解　説

最大流アルゴリズムの最近の発展とその背景—II†

岩　野　和　生††

3. 最近のプリフロー・プッシュのアルゴリズム

この章では，Goldberg と Tarjan によるプリフロー・プッシュのアルゴリズムについて解説する．そして，このアルゴリズムに含まれる任意性がどのように解消されて，最近のアルゴリズムの発展がなされたのかをみていく．

まず，アルゴリズムの設計と解析において有用な手段となるアルゴリズムの不変性とポテンシャルについて説明しよう．

3.1 アルゴリズムにおける不変性とポテンシャル

あるアルゴリズムの不変性 (invariants) とは，そのアルゴリズムが，実行中に保っている一つの性質である．たとえば，Dinic 以前の最大流アルゴリズムは，アルゴリズムの実行中のいかなる時点においても，“それまでに求められているのはフローである”という性質（不変性）を保ちながら，もうこれ以上流量を増加できなくなるまで，フローの増加を繰り返している．また，プリフローを使った Karzanov や Goldberg と Tarjan のアルゴリズムでは，“現在求められているプリフローに対する残余グラフには，始点 s から終点 t への増加パスがない”という不変性が保たれている．このように，問題がある二つ以上の条件を満たす解を求めるものであるとき，一部の条件をつねに満たし（不変性を保ち）ながら，残りの条件を満たすように，解を変化（改善）させていくという方法がよく用いられる．このとき，アルゴリズムにおいて用いられる各操作が，この不変性を保つように設計しなければならない．

ポテンシャル (potential) の概念は，アルゴリズムの進行度（直感的には，途中の解が，どの程度最適解に近いのかを示す目安）や計算複雑度の解析に用いられる．

たとえば，Dinic のアルゴリズムにおいて，ϕ＝(残余グラフにおける s から t までの距離) と定義すると，あきらかに，$0 \leq \phi \leq n-1$ であり，高々 m 回のフローの増加ごとに，ϕ は少なくとも 1 増加し，途中で減少することはない．したがって，高々 $m(n-1)$ 回のフローの増加があることが分かり，各フローの増加に $O(n)$ 時間かかるので全体で $O(mn^2)$ 時間かかる．このように，ポテンシャルをうまく定義することによって，アルゴリズムの解析がみやすくなる場合がある．また，逆に，妥当なポテンシャルを考え，それに応じた操作を設計してアルゴリズムを構築することも可能になる．Karzanov のアルゴリズムでは，$\phi = \sum e(v)$ などが考えられる．Goldberg と Tarjan 以降のアルゴリズムでは，これらの概念がうまく使われているので以下の節で詳しくみていきたい．

3.2 Goldberg と Tarjan のプリフロー・プッシュのアルゴリズム

Goldberg と Tarjan のプリフロー・プッシュ (preflow push) のアルゴリズム[GolTarj86]は，Karzanov のアルゴリズムのように，最初は始点 s に関するプリフロー f_{source} から始める．次に，任意の活動点から残余グラフの辺に沿って t に“もっとも近い”と判断される点に向けてフローを押し出しながら，t にフローを集めていく．t に押し出せないフローは，やはり，残余グラフの辺に沿って s に“もっとも近い”と判断される点に向けてフローを押し戻す．

このとき“もっとも近い”の判断に用いられるのが，距離ラベル (distance label) と呼ばれる点集合 V から非負整数への関数である．また，$d(v)=d(w)+1$ であるような辺 (v, w) を可能辺(eligible edge) と呼ぶ．

Goldberg と Tarjan のアルゴリズムは，次の始点 s に関するプリフロー f_{init} と距離ラベル d_{init} から始めて，活動点がなくなるまで，以下に説明される push と relabel の操作を繰り返すことから成り立つ．

† Recent Development of Maximum Flow Algorithms and Their Background—II 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

s から出る辺 $[s, v]$ について，$f_{init}(s, v)=u(s, v)$，$f_{init}(v, s)=-u(v, s)$ であり，それ以外の辺 $[v, w]$ について，$f_{init}(v, w)=0$ である．$d_{init}(s)=n$ であり，それ以外の点 v について，$d_{init}(v)=0$ とする．このとき，

```
preflow-push :
  begin
    do while (活動点 v が存在する)
      push(v, w) または，relabel(v)
    end
  end
```

