解　説

最大流アルゴリズムの最近の発展とその背景—I†

岩　野　和　生††

1. はじめに

最近のネットワーク・フロー問題のアルゴリズムの進展は目覚ましいものがある．それらのなかには，昨年8月に中央大学で開かれたMathematical Programming Symposium で Fulkerson 賞を受賞した E. Tardos の最小費用流問題の強多項式時間アルゴリズム[Tard85]，Karmarkar の線形計画法における Karmarkar法と呼ばれる多項式時間アルゴリズム[Karm84a, Karm84b] や Tucker 賞を受賞した A. V. Goldberg の最大流問題と最小費用流問題のプリフロー・プッシュのアルゴリズム[Gol85a, Gol85b] などの重要な仕事が含まれる．

これらの最近のアルゴリズムの進展の背景には，古典的なアルゴリズムで使われた着想・技術（プリフロー，スケーリング法，データ構造など）と新しい着想・技術（距離ラベル，ε—最適性，ポテンシャル，アルゴリズムの不変性など）の巧妙な融合がみられる．この解説では，これらの着想・技術を説明し，それらがどのように融合されて新たなアルゴリズムが作られてきたのかを最大流問題についてみていく．

2. 古典的アルゴリズムにみられる着想と技術

この章では，ネットワーク・フロー問題のうち，代表的な問題の一つである最大流問題を定式化する．そして，最近のアルゴリズムに関係の深い古典的なアルゴリズムの着想と技術についてみていく．

2.1 ネットワークに関する諸定義

まず，ネットワーク $N=(G, s, t, u)$ を次のように定義する．$G=[V, E]$ は，V を点集合，E を辺集合とする有向グラフで，$|V|=n$，$|E|=m$ とし，$m \geqq n-1$ と仮定する．s と t は，V の特別な点で，おのおの始点 (source) と終点 (sink) と呼ばれる．容量関数 u とは，V の各点の順序対 $[v, w]$ に対して定義される非負の実数値関数である．$u(v, w)$ は，辺 $[v, w]$ を通過しうる流れの上限値である．E の辺でない点対 $[v, w]$ については，$u(v, w)=0$ と定義する．$U=\max\{u(v, w) | [v, w] \in V \times V\}$ とする．このとき，G のフロー（または，流）f は，V の各点対に対して定義され，次の三つの性質をもつ実数値関数である．

（1） 歪対称性．各点対 $[v, w]$ について，$f(v, w)=-f(w, v)$ である．

（2） 容量条件．各点対 $[v, w]$ について，$f(v, w) \leqq u(v, w)$ である．

（3） フロー保存則．始点 s と終点 t 以外のすべての点 v において，$\sum_w f(u, v)=0$ である．

フロー f の流量 $|f|$ を始点 s からの流出量，つまり $\sum_v f(s, v)$ で定義する．

一般には，（1）の歪対称性を必ずしも仮定せず，（3）のフロー保存則を $\sum_w f(u, v)=\sum_w f(v, u)$，つまり，$v$ に入ってくる流量と v から出ていく流量が等しいと表現したりする．しかし，歪対称性を導入しても本質は変わらず，むしろ下記に定義される残余容量の取り扱いなどが簡単になる．

（3）のフロー保存則をゆるめて，次の（3′）を採用したものをプリフロー (preflow) と呼ぶ．

（3′） フロー残存則．始点 s と終点 t 以外のすべての点 v において，$\sum_w f(u, v) \geqq 0$ である．

いま，$e(v)=\sum_w f(u, v)$ と表し，$e(v)$ を v におけるプリフロー f の残存量 (excess) と呼ぶ．定義よりフローは，プリフローである．プリフロー f が与えられたとき，点対 $[v, w]$ の残余容量 (residual capacity) とは，$u_f(v, w)=u(v, w)-f(v, w)$ であり，$u_f(v, w) \geqq 0$ である．このとき，$f(v, w)=u(v, w)$ なる点対 $[v, w]$ は，f によって飽和されている (saturated) という．また，f に関する G の残余グラフ (residual

† Recent Developments of Maximum Flow Algorithms and Their Background—I by Kazuo IWANO (Tokyo Research Laboratory, IBM Japan Ltd.).

†† 日本アイ・ビー・エム(株)東京基礎研究所

graph) $G_f=(V, E_f)$ を $E_f=\{[v, w] | u_f(v, w)>0\}$ と定義する.

始点 s から出るすべての辺を飽和させ，それ以外の辺での流量が0であるようなプリフローを，この解説では特別に始点 s に関するプリフローとよび f_{source} と書き表すことにする．この始点 s に関するプリフロー f_{source} は，2.5 の Karzanov のアルゴリズムや 3.(次号) の Goldberg と Tarjan のプリフロー・プッシュのアルゴリズムで最初のプリフローとして使われる．f_{source} について残存量が正である点 v は，辺 $[s, v]$ が G にあるような点のみである．図-1 はプリフロー f_{source} とその残余グラフを示したものである.

