特　別　研　究　報　告

一般化最大フロー問題の多項式アルゴリズム

指導教官

藤重　悟 教授

岩田　覚 助教授

報告者

野上　裕介

平成１２年２月２５日

大阪大学 基礎工学部 システム工学科

概要

私たちが，現在生活している社会において，インターネットなどによるネットワーク化が盛んに行われている．そのようなネットワーク社会において，いかに損失を少なくデータなどを送るか，いかに速くデータを安全に送るかといったことが重要な問題になってきている．このような要求に答えるために，どのような経路でデータのやりとりを行えばよいのかといった問題に最適な解を与えてくれるのが，一般化最大フロー問題である．そのため，一般化最大フロー問題の重要性がますます大きくなってきている．

一般化最大フロー問題とは，ネットワークの始点と終点の間にできるだけ多くの流量 (フロー）を流す問題です．ネットワーク間にはいくつかの点が存在し，それらは枝で結ばれている．枝にはフローを流すことができる限界を示す枝容量や，流れたフローを大きくしたり，小さくしたりするゲインが存在する．点には超過が存在する．フローを流す前の超過を初期超過という．これらのファクタをうまく使って，一般化最大フロー問題を解くアルゴリズムが研究されている．このような研究の背景には，数理計画を研究している人たちが，現在のようなネットワーク社会を意識し，より計算時間の少ないアルゴリズムの開発に力を注いできた結果がある．そのようなアルゴリズムは現在の社会において，とても有益なものとなっている．

本報告では，一般化最大フロー問題を高速に解くアルゴリズムの開発に関する最新の成果をまとめるとともに，手数料を含む一般化最大フロー問題及び凹関数のゲインを持つ一般化最大フロー問題に関する問題提起，モデル化を行い，今後の研究の第一歩としたい．

目 次

1　序論

現在，私たちが生活する社会において，様々なネットワークに遭遇する．たとえばインターネット，飛行ルート，銀行業務などが，そういったものである．そのネットワーク内でより高速化されたデータ交換をめざすのが一般化最大フロー問題といえるであろう．もし，一般化最大フロー問題が定義されていなくて，各ホスト間（この論文の場合は各ノード間になる.）の通信速度のようなものが最適な経路でなければ，ネットワーク社会において，多大なロスになるのはもちろん，人命にかかわることさえでてくるだろう．そんな問題を解決するために，現在では，様々なアルゴリズムが作られて役にたっている．今回の論文では，まずそのような，よく使われているアルゴリズムの紹介をしてみようと思う．その前には，一般化最大フロー問題とは何か，ということについて，各変数を定義しながら，一般例まで紹介しようと思う．そしてそのアルゴリズムによる，計算量はどれぐらいになるのか，ということについてもまとめてみた．

そして，最後に自身が考えた，ネットワークにおいて，各道に付加的な要素がかかってきた場合及び凹関数で与えられるゲインを持つ一般化最大フロー問題はどのように考えていけばよいか，ということについても触れておく．今回，この研究を進めるにあたって，主に参考にした論文はÉva Tardos と Kevin Wayne による参考文献 [6] と [9] である．その他の [1],[2],[3],[4],[5],[7],[8] は，これと同様に最後にまとめておく．研究を進めていく過程で出てきた問題などは結論にまとめておいた．それらが，今後の研究課題になるであろう．ではまず一般化最大フロー問題の定義から見ていこうと思う．

2　一般化最大フロー問題の定義

2.1　グラフ，変数の定義

本論文において,使用するグラフ，変数を定義する．まず，一般化ネットワーク（generalized network）$G = (E, V, t, u, \gamma, e)$ を以下の様に定義する：

E: m 本の枝の集合　(図 2.1)

V: n 個の点集合

t: シンクと呼ばれる区別された点

$u(:E \to \mathbf{R}_+)$: 容量関数で各枝に非負の実数を与える

$\gamma(:E \to \mathbf{R}_{++})$: ゲイン関数

$e(:V \to \mathbf{R}_+)$: 初期超過関数　(図 2.3)

である．ただし，$\mathbf{R}_+$ は非負実数の全体，$\mathbf{R}_{++}$ は正実数の全体を意味する．

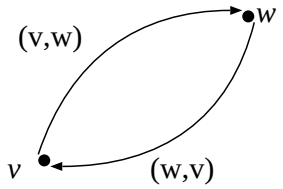

図 2.1: 点と枝

容量 $u(v, w)$ は点 v と w を結ぶ枝の容量を表す．これはその枝に流すことのできるフローの上限を表している．つまり容量制約のことである．$f(v, w)$ を枝 (v, w) に流れるフローとすると，$f(v, w)$ は