である．

push(v, w) は，v が活動点 $(e(v)>0)$ で，$[v, w]$ が可能辺 $(d(v)=d(w)+1)$ のときに適用され，v から w に $\delta=\min\{e(v), u_f(v, w)\}$ だけフローを押し出す．$[v, w]$ がこの push によって飽和するとき，この push を飽和 push (saturating push)，そうでないとき，非飽和 push (nonsaturating push) と呼ぶ．

relabel(v) は，v が活動点であるが，v から出る可能辺が一つもないときに適用され，$d(v)$ を $\min\{d(w)+1 | u_f(v, w)>0\}$ で置き換える．

プリフロー f に対する正当な距離ラベルとは，次の二つの性質を満たすものである．

(D 1)　$(v, w)\in E_f$ で $u_f(v, w)>0$ のとき，$d(v)\leqq d(w)+1$.

(D 2)　$d(s)=n, d(t)=0$.

最初の距離ラベル d_{init} は正当であり，**3.4** に示されるように，このアルゴリズムの途中の距離ラベルも正当である．また，このアルゴリズムの途中の距離ラベル $d(v)$ について，$d(v)<n$ のとき，$d(v)$ は v から終点 t への最短路の長さの下界であり，$d(v)\geqq n$ のとき，$d(v)-n$ は，始点 s への最短路の長さの下界であることが分かる．

このアルゴリズムは，Orlin が指摘しているように，次のように考えると分かりやすい．**図-6** のように，各点 $v\in V$ に十分な大きさのバケツがあり，バケツ同士は，残余グラフでの辺に相当するホースでつながれている．$d(v)$ はこれらのバケツの位置する高さに相当し，$d(s)=n, d(t)=0$ である．ホース (v, w) には止金があり，この止金は $d(v)=d(w)+1$ のとき（つまり，v のバケツが w のバケツよりも 1 だけ高い位置にあるとき）にのみ開かれているとする．このとき，relabel(v) は，まわりのバケツがすべて自分の高さ以上にあるために，水を流せない状態の v に対して，少なくとも 1

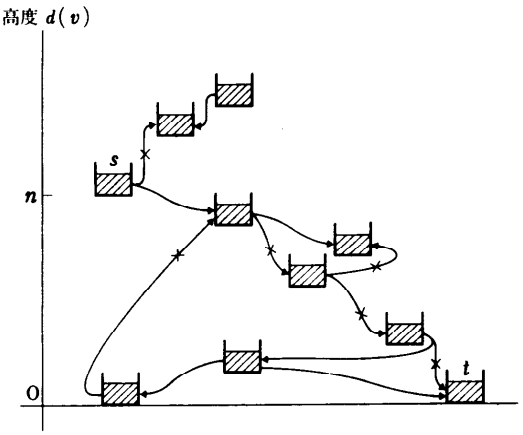

図-6　バケツ s からバケツ t へ水を流す．×印は止金を示す．

つのバケツだけには流せるように，v のバケツを持ち上げる操作である．最終的に t に流せなかった水は，s より高く $(d(v)>n)$ 持ち上げることによって，s に返されることになる．

図-7 は，このアルゴリズムの適用例である．

このアルゴリズムのすばらしい点は，距離ラベルの採用によって，従前のアルゴリズムのような始点 s から終点 t への道を求めるという全体的な操作を必要とせず，各点 v の局所的な情報によってのみ，次の操作が決められることである．この特長は，アルゴリズムの並列化に適している．また，push と relabel という二つの基本操作によって成り立っているので，データ構造に対する操作との対応がより簡単になった．また，アルゴリズム自体が単純なために，実用的である可能性が高いともいえる．さらに，もっと重要なのは，このアルゴリズムが，いろいろな任意性をもっていることである．このことによって，次の **3.4** にみられるように，いろいろな面での改善が（実用，および，理論面からみて）図れることになる．たとえば，次のような任意性が考えられる．

