最大流量の算法の最近の話題

真鍋　龍太郎

枝に容量のあるネットワークの入口の点から出口の点までに流せる最大流量を求める算法は，1956年に Ford と Fulkerson が提案した，ラベリングで流れを追加できる道を見つけてゆく方法が最もポピュラーで，最大流の計算法はこれだけだと思っている人が多いほどになっている．この算法は，実用上は問題なく使われてはいるようだが，枝の容量が無理数だと有限回では終了しない例は Fulkerson らも示しているし，たとえ整数値容量でも，馬鹿げた回数の繰り返しをすることもある．このラベリング法の欠点は，計算量が問題のサイズではなく，最終的に得る最大流による点にある．

これに対し，ラベリング法を有限回で終了させる改善と，計算量の上界を押さえた，Iri [12]，Tucker [17]，Galil ら [15] の仕事がある．この段階での計算量の上界は，途方もなく大きく，実用のネットワークではそうなることはないまでも，最大流問題が相当複雑なものであることがうかがわれる．

ところが Edmonds-Karp は，ラベリング法で最悪のケースでも点の数の5乗のオーダーの計算量で済む規則を示したし，流れを追加する道の情報を整理した補助的なネットワークを用いて，問題の複雑さを改善し，計算量のオーダーがだんだん下ってきた．今では，理論上はかなり早いという算法がいくつも発表されている．算法を速くする研究自体は貴重であるが，今ただちに，ラベリング法から乗り換えてしまうほどのこともなかろう．

このあたりの進歩は，ORと計算機科学の接点に当るためか，計算機科学の人たちによるものが多い．[18] には，すぐれた技術的な解説があるが，あまり広くは知られていないようだ．それとの重複もあろうが，ここでは，最近の発展の展望を試みる．§1 で問題を述べ，§§2～3 でラベリング法を，そのあと，§4 ではその10年の発展のきっかけとなった，Dinic と Karzanov の方法，さらにそれからの方法を §5 で触れる．

1. 最大流量問題

n 個の点の集合 V と，m 本の有向な枝の集合 A から成るネットワーク G を考える．枝はどの2点間にも一方向には高々1本とし，各枝 $a=(u, v)$ には正実数の容量 $c(a)$ が与えられているとする．

このネットワーク上で，入口の点 s から出口 t への**流れ**とは，各枝 a で下の $(C1)$ の**流量制限**以内であり，各点 v での流入量と流出量とが等しいという**連続条件** $(C2)$ を満たすような $\{f(a)\}$ のことを言う：

$(C1)$　$0 \leq f(a) \leq c(a)$，すべての枝 a で，

まなべ　りゅうたろう　神戸商科大学　管理科学科

(C2) $\sum_{a\in \mathrm{in}(v)} f(a)=\sum_{a\in \mathrm{out}(v)} f(a)$, $v\neq s,t$ で.

ここでin (v), out(v)は，それぞれ，点vに入る枝，vから出る枝の集合を表わす．そして次の量を**流量**Fという：

$$F=-\left(\sum_{a\in \mathrm{in}(s)} f(a)-\sum_{a\in \mathrm{out}(s)} f(a)\right)=\sum_{a\in \mathrm{in}(t)} f(a)-\sum_{a\in \mathrm{out}(t)} f(a) \quad (1)$$

(C1)，(C2)を満たし，流量がFの流れ$\{f(a)\}$を，便宜上，**流れ**Fともよぶ．**最大流問題**とは，Fを最大にする流れ$\{f(a)\}$を求めることである．

2. Ford-Fulkerson のラベリング法

(a) ラベリング：ある流れFから出発し($F=0$でもよい)，枝に残っている容量を調べて，sからtへ**流れを追加できる道**をみつけ，その道にできる限りの流れを追加するという方法である．流れを追加できる道がなくなったら，最大流を得ている [7].