容量制約条件:

$$\forall (v, w) \in E : f(v, w) \leq u(v, w) \tag{2.1}$$

を満たさなければならない．さらに，フロー f は歪対称性条件をみたす．つまり，

$$\forall (v, w) \in E : f(v, w) = -f(w, v). \tag{2.2}$$

ゲインは $\gamma(v, w)$ で表されて，点 v から出ていくフローは $\gamma(v, w)$ 倍だけされて点 w に入

る．またゲインが高々1であるようなネットワークは損失ネットワークと呼ばれる．そしてゲインには対称性条件

$$\gamma(v,w) = \frac{1}{\gamma(w,v)} \tag{2.3}$$

を満たすと仮定する．今回の論文では，任意の枝 (v,w) に対して，反対向きの枝 (w,v) が必ず存在し，そのゲインは上記のようになると仮定する（図 2.2）．

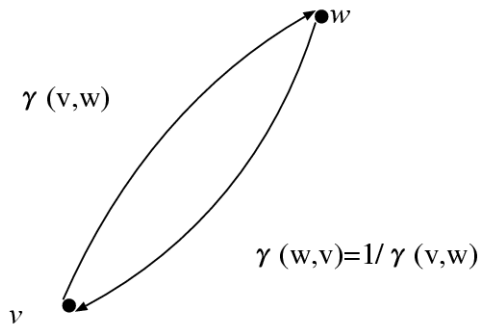

図 2.2: 対称制約

初期超過は各点 v での超過であり，$e(v)$ で表される．

次に実際に流れるフローに関してであるが，$f(v,w)$ を枝 (v,w) に流れるフローとしたが，これからは一般化擬似フロー（generalized pseudoflow) を定義して考えていく．この関数は g で $g : E \to \mathbf{R}$ となる．これは容量制約 (2.1) を満たしていて，歪対称性制約

$$\forall (v,w) \in E : g(v,w) = -\gamma(w,v)g(w,v) \tag{2.4}$$

を満たすとする．以下の図はこの簡単な例を示している (図 2.4)．

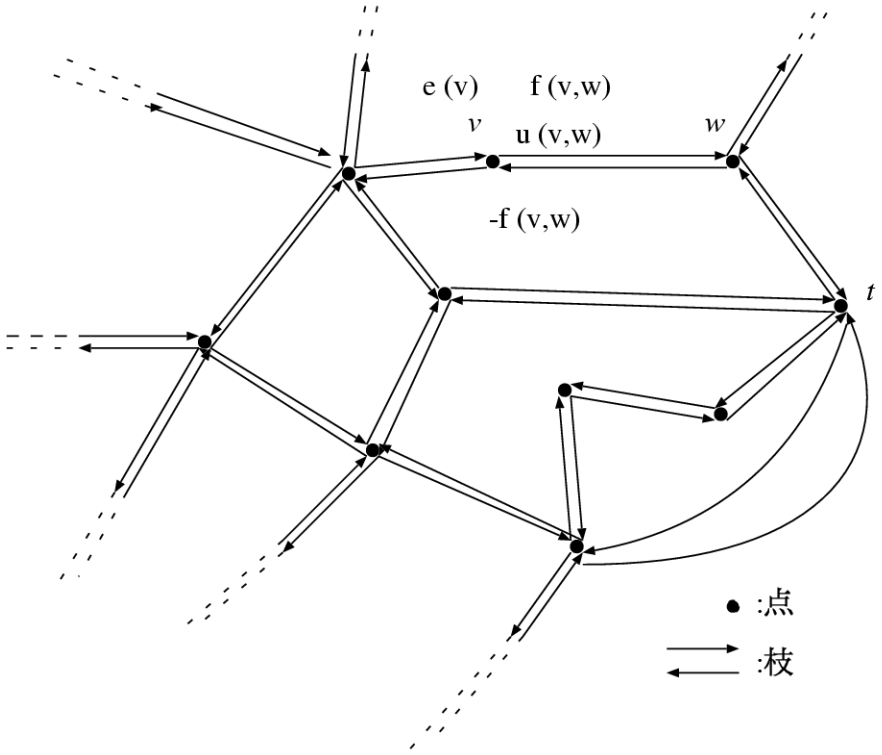

図 2.3: 一般化ネットワーク $G = (V, E, t, u, \gamma, e)$

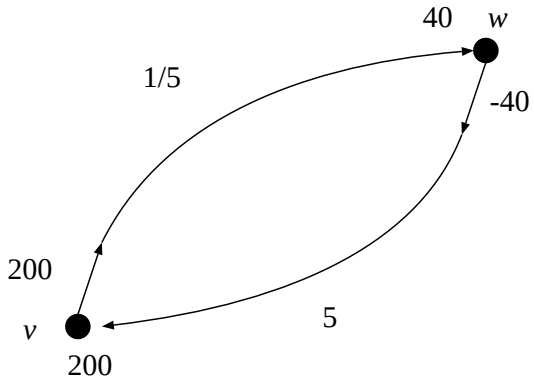

図 2.4: 歪対称性条件の例

図 2.4 において $g(v,w) = 200,\ \gamma(w,v) = 5,\ g(w,v) = -40$ であるから，(2.5) を満たしていることは，すぐに分かるだろう．

