大規模な多種結線実現問題の発見的解法†

丸本 悟†† 岸本一男†† 翁長健治††

枝の使用回数に制限のあるネットワークにおいて，q 個のソース・シンク対間のおのおのに要求された本数の結線群を実現する問題を，多種結線実現問題という．本論文は，「翁長の多重フロー定理」に立脚した発見的アルゴリズムを考案し，(i)大規模ネットワークへの適応と高速化，(ii)アルゴリズムの多様化の2点を重視して開発した実用的プログラムの精度と計算時間の特性を計算実験により明らかにしたものである．ネットワーク規模の制約より LP, IP の適用が不可能であるため，結線実現の難易度は高いが結線解の存在が自明な問題に対する結線復元率を精度の目安とすることにした．主要な結果として，計算時間が $|V|^{1.3}q\log R$ に比例することおよび平均復元率 96% を得た．ただし，$|V|$ は節点数，q はソース・シンク対数，R は結線要求総本数である．

1. まえがき

節点集合 V，枝集合 E をもつグラフ $G=(V,E)$ の各枝 $e\in E$ に容量 c_e を与えたものをネットワーク $N=(G,C)$, $(C=[c_1,c_2,\cdots,c_{|E|}])$ と書く．多種フローとは，ソースおよびシンクと呼ぶ節点対 (s_i,t_i) 間に存在する端子流量 F_i をもつフロー f^i $(i=1,2,\cdots,q)$ の集合である．このとき，枝 e を流れるフローの和は，枝容量 c_e を越えないという制約 $\left(0\leq\sum_{i=1}^{q}f_e^i\leq c_e\right)$ を満たしていなければならない．

ネットワーク上で，与えられたソース・シンク対 (s_i,t_i) 間に要求量 r_i 以下のフローを $\sum F_i$ が最大となるように割り当てる最大多種フロー問題は，線形計画法に定式化でき古くから多くの研究がなされてきた[4]．その結果，1種または無向ネットワーク上の2種フロー $(q=1,2)$ の場合，いわゆる最大フロー・最小カット定理[1),2)]が成立しグラフ論的手法による効率のよいアルゴリズムが種々工夫されている[1),10)]．多種フローの存在に対し，ネットワーク的特徴をもつ一般的必要十分条件は，翁長により示されている[8)]．しかし，有向ネットワーク上の2種，および3種以上のフローの場合，ごく限られた形状をとるネットワークに対するものを除き[5),9),11)]，グラフ論的手法による効率のよいアルゴリズムは知られておらず線形画法にたよらざるをえない．

一方，本文が対象とする多種結線実現問題とは，(s_i,t_i) 間に要求本数 r_i 以下の結線を，総本数を最大とするように割り当てるもので，最大多種フロー問題にフロー f^i が整数値を取るという制約を加えたものである．こうした問題は，交通網・通信網等の制御，VLSI の配線など多くの応用分野をもち実用上重要であるが，整数計画法がアルゴリズム的に NP-困難であり，ネットワーク規模に対して指数関数的に増大する計算時間を必要とするため，最適解でなくともそれに近いものが短時間で得られる発見的アルゴリズムが要求されている．

本論文は，著者らが先に発表した発見的アルゴリズム[7)]を多様化・高速化させ，ネットワークの大規模化と高精度化への適応をはかったものである．本文のアルゴリズムは，翁長の「多重フロー定理」をもとにしたもので，その要点を次にまとめる．

1) 結線追加の際，多くの結線で混み合うカットセットをボトルネックと呼び，これに含まれる枝に混雑の度合に応じた重み w を与える（枝重み生成）．

2) 各ソース・シンク対間の結線を，w を枝の長さとした最短径路に沿っておこなう．

3) 残留要求本数の一定割合（100α%）に相当する結線を追加するごとに枝重みを更新する（逐次割当）．

4) 結線追加ができなくなった時点で，割当済み結線の一部（100β%）を解除し，使用されていた枝容量をネットワークに戻した上で再結線をおこなう（緩和－再結線）．

上記 1) ～4) は，アルゴリズムの基本的な構造を示したもので，ボトルネックの発見，結線割当順序，解除される結線の選択等に多くの変種が存在する．HITAC M 200 H 上で計算時間と精度の計測をおこない，一例として節点数 782，枝数 2,260 の無向ネットワーク上で 220 種，765 本の結線に対し全計算時間

† A Heuristic Algorithm for Solving Large Scale Multi-Connection Assignment Problems by SATOSHI MARUMOTO, KAZUO KISHIMOTO and KENJI ONAGA (Faculty of Engineering, Hiroshima University).