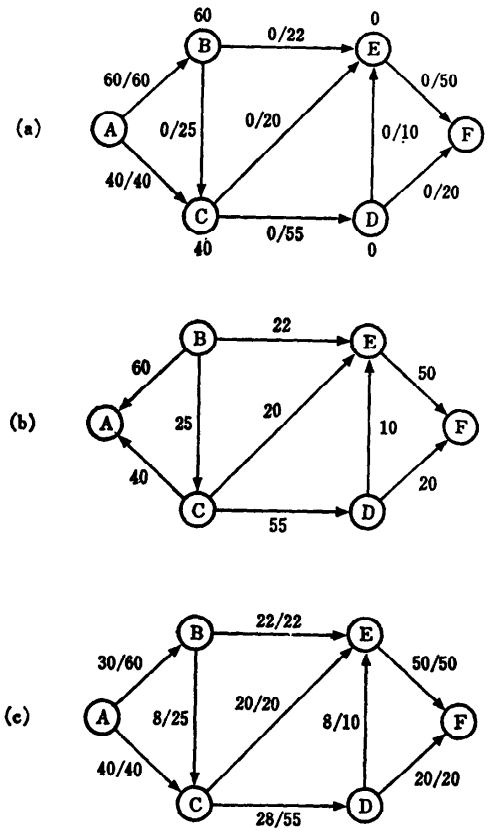

図-1　(a)ネットワーク N，始点 s に関するプリフロー f_{source}．始点は A，終点は F である．各辺 $[u, v]$ の分数の分母はその辺の容量 $u(v, w)$ を，分子は f_{source} の流量 $f_{source}(v, w)$ を表す．また，点 B, C, D, E に示されている数字は各点における残存量を示す．(b) f_{source} に対する残余グラフ G_f．各辺の数字は容量を示す．(c)ネットワーク N における最大流 f^*．f^* の流量は 70 である.

G_f における始点 s から終点 t までの道を，f に対するフロー増加可能パス（または，略して，増加可能パス，増加パスと呼ぶ.）(augmenting path) という．f に対するフロー増加可能パス p の辺の残余容量の最小値を Δp とし，p の f に関する残余容量と呼ぶ．δ を $\delta \leqq \Delta p$ なる非負の数とするとき，p の各辺 $[v, w]$ の流量を δ だけ増やして $f(v, w)+\delta$ にして，辺 $[w, v]$ の流量を δ だけ減らして $f(w, v)-\delta$ とすることを，p に沿ってフローを δ だけ増加するという.

ネットワーク N における点集合の二つの部分集合への分割 $X, \bar{X}=V-X$ で，$s \in X, t \in \bar{X}$ であるようなものを，カット (cut) と呼び，カットの容量を $u(X, \bar{X})=\sum_{v \in X, w \in \bar{X}} u(v, w)$ で定義する．プリフロー f に対して，カット $(X, \bar{X})$ を横切る流量 $f(X, \bar{X})$ を $\sum_{v \in X, w \in \bar{X}} f(v, w)$ で定義する．このとき，次の定理が成り立つ [参照 IriFujOhya86, Tarj83 など].

定理 2.1.　任意のフロー f と，任意のカット $(X, \bar{X})$ について，$f(X, \bar{X}) \leqq u(X, \bar{X})$ である.

2.2 最大流問題

最大流問題 (the maximum flow problem) とは，ネットワーク $N=(G, s, t, u)$ が与えられたとき，始点 s から終点 t までの最大の流量をもつフロー（最大流と呼ぶ）をみつける問題である.

図-1(c)は，ネットワーク N における最大流を示している．また，図-2 は，最大流を求めるアルゴリズムの歴史を示したものである.

図-2 にみられるように，最大流問題は 1956 年に Ford と Fulkerson が“ラベリング法”によるアルゴリズムを示して[ForFul56,62] 以来，活発に研究されてきた．1982 年以前のアルゴリズムについての詳しい説明は，[Imai86, Iri69, IriFujOhya86, IriKo76, Law76, PaSt82, Tarj83, Tarj84, Tarj86] を見られたい．次の諸定理は最大流アルゴリズムの基本である.

定理 2.2.　フロー f は，その残余グラフ G_f がフロー増加可能パス（すなわち，始点 s から終点 t への有向道）を含まないとき，そして，そのときに限り，最大流である.

定理 2.3 (最大流・最小カット定理).　フロー f は，あるカット $(X, \bar{X})$ があり，$f(X, \bar{X})=u(X, \bar{X})$ であるとき，そして，そのときに限り，最大流である.

定理 2.4 (整数定理).　容量関数が整数値を取るならば，すべての辺の上での流量が整数であるような最大流が存在する.