さらに後のアルゴリズムの説明で使用される，残余ネットワーク，再ラベルネットワークも定義しておく．一般化残余ネットワーク（generalized residual network) において，擬似フロー g が流れた後に残る残余超過 (residual exess) の定義は

$$e_g(v) = e(v) - \sum_{(v,w)\in E} g(v,w) \tag{2.5}$$

となる (図 2.5)．そして $e_g(v) \geq 0$ であるような一般化疑似フローを一般化フローとする．残余ネットワークは $G_g = (V, E, s, u_g, \gamma, e_g)$ と定義される．u_g は残余容量で

$$u_g(v,w) = u(v,w) - g(v,w) \tag{2.6}$$

と定義される．また $u_g(v,w) > 0$ であるような枝の集合を E_g とする．

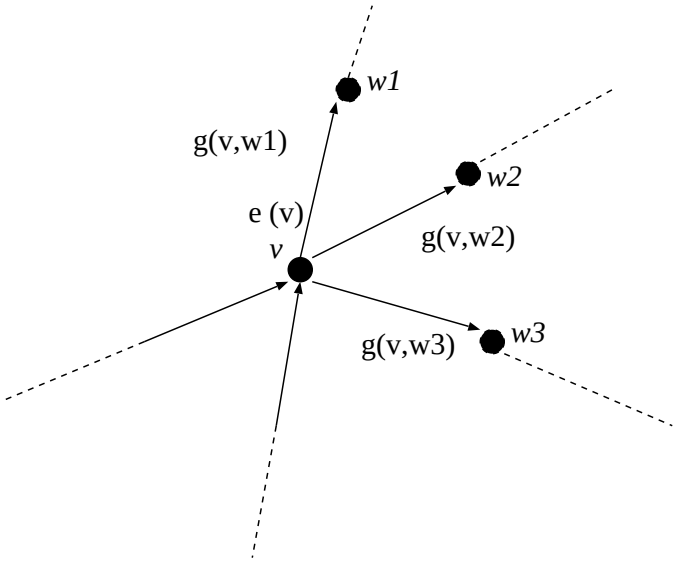

図 2.5: 残余超過

続いて再ラベルネットワーク (relabeled network) $G_\mu = (V, E, t, u_\mu, \gamma_\mu, e_\mu)$ を定義する．ここで u_μ,γ_μ,e_μ はそれぞれ，再ラベル容量（relabeled capacity), 再ラベルゲイン（relabeled gain), 再ラベル初期超過 (relabeled initial excess)，μ は再ラベル関数 (relabeling function) である．それぞれの定義は $u_\mu(v,w) = \frac{u(v,w)}{\mu(v)}$, $\gamma_\mu(v,w) = \frac{\gamma(v,w)\mu(v)}{\mu(w)}$, $e_\mu(v) = \frac{e(v)}{\mu(v)}$, $\mu \to \mathbf{R}_{++}$ と定義される．また再ラベルされた残余ネットワークは $G_{g,\mu} = (V, E, t, u_{g,\mu}, \gamma_\mu, e_{g,\mu})$ で表され，$u_{g,\mu}$ は再ラベル残余容量（relabeled residual capacity) で $u_{g,\mu}(v,w) = \frac{u_g(v)}{\mu(v)}$, $e_{g,\mu}$ は再ラベル残余超過で，$e_{g,\mu}(v) = \frac{e_g(v)}{\mu(v)}$ と定義される．

この論文では標準的なラベルとして，v からシンクへの道の最高ゲインの逆数として定義する．フロー生成閉路がなければ, $c(v,w) = -\log \gamma(v,w)$ として，これを枝 (v,w) の長さとして最短路を計算することによって，標準的ラベルは求まる．

次に $OPT(G)$ とはネットワーク G 中でのフローの最大値と定義する．またネットワーク中のフロー g の値は $|g| = e_g(t)$ (シンクでの残余超過) とする．また ξ- 最適とは，$|g| \geq (1-\xi)OPT(G)$ をみたすフローのことである．

フロー生成閉路は図 2.6 に示すように，$\prod_{e \in C} \gamma(e) > 1$ であるような閉路である (ここで，$e \in C$ はサイクルに含まれる枝の意)．