†† 広島大学工学部第2類（電気系）

276 秒を得ている．整数計画法はぼう大な計算時間と記憶領域を必要とするため十分大きなネットワークに対し適用不能である．精度測定として，前もって定めた結線径路を多数重畳して得られるネットワークをサンプルとして用い，結線が復元される割合を求め平均値 96% を得ている．結線復元率は精度のよい目安，あるいはその代替として用いることができる．

2. 多種結線実現アルゴリズムの基本構造

一般グラフにおける多種フロー存在の必要十分条件は，翁長により Ford-Fulkerson の最大フロー・最小カット定理を拡張した供給－消費不等式の形で与えられることが示された[8]．すなわち，枝容量 $C=\{c_e|e\in E\}$ をもつ有向または無向グラフ上で，与えられたソース・シンク対 (s_i, t_i) 間に要求量 $r_i\ (i=1,2,\cdots,q)$ のフローが割当可能であるためには，任意の非負整数からなる枝重み $W=[w_1, w_2, \cdots, w_{|E|}]$ に対し，次の供給－消費不等式の成立が必要十分条件となる．

$$\begin{cases}\sum_{e\in E} c_e w_e \geq \sum_{i=1}^{q} r_i d_i \\ d_i : w_e \text{ を長さとした } s_i\text{-}t_i \text{ 間の最短距離}\end{cases} \quad (1)$$

個々の枝重みはあらゆる非負整数値をとり，その大部分は冗長であるが供給－消費不等式を満たすフローの割当法は現在のところ不明である．

本アルゴリズムでは枝重み W を，冗長が少なく有効に使用すべき枝ほど大きな値をもつように選び，その上で各ソース・シンク間の最短径路に沿って結線を割り付ける方法によって式(1)の関係を保存するか，または超過分を小さくおさえる方針をとる．このとき，枝容量消費によるネットワークの変化を新しい結線に反映させるため，結線を少量追加するたびに枝重みを新たに求める逐次割当法を採用する．実現結線本数は結線の履歴に依存するため，結線の追加が不能となった時点で割当済みの結線の一部を未結線とし，それらが使用していた枝容量をネットワークに戻す操作すなわち結線緩和，をおこなった上で再び結線をおこない，結線率の改善を図る．この方法は交通網計画の分野でよく利用される改良逐次割当法 IA*[6] に準じたもので，緩和以前の結線を初期結線，以後のものを再結線と呼び区別する．

本アルゴリズムで最も重要な枝重みの生成法は，節点集合 V の部分集合 X_i が定める有向カットセット $K_i=(X_i, \bar{X}_i)$, $(\bar{X}_i \triangleq V-X_i)$ の組合せ $\mathcal{K}=\{K_i|i=1, 2, \cdots, k\}$ を用いて次のようにおこなう．

$$w_e=\sum_{j=1}^{k} \|\{e\}\cap K_j\| \quad (2)$$

式(2)は w_e を，枝 e を含む $\mathcal{K}$ 中のカットセットの個数に等しいとしたもので，$k=1$ の場合式(1)は

$$\sum_{e\in(X,\bar{X})} c_e \geq \sum_{i;\ s_i\in X, t_i\in\bar{X}} r_i \quad (3)$$

と変形され，結線本数は切断容量を越えられないというよく知られた結果を得る．$k\geq 2$ の場合には，各ソース・シンク間の結線が $\mathcal{K}$ 中のカットセット群を最小回数で横断する径路に沿うべきことを示している．このように，枝重みが結線を追加する際の枝の貴重さを表すことから，飽和しやすいカットセット，すなわちボトルネックを $\mathcal{K}$ の構成要素として採用する．ただし，式(2)による枝重みは枝が $\mathcal{K}$ 中のカットセットに含まれるか否かのみを評価したもので，カットセットの飽和の度合を評価しておらず一般的でない．値1付近で急速に増大するボトルネック関数 g を用いて次式のように変更すれば，混み合うカットセットを必要以上に横断する結線を抑制する効果があり，枝容量の有効利用が促進される．

$$\begin{cases}\sigma_i=\sum_{i;\ s_i\in X, t_i\in X_i} r_i \Big/ \sum_{e\in(X_i, X_{\bar{i}})} c_e \\ w_e=\sum_{j;\ e\in K_j} g(\sigma_j)\end{cases} \quad (4)$$