By	Complexity
Ford and Fulkerson (1956)	$O(nmU)$
Edmonds and Karp (1969)	$O(nm^2)$
Dinic (1970)	$O(n^2m)$
Karzanov (1974)	$O(n^3)$
Cherkasky (1977)	$O(n^2m^{1/2})$
Malhotra, Kumar and Maheshwari (1978)	$O(n^3)$
Galil (1978)	$O(n^{5/3}m^{2/3})$
Galil and Naamad (1978) ; Shiloach (1978)	$O(nm\log^2 n)$
Sleator and Tarjan (1980)	$O(nm\log n)$
Shiloach and Vishkin (1982)	$O(n^3)$
Gabow (1983)	$O(nm\log U)$*
Tarjan (1984)	$O(n^3)$
Goldberg (1985)	$O(n^3)$
Goldberg and Tarjan (1986)	$O(nm\log(n^2/m))$
Cheriyan and Maheshwari (1987)	$O(n^2m^{1/2})$
Ahuja and Orlin (1987)	$O(nm+n^2\log U)$*
Ahuja, Orlin and Tarjan (1987)	$O\left(nm\log\left(\frac{n\log U}{m\log\log U}+2\right)\right)$*

図-2 最大流アルゴリズムの歴史
ネットワークを $N=(G=(V,E),s,t,u)$ とするとき，$n=|V|, m=|E|, U=\max\{u(v,w)\}$ である．* は整数容量を仮定している．

2.3 Ford と Fulkerson, Edmonds と Karp のアルゴリズム

定理 2.2 の帰結として，次のようなアルゴリズムが考えられる．すなわち，すべての辺で流量が 0 のフロー（零フローと呼ぶ）から始めて，フロー増加可能パスを見つけ，この道に沿ってフローの流量を増やす．求められたフローの残余グラフを考えて，上記のステップを最大流が得られるまで繰り返すのである．

ところが，このアルゴリズムには，次の二つの任意性がある．ここでアルゴリズムの任意性とは，アルゴリズム設計上の決定事項で取りうる選択に余地があるとき，その選択における任意性を指す．アルゴリズムにおける自由度をもった性質ともいえる．

『流量の任意性』求められた増加パスに沿ってフローを増加する際の増加流量についての任意性．通常は，増加パスの残余容量 Δp を取る．

『パス選出の任意性』いくつかの増加パスの選択があるときにそのなかから一つ選ぶ際の任意性．この任意性には『パスの残余容量』に関するものと『パスの長さ』に関するものとの二通りが考えられる．たとえば，『パスの残余容量』については，できるだけ大きな残余容量をもつパスを選ぶなどである．

もし，すべての容量 $u(v,w)$ が整数であるとき，どちらの任意性も任意にするとしよう．すると，一つの増加パスごとに少なくとも一だけフローを増やすことができるから，高々 $|f^*|$ 回の増加パスを求めればよいので，全体の時間は，$O(m|f^*|)$ となる．ここで f^* は，最大流である．ところが，容量が無理数であれば，順次求まるフローの流量が，収束しないかもしれないし，収束しても最大流の流量とは異なる値に収束する場合がある[Zad73a,Zad73b]．

『パス選出の任意性（残余容量）』を“最大の残余容量をもつ増加パスに沿ってその残余容量を増加する（最大流量増加）”ことにすれば，順次求まるフローの流量が収束する[Que80]．さらに，容量関数が整数値を取る場合，$O(m\log U)$ 回のフローの増加によって最大流をみつけることができる[Tarj83]．また，各増加パスをみつけるのに，ボトルネック最短路問題を解けばよく，これは，$O(m\log_{(2+m/n)}n)$ 時間かかり，全体で $O(m^2\log_{(2+m/n)}n\log U)$ かかる[Tarj83]．有向グラフ G の各辺 $[v,w]$ に実数値のコスト $\mathrm{cost}(v,w)$ が定義され，G に特別な 2 点 s と t があるとする．このとき，ボトルネック最短路問題とは，s から t までの最短路のうち，その道上の最大の辺のコストが最小であるものを求める問題である．

『最大流量増加』にくわえて，さらに『パス選出の任意性（パスの長さ）』を“始点 s から終点 t までの辺の数に関する最短路を取る（最短路増加）”ことにすると，全体で高々 $(n-1)m$ 個の増加パスをみつけることになり，全体で $O(nm^2)$ 時間かかる[EdmKar72]．図-3 は『最大流量増加』と『最短路増加』を用いて最大流を求めている．

2.4 レベルグラフと極大流 (Dinic)

前節でみたように，一つの着想に対していくつかの任意性がみつけられ，それらの任意性の解消法の組み合わせによって，より有用なアルゴリズムがつくられてきた．

このことは，Dinic による着想（レベルグラフと極大流）についてもいえる．いまフロー f に対して f の残余グラフ G_f の部分グラフで f に関するレベルグラフ (level graph) と呼ばれるもの $L_f=(V_{1f},E_{1f})$ を，次のように定義する．V_{1f} は G_f において s から到達可能な点の集合からなり，各点 v について，level (v) を点 s から v への辺の長さに関する最短路の長さとする．このとき，