2.2 一般化最大フロー問題

本論文で考察する一般化最大フロー問題の目的は，容量条件を満たしながら，シンク t へできる限り多くの流量のフローを流し込むことである．

もとのネットワークで問題を解くことと，G_g(residual network) で問題を解くことは等しいということが，次の補題で証明される．

補題 **2.1**: g はグラフ G での一般化フロー, g' はグラフ G_g でのフローとする．すると，

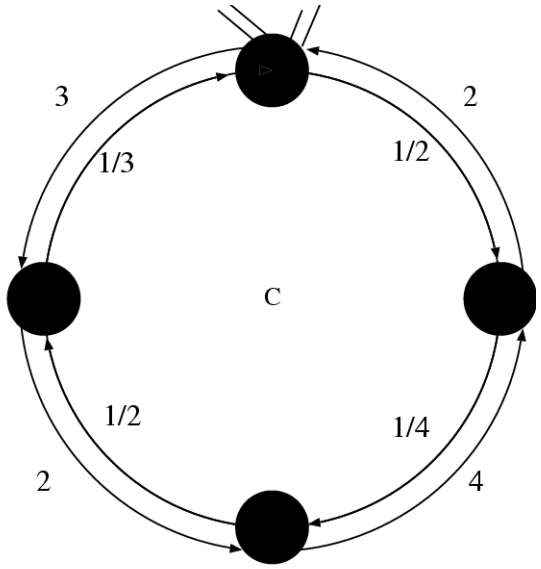

図 2.6: フロージェネレイティングサイクル

$OPT(G) = |g| + OPT(G_g)$ となる．よって g' が G_g において最大フローになる必要十分条件は $g + g'$ が G において最大フローになることである．

また再ラベルネットワークに関しても，同じことが言える．

次に一般化フロー g が最大フローであるかどうか，判定するための条件について述べる．

最適性条件 **(Optimality Conditions)**

定理 2.1: ネットワーク G においてフロー g が最適であるための必要十分条件は，G_g において，増加道が存在しないことである．

ここで，増加道 (generalized augmenting path) とは，残余超過（residual excess）をもった点からシンクへの residual path のことである．また増加道は，残余的なフロー生成閉路の

ことであり，以下これを G.A.P と略記する．上の定理は，Onaga [4] によって証明されている．

つまり，増加道があると，その点の残余超過がまだ流すことができたり，ゲインでフローがまだ増加させることができる余地があるので， 現在のフローは最適ではないということになる．

さて，問題を解くためのアルゴリズムを考える前に，前準備をしておく．それは初期フローである程度，理想値に近いフローをみつける．もう一つはフロー生成閉路を消しておくことである．まず，理想値に近いフローをみつけることであるが，これは Radzik[5] のアルゴリズムによって行う．反対向きの枝を使わずに，最大ゲインの道にフローを流すと，$OPT(G) \leq \Delta_0 \leq nOPT(G)$ となる，パラメタ Δ_0 を見つける．次にフロー生成閉路を消すことが，Goldberg-Plotkin-Tardos [1], Goldberg-Tarjan[2] の cancel-tighten アルゴリズムによって行われる．

2.3 一般化最大フローの応用例

以下では一般化最大フロー問題が実社会でどのように使われているのか，いくつかの例をあげて紹介する．

例：通貨の換金

現在海外への旅行，仕事などで，出かける人々が多くなり，日本の円だけでなくさまざまな国の通貨が使用されるようになってきた．また銀行などで貯金する際にドルで貯金したりもする．それは換金する時に 2 つの通貨間での独自のレートが存在するためで，より有効なレートで換金したいときに，この最大フロー問題が使われる．簡単な例を図 2.7 を使って考えてみよう．