1.『活動点の選択の任意性』次回の操作の対象となる活動点の選択は自由である．たとえば，

（a）　一度選ばれた点 v に対しては，v が非活動点になるか，v からの可能辺がなくなるまで v を選択し続けるとか，

（b）　活動点を順番に選んでいくとか，

（c）　push が可能な活動点があればそれを選ぶなど，いろいろな任意性の解消が考えられる．

2.『流量の任意性』操作 push においてどれだけの

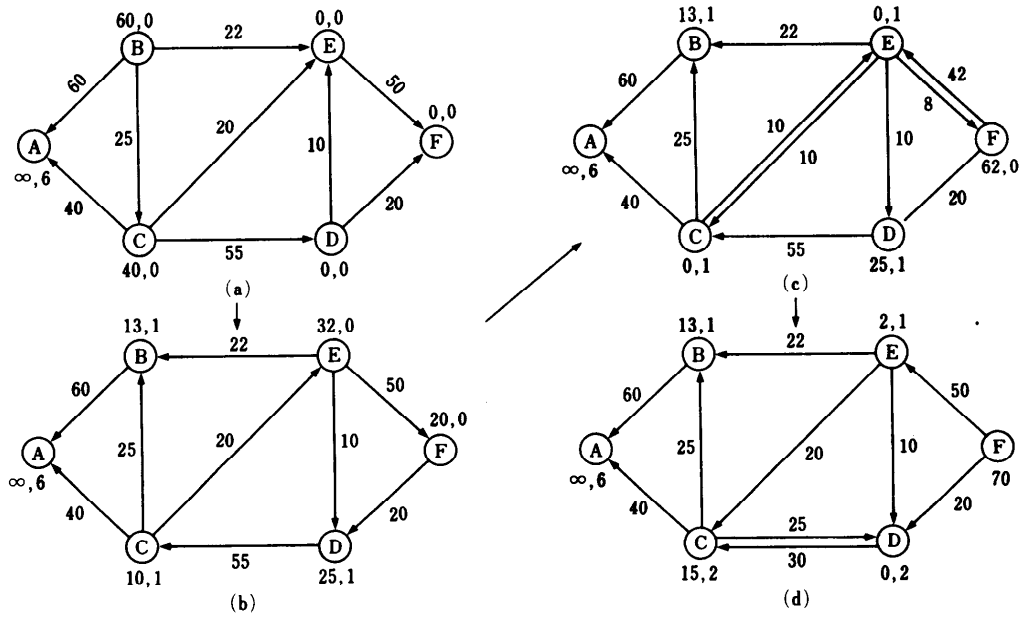

図-7 プリフロー・プッシュアルゴリズムの適用例.

(a) 最初のプリフロー．点Aから出る辺は何もないことに注意．各点についている最初の数字は，その点の残存量，二番目の数字は，その点の距離ラベルを示す．辺の数字は，その辺の残存容量を示す．

(b) relabel(B), push(B, E), push(B, C), relabel(C), push(C, D), relabel(D), push(D, F), push(D, E) の後の状態.

(c) push(C, E), relabel(E), push(E, F) の後の状態.

(d) relabel(D), push(D, C), relabel(C), push(C, E), push(E, F), ここまでで，流量 70 (最大流の流量) のプリフローが求められており，後は各点の残余量を始点 A に戻すプロセスである.

流量を押し出すのかは自由である．(上記では，$\delta = \min\{e(v), u_f(v, w)\}$ を採用している.)

3.『データ構造の選択の任意性』push と relabel をあるデータ構造の基本操作によって実現する際のデータ構造の選択の任意性.

4.『他の技術との併用の任意性』たとえば，スケーリング法の併用などが考えられる.

3.3 アルゴリズムの解析

3.2 のプリフロー・プッシュのアルゴリズムは，次の不変性を保っている.

『距離ラベルの正当不変性』どの時点においても距離ラベルの正当性は，保たれている.

『極大不変性』現時点のプリフロー f に対して，f の残余グラフで始点 s から終点 t への増加パスはない.

『距離ラベルの正当不変性』は，最初の距離ラベル d_{init} が正当であり，各 push, relabel が距離ラベルの正当性を壊さないことによる．また，どの relabel(v) も $d(v)$ を増加させ，$d(s)=n$ であるので，残余グラフで辺 $[s, v]$ が現れるときは，必ず $d(v)>n$ となり，『極大不変性』が証明される.

このアルゴリズムでは活動点があるかぎり push または，relabel が適用できる．なぜなら，v が活動点で push が適用できなければ，v から出る可能辺が一つもないので relabel(v) が適用できる．またこのアルゴリズムは，後ほど示されるように，有限回の push または relabel の操作の後停止する．したがって，最終的に活動点がなくなった状態で停止するので，フローが求められる.『極大不変性』をもったフローは最大流であるので，次の定理がえられた.