流れを追加する道は，ラベリングという手順で探す．まず入口sにラベル(しるし)をつける．次に，ラベルのついた点uから，次のような枝aがあるときに，未だラベルのない点vにラベルをつける：

$$f(a)<C(a) \quad \text{の} \quad a=(u,v)\in A \quad (2)$$

$$f(a)>0 \quad \text{の} \quad a=(v,u)\in A. \quad (3)$$

(2)は，まだ容量に余裕のある枝であり，(3)はすでに存在している流れを取り消そうという逆向きの枝である．この2種類の枝を，流れの追加に**役に立つ枝**（実際に流れの追加に使うかどうかは問わず）とよぶ [5].

ラベルをつけた点から，そこに出入りする各枝が役に立つかどうかを調べることを**走査**といい，これが終わったかどうかで，ラベル済みの点を**既走査**，**未走査**の2つの状態に分ける．ラベルには，どの点からラベルがついたか，枝の方向の向きが逆向きにラベルがついたか，sからvまではいくら流れを追加できるかの，3つの情報をもたせておく．

この結果，出口tにラベルがついたら，tからsへラベルを遡って，sからtへの流れを追加できる道pを見つけられ，その道に追加できる最大限の流れΔfをsからtに加える．この値は，

$$\Delta f=\min\begin{cases}\min\{c(a)-f(a)\}, & \text{前向きの枝 } a\\ \min\{f(a)\}, & \text{逆向きの枝 } a\end{cases} \quad (4)$$

これを用いて，p上の各枝の流れを修正し，

$$\begin{aligned} &f(a)\leftarrow f(a)+\Delta f, \quad a:\text{前向きの枝}\\ &f(a)\leftarrow f(a)-\Delta f, \quad a:\text{逆向きの枝}\end{aligned} \quad (5)$$

流量Fも追加し，$F+\Delta f$とする．

このあと，s以外の全点のラベルを消して，tにラベルがつかなくなるまで（流れを追加できる道がなくなるまで），繰り返す．

(b) 問題点：この算法は，枝の容量が整数なら，毎回の流れの追加量は整数であり，有限回の流れの追加で，整数値の最大流量を得る．容量が有理数でも有限回で終わる．しかし，無理数になると，有限回で収束しないばかりか，無限回で収束した値も最大流になってない例もある[p. 217].

たとえ整数データでも，おかしな現象が起こることがある．図1の例で[4]，流れを追加する道として，(s,u,t)，(s,v,t)の2つの道がこの順に選ばれたら，2回の流れの追加で最大流量$2M$を得る．ところが，(s,u,v,t)と(s,v,u,t)とが交互に選ばれることになると，毎回の追加量は1で，$2M$回もの反復が要る．つまり，必要な反復数が，計算をしてからわかる最大流量に依存しており，ネットワークの大きさで決まるのではない，という欠点があるのだ．

3. ラベリング法の有限回での収束

ラベリング法で前記のような問題が起こるの

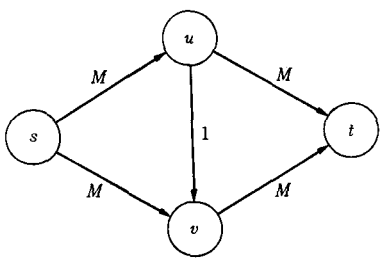

図 1 （枝の記号，数字は容量）

は，ラベルがついた点の中から，次に走査をする点を選ぶルールが確定してなかったためであることに注目して，有限回で終わるような改善がいくつか示された．その最初は Edmonds と Karp によるもの[4]と書いたものが多いが（実は筆者もそう書いてしまった1人だが），Iri が有限回で終わるための策を提案している［pp. 140～142, 12］．さらに，最近は Tucker も提案し［17］，Galil ら［15］がそれに計算量の上界を与えた．

では，有限回での収束の保証なく，ラベリング法を用いていて，実用上困ったという報告がほとんどないのはなぜであろうか．その理由のひとつは，実際のネットワークには，それほどタチの悪い問題がなかったこと，それに，以下に述べる算法上の改善が，プログラムする段階で，有限性の認識はないままで，用いられていたためと思う．

(a) Iri のジグザグ現象防止策：図1の例のように，ある枝が，流れの追加ごとに正逆向きに交互に飽和するジグザグ現象を避けるために，次の規則が提案された[12]：

"流れを追加する道を探すときは，まず，正逆いずれの向きにも飽和してない枝のみからなる道を探す．そういう道がないときに限り，いずれか一方の向きに飽和している枝を含むものをさがす．"