3. 多種結線実現アルゴリズム

多種結線実現アルゴリズムは，枝重み生成，結線割当，結線緩和の三つのサブアルゴリズムにより構成される．精度と計算時間は，扱うネットワークのトポロジ，ソース・シンク対の分布，要求量等と密接に関連して変化することが予想される．したがって各サブアルゴリズムは，扱う問題や必要とする精度に応じて適宜選択使用されるべきであり，唯一存在するというものではない．アルゴリズムの効率と扱うネットワークのトポロジ等との関連は，より多くの理論的考察と実験的データを必要とし現時点では明確になっていない．以下では，種々考えられるサブアルゴリズムのうちで一般的と思われるものを二，三示す．

3.1 枝重み生成アルゴリズム

枝重みが枝の貴重さを反映するには，$\mathcal{K}$ を結線追加時に混雑しやすいカットセットで構成することが重要である．カットセットの混雑度は，カット容量とそこを通過する結線の要求量によって決まる．**図1**に示す一般的アルゴリズム MFW(Maximum Flow Weighting) は，m 個以下のソース・シンク対の組合せに対

```
procedure MFW (m);
    begin
        for e ∈ E do w_e : =0;
        V にダミーのソース v_s, シンク v_t を加える;
        for n : =1 to m do begin
            n 個のソース・シンク対の組を PC_ni,
            (i=1, 2, ···, qC_n) とする;
            for i : =1 to qC_n do begin
                rt : =0;
                for (s_j, t_j) ∈ PC_ni do begin
                    ダミーの枝 (v_s, s_j), (t_j, v_t) を E に
                    加え枝容量を ∞ とする;
                    rt : =rt+r_j
                end;
                v_s, v_t 間に最大フローを流し, 最小カッ
                トK と最大フロー値 MF を求める;
                for e ∈ K do w_e : =w_e+g(rt/MF);
                ダミーの枝を除く
            end;
        end
    end;
```

図 1 アルゴリズム MFW
Fig. 1 Algorithm MFW.

```
procedure SW;
    begin
        for v ∈ V do begin
            r_v^± : =v をソース (+), シンク (−) とする
                要求の和;
            c_v^± : =v の出枝 (+), 入枝 (−) の容量の和
                end;
        for e=(u, v) ∈ E do
            w_e : =g(r_u^+/c_u^+ + r_v^-/c_v^-)
    end;
```

図 2 アルゴリズム SW
Fig. 2 Algorithm SW.

する最小カットで $\mathcal{K}$ を構成している．m 個以下のソース・シンク対の組合せは，${}_qC_1+{}_qC_2+\cdots+{}_qC_m \triangleq A$ 個存在するため MFW の計算量は $O(A \cdot MF)$ となる．ただし MF は最大フローの計算量とする．m を大きくとれば，MFW は多大な計算時間を必要とし現実的でないので以後とくにこだわらない限り $m=1$ とし，$\mathcal{K}$ を各ソース・シンク間の最小カットで構成する．

ネットワークのトポロジと枝容量が一様な場合，最小カットはソースまたはシンクに接続する枝集合上に発生しやすいため，**図 2** に示す $O(|E|)$ のアルゴリズム SW (Simple Weighting) は，とくに初期結線に対して有効である．

3.2 結線割当アルゴリズム

枝重みを生成した後，総残留要求の一部 (100α%) にあたる結線をソース・シンク間の最短径路に沿って割り付ける．結線の追加をおこなうソース・シンク対

```
procedure NPFC (cn, cr, γ);
    begin
        cr : =0; SP : =φ;
        while cn−cr>0 do begin
            for i : =1 to q do begin
                (s_i, t_i) 間の最短径路 S_i と距離 l_i を
                求める;
                if l_i<∞ then SP : =SP ∪ {S_i};
            end;
            if SP=φ then go to exit;
            while cn−cr>0 and SP≠φ do
                begin
                    l_m : =min {l_j | S_j ∈ SP};
                    mcap : =min {c_e | e ∈ S_m};
                    if mcap>0 then begin
                        Δ : =min (cn−cr, ⌈γ * mcap⌉);
                        k : =k+1; p_k : =S_m;
                        np_k : =Δ; P : =P ∪ {p_k};
                        r_m : =r_m−Δ; cr : =cr+Δ;
                        for e ∈ S_m do c_e : =c_e−Δ
                    end;
                    SP : =SP−{S_m}
                end
        end;
    exit:
    end;
```

図 3 アルゴリズム NPFC
Fig. 3 Algorithm NPFC.