定理 3 1. プリフロー・プッシュのアルゴリズムは正しく最大流を求める.

以下の定理と補題は [GolTarj 86] による．詳しくは，原論文を見られたい.

アルゴリズムの途中のプリフローを f とすると，活動点 v から始点 s への有向道が f の残余グラフ

G_f にあることが証明される．したがって，任意の点 v について，v から始点 s への道の長さは，高々 $n-1$ なので $d(v) \leqq d(s)+n-1<2n$ である．次の補題が証明された．

補題 3.2. すべての点 v について，$d(v)<2n$ である．

プリフロー・プッシュのアルゴリズムの解析に各操作 relabel, 非飽和 push, 飽和 push の数が必要であるが，以下の定理が導かれる．まず，補題3.2より次の定理は明らかである．

定理 3.3. 操作 relabel の数はアルゴリズム全体で，高々 $2n^2$ である．

定理 3.4. 飽和 push の数は，高々 $2nm$ である．

証明. 辺 $[v, w]$ に沿って行われる一連の飽和 push に着目する．ある飽和 push-push(v, w) が行われたとき，$d(v)=d(w)+1$ であり，辺 $[v, w]$ は残余グラフから消える．次の飽和 push-push(v, w) が行われるためには，push(w, v) によって，辺 $[v, w]$ が，再び残余グラフに現れることが必要である．このとき，新たな距離ラベル $d'(v)$ と $d'(w)$ について，$d'(w)=d'(v)+1 \geqq d(v)+1 \geqq d(w)+2$ である．したがって，次の飽和 push-push(v, w) のまえに w の距離ラベルは少なくとも2だけ増加する必要がある．補題3.2より，辺 (v, w) に関する飽和 push は，両方向数えると，高々 $2n$ 回である □

定理 3.5. 非飽和 push の数は，高々 $4n^2m$ である．

証明. ポテンシャル $\Phi=\sum\{d(v) \mid v$ は活動点$\}$ を用いる．最初は s 以外のすべての点の距離ラベルは0なので $\Phi_{init}=0$ であり，最後は活動点が何もないので $\Phi_{last}=0$ である．

relabel(v) によって $d(v)$ が a 増加すると，Φ も a 増加し，補題3.2より，relabel 全部による Φ の増加分は，Φ_{delta}(relabel)$\leqq(2n-1)(n-2)$ である．

push(v, w) が飽和 push のとき，w は新たに活動点に変わるかもしれないので，Φ の増加分は高々 $d(w)$ である．定理3.4より，飽和 push による増加分は，Φ_{delta} (飽和 push)$\leqq 2nm(2n-1)$ である．push(v, w) が非飽和 push のとき，v は必ず非活動になり，w は新たに活動点になるかもしれない．ところが，$d(v)=d(w)+1$ なので，一回の非飽和 push によって，Φ は少なくとも1は減少する$(-d(v)+d(w)=-d(v)+d(v)-1=-1)$．$\Phi$ の増加分は，$m \geqq n-1$ という仮定を用いて，全体で高々 Φ_{delta}(relabel)$+\Phi_{delta}$(飽和 push)$\leqq(2n-1)(n-2)+2nm(2n-1) \leqq(2n-1)+m+2nm(2n-1) \leqq 4n^2m$ なので，非飽和 push の数は $O(n^2m)$ となる． □

3.4.1 でみるようにプリフロー・プッシュに基づくアルゴリズムでは，非飽和 push の数がアルゴリズムの速さのネックになることが知られている．

以上がプリフロー・プッシュのアルゴリズムの基本である．

3.4 アルゴリズムの改良

この節では，プリフロー・プッシュのアルゴリズムに内在するいろいろな任意性の解消がどのように図られて，前号図-5 に見られるようなアルゴリズムの最近の発展がなされたのかをみていこう．

3.4.1 FIFO, 操作 discharge の導入

Goldberg と Tarjan[GolTarj86] は，まず，**3.2** の『活動点の選択の任意性』を次のように解消した．各点 v の隣接辺をあらかじめある順に並べておき，隣接辺リスト $A(v)$ に貯える．選択された活動点 v に対して，リスト $A(v)$ から辺 $[v, w]$ を順に取り出し push(v, w) を可能なかぎり適用していく．もし，v の残存量がなくなれば v に対する操作を終了する．また，$A(v)$ の最後の辺まで処理して，v の残存量がまだあれば（v から出る可能辺がなくなった状態）relabel(v) を施し v に対する操作を終了する．この一連の v に対する操作を push/relabel(v) と呼ぶ．そして，新たな活動点を選択する．このとき，次の活動点の選択に任意性があることに注意されたい．Goldberg と Tarjan[GolTarj86] は，この push/relabel に基づいたプリフロー・プッシュのアルゴリズムについて次の定理を証明した．