$$E_{1f}=\{[v,w]\in E_f \mid \mathrm{level}(w)=\mathrm{level}(v)+1\}$$

である．図-4 はレベルグラフの例である．レベルグ

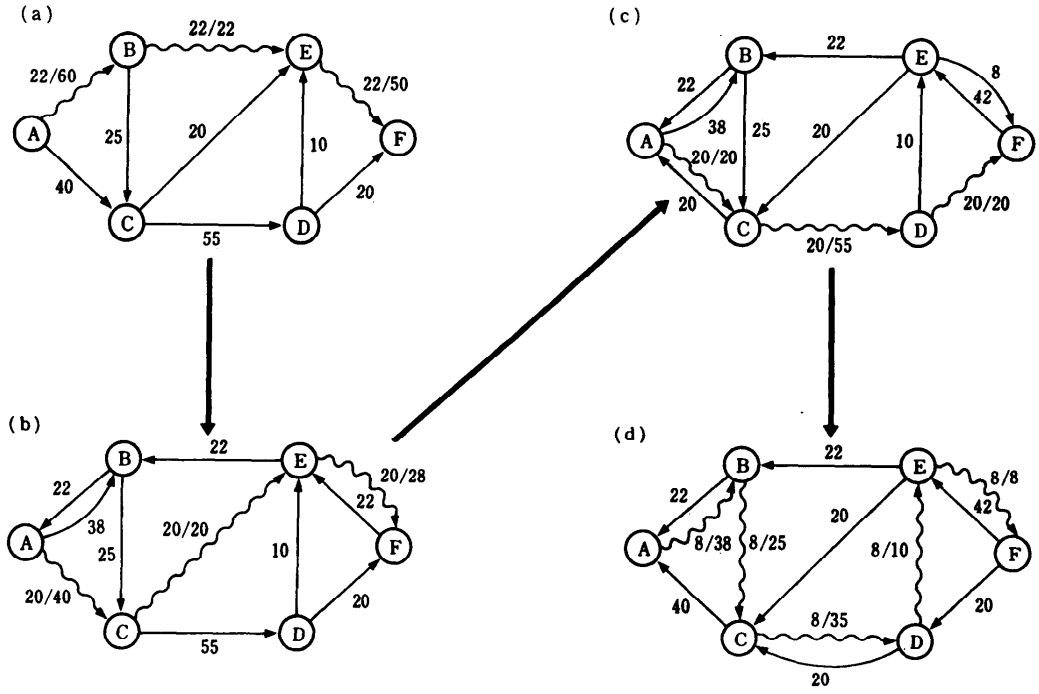

図-3 『最大流量増加』と『最短路増加』を用いた最大流アルゴリズム．各図で波線の施された辺はフロー増加可能パスに含まれる．
(a), (b), (c)で，最短路の長さのものを残余容量の大きい順にみつけ，(ABEF, ACEF, ACDF)，最後に(d)で，長さが5のもの (ABCDEF) をみつけている．

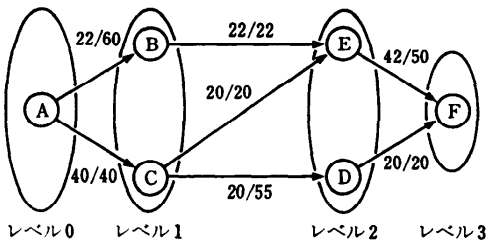

図-4 レベルグラフと極大流

ラフは，層構造をもっており，別名，層別ネットワーク (layered network) とも呼ばれる．レベルグラフ L_f の一つの特長は，残余グラフ G_f での始点 s から終点 t までのすべての最短路は L_f に含まれることである．レベルグラフは，幅優先探索を用いて $O(m)$ 時間で求められる．ネットワーク $N=(G, s, t, u)$ のフロー f は，始点 s から終点 t までのどの有向道に沿っても（残余グラフ G_f ではなく，もとのグラフ G で），もうこれ以上流量を増やすことができないとき，極大流 (maximal flow, blocking flow) と呼ばれる．

Dinic のアルゴリズムは，零フローから始めて，現在までに求まったフローを f とするとき，f に関するレベルグラフ L_f で極大流 f' をもとめ，f を $f+f'$ で置き換えて，最大流が求められるまで，このステップを繰り返していく．このステップを1回繰り返すたびに level(t) は少なくとも1増加し，高々 $(n-1)$ 回のステップの後，可能な長さのすべての増加パスは調べられることになり，最大流が求められる[Tarj83]．

Dinic のアルゴリズムの改良は各レベルグラフにおいて極大流を求める際の改良によってなされてきた．レベルグラフは，非巡回的 (acyclic) なので，問題は非巡回グラフで極大流を求める問題になる．

やはり，2.3 と同じように，『パス選出の任意性（残余容量）』があり，『最大流量増加』を用いれば，極大流は，$O(n)$ 時間かかる深さ優先探索を高々 m 回行えば求められるので，全体で $O(n^2m)$ 時間かかる[Di70]．

Dinic のアルゴリズムは，全体でみると，始点 s から終点 t への道の長さの短い順にフローの増加を行っているので，『パス選出の任意性（パスの長さ）』は，『最短路増加』が暗黙のうちに採用されている．こ