の選び方は，実現結線本数を大きく左右するため注意が必要である．一般に距離の短い結線ほど他の結線を妨害する可能性が小さいので，**図 3** に示すアルゴリズム NPFC (Nearest Pair First Connection) が基本となる．NPFC は全ソース・シンク対間の最短径路集合 SP から短い径路を順に選んで結線を割り当て実現結線集合 P に追加する．cn は結線すべき本数，γ は結線追加のペースを表し区間 (0, 1] 内の値をとる．P は結線径路 p_i $(i=1, 2, \cdots, k)$ の集合，対応する np_i は p_i に割り当てられた結線の本数とする．

NPFC を用いて逐次割当をおこなうには $cn=\alpha R$ (R; 総残留要求) として繰り返し呼び出せばよい．結線の追加が継続不能かつ $cn>cr$ ならば，引き続いて結線緩和をおこなわねばならない．NPFC は最悪の場合，1 本の結線を割り当てるのに全ソース・シンク対間の最短径路を求めるので $O(\alpha R \cdot q \cdot ST)$ の計算量をもつ．ただし ST は最短径路の計算量とする．

計算時間の節約を目的とすれば，ランダムに選択したソース・シンク対間の最短径路に沿って結線追加を繰り返すアルゴリズム RC (Random Connection) が最速である．この場合の計算量は $O(\alpha R \cdot ST)$ である．

```
procedure SRX (rn, rr, δ);
    begin
        Set: =φ; rr: =0;
            while rn−rr>0 do begin
                for i: =1 to q do begin
                    (si, ti) 間で飽和枝を最も多く含む
                    結線径路を spi, その結線本数を
                    sni, 含まれる飽和枝数を sei とす
                    る; if sei>0 then Set: =Set ∪
                    {spi}
                end;
                if Set=φ then go to exit;
                while Set=φ and rn−rr>0 do
                    begin
                        sem: =max {sek|spk ∈ Set};
                        Δ: =min (⌈δ*snm⌉, rn−rr);
                        snm: =snm−Δ; rm: =rm+Δ;
                        rr: =rr+Δ;
                        for e ∈ spm do ce: =ce+Δ;
                        Set: =Set−{spm};
                        if snm=0 then spm を除き P
                        を修正する
                    end
            end;
exit:
    end;
```

図 4 アルゴリズム SRX
Fig. 4 Algorithm SRX.

3.3 結線緩和アルゴリズム

緩和されるべき結線は枝の使用効率の悪いもの，いい替えればボトルネックを不用意に横断し他の結線を妨害しているものである．しかし，結線の実現本数を改善するためには，複数の結線を同時に緩和しなくてはならない場合もあり，枝の使用効率の明確な基準を定めることはむずかしい．

図 4 に示すアルゴリズム SRX (Saturation Relaxation) は，飽和枝を含む結線ほど枝の使用効率が悪いとしたものである．このとき，結線緩和がネットワークの一部で局所的に生じても，改善への寄与は小さいため多くのソース・シンク対間で緩和をおこなう．SRX において，rn を緩和すべき本数，δ を結線緩和のペースを表す区間 (0, 1] 内の実数とする．Set には，各ソース・シンク対間の飽和枝を最も多く含む結線を格納する．

SRX は，各ソース・シンク対間で飽和枝数最大の結線を求めるので $O(\theta \cdot \max(R|V|, q^2))$ の計算量をもつ．ただし，$\theta \triangleq \beta R/q$ とした．

一方，枝の使用効率に明確な基準がない以上，ランダムに結線を選択しこれを緩和する RRX (Random Relaxation) も考えられる．RRX は，各結線に対し均等に緩和の機会を与えるもので，$O(\beta R|V|)$ の計算量をもち最速である．

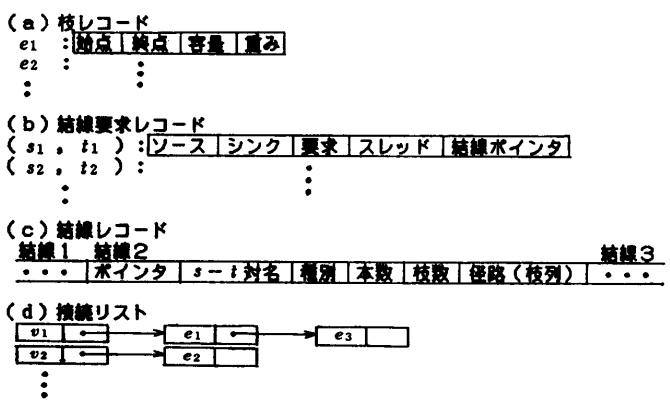


図 5 データ構造
Fig. 5 Data structures.