定理 3.6. push/relabel に基づいたプリフロー・プッシュのアルゴリズムは，$O(nm+k)$ 時間かかる．ここで k は，非飽和 push の数である．

前節の定理3.5より，非飽和 push の数は $O(n^2m)$ なので，定理3.6より最大流が $O(n^2m)$ で求められることが分かる．Goldberg と Tarjan[GolTarj86] は，さらに，上記の push/relabel に対する活動点の選択を FIFO (first in, first out) のキュー Q を導入することによって解決すると，非飽和 push の数が $O(n^3)$ に減らせることを示した．

アルゴリズムの詳細は次のとおりである．Q に活動点を貯えておき，Q から順次取り出される点 v に対して，push/relabel(v) を施す．その最中に新たに活動点になった点は Q の最後に加え，v がこの操作の後も活動点であれば，Q の最後に v を加える．そして

Q が空になるまで上記のステップを繰り返すのである.

このとき, ポテンシャル $\Phi=\max\{d(v)|v$ は活動点$\}$ を用いると, キューを高々 $4n^2$ 巡りしかしないことが証明できる. ここで, Q の最初の巡りは, 最初に Q に含まれる点 (つまり, 辺 $[s,v]$ が G にあるような s の隣接点) に対する push/relabel 操作全体からなり, 二巡目以降は, 前の巡りの最中に Q に加えられた点に対する push/relabel 操作全体からなる. すると, 各巡りにおいて, 各点ごとに, 高々1回の非飽和 push しかないので, アルゴリズム全体で, 非飽和 push の数は $O(n^3)$ となる. したがって, 定理 3.6 より最大流が $O(n^3)$ 時間で求められる.

3.4.2 Cheriyan と Maheshwari の最遠点選択

ポテンシャル $\Phi=\max\{d(v)|v$ は活動点$\}$ に着目すると, 『活動点の選択の任意性』の解消を, $d(v)$ のもっとも大きい点 (すなわち, t から最遠の活動点) の選択に求めることは自然であろう.

Cheriyan と Maheshwari は, 最短距離ラベルと呼ばれる特殊な距離ラベルを用いて, 次の定理を証明し, 最大流が $O(n^2\sqrt{m})$ 時間で求められることを示した[CheMah88]. ここで, 最短距離ラベルは, 点 v から終点 t への道があれば, v から t への最短路の長さで, そうでなければ, v から始点 s への最短路の長さに n をたしたものであると定義される.

定理 3.7. 最短距離ラベルを用いた最遠点選択によるプリフロー・プッシュのアルゴリズムには, 高々 $O(n^2\sqrt{m})$ 回の非飽和 push がある.

3.4.3 Ahuja と Orlin の残存量スケーリング法

Ahuja と Orlin は, プリフロー・プッシュのアルゴリズムに, Gabow の容量スケーリング法の着想を取り入れ非飽和 push の数が $O(n^2\log U)$ になることを示した[AhuOrl87]. ここで, U は最大の辺の容量であり, $U=O(n^k)$ のとき, $O(n^2\log n)$ となり, Cheriyan と Maheshwari の $O(n^2\sqrt{m})$ よりかなり少なくなる. スケーリング法を使うので容量関数は整数値を取ると仮定する.

かれらの基本的な着想は, 残存量の多い点から少ない点にフローを流し, ネットワーク上にできるだけフローの残存量を均一に分散し, それらの最大の残存量を段階的に減らしていこうというものである. これを残存量スケーリング法 (excess scaling) と呼ぼう.

段階 i における最大残存量の上界を Δ_i とする. このとき, 次の不変性を保つようにする.

『残存量不変性: 段階 i では, すべての点 v について, $e(v)\leqq\Delta_i$ である.』

最初は, $\Delta_0=2^{\lceil\log U\rceil}$ とし, $\Delta_{i+1}=\Delta_i/2$ とする. すると last$=(\lceil\log U\rceil+1)$ 段階の後に Δ_{last} は1より小さくなる. 残存量不変性が保たれていて, 整数容量を仮定しているので, すべての点の残存量が0になり, 最大流が求められる.