まず，手元に 1,000US ドルあったとしよう．そして目的は，フランに換金することであ

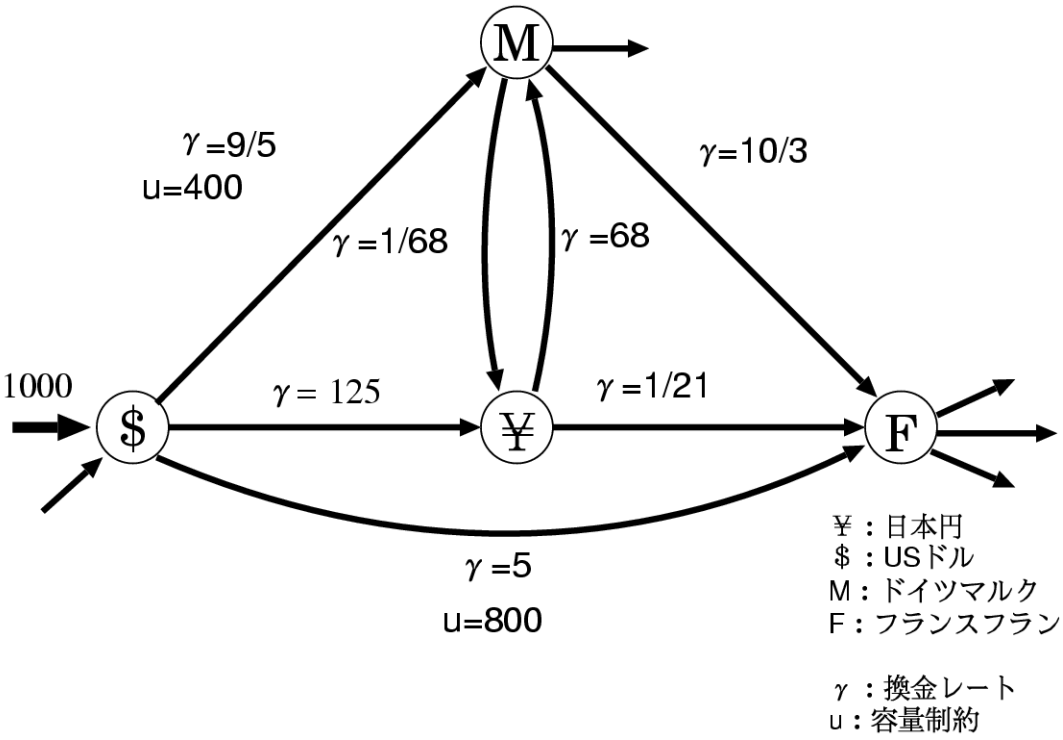

図 2.7: 換金問題

る．直接ドルからフランに換金するとレートは，1ドル5フランになる．しかし，ドイツマルクを経由してフランにすると γ=6 という最大のレートを得ることになる．今の例は簡単ではあるが，通貨の種類が多くなったときに，効果を発揮する．

最大フロー問題を解くことで便利になる, 以下に他の応用例を表 2.1 にまとめる．

3　アルゴリズム

この章では，実際に一般化最大フロー問題を解くためのいくつかのアルゴリズムの説明を行う．

3.1　ゲインスケーリング

この節ではゲインスケーリングについて説明する．ゲインスケーリングとはゲインの丸め (rounding) とスケーリングにより行われる．これをなぜ行うのかというと，一般化フロー

表 2.1: 応用例

ネットワーク	ノード（点）	枝	フロー
会話	コンピュータ，衛星 電話交換所	ケーブル，m.r.l 光ファイバー	ビデオ，データ 音声メッセージ
水力学	貯水池，湖 ポンプ所	パイプ	液体燃料，水 ガス，石油
金融	通貨，貯蓄	取引，処理	お金
運輸	空港，駅 交差点	高速道路，線路 飛行ルート	車，乗客 飛行機

を計算する際に行う様々な計算（例えば，フロー生成閉路を消したりすること）の計算の計算量を改善することができるからである．その結果，丸めによる近似により，計算誤差が生じるが，ゲインスケーリングのアルゴリズムを反復することで，より精度のよい近似を得ることができる．

3.1.1 ゲインの丸め

この論文において，ゲインを丸めるというと，ゲインを，底 $b = (1+\xi)^{\frac{1}{n}}$ に丸め落とすということである（このネットワークを ξ-丸めネットワークとする）．この丸めの考え方は損失ネットワークに適応させる．具体的には，残余的な枝のゲインを $\bar{\gamma}(v,w) = b^{c(v,w)}$ に丸め落とす．すなわち，$c(v,w) = \lfloor \log_b \gamma(v,w) \rfloor$ とする．この丸めを行った後でも，歪対称性条件 $\bar{\gamma}(v,w) = \frac{1}{\bar{\gamma}(v,w)}$ を満たしていて，(v,w),(w,v) がともに残余枝として存在するならば，それぞれのゲインは 1 である (なぜならば，$\bar{\gamma}(v,w) \cdot \bar{\gamma}(w,v) = 1$ となるため)．次に，丸められたネットワークでのフローがどういう意味をもつか考えよう．まず，H を，ξ-丸めネットワークで損失ネットワーク．G と同じ残余枝の集合を持つとしよう．$c = \max_{e \in E} c(e)$ とすると，

$$c \leq 1 + \log_b B = 1 + \frac{\log B}{\log(1+\xi)^{\frac{1}{n}}} \leq 1 + \frac{n \log B}{\xi} \tag{3.7}$$

が成り立つ．

h をネットワーク H を流れるフローとしよう．h のネットワーク G でのフロー g としての解釈は式 (3.8) のようになる．

$$g(v,w) = \begin{cases} h(v,w) & \text{if } g(v,w) \geq 0 \\ -\gamma(w,v)h(w,v) & \text{if } g(v,w) < 0 \end{cases} \tag{3.8}$$

すなわち，フロー g は正のフローをもつ枝上のフロー h と一致する．しかし歪対称性条件を満足するために，それらの反対向きの枝（負のフロー）上の h とは異なる．そしてこの考え方は付加的な超過を生むが，不足になることはない．それは次の図 3.8 でわかるであろう．