4. プログラムの設計

4.1 データ構造

各サブアルゴリズムに共通するデータは，ネットワーク，結線要求（ソース・シンク対も含めて考える），実現結線に大別できる．これらは，図 5 に示す 3 種類のレコード集合と，グラフ表現の接続リストにより保持される．

枝および結線要求レコードは，対応する枝およびソース・シンク対番号により参照される．結線要求レコード中のスレッドは，残留要求が存在する次のソース・シンク対番号を格納し，結線ポインタは割り当てられている最新の結線レコードの位置を示す．結線レコードは，異なる長さの結線径路（枝列）を含むので，一次元配列中に格納され，ソース・シンク対ごとにポインタにより割当順につながれる．種別というタグは，緩和 - 再結線を通じて最大の実現総本数を示したときの結線，現在の結線，最短径路を区別するものである．

4.2 機能分割

プログラムは，2 章で述べた基本構造に沿って各サブアルゴリズムを組み合わせることにより構成される．利用するサブアルゴリズムの交換を容易にするため，おもなプログラム単位の機能を図 6 に示すように設計した．3 章で述べたものはすべて，図 6 の最下位機能を利用し容易に実現できる．

4.3 主プログラム

基本的なアルゴリズムは，まず精度は高く

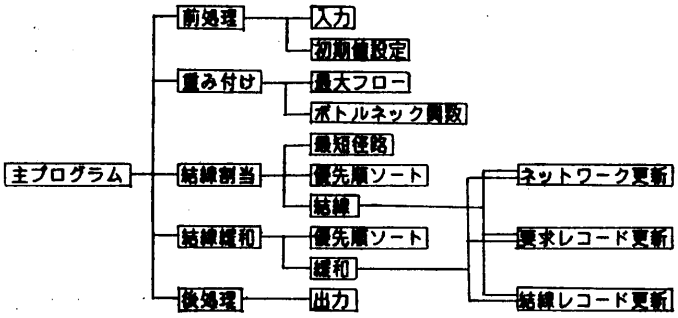


図 6 プログラムの機能樹
Fig. 6 Hierarchy of program units.

```
program MCA;
  begin
    i: =0; pcn: =0; ccn: =0; itr: =0;
    repeat
      SW;
      cn: =⌈α1*R⌉; {R: 初期値は総要求本数}
      NPFC (cn, rn, r); R: =R−rn
      ccn: =ccn+rn; pcn: =ccn
    until cn>rn or R<=0;
    while i<n and R>0 do begin
      i: =i+1; rc: =⌈β*ccn⌉;
      RRX (rc, rr, δ);
      R: =R+rr; ccn: =ccn−rr;
      repeat
        MFW (1);
        cn: =⌈α2*R⌉;
        NPFC (cn, rn, r);
        R: =R−rn; ccn: =ccn+rn
      until cn>rn or R<=0;
      if pcn<ccn then begin
        pcn: =ccn; itr: =i end
    end;
    itr 回目の再結線を解とする
  end.
```

図 7 アルゴリズム MCA
Fig. 7 Algorithm MCA.

ないが高速な方法で初期結線を作り上げ，続いて高精度な方法で緩和－再結線を繰り返し結線実現率の向上を図るものである．逐次割当ての繰返し回数は，入力パラメータ α により左右されるので，初期結線用と再結線用を区別して α_1, α_2 とし，具体的なアルゴリズム MCA (Multi-Connection Algorithm) を図 7 に示す．

MCA および各サブアルゴリズムのプログラム化は，FORTRAN 77 を用いて HITAC M 200 H 上でおこなった．MCA の大きさは，ライン数が約 1,500 行，実行形式で 171 kB である．必要とする変数領域をネットワークの大きさで表せば，$13|V|+13|E|+7q+(l+6)R$ となっている．ただし l は平均結線長とする．

5. 計算実験

計算実験は，以下に示す系統的生成が容易な 3 種類のネットワークに対し，計算速度と精度の特性を調べたものである．

G型) 計算時間の計測用

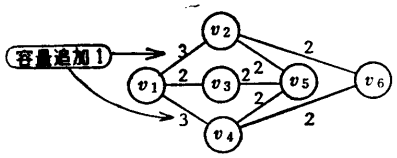
$m\times n$ 無向格子グラフ上の任意の節点対間に枝（ジャンパ枝と呼ぶ）を，0 または 50% の割合で付加する．q, R をパラメータとし，q 個のソース・シンク対の選択と総結線要求 R の分配をランダムにおこなう．枝容量は1～5 をランダムに割り付ける．