問題は各段階におけるアルゴリズムの設計である. 以後簡単のために, Δ_i のかわりに Δ を用いる. Ahuja と Orlin は, 上記の『残存量不変性』を保つために, push(v,w) は $\min\{e(v), u(v,w), \Delta-e(w)\}$ のフローを流すことにした. このことによって, この push の後, $e(w)\leqq\Delta$ が保証される. 次に, 非飽和 push の数を制限するために,

『流量不変性: すべての非飽和 push は, 少なくとも $\Delta/2$ だけフローを流す.』

を導入した. 『流量不変性』を満たすとき, 非飽和 push は, $\min\{e(v), u(v,w), \Delta-e(w)\}=\min\{e(v), \Delta-e(w)\}\geqq\Delta/2$ なので, $e(v)\geqq\Delta/2$ かつ $e(w)\leqq\Delta/2$ を満たさなければならない. そこで, かれらは, 活動点の選択を "最近点選択", すなわち, $e(v)\geqq\Delta/2$ を満たす点のうち t にもっとも近い点を選択することにした. そして, 非飽和 push の数について, 次の定理を証明した.

定理 3.8. 各段階ごとに, 非飽和 push の数は, $8n^2$ 以下である.

証明. 今, 第 i 段階であり, $\Delta=\Delta_i$ とする. このとき, ポテンシャル $\Phi=\sum e(v)d(v)/\Delta$ を用いることにする. 『残存量不変性』より, $e(v)\leqq\Delta$ なので, 補題 3.2 より, 第 i 段階の最初のポテンシャルを Φ_{init} とすると, $\Phi_{init}<2n^2$ になる.

relabel(v) によって, $d(v)$ が a 増加したとすると, Φ も a 増加するので, relabel 全体による Φ の増加分は, $\Phi_{delta}(\text{relabel})<2n^2$ である.

また, push によって Φ は減少し, とくに非飽和 push は『流量不変性』により必ず $\Delta/2$ だけフローを流すので, 一回の非飽和 push ごとに Φ を少なくとも 1/2 だけ減少する. したがって, 非飽和 push の数は, 高々 $(\Phi_{init}+\Phi_{delta}(\text{relabel}))/(1/2)=8n^2$ である. □

定理 3.8 より, 全体で $O(n^2\log U)$ 回の非飽和 push があることが分かる. さらに, このとき Ahuja と Orlin は, 最大流が $O(nm+n^2\log U)$ で求められることを示した. 図-8 は, Ahuja と Orlin の残存量ス