このルールを用いることにより，$O(m3^m)$ 回の流れの追加で，最大流量が求められる．

(b) Tucker の有限性の証明：次に走査する点を，"ラベル済みで未走査の点の集合から，前の反復までで行なったのと同じ順に選んでゆく"と有限回で収束する［17］．次の Edmonds と Karp のものより数年遅れて出たし，計算量の上界も示してはないが，簡単な策で有限性が保証されるという点で意味がある．

Meggido と Galil が，この手順に従うと，$O[\min(n!, 3^m)]$ の手間で収束することを証明した［15］

この上界も，Iri によるものも，大き過ぎる値で実用的な評価には遠い．実際のネットワークでは，各点に出入りする枝の数は高々数本のことが多いから，これほど大きくはあるまい．

(c) 流れ追加の最短の道：Edmonds と Karp は[4]，流れを追加する道として，"含まれる枝の数を道の長さとしたときに，流れを追加できる道の中で最も短いものを採用する"という規則で，高々 mn^3 の手間で，最大流が求められることを示した．では，流れを追加できる道の中で最短のものをどうやってみつけるかだが，これは，"最も早くラベルがついた点から走査する"という簡単な規則でできる．

4. 補助ネットワークを用いる方法

最大流量の計算量の上界をラベリング法よりも1桁以上も小さくする算法が，ソ連の Dinic と Karzanov により提案された．彼らは，閉路を含んだ一般のネットワーク上の問題を，s から t への流れを追加できる道の集合でつくった閉路のないネットワーク上の問題に移しかえて，問題の構造を簡単にして，計算量を減らしたと言える．Dinic［3］は，Edmonds-Karp よりも早く発表されたが，説明は［13］とまとめるためにあとにした．

Dinic の方法は，2段階から成る［3］．第1段階は，ある流れから始めて，流れを追加できる最短路をすべて列挙した別のネットワーク（以後補助ネットワークとよぶ）をつくる．第2段階では，補助ネットワーク上で，極大流という流れを求める．この極大流は，元のネットワークの流れを追加できるすべての道で追加する流れに対応するので，極大流を元のネットワークの対応する枝に加えてやる．これで，第1段階にもどる．以上を，第1段階で，流れを追加する道ができなくなるまで，反復する．これらの手順をもう少し説明しよう．

(a) 補助ネットワーク：流れ $\{f(a)\}$ がネットワーク G にあるとき，**補助ネットワーク** $\tilde{G}$ は，点

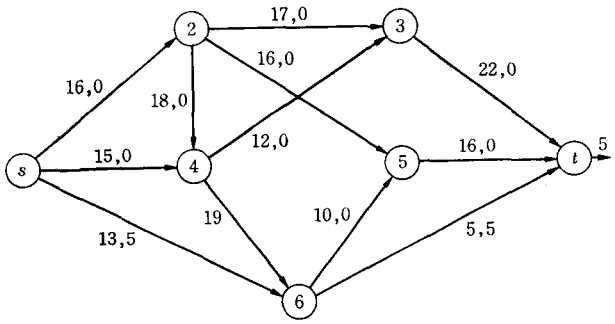

図 2 ネットワーク G（枝の数字は容量と流れ）

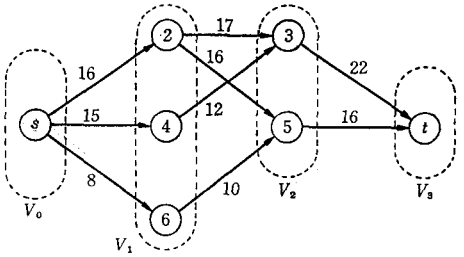

図 3 図2の G と流れに対する補助ネットワーク $\tilde{G}$（数字は容量）

の集合は G と同じだが，s からその点までの流れの追加の最短路の長さによって層別したもの，枝は G で流れの追加に役に立つものに対応して，つくったものであり，次のようにつくる．s からの最短路に i 本の枝が含まれるような点の集合を**層**といい V_i と表わす．まず $V_0=\{s\}$ だけで $\tilde{G}$ とする．次に s から役に立つ枝で結ばれる点の集合 V_1 を $\tilde{G}$ に加える．この役に立つ枝も $\tilde{G}$ に加える．次に，V_1 の点から，役に立つ枝で結ばれている点の集合 V_2 を $\tilde{G}$ に加え，この役に立つ枝も $\tilde{G}$ に加える．こうして，t が $\tilde{G}$ に加えられたら止める．