こで，『最大流量増加』と『最短路増加』という同じ任意性の解消を図っているのに，なぜ Edmonds と Karp のアルゴリズムは $O(nm^2)$ かかり，Dinic のアルゴリズムは $O(n^2m)$ ですむのかという疑問が生ずる．それは，Edmonds と Karp のアルゴリズムは，事前に知り得た情報をまったく用いずに各最短路を毎回 $O(m)$ 時間かけて求めているためである．一方，Dinic では，対象となるグラフが，あらかじめすべての最短路を含む非巡回グラフに構造化されているために，各最短路が $O(n)$ 時間で求められることによる．ここにもう一つの任意性が考えられる．

『共有情報の任意性』各段階で共有できる情報をなんらかの形で貯える．

この貯え方には，Dinic のようにグラフの構造に貯えたり，なんらかのデータ構造（たとえば，2-3木，二色木 (red-black tree)[GuibSed78,Sed88]，動的木 (dynamic tree)[SlTarj83,Tarj83] を使用することなどが考えられる．

これまでの方法は，基本的には，“増加パスをみつけ，この道に沿ってフローの流量を増やし，新たなフローに対する残余グラフを求める”というステップを繰り返すものである．この過程で一度みつけられ使われた増加パスの情報は，すべて捨て去られて，それ以降に使われることはない．ところが，増加パスから飽和された辺を取り除くといくつかの道の集合ができるが，これらは他の増加パスの部分として使われうるかもしれない．この意味での『共有情報の任意性』に着目したのが，Galil と Naamad[GalNaa80] や Shiloach[Si78] である．かれらは，これらの断片的な道を 2-3 木などに覚えこませて，極大流が $O(m(\log n)^2)$ 時間で求められることを示した．さらに，Sleator と Tarjan は，極大流が，動的木の操作 (maketree, findroot, findcost, addcost, link, cut) を高々 $O(m)$ 回使用することで求められることを示した[Sl80,SlTarj83]．ここで，動的木の操作を k 個取り混ぜて実行するのに要する時間は全体で $O(k\log n)$ である．これは，各木の操作がある意味で，平均して $O(\log n)$ 時間かかることを示している．Tarjan は，これを各操作が $O(\log n)$ の償却時間 (amortized time) をもつと表現した[Tarj85]．したがって，動的木を使用すると $O(m\log n)$ の償却時間で極大流が求められ，最大流が $O(nm\log n)$ 時間で求められることになる．

2.5 プリフロー (Karzanov)

最大流には，(1)フローであることと，(2)もうその残余グラフで増加パスが存在しないという意味で“極大”であるという二つの特性がある．いままでのアルゴリズムは，(1)を保ちながら(2)を実現するというものであった．すると，逆に，(2)を保ちながら(1)を実現するというアルゴリズムが考えられる．この着想の出発点となったのが，Karzanov のアルゴリズムである．Karzanov のアルゴリズムの詳細は，[Gal80, Imai86, Karz74, Tarj84] などを見られたい．ここでは，最近のアルゴリズムの発展に関連のある Karzanov のアルゴリズムの二つの特長について述べよう．

このアルゴリズムは，現在まで求められているフローを f とするとき，f のレベルグラフ L_f 上で次の push と balance の二つの基本操作を施しながら極大流を求める．活動点 v（すなわち，残存量が正である）に適用される push(v) は，v から出る辺に可能なかぎりフローを押し出し，v から出るすべての辺が飽和するか，v の残存量が 0 になるまで続ける．活動点 v に適用される balance(v) は残存量が 0 でない点に適用され，v に入ってくる辺の流量を減らすことによって，v の残存量を 0 にする．そして，v と v の隣接辺をレベルグラフから削除する．

このとき Karzanov のアルゴリズムは，次のようになる．最初のプリフローを始点 s に関するプリフロー f_{source}（参照 2.1）とする．まず，始点 s に近い点から順に，push(v) を用いて，可能なかぎりフローを終点 t のほうへ押し出していく．次にある点 v に残存量があるが，もうどの道に沿ってもフローを t に押し出せないとき，そのような v のうちもっとも終点 t に近い点 x に対して，balance(x) によって，x に残存しているフローを x の一つ手前の点に戻し，再び上記の push と balance のステップを繰り返す．このようにして，プリフローを最終的にフロー（すべての点の残存量が 0 であるプリフロー）に変えて，最大流を求める．

最初は，s から出る辺はすべて飽和しているので，上記特性(2)を満たした状態にある．始点 s にフローが戻るのは，操作 balance(v) によってのみであり，この操作が v に施されるときには，残余グラフにおいて v から t への道はない．そのため，このアルゴリズムの実行中，特性(2)は保たれる．そして，最終的にフローが求められ，つまり，特性(1)が満たされ最大流を得る．

このように特性(2)を保ちながら，プリフローをフローに変えていくという方針は，最近のアルゴリズム

でも用いられている[AhuOrl87, AOT87, Gol85a, Gol85b, GolTarj86, OrlAhu88, Si78, SiVi82, Tarj84].