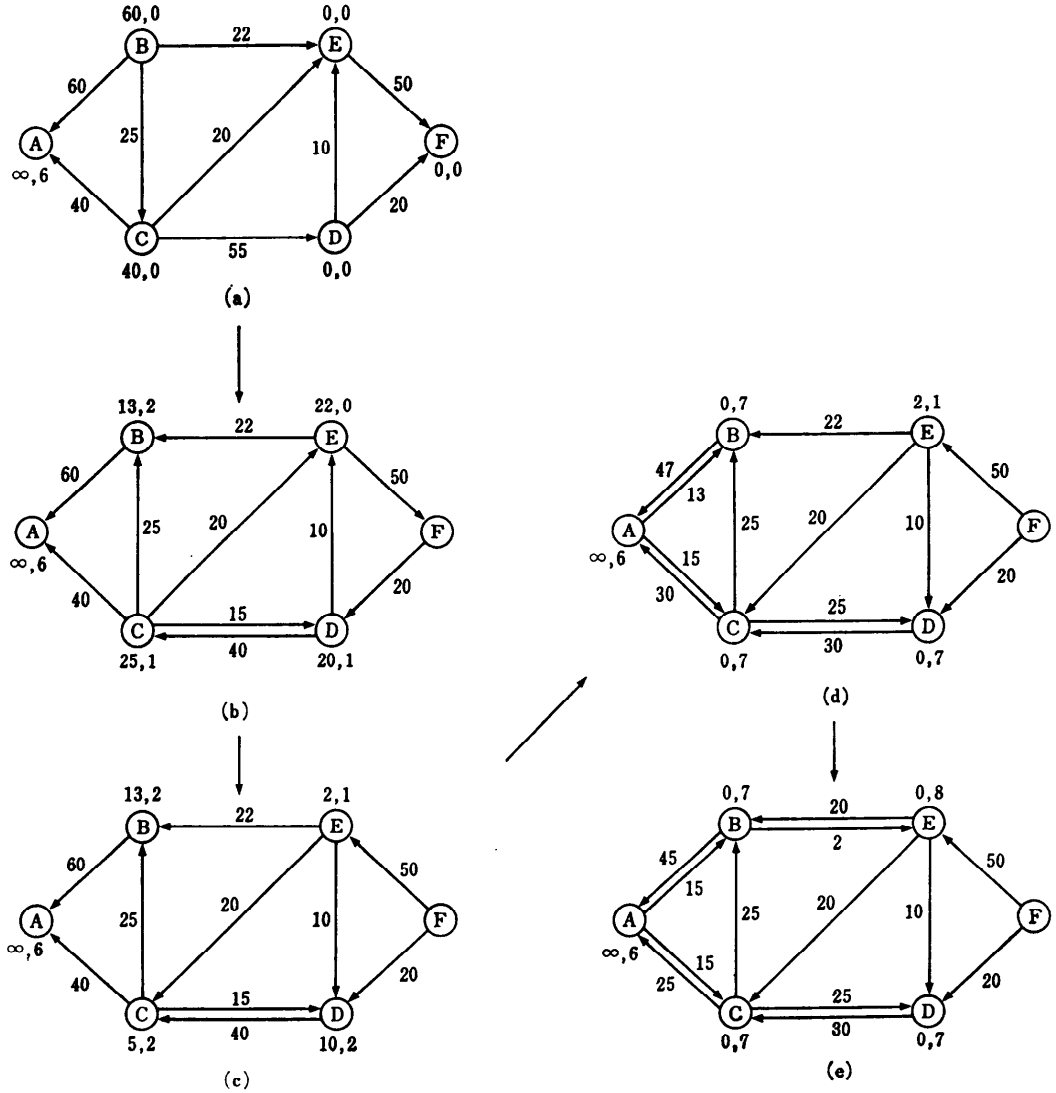

図-8　Ahuja と Orlin の残存量スケーリング法．以下の説明で push(x, y, z) は点 x から点 y にフローを z だけ push することを示す．relabel(x, z) は点 x の距離ラベルを z にすることを示す．

(a) Δ=64 の段階．最初，点 B と C の残存量が 32 以上なので対象となる．relabel(B, 1), relabel(C, 1), push(C, D, 40), relabel(D, 1), push(D, F, 20), push(B, E, 22), relabel(B, 2), push(B, C, 25) の後．

(b) Δ=32 の段階．relabel(E, 1), push(E, F, 22), relabel(D, 2), relabel(C, 2), push(D, E, 10), push(C, E, 20), push(E, F, 28) の後．ここまでで，流量 70（最大流の流量）のプリフローが求められており，後は各点の残存量を始点 A に戻すプロセスである．

(c) Δ=16 の段階．relabel(D, 3), relabel(B, 7), push(D, C, 10), relabel(C, 4), push(C, D, 15), relabel(D, 5), push(D, C, 15), relabel(C, 6), push(C, D, 15), relabel(D, 7), push(D, C, 15), relabel(C, 7), push(C, A, 15), push(B, A, 13) の後．

(d) Δ=8 の段階では，対象となる点がない．

Δ=4 の段階．relabel(E, 8), push(E, B, 2), push(B, A, 2) の後（図 e の状態），すべての点の残存量が 0 になりフロー（すなわち，最大流）が求められた．

ケーリング法の適用例である.

さらに Ahuja と Orlin は, スケーリングの倍率を 1/2 の代わりに $1/k$ を採用すると非飽和 push の数が全体で $O(kn^2(\log_k U+1))$ になることを示した.

3.4.4 Ahuja, Orlin, Tarjan の残存量スケーリング法+最遠点選択

Ahuja と Orlin の最近点選択による残存量スケーリング法の動きに着目すると, push(v, w) が行われるときは, つねに, 多くの残存量を v の背後 (t から見て v より遠い点) に残している. Ahuja, Orlin, Tarjan はこの点に注目し, Cheriyan と Maheshwari のように最遠点選択を採用すると, さらに改善が図れることを示した[AOT87]. かれらは, スケーリングの倍率を $1/k$ とし, 最後に最適な k を求めた.