A型) 精度の計測用 I

まずジャンパ枝付無向格子グラフ G を用意し，q 個のソース・シンク対 (s_i, t_i) をランダムに選択する．次に s_i-t_i 間で r_i 本の結線を選択し，各枝が結線に使用された回数を G の枝容量とする．結線の選択方法としては，枝の長さをランダムに与えたときの最短径路を採用することもある．A型ネットワークでは，その作り方より要求が 100% 実現可能である．

H型) 精度の計測用 II

Hu の 3 種フロー (f_1, f_2, f_3) 問題で使用された反例ネットワークに，容量 2 の追加をおこなえば要求は実現可能であること，2 は最小の容量追加であることが知られている[8]．図 8 に示す反例に容量 2 を追加したネットワーク H (以後 Hu-ネットワークと呼ぶ) は，結線の相互干渉が最も厳しいもののひとつである．H型ネットワークの生成法として，まずジャンパ枝付無向格子ネットワーク G と複数のネットワーク H を別々に用意する．次に，H の各節点を G の節点とランダムに対応づけた後，H の枝を G 上の適当な道に対応させ枝容量と要求を G 上に移し替える．H型ネットワーク上の結線要求は生成法より 100% 実現



	ソース	シンク	要求
f_1	v_1	v_5	2
f_2	v_2	v_4	1
f_3	v_3	v_6	4

図 8 ネットワーク H と結線要求
Fig. 8 Network H and multi-connection requirements.

表 1 G型ネットワークの計算結果
Table 1 Computational experiments for G-networks.

50% ジャンパ付無向格子ネットワーク						
ネットワーク	節点数	枝 数	s-t 対 数	総要求	実現結線	CPU 時間(秒)
G-1	96	258	30	150	140	4.7
G-2	187	519	60	500	352	14.8
G-3	384	1,092	150	600	440	69.5
G-4	600	1,726	180	850	629	160.5
G-5	782	2,260	220	1,000	765	276.2

表 2 A型，H型ネットワークの計算結果
Table 2 Computational experiments for A- and H-networks.

ネットワーク	節点数	枝 数	s-t 対数	総要求	実現結線	復元率(%)	CPU 時間 (秒)
H-1	100	210	120	160	158	98.7	10.8
H-2	210	440	118	180	178	98.8	18.7
H-3	400	779	148	230	226	98.2	46.0
H-4	600	1,231	173	270	270	100.0	24.1
H-5	812	1,615	195	310	308	99.3	120.2
H-1	179	293	42	98	97	98.9	5.3
H-2	181	263	27	63	63	100.0	1.6
H-3	189	287	33	77	74	96.1	4.2
H-4	196	328	51	121	117	96.6	7.0
H-5	203	330	45	107	106	99.0	6.9

可能である．

以上のサンプルネットワークに対する MCA の計算結果を**表 1, 2** にまとめる．制御用パラメータは，$\alpha_1=1.0$, $\alpha_2=0.1$, $\beta=0.3$, $\gamma=\delta=0.5$, $n=5$ と設定している．

計算実験の主要結果は，次のようにまとめられる．

（1） ネットワークの規模と計算時間

MCA の計算量は，初期結線 $O(ST\cdot q\log R)$，再結線 $O((MF+ST)q\log R)$ である．ただし，$\alpha_1=1.0$ とすると初期結線は $O(ST\cdot q)$ となり MCA 全体の計算時間は緩和－再結線に支配される．G 型ネットワークに対する $\mathcal{T}\triangleq T/(q\log R)$, ($T$: MCA の全計算時間) と節点数の関係を**図 9** に示す．グラフの勾配より，$\mathcal{T}$ はジャンパ枝の割合にかかわらず節点数の約 1.2 乗に比例することがわかる．

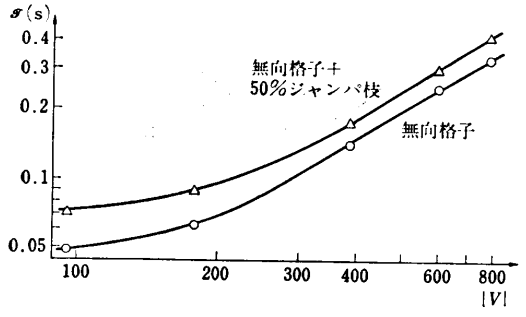


図 9 G型ネットワークの CPU 時間 $\mathcal{T}=T/(q\log R)$
Fig. 9 CPU-time $\mathcal{T}=T/(q\log R)$ for G-networks.