たとえば，図2のように枝の容量と流れが与えられていたら，補助ネットワークは図3のようになる．図4のように流れがあると，これに対する補助ネットワークは，図5のようにつくれる．図5の枝 (3, 2) は，元のネットワーク（図4）の枝 (2, 3) の流れを打ち消す逆向きの枝であることに注目されたい．

G で，流れを追加できる最短の道の長さが l であると，$\tilde{G}$ では点の集合 $V_0=\{s\}$, $V_1, V_2, \cdots, V_l=\{t\}$ を得，枝はすべて V_i から次の V_{i+1} へ向うもののみになっている．

t を含む $\tilde{G}$ がつくれなかったら，G に流れを追加できる道がないわけで，最大流を得ている．

(b) 補助ネットワーク上の極大流：補助ネットワーク上の枝の容量，流れには記号の上に波印をつけて表わすことにする．$\tilde{G}$ の枝 a の**容量**を次のように定義する．すなわち，$\tilde{G}$ の枝 $a=(u, v)$ に対応する G の枝が，

前向きなら，　$\tilde{c}(a)=c(a)-f(a)$,

逆向きなら，　$\tilde{c}(a)=f(a)$.

また，**G 上の極大流**を，s から t へのどの道も飽和した枝を少なくとも1つは含んでいるような流れ $\{f(a)\}$ と定義する．この $\tilde{G}$ 上の極大流は，普通の意味の最大流とは異なる．たとえば，すべての枝の容量が1の図6で，道 $(s, 1, 4, t)$ 上の値1の流れは，極大流だが，最大流ではない．

こう定義した極大流 $\tilde{F}$ が $\tilde{G}$ 上で求められると，これは元の G で流れを追加できる長さ l の道をすべて用いて増加できる流れであり，G の流量を $\Delta f=\tilde{F}$ 増やし，各枝ごとの流れ $\tilde{f}(a)$ を対応する G の枝の Δf として(5)に用いて，新しい流れを求め

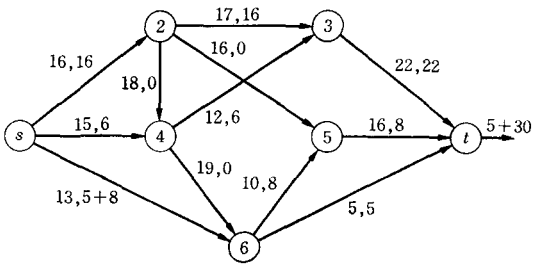

図 4 ネットワーク G（枝の数字は容量と流れ）

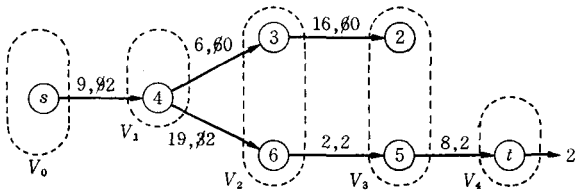

図 5 図4のネットワークの補助ネットワーク $\tilde{G}$（枝の数字は容量と流れ，消した数字は押し流しのときに修正した量）

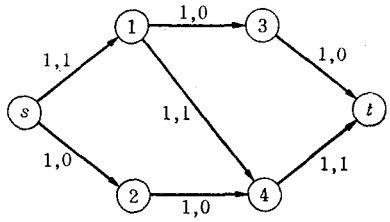

図 6 $\tilde{G}$ (枝の数字は容量と流れ)

る．こうしてから，第1段階にもどる．

$\tilde{G}$における極大流を，Dinic [3] は，深さ優先の探索で，流れを追加する道を1つずつ見つけて求めたが，Karzanov [13] が，流れ追加の道という概念を破った方法を示した．

(c) **Karzanov の方法**：彼は，各点での流れの連続条件 $(C1)$ を緩めた，次の $(C3)$ を満たす流れを導入し，これを**準流れ** preflow とよぶ：

$(C3)$ $\sum_{a\in \mathrm{in}(v)} \tilde{f}(a) \geq \sum_{a\in \mathrm{out}(v)} \tilde{f}(a)$.