しかし, 最遠点選択によっては, Ahuja と Orlin の『流量不変性』つまり, "すべての非飽和 push は, 少なくとも Δ/k だけフローを流す" が, 保たれないことがある. Ahuja, Orlin, Tarjan は, スタック (last-in, first-out) を導入して, この解決を図った. まず, 次のようにスタックの操作を決めた. 非飽和 push-push(v, w) によって, $e(w)=\Delta$ になるかぎり, w をスタックに積み, w は $e(w)\leqq\Delta/k$ になるか, 操作 relabel(w) が施されたときに, スタックから取り除かれる. スタックから点が取り除かれるときは, その下にある点 x が $e(x)\leqq\Delta/k$ であれば, x も取り除かれることにする. かれらは, "スタックを空にする非飽和 push は, 少なくとも Δ/k はフローを流す" という不変性が, このスタックの導入によって保たれ, この不変性を用いても Ahuja と Orlin と同様の解析ができることを示した.

さらに, かれらは最遠点選択を採用することによって, どの段階においても, t からある距離離れたすべての点で残存量が, Δ/k 以下になることを利用して次の定理を証明した.

定理 3.9. スタックを利用した最遠点選択による残存量スケーリング法での非飽和 push の数は, 全体で $O(nm+kn^2+n^2(\log_k U+1))$ である.

そして, $k=\lceil \log U/\log\log U\rceil$ にすると, 最大流が $O(nm+n^2(\log U/\log\log U))$ で求められることを示した.

3.4.5 動的木による改良

前号 **2.4** でも触れたが, 適切なデータ構造を用いることによって, 全体の計算時間を減らしうる. Ahuja, Orlin, Tarjan[AOT87]は, プリフロー・プッシュのアルゴリズムで push の数が, $p\geqq nm$ であるものは, 動的木を使うことによって, 全体の計算時間を $O(p)$ から $O(nm\log(1+p/nm))$ に減らせるのではないかという予想を立てている. 実際, Goldberg と Tarjan は, **3.4.1** の $O(n^3)$ の時間を $O(nm\log(n^2/m))$ に減らし[GolTarj86], Ahuja, Orlin, Tarjan は, **3.4.4** の $O(nm+n^2(\log U/\log\log U))$ の時間を $O(nm\log((n\log U/m\log\log U)+2))$ に減らしている[AOT87].

このように動的木を使うと計算時間が減ることが示されてきたが, 実用面についていえば, かなり大きなネットワークを取り扱うときに, 動的木のような複雑なデータ構造を使うことが有効になる場合があるのかもしれない. しかし, 現段階では, 動的木の使用は, 理論的興味によるものである.

4. おわりに

この解説では, 最大流問題のアルゴリズムが, いろいろな着想に基づいて, どのように発展してきたかをみてきた. 単純で, しかも, 並列化に向いており拡張性の高いプリフロー・プッシュのアルゴリズムは, さらなる理論面, 実用面での研究の余地があるように思われる. 理論的興味としては, $O(nm)$ 時間の最大流アルゴリズムを求めることは, あながち無鉄砲な試みとはいえないと Ahuja, Orlin, Tarjan が述べている[AOT 87].

実用性の観点については, この解説で触れることができなかったが, プリフロー・プッシュのアルゴリズムは単純なので, 理論的興味のほかにも, 実用面でも期待できそうである. さらに, 最近の最大流のアルゴリズムについて, [Imai 83] のような実証的な研究が望まれる. また, 最大流問題の並列・分散アルゴリズムも, かなり研究されているようである[Awe85, BerEck87, SiVi82]が, この面においてもプリフロー・プッシュのアルゴリズムは重要だろう[Gol85a, Gol85b, GolTarj86].

最小費用流問題についても, 似たようなアルゴリズムの発展があり, そこにおいても, プリフロー・プッシュ, ポテンシャル, 不変性などが縦横に使われており, 関連論文などを見ると興味深いと思われる[IriFuj Ohya86, Tard85].

最後に, 草稿を読んでいただき貴重な意見をいただいた上智大学の浅野孝夫教授, 日本アイ・ビー・エム(株)の福永光一, 手塚集, 徳山豪各氏に感謝いたします. (平成元年3月14日受付)