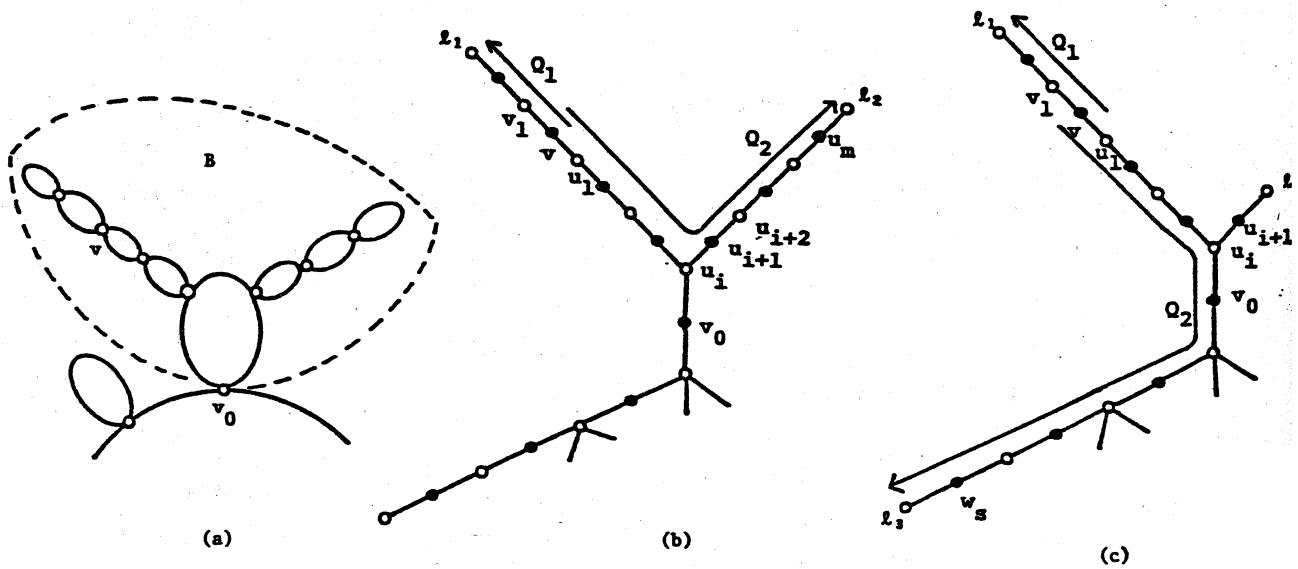


図8 $d = 2$, かつ $p > 3$ の場合の処理

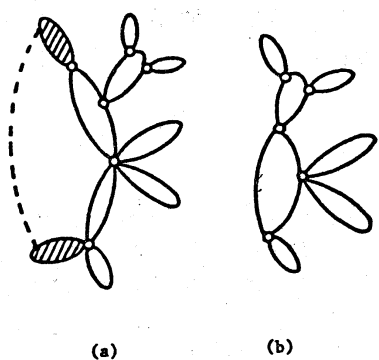


図9 GとG'
(a) G (b) G'

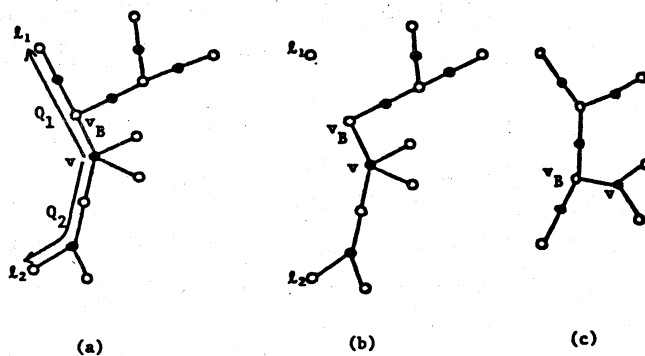


図10 T(G)の更新
(a) T(G)
(b) 次数2のC点を除去
(c) T(G')

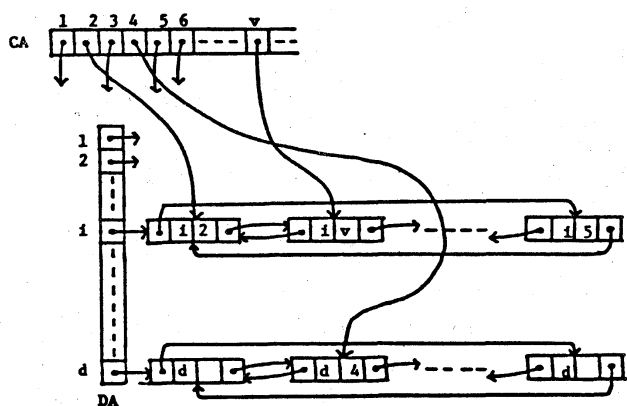


図11 DAのデータ構造

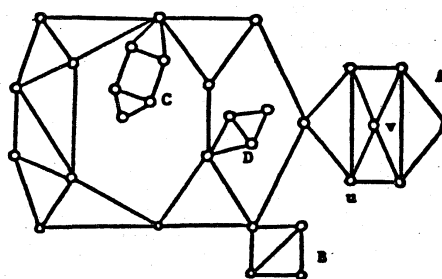


図12 平面グラフG

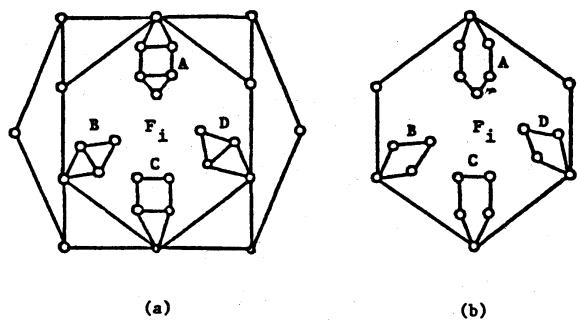


図13 ペンダントブロックの選び方
(a) G (b) Gi

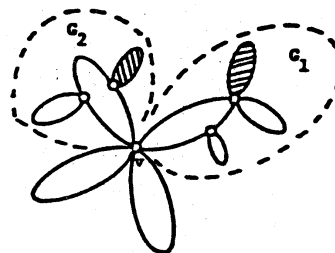


図14 条件(1)(2)の説明

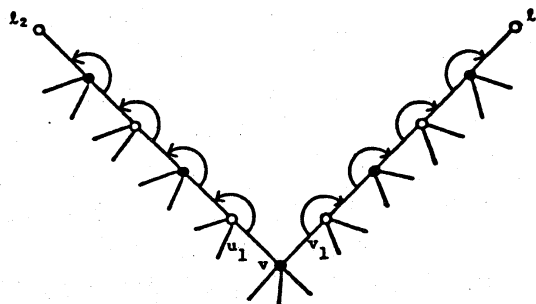


図15 枝の探索方法