そして，各点で$(C3)$の両辺の差を超過量 $e(v)\geq 0$ と定義しておく．

Karzanov の考え方は，s から流量制限を満たす範囲でできる限りの流れを逐次割り当ててゆき，あとで，各点で連続条件を満たすように修正するものである．

まず s から出る各枝に容量一杯の流れを与える．次に，V_1 の各点で，そこへの流入量の範囲内で，そこから出る各枝と容量内で可能な限りの量を流すように割り当てる．この手順を**押し流し** PUSH とよび，順次 t まで行なう．すると，釣り合ってない$(e(v)>0$ の$)$点ができてしまう．そこで，$e(v)>0$ の点を含む，t に最も近い層 V_i から逆に，$e(v)>0$ の点を $e(v)=0$ とするように，v に入る準流れを必要なだけ取り消してゆく．この手順を BALANCE とよぶ．$e(v)=0$ になった点に入る枝の流れは，以後はいじらない．

この PUSH と BALANCE の手順を，V_i で PUSH が V_{i+1} に少しでも流せたら，PUSH を V_{i+1} に用い，失敗したら $e(v)>0$ の v を含む t に最も近い V に BALANCE を行なう．BALANCE ができたら，PUSH を，と繰り返し，s, t 以外の全点で $e(v)=0$ となるまで行なう．

(d) **計算の手間**：本節の方法では，最大流量が求められるまでに，補助ネットワークを作り，そこで極大流を求めるという手順を，何度繰り返すだろうか．

k 回目の反復で流れを追加できる s から t への道の長さ l_k は，毎反復では長さ l_k の道はすべて飽和するように流れを増やすので，単調増加で $l_{k+1}>l_k$．道の長さは，高々 n-1 だから，反復数は n-1 以下である．

補助ネットワークを作るには，元のネットワークの各枝を正逆向きに高々2回調べるだけだから，手間は合わせて，枝の総数に比例して $O(m)\leq O(n^2)$.

第2段階で，Karzanov の方法で $\tilde{G}$ の極大流を求めると，各点は高々1度だけ BALANCE され，この手順を高々(中間点の数) n-2 回通って終了する．個々の BALANCE の手間は，流れの取消しの手間で決まり，さらにこれは PUSH のときに枝への流れの追加の回数で決まる．点ごとには，そこの out(v) の数だが，これは高々 n-1 本，したがって総計では $(n$-1$)(n$-2$)$ 回の手間で，上界は $O(n^2)$ である．

以上から，第1，2段階ともに $O(n^2)$ で，これを n-1 回反復するのだから，Dinic-Karzanov の方法は，$O(n^3)$ の計算量が要る．

ところで，計算量の上界の飛躍的改善は，本節始めに，流れの問題を閉路を含まないネットワーク上のものに直したことによると書いた．実は，PUSH, BALANCE の手順は，Ford-Fulkerson 以前に提案された最大流問題のヒューリスティック法と同一のアイデアであることは興味深い．Boldyreff [1] は，PUSH の要領で，入口から順に流量制限一杯に流れを割り当て，あとで出口から逆に，各点の出入りを調整する，**洪水法**というものを示した．洪水法は，与えられたネットワーク上で実行するので，閉路があるとなかなかう

まく系統的にはできない．Karzanovは，[1]を引用してはいないのだが，最短の流れ追加の道に注目して，閉路のない補助ネットワークを作ってから洪水法を用いるということで，問題の複雑さを一段緩めることによって，初等的な考え方にもどれて成功したと言えるのではなかろうか．

5. その後の発展

この数年に発表されている最大流量問題の方法は，いずれも，Dinic と Karzanov の算法から出発しているといってよい．計算量の上界を小さくしてゆくことは，問題の特殊構造を利用して，ある部分のみを探索できるようにしていると言えようが，そのためにプログラム技法上で，かなり高級なことをして，プログラムのほうの複雑さが増してくる．したがって，実際に計算するには，問題の大きさや複雑さと，プログラムの手間とのトレードオフで使う算法が決まるといってよい．