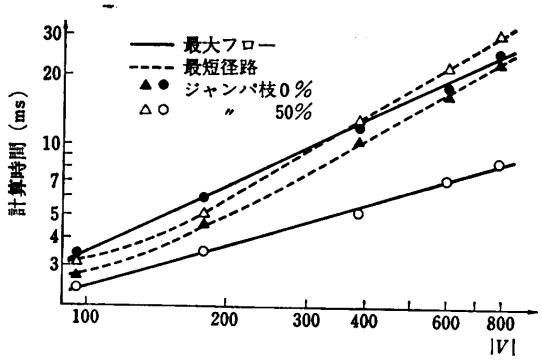


図 10 最大フローおよび最短径路のサブルーチン呼出し1回当りの平均計算時間
Fig. 10 Mean CPU-time per subroutine call of maximum-flow and shortest-path.

表 3 最大フロー，最短径路の呼出回数と計算時間
Table 3 Computation time for maximum-flow and shortest-path and total number of their routine calls.

ネットワーク	最大フロー		最短径路	
	呼出回数	全 CPU 時間(秒)	呼出回数	全 CPU 時間(秒)
G-1	790	1.9	616	2.0
G-2	1,227	4.3	1,332	6.8
G-3	2,238	11.6	3,215	41.0
G-4	2,943	21.3	4,633	103.6
G-5	3,698	32.3	5,944	183.9

表 3 は，G 型ネットワークに対し MCA が最大フローおよび最短径路ルーチンを呼び出した回数とその全所要時間をまとめたものである．**図 10** は呼出し1回あたりの計算時間と節点数の関係を示したものである．使用したアルゴリズムは，最大フローが Dinic 法，最短径路が Heap により一時ラベルを管理した Dijkstra 法である．**表 3** および図 10 は，MCA の全計算時間が最短径路の計算により支配されていることを示している．最大フロー計算に要する時間が意外と少ないのは，最小カットがソースまたはシンクに近い位置に存在し，補助ネットワークを構成する回数が小さくおさえられるためと思われる．

（2） 緩和－再結線の反復回数

緩和－再結線の反復回数 n は，MCA の計算時間

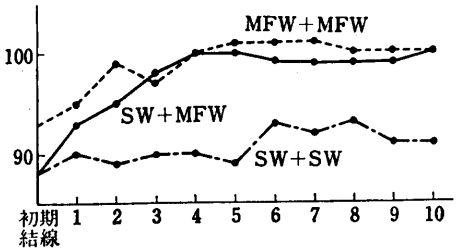


図 11 SW, MFW による実現結線本数の推移
Fig. 11 Change of realized connections by SW and MFW.

を直接左右する重要なパラメータである．有効な n の上限は，実現結線に改善がなくなる点により与えられる．図 11 の実線は，MCA における n と実現結線本数の推移を表す特徴的な例である．5〜6 回の緩和－再結線以後顕著な改善はなく，実現結線本数は震動する傾向をもつ．

（3） 結線復元率

A 型および H 型ネットワークは，生成法より全要求が 100% 実現可能である．H 型は A 型よりも高い難易度をもつが，それぞれの平均結線復元率は A 型 98%，H 型 96% を示し実用上十分高い精度が確保できた．ただし，計算時間は増加するが，α, γ, δ の各パラメータの値を小さく設定することにより精度の向上を図ることができる．

表 4 サブアルゴリズムの比較
Table 4 Comparison of subalgorithms.

ネットワーク	節点数	SW		MFW	
		実現結線	CPU 時間（秒）	実現結線	CPU 時間（秒）
G-6	96	89	2.1	92	3.4
G-7	187	201	17.5	226	23.7
G-8	384	301	33.5	311	60.5
G-9	600	421	82.6	423	120.0
G-10	782	404	99.7	406	161.4
		RC		NPFC	
G-11	96	25	0.7	27	1.3
G-12	187	59	3.0	76	10.1
G-13	384	83	8.0	114	29.4
G-14	600	96	16.3	140	64.8
G-15	782	116	28.0	164	124.5
		RRX		SRX	
G-16	96	49	2.4	50	2.5
G-17	187	83	9.5	83	14.4
G-18	384	105	29.8	108	42.6
G-19	600	140	64.8	139	95.8
G-20	782	149	115.9	152	186.0

（4） 枝重み生成法の比較

G 型ネットワークに対し，初期結線および再結線として，ⅰ)ともに MFW（破線），ⅱ)ともに SW（一点鎖線），ⅲ）それぞれ SW と MFW（実線）を利用した三つの場合の実現結線本数推移の一例を**図 11** に示す．再結線時に SW を使用すれば精度は低下するが，計算時間はⅲ)の場合の約 60% であった．初期結線のみを比べれば，MFW と SW の差は比較的小さく SW は計算速度（この場合 MFW の約 2.5 倍）の点で，よい初期結線を与えてくれる．**表 4** 上段にⅱ），ⅲ）の結果をまとめる．