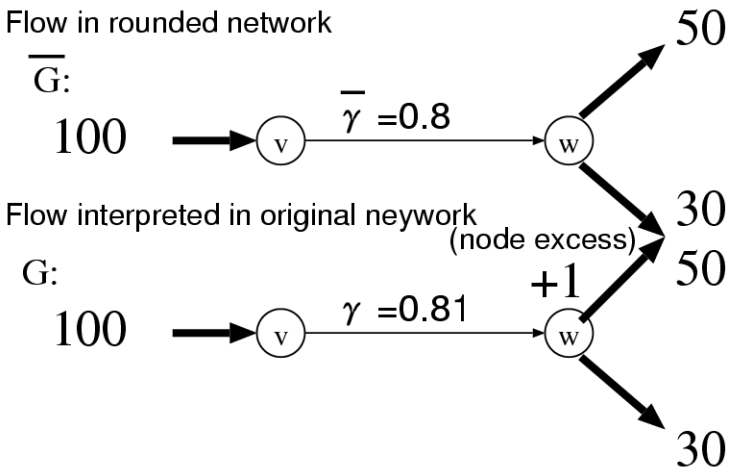

図 3.8: フローの解釈

同時に，残余的なフロー生成閉路をもつネットワークに応用したい時には，最初にフロー生成閉路を消しておかなければならない．それの方法は Goldberg-Tarjan[2] によって述べられている．

次に，丸められたネットワークでの近似フローは元のネットワークでの近似フローを減少させることを証明したい．そうなることで，丸められたネットワークでの近似フローを求めることの有効さが分かる．まず，次の定理 3.1 で丸められたネットワークと元のネットワークは近いことが示されている．

定理 **3.1**: $0 < \xi < 1$ とする．G は損失ネットワーク，H は上記のようにつくられた丸められたネットワークとしよう．すると，$(1-\xi)OPT(G) \leq OPT(H) \leq OPT(G)$ である．

（証明）まず $OPT(H) \leq OPT(G)$ は G における残余枝のゲインを丸めたのが H なので，明らかに成り立つ．次にもう一方の不等式であるが，ネットワークにおいて，増加道を P，そこを流れるフローの量を x_p とし，x^* を G における最適フローとしよう．x^* は H では実行可能である (ここで，G で $\gamma(P)x_p^*$ という量がシンクにつくと，H では $\bar{\gamma}(P)x_p^*$ だけとどく)．すると，$\bar{\gamma}(v,w) = b^{\bar{c}(v,w)}$ だから $\bar{\gamma}(v,w)b = b^{\bar{c}(v,w)+1} \geq b^{\log_b \gamma(v,w)} = \gamma(v,w)$ が成り立つのは明らかである．これを今，道 P で考えると，$\bar{\gamma}(P)b^{|P|} = \gamma(P)$ となる．($|P|$ は $path$ の長さ) よって，$\bar{\gamma}(P) = \frac{\gamma(P)}{b^{|P|}} \geq \frac{\gamma(P)}{b^n} = \frac{\gamma(P)}{1+\xi} \geq \gamma(P)(1-\xi)$ (n は全枝数) が成り立つのは明らか．以上より，$(1-\xi)OPT(G) \leq OPT(H)$ も成り立つ．　（証明終り）

そして，次の系 (3.1) がでる．

系 3.1: G を損失ネットワーク，H を ξ-丸めネットワークとしよう．$0 < \xi < 1$ ならば，H における ξ'-最適フローは G において $(\xi + \xi')$-最適フローと解釈される．

（証明）h を H における ξ'-最適フローとしよう．また g を G におけるフロー h の解釈とする．すると $|g| \geq |h|, |h| \geq (1-\xi')OPT(H)$ は明らかである．また，定理 (3.1) より，$(1-\xi')OPT(H) \geq (1-\xi')(1-\xi)OPT(G)$ となることもわかる．さらに，$(1-\xi')(1-\xi)OPT(G) = \{1-(\xi'+\xi)+\xi'\xi\}OPT(G) \geq (1-\xi'-\xi)OPT(G)$ となる．だから，$|g| \geq |h| \geq (1-\xi')OPT(H) \geq (1-\xi)(1-\xi')OPT(G) \geq (1-\xi-\xi')OPT(G)$ となり，g は $(\xi'+\xi)$-最適フローとなる．　（証明終り）

3.1.2　誤差スケーリング

ここでは，誤差スケーリングという考え方が，どのようにして計算速度を上げているかを説明する．今，ネットワーク G において $\frac{1}{2}$-最適フローをみつけるサブルーチンがあるとしよう．誤差スケーリングによって，このサブルーチンを $\log \frac{1}{\xi}$ 回呼ぶことで，ネットワーク G において ξ-最適フローをみつけることができる．これを行うためには，最初にネットワーク G で $\frac{1}{2}$-最適フロー g をみつけなければならない．そして G_g において，$\frac{1}{2}$-最適フロー h をみつける．すると，$g+h$ はネットワーク G において $\frac{1}{4}$-最適フローになる．一般的に ξ-最適フローをみつけるのに，サブルーチンを $\log \frac{1}{\xi}$ 回呼ばなければならない．次のものは，