以下に，この数年の若干の進展を示す．

(a) 両方向からのラベリング：ラベリング法にしろ，Dinic-Karzanov の方法にしろ，入口 s から進むだけでなく，出口 t からも後向きに進めてきて，両方向からのラベルが接したところで，s から t への道をつくるという方法が考えられる．すでに知られてはいるようだが，Pohl [17]が，前節の方法に適用し，$n=1000, m=5000$ までのランダムに作ったネットワークで計算実験をした．平均計算時間が $O(n^2)$ でしか増えないと報告している．

(b) ネックの点からの押し流し：Karzanovの押し流しを，入口から始めずに，補助ネットワーク中の最もネックになっている点を見つけて，そこから出口と入口に向けて押し流してゆこうという方法を3人のインドの研究者が提案した [14]．流れ $\tilde{f}$ があるときに，各点に入る枝の容量の和，出る枝の容量の和の小さいほうで，各点の容量 $\rho_{\tilde{f}}(v)$ を次のように定義する：

$$\rho_{\tilde{f}}(v)=\min\left[\sum_{a\in \mathrm{in}(v)}\tilde{c}(a),\ \sum_{a\in \mathrm{out}(v)}\tilde{c}(a)\right]$$

ただし，s, t では［ ］の中は一方の項のみとなる．$\rho_{\tilde{f}}$ が最小の点をネックの点とよんだ．

この方法の計算の手間の上界は，Dinic-Karzanov と同じく，$O(n^3)$ だが，実際には，もっと簡単だし，速くもなかろうか．

(c) Cherkasky, Galil らの方法：この辺にくると，算法の開発であるか，すでに発表された Dinic らの方法のプログラミング技法上の開発なのか区別がつけ難くなってくる．Cherkasky（原論文ではなく Galil の記述 [9] によるのだが）は，補助ネットワークの点を分割した層を個別には扱わず，いくつかの層をまとめて面倒をみる超層という概念に対して，Karzanov の押し流しや釣り合いの算法を適用している．

Galil [9] は，超層の記憶の方法に，グラフの森の概念を応用した構造を導入して，また [11] では，道に飽和した枝ができて中断されても，残りの部分を上手に記憶して用いるなどして，大型のネットワークに対して効果を上げようと計っている．

6. まとめ

各算法の計算量の上界の比較を，Galil [9, 10] から引用して，表1にまとめる．枝の数の両極端のケースも示してある．ラベリング法は，他の算法より簡単で，プログラムも楽だから，§3の方法のひとつを用いて，有限性を確保しておくと安全だ．少し大きいネットワークや，複雑なものには，Dinic の方法が，[18]に標準的とも言えるプログラム技法も示されており，有利だろう．

しかし，Karzanov の方法あたりからは，プログラムも面倒だし，§5の後半のものになると，プログラムの手間で，ちょっと考えてしまう．

Galil 教授は，本年5月から3か月間，学術振興会の交換で，伊理教授のもとに滞在し，筆者も面識を得たが，彼の方法を $n=200\sim1000$ のネットワークに用いたら(450ステートメントのプログラムで)，残念ながら，Cherkasky の方法のほう

表 1　各算法計算量の上界の比較

算　　法（発表年）	n：点の数，m：枝の数	$m=n^2$	$m=n$
Ford-Fulkerson ('56)	—	—	—
Iri (1969)	$m3^m$	$n^2 3^{n^2}$	$n3^n$
Tucker ('77), Galil ('79)	$\min(n!, 3^m)$	3^{n^2}	3^n
Edmonds-Karp ('73)	m^2n	n^5	n^3
Dinic (1970)	mn^2	n^4	n^3
Karzanov (1974)	n^3	n^3	n^3
Malhotra, et al. ('78)	n^3	n^3	n^3
Cherkasky (1977)	$\sqrt{m}\,n^2$	n^3	$n^{2.5}$
Galil [9] (1978)	$m^{2/5}\,n^{5/3}$	n^3	$n^{2.33}$
Galil [11] (1979)	$mn\,(\log n)^2$	$n^3(\log n)^2$	$n^2(\log n)^2$
Tarjan (?) (1980)	$mn\log n$	$n^3\log n$	$n^2\log n$