（5） 結線割当て法の比較

RC と NPFC を同じ G 型ネットワークに適用した結果を表 4 中段に示す．RC は，非常に高速であるが精度的にはきわめて悪い．

（6） 結線緩和法の比較

RRX と SRX を G 型ネットワークに適用した一例を表 4 下段に示す．結線緩和自体に要する計算時間は，枝重み生成や結線割当に比べて非常に小さい．SRX は精度的に優れているが，MCA 全体の計算時間を RRX 使用時の約 1.5 倍に増加させる．結線が混み合う部分の集中的緩和が，再結線時に最短径路を求める回数を増加させるためと思われる．また，SRX は比較的早い時期に結線本数の震動があり以後安定する．これは，同一径路の緩和－再結線が反復されるためであろう．

6. むすび

多種結線実現アルゴリズム MCA とその変形の特性を調べた．MCA の計算時間はジャンパ枝の割合にかかわらず $|V|^{1.2}q \log R$ に比例している．最大フローの計算時間はジャンパ枝の付加により大きく変化するが，最短径路の計算時間は比較的安定している．文献 3) では，種々の最大フローおよび最短径路アルゴリズムに対し，実用的見地に立った実験的な評価がおこなわれている．MCA の最大フローおよび最短径路部分を独立させたプログラムの計算時間は，格子ネットワークの場合，文献 3) の結果と非常によく一致する．しかし，これらを MCA の環境に組み込んだ場合（図 10）は，計算速度が約 1/5 に低下する．原因として，プログラムの規模および扱うデータの数が大きくなったことにより，ページングその他のオーバヘッドが増大することが考えられる．

精度は，入力パラメータがネットワークのトポロ

ジ等が複雑に影響し合い多様な変化を示す．MCA は，$\alpha_1=1.0$, $\gamma=\delta=0.5$ という比較的厳しい条件の下でも，平均復元率 96% を示し実用的に十分なレベルに達している．

今後の課題を次にまとめる．

（1） 各入力パラメータと計算速度，精度の関係を明確にする．とくに精度の計測を，Hu ネットワーク以外にも難易度の高いものを利用し精密化する．

（2） VLSI，通信・交通網の分野で幅広い応用を可能とするために，節点容量，最小コスト多種結線を扱えるようにする．

（3） 多種結線問題を扱う汎用システムを作り上げる．

開発したプログラムは入手可能である．希望者は，

〒724 東広島市西条町下見

広島大学工学部第2類システム工学基礎論研究室

まで連絡されたい．

参 考 文 献

1) Ford, L.R. and Fulkerson, D.R.: *Flow in Networks*, Princeton Univ. Press, Princeton, N.J. (1962).
2) Hu, T.C.: Multicommodity Network Flows, *Oper. Res.*, Vol. 11, No. 3, pp. 344-360 (1963).
3) 今井 浩：ネットワーク理論の現状，第3回数理計画シンポジウム論文集，pp. 31-44 (1982).
4) Kennington, J. L.: A Survey of Linear Cost Multicommodity Network Flows, *Oper. Res.*, Vol. 26, No. 2, pp. 209-236 (1978).
5) 松本，西関，斉藤：平面グラフの多種フローを求める多項式時間アルゴリズム，信学論誌，Vol. J65-A, No. 12, pp. 1205-1212 (1982).
6) 森口，伊理，長谷：多種流輸送問題に対する一つの逐次解法，1970年 OR 学会秋期研究発表会論文集 1-1-7, pp. 21-22 (1970).
7) 翁長，有本，岸本：ネットワークにおける多種結線実現問題とその近似算法，信学論誌，Vol. J65-A, No. 5, pp. 431-438 (1982).
8) 翁長健治：多重フロー定理，信学論誌，Vol. 53-A, No. 7, pp. 350-356 (1970).
9) Sakarovitch, M.: The Multicommodity Flow Problem, Doctoral Thesis, Oper. Res. Center, University of California, Berkeley (1966).
10) Sleator, D. D.: An $O(mn\log n)$ Algorithm for Maximum Network Flow, Ph. D. Thesis, Computer Science Department, Stanford University, Stanford, California (1980).
11) Tang, D. T.: Bi-Path Networks and Multicommodity Flows, *IEEE Trans. Circuit Theory*, Vol. CT-11, No. 4, pp. 468-474 (1964).

(昭和58年7月21日受付)
(昭和58年9月13日採録)