もう一つの特長は，このアルゴリズムが push と balance という二つの局所的な操作によって構成されている点である．アルゴリズムの構成要素をいくつかの局所的な基本操作のみで表現することによって，データ構造に対する基本操作との対応付けが容易になり，また，並列化への視点も生まれてきた．これは，[Gol85a, Gol85b, GolTarj86, SiVi82] の特長でもある．

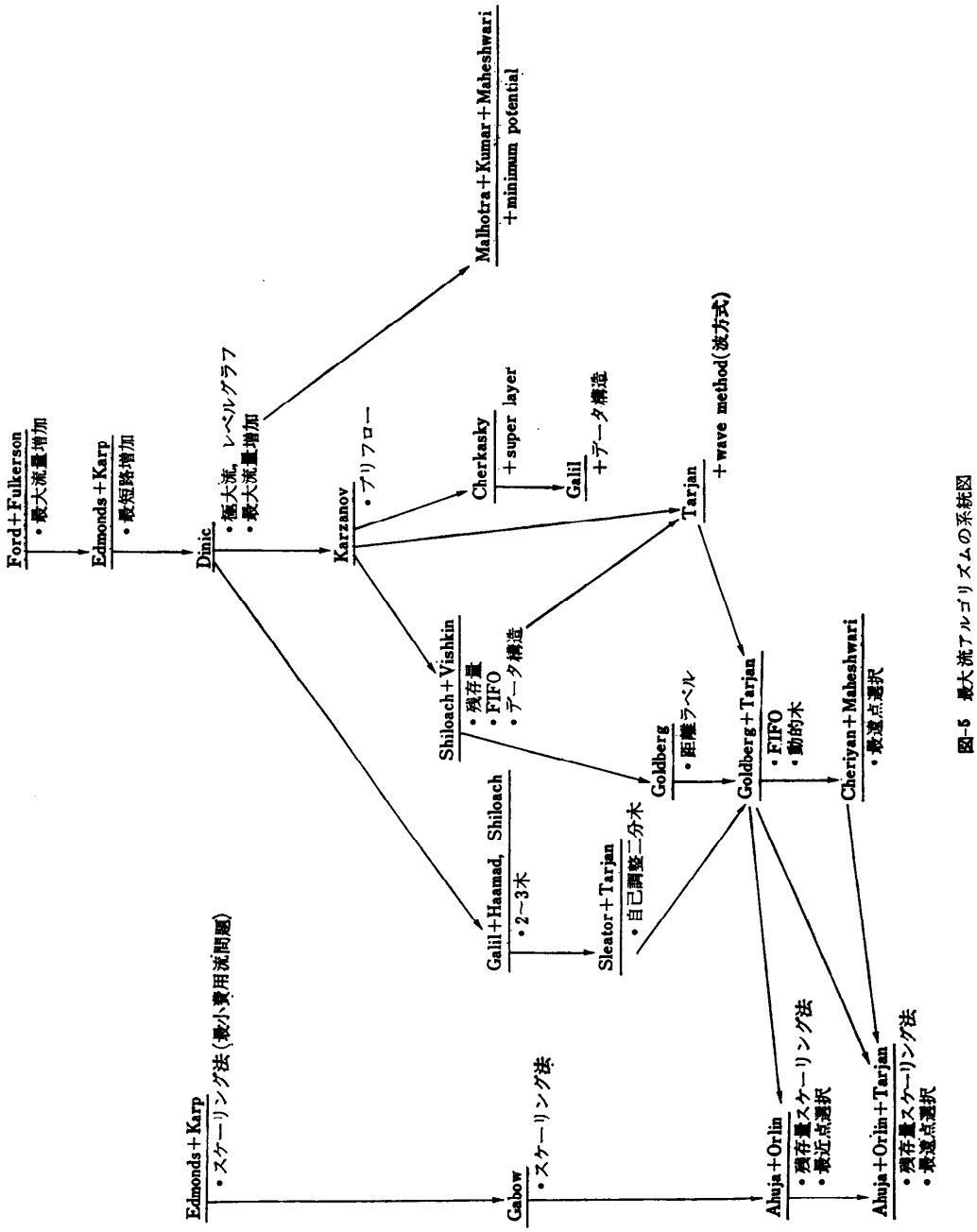

図-5　最大流アルゴリズムの系統図

2.6 スケーリング法 (Gabow)

最近のアルゴリズムの進展に関係のあるもう一つの重要な着想を紹介しておこう．これは，スケーリング法 (scaling method) と呼ばれるもので，Edmonds と Karp が，整数容量を仮定したときに最小費用流問題を解くのに用いたのが最初である[EdmKar72]．ここでは，Gabow による最大流問題へのスケーリング法の適用を紹介しよう．