が優れていたという．数千の点のネットワークにならないと自分のものの有利さが出ない，と言ってた．

また Galil 氏は，E. Tarjan が，$O(mn\log n)$ の算法を作ったと本人から聞いたと言っている．ということは，ほとんど反復がないような方法か，毎反復の手間がごく減らされたわけで，論文が早く出ることが期待される．

謝辞：この特集のコーディネーター伊理正夫教授には，本稿に対しご意見をいただき改善できたことを感謝いたします．また，Galil 教授など多くの方々から，資料や情報をいただいた．

参考文献

[1] Boldyreff, Alexander W., "Determination of the Maximal Steady State Flow of Traffic through a Railroad Network, "*Opns. Res.* **3** (1955), 443-465.

[2] Cherkasky, B. V., "Algorithm of Construction of Maximal Flow in Networks with Complexity of o ($V^2\sqrt{E}$) Operations, "*Mathematical Methods of Solution of Economic Probleus*, **7** (1977), 117-125 (in Russian).

[3] Dinic, E. A., "Algorithm for the Solution of a Problem of Maximum Flow in a Network with Power Estimation," *Soviet Math. Dokl.*, **11**(1970), 5, 1277-1280.

[4] Edmonds, Jack and Richard Karp, "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems," *J. Asso. Comp. Mach.*, **19**(1973), 2, 248-264.

[5] Even, Shimon, "The Max Flow Algorithm of Dinic and Karzanov, An Exposition," MIT, LCS, TM-80, August 1976.

[6] —— and R. Endre Tarjan, "Network Flow and Testing Graph Commectivity," *SIAM. J. on Computing*, **4** (1975), 507-518.

[7] Ford, L. R. and D. R. Fulkerson, "Maximal Flows through a Network," *I. R. E. Trans. on Information Theory*, **IT-2** (1956), 117-119.

[8] —— and ——, *Flows in Networks*, Princeton Univ. Press, Princeton, New Jersey, 1962.

[9] Galil, Zvi, "A New Algorithm for the Maximal Flow Problem," *Proc. of the 19th IEEE Symposium on Foundations of Computing Science*, Ann-Arbor, Mich., Oct. 1978, 231-245.

Als to appear as "An $O(V^{5/3} E^{2/3})$ Algorithm for the Maximal Flow Problem," in *Acta Informatica*, (1980).

[10] ——, "On the Theoretical Efficiency of Various Network Flow Algorithms," Research Report (# 31506), IBM Research Div., York Heitght, N.J., Sep. 1978.

[11] ——, and Amnon Naamad, "Network Elow and Generalized Path Compression," *Proceedings 11th Annual ACM Symposium on Theory of Comput. System Sci.*

[12] Iri, Masao, *Network Flow, Transportation and Scheduling Theory and Algorithms*, Academic Press, New York, 1969.

[13] Karzanov, A.V., "Determining the Maximal Flow in a Network by the Method of Preflow," *Soviet Math. Dokl.*, **15**(1974), 2, 434-437.

[14] Malhotra, V.M., M. Pramodh Kumar, and S.N. Maheshwari, "An $O(|V|^3)$ Algorithm for Finding Maximum Flow in Network," *Information Processing Letters*, **7** (1978), 6, 277-279.

[15] Megiddo, N. and Z. Galil, "On Fulkerson's Conjecture about Consistent Labeling Processes," *Math. of Opns. Res.* **4** (1979), 3, 265-267.

[16] Pohl,Ira, "Improvements to the Dinic-Karzanov Network Flow Algorithm," Information Science, Univ. of California, Santa Cruz. Calif., May 1977.

[17] Tucker, Alan, "A Note on Convergence of the Ford-Fulkerson Flow Algorithm," *Math. of Opns. Res.*, **2** (1977), 2, 143-144.

[18] ネットワーク構造を有するオペレーションズ・リサーチ問題の電算機処理に関する基礎研究, (社)日本オペレーションズ・リサーチ学会, 1973年3月.