誤差スケーリングの devide-and-conquer バージョンです．それは 2 つの $\sqrt{\xi}$-最適フローをくりかえしあわせることで，より速いアルゴリズムになる (補題 (3.1) 参照)．

補題 **3.1**: g をネットワーク G において，$\sqrt{\xi}$-最適フロー，h を G_g における $\sqrt{\xi}$-最適フローとしよう．すると，フロー $g+h$ は G において ξ-最適になる．

（証明）$OPT(G) - |g+h| = OPT(G_g) - |h| \leq \sqrt{\xi}OPT(G_g) = \sqrt{\xi}(OPT(G) - |g|) \leq \xi OPT(G)$ となるので，$OPT(G) - |g+h| \leq \xi OPT(G)$ より $(1-\xi)OPT(G) \leq |g+h|$ となる．よって $|g+h|$ は ξ-最適フローである (ただし，2 つの不等式には $\sqrt{\xi}$-最適フローの定義をつかっている)．（証明終り）

3.2 Truemper のアルゴリズム

この節では Truemper のアルゴリズムについて説明する．このアルゴリズムは一般化最大フロー問題を解くための最も簡単なアルゴリズムの 1 つであり，増加道アルゴリズムに基づいている．そして，ゲインスケーリングの技術をこのアルゴリズムに適用することによって，最も簡潔な多項式時間アルゴリズムが構成できる．まず，Truemper[7] のアルゴリズムを紹介し，その次にそのアルゴリズムの変形版を丸められたネットワークで実行させる．これにより $\xi > 0$ であれば，多項式時間で ξ-最適フローをみつけることができる．しかし最適なフローをみつけるためには，指数時間になるので，残余グラフスケーリングの考えを合わせることにより，多項式時間で最適フローをみつけられるようになる．

ゲイン無しの通常のネットワークにおける自然なアルゴリズムは，超過のある点からシンクへの最高ゲインの道を通して送ればよいとわかるであろう．Onaga はフロー生成閉路がなければ，アルゴリズムはこの特性をもつことを証明した．最高ゲインをもつ道は $cost(v,w) = -\log \gamma(v,w)$ の計算をして最短路をみつけることで求まる．Onaga[4] により，標準的なラベルをつけられたネットワークでゲイン 1 の道は，元のネットワークで最高ゲインの道に対応する．Truemper のアルゴリズム [7] は同時に全ての最高ゲインの増加道に沿ってフローを増やすために最大フローの計算に使う．

定理 **3.2**: Truemper のアルゴリズムにおいて最大フローの計算の繰り返し数は，高々（点の数 n）+（元のネットワークでの異なるゲインの道の数）である．

Input : no-gain network G,error parameter $0 < \xi < 1$

Output : ξ-optimal flow g

Set base $b = (1+\xi)^{\frac{1}{n}}$ and round gains in G to powers of b.

Let H be resulting network.

Initialize $h \leftarrow 0$

while $\exists$ augmenting path in H_h **do**

$\mu \leftarrow$ canonical labels in H_h

$f \leftarrow$ max flow excess nodes to the sink in $H_{h,\mu}$ using only gain one relabeled residual arcs

$h(v,w) \leftarrow h(v,w) + f(v,w)\mu(v)$

end while

$g \leftarrow$ interpretation of flow h in G

図 3.9: RT(G, ξ)

（証明）それぞれの最大フローの計算のあと，$\mu(v)$ はそれぞれの超過点 $v \neq t$ に対して厳密に増加する． （証明終り）

3.2.1 Rounded Truemper (RT)

アルゴリズム RT は丸められたネットワーク上で Truemper のアルゴリズムを実走させることにより，ξ-最適フローを計算する．入力はゲイン無しのネットワーク G と誤差パラメタ ξ である．H は丸められたネットワークとしよう．RT は H の中で最適フローを計算し，そのフローを元のネットワーク上でのものに解釈して出力する．図 3.9 にアルゴリズム RT をのせる．

計算時間の考察は 4 章で行う．

3.2.2 Iterative Rounded Truemper (IRT)

RT は，$\xi = B^{-m}$ だから，多項式時間で最適フローを計算しないと考えられる．IRT は 3.1 節での誤差スケーリングを使う．IRT は反復して $\frac{1}{2}$ の誤差パラメタと現在の残余ネットワークをもった RT を呼ぶ．RT は残余的なフロー生成閉路を生むので，RT を呼ぶまえに，全ての残余フロー生成閉路をキャンセルしておく（なぜならば，入力はゲイン無しのネットワークのため）．すると，最適フローを計算できる．