まず，ネットワーク $N=(G, s, t, u)$ は，やはり整数値の容量をもつと仮定する．このとき，各辺 $[v, w]$ の容量 $u(v, w)$ を $u'(v, w)=\lfloor u(v, w)/2 \rfloor$ とほぼ半分にして，新しい容量をもつネットワーク $N'=(G, s, t, u')$ について，このスケーリング法を再帰的に適用して，最大流 f' を求める．次に求められたフロー f' の各辺の流量を2倍にしてネットワーク N でのフロー f を得る．このとき N における f の残余グラフでは，どの増加パスの残余容量も1である．したがって，f から Nの最大流を求めるのに，高々 m 回のフローの増加を行えばよい．Dinic のアルゴリズムを用いれば，m 回のフローの増加は $O(nm)$ 時間かかり，高々 $\lceil \log U+1 \rceil$ 回の再帰による繰り返しがあるので，全体で $O(nm \log U)$ 時間かかることになる[Gab85]．$U=O(m^k)$ のときは，Sleator と Tarjan のアルゴリズム[SlTarj83] に匹敵する．

スケーリング法の特長は，アルゴリズムが単純であることと，もう一つは，『問題を大体のところで解いて，あとで微調整を施す』という点である．

以上，最近の最大流問題のアルゴリズムの発展の基礎となる着想や任意性について触れてきた．図-5 は，これらの着想がどのように吸収されてアルゴリズムが発展してきたかを示したものである．次号では，Goldberg 以降のアルゴリズムについて，図-5 に基づいて説明していこう．

参 考 文 献

[AhuOrl87] Ahuja, R.K. and Orlin, J.B.: A Fast and Simple Algorithm for the Maximum Flow Problem, Technical Report 1905-87, Sloan School of Management, M.I.T. (1987).

[AOT87] Ahuja, R.K., Orlin, J.B. and Tarjan, R.E.: Improved Time Bounds for the Maximum Flow Problem, Sloan W.P. No. 1966-87, Sloan School of Management, M.I.T. (1987).

[Awe85] Awerbuch, B.: Reducing Complexities of the Distributed Max-flow and Breadthfirst-search Algorithms by Means of Network Synchronization, Networks, 15, pp. 425-437 (1985).

[BerEck87] Bertsekas, D.P. and Eckstein, J.: Distributed Asynchronous Relaxation Methods for Linear Network Flow Problems, Proc. IFAC '87, Munich, Germary (1987).

[CheMah88] Cheriyan, J. and Maheshwari, S.N.: Analysis of Preflow Push Algorithms for Maximum Network Flow, Proc. Eighth Conference on Foundations of Software Technology and Theoretical Computer Science, Lecture Notes in Computer Science 338, Edited by Nori, K.V. and Kumar, S., Springer-Verlag, pp. 30-48 (Dec. 1988).

[Che77] Cherkasky, R.V.: Algorithm of Construction of Maximal Flow in Networks with Complexity of $O(V^2\sqrt{E})$ operations (in Russian), Mathematical Methods of Solution of Economical Problems, 7, pp. 112-125 (1977).

[Di70] Dinic, E.A.: Algorithm for Solution of a Problem of Maximum Flow in a Network with Power Estimation, Soviet Mathematics Doklady, 11, pp. 1277-1280 (1970).

[EdmKar72] Edmonds, J. and Karp, R.M.: Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems, Journal of the ACM, 19, pp. 248-264 (1972).

[Eve79] Even, S.: Graph Algorithms, Computer Science Press, Potomac, MD. (1979).

[ForFul56] Ford, L.R. and Fulkerson, D.R.: Maximal Flow through a Network, Canad. J. Math. 8, pp. 399-404 (1956).

[ForFul62] Ford, L.R. and Fulkerson, D.R.: Flows in Networks, Princeton University Press, Princeton, NJ (1962).

[FreTarj84] Fredman, M.L. and Tarjan, R.E.: Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithms, Proc. 25th IEEE Symp. on Foundations of Computer Science, pp. 338-346 (1984); also in J. Assoc. Comput. Mach., 34, pp. 596-615 (1987).

[Gab85] Gabow, H.N.: Scaling Algorithms for Network Problems, J. Computer and Systems Sciences, 31, pp. 148-168 (1985).

[GabTarj87] Gabow, H.N. and Tarjan, R.E.: Faster Scaling Algorithms for Network Problems, Research Report, Computer Science Dept., Princeton University, NJ (1987).

[Gal80] Galil, Z.: An $O(V^{5/3}E^{2/3})$ Algorithm for the Maximal Flow Problem, Acta Informatica, 14, pp. 221-242 (1980).

[Gal81] Galil, Z.: On the Theoretical Efficiency of Various Network Flow Algorithms, Theoretical Computer Science, 14, pp. 103-111 (1981).

[GalNaa80] Galil, Z. and Naamad, A.: An $O(EV \log^2 V)$ algorithm for the Maximal Flow Problem, Journal of Computer and System Sciences, 21, pp. 203-217 (1980).

[Gol85a] Goldberg, A. V.: A New Max-flow Algorithm, Technical Report MIT/LCS/TM-291, Laboratory for Computer Science, M. I. T., Cambridge, Mass. (1985).

[Gol85b] Goldberg, A. V.: Efficient Graph Algorithms for Sequential and Parallel Computers, Ph. D. Dissertation, M. I. T., Cambridge, Mass. (Jan. 1987). Also availabe as Technical Report TR-374, Laboratory for Computer Science, M. I. T., Cambridge, Mass. (1987).

[GolTarj86] Goldberg, A. V. and Tarjan, R.E.: A New Approach to the Maximum Flow Problem, in proc. 18th ACM Symposium on Theory of Computing, pp. 136-146 (1986); Also in J. Assoc. Comput. Mach., Vol. 35, No. 4, pp. 921-940 (Oct. 1988).

[GuibSed86] Guibas, L. J. and Sedgewick, R.: A Dichromatic Framework for Balanced Trees, in Proc. 19th Annual IEEE Symposium on Foundations of Computer Science, pp. 8-21 (1978).

[Imai83] Imai, H.: On the Practical Efficiency of Various Maximum Flow Algorithms, J. Oper. Res. Japan, 26, pp. 61-83 (1983).

[Imai86] Imai, H.: Network Flow Problems, Chapter 5 in Computational Geometry and Geographical Inforamtion Processing, Edited by Iri, M. and Koshizuka, T., Kyoritsu Shuppan, Tokyo (1986).

[Iri69] Iri, M.: Network Flow, Transportation and Scheduling Theory and Algorithms, Academic Press, New York (1969).

[IriFujOhya86] Iri, M., Fujishige, S. and Ohyama, T.: Graph, Network, Matroid, Sangyou Tosyo (1986).

[IriKo76] Iri, M. and Kobayashi, T.: Network Theory, Nikka-Giren (1976).

[Karm84a] Karmarkar, N.: A New Polynomial-time Algorithm for Linear Programming, in Proc. 16th ACM Symposium on Theory of Computing, pp. 302-311 (1984).

[Karm84b] Karmarkar, N.: A New Polynomial-time Algorithm for Linear Programming, Combinatorica, Vol. 4, No. 4, pp. 373-395 (1984).

[Karz74] Karzanov, A. V.: Determining the Maximal Flow in a Network by the Method of Preflows, Soviet Mathematics Doklady, 15, pp. 434-437 (1974).

[MKM78] Malhotra, V. M., Kumar, M. P. and Maheshwari, S. N.: An $O(|V|^3)$ algorithm for Finding Maximum Flows in Networks, Information Processing Letters, 7, pp. 277-278 (1978).

[Law76] Lawler, E. L.: Combinatorial Optimization: Networks and Matroids, Holt, Rinehart, and Winston, New York (1976).

[PaSt82] Papadimitriou, C. H. and Steiglitz, K.: Combinatorial Optimization: Algorithms and Complexity, Prentice-Hall, Englewood Cliffs, NJ (1982).

[Que80] Queyranne, M.: Theoretical Efficiency of the Algorithm 'capacity' for the Maximum Flow Problem, Math. Oper. Res., 5, pp. 258-266 (1980).

[Sed88] Sedgewick, R.: Algorithms Second edition, Addison-Wesley (1988).

[Si78] Shiloach, Y.: An $O(nI \log^2 I)$ maximum-flow Algorithm, Technical Report STAN-CS-78-802, Computer Science Department, Stanford University, Stanford, CA (1978).

[SiVi82] Shiloach, Y. and Vishkin, U.: An $O(n^2 \log n)$ parallel Max-flow Algorithm, Journal of Algorithms, 3, pp. 128-146 (1982).

[Sl80] Sleator, D. D.: An $O(nm \log n)$ algorithm for Maximum Network Flow, Technical Report STAN-CS-80-831, Computer Science Department, Stanford University, Stanford, CA (1980).

[SlTarj83] Sleator, D. D. and Tarjan, R. E.: A Data Structure for Dynamic Tress, Journal of Computer and System Scinces, 26, pp. 362-391 (1983).

[Tard85] Tardos, E.: A Strongly Polynomial Minimum Cost Circulation Algorithm, Combinatolrica, 5, pp. 247-255 (1985).

[Tarj83] Tarjan, R. E.: Data Structures and Network Algorithms, Society for Industrial and Applied Mathematics, Philadelphia, PA (1983).

[Tarj84] Tarjan, R. E.: A Simple Version of Karzanov's Blocking Flow Algorithm, Operations Research Letters, 2, pp. 265-268 (1984).

[Tarj85] Tarjan, R. E.: Amortized Computational Complexity, SIAM J. Alg. Disc. Meth., 6, pp. 28-40 (1985).

[Tarj86] Tarjan, R. E.: Algorithms for Maximum Network Flow, Mathematical Programming Study, 26, pp. 1-11 (1986).

[OrlAhu88] Orlin, J. B. and Ahuja, R. K.: New Distance-directed Algorithms for Maximum Flow and Parametric Maximum Flow Problems, Tehnical Report OR 192-88, Operations Research Center, M. I. T. (Mar. 1988).

[Zad73a] Zadeh, N.: More Pathological Examples for Network Flow Problems, Math. Prog., Vol. 5, pp. 217-224 (1973).

[Zad73b] Zadeh, N.: A Bad Network Flow Problems for the Simplex and Other Minimum Cost Flow Problems, Math. Prog., Vol. 5, pp. 255-266 (1973).

(平成元年3月14日受付)