3.3 Preflow-push

この節では，Goldberg と Tarjan の論文 [3] で示された preflow-push アルゴリズムを一般化最大フローに適応させる．これは，一般化ネットワークフローには多項式時間で対応できた．Tseng と Bertsekas[8] は，最小費用フロー問題に対応する preflow-push によく似たアルゴリズムをつくった．しかし，これは B^n 以上のくりかえしを必要とする．そこで，丸めの技術を，この preflow-push アルゴリズムに使うことで，どんな定数 $\xi > 0$ に対しても ξ-最適フローを計算することが可能になる．そして，残余グラフをスケーリングすることで，最適フローを多項式時間でみつけられる．

3.3.1 Rounded Preflow Push (RPP)

RPP は費用 $c(v,w) = -\log \gamma(v,w)$ と誤差パラメタ $\epsilon = \frac{1}{n} \log b$ (ただし，$b = (1+\xi)^{\frac{1}{n}}$) として，最小費用フローのアルゴリズムを実行する．アルゴリズムは図 3.10 にのせる．許容枝とは，再ラベルゲインが 1 以上の残余枝のことである．許容グラフとは許容枝を含んだグラフのことです．活性点 (active node) とは，正の残余超過とシンクへの残余的な道をもつ点のことである．もし残余的な道がなくて，最適な解がこの点を通ってフローを送るならば，そのフローはシンクにとどかない．よって，この不必要な残余超過を無視できる．RPP はフロー h と点ラベル μ をもっていて，繰り返し活性点 v を選ぶ．その点が許容枝をもてば，IPP は v から w へ $\delta = u_h(v,w)$ ならば “飽和”，そうでなければ “非飽和” と呼ばれる．もし許容枝がなければ $2^\epsilon = b^{\frac{1}{n}}$ で点 v のラベルを増加させる．この過程が再ラベルである．再ラベルされた点 v は v から出る許容枝を生ずる．RPP の入力はゲイン無しのネッ

Input : no-gain network G, error parameter $0 < \xi < 1$

Output : ξ-optimal flow g

Set base $b = (1+\xi)^{\frac{1}{n}}$ and round gains in network G to powers of b.

Let H be the resulting network.

Initialize $h \leftarrow 0, \mu \leftarrow 1$

while $\exists$ active node v **do**

  $\exists$ admissible arc (v, w) **then**

    Push $\delta = \min\{e_h(v), u_h(v, w)\}$ units of flow from v to w and update h {push}

  else

    $\mu(v) \leftarrow b^{\frac{1}{n}}\mu(v)$ {relabeled}

  end if

end while

$g \leftarrow$interpretation of flow h in G

図 3.10: RPP(G, ξ)

トワーク G と，誤差パラメタ ξ である．このアルゴリズムを適用する前に，前章でのべたように，ゲインを $b = (1+\xi)^{\frac{1}{n}}$ の整数乗に丸める．この RPP は丸められたネットワーク H に適用される．

3.3.2 Iterative Rounded Preflow Push (IRPP)

RPP は多項式時間で最適フローを計算しない．そのため IRT のようなアルゴリズムに似た IRPP によって多項式時間で最適フローを計算できるようになる．

3.4 Rounded Fat Path

この節では Radzik[5] による Fat-Path の変形と同様の計算量をもつ．Goldberg-Plotkin-Tardos[1] の Fat-Path アルゴリズムの簡単な変形を示す．Fat-Path のアルゴリズムは最初に残余的なフロー生成閉路を消すことから始まる．Radzik のアルゴリズムは閉路をキャンセルすると，解の正確さの分析が困難になるため，ここでは丸めの技術をつかって，ゲイン因子を小さくする．まず最初に，十分に大きな容量をもつ増加道をみつける Fat Augmentations のサブルーチンを説明する．そして，丸められたネットワークで計算量を改善するために実行する Fat-Path アルゴリズム RFP を紹介する．さらに，多項式時間で最適解をみつけるための RFP の繰り返し版も紹介する．

3.4.1 Fat Augmentations

このアルゴリズムは Goldberg, Plotkin, Tardos によって，彼らの Fat-Path アルゴリズムのために作られたサブルーチンである．詳細は文献 [1] で述べられている．入力はゲイン無しのネットワークと fatness パラメタ δ である．サブルーチンは，くりかえし最高ゲインの δ-Fat 道に沿ってフローを増やす．すなわち，道の最初の点で十分な超過があってシンクでの超過を δ だけ増やすための十分な残余容量をもつ道の間の最高ゲインをもつ増加道のことである．このプロセスは δ-Fat-Path がなくなるまで繰り返される．高々 $\frac{n+OPT(G)}{\delta}$ 回の増加があり，最終的に最後のフローは，少なくとも $OPT(G) - m\delta$ という流量をもつ．

3.4.2 Rounded Fat Path (RFP)

RFP のアルゴリズムは，丸められたネットワーク上で元々の Fat-Path アルゴリズムを実行する．元々の Fat-Path のアルゴリズムは Fat Augmentations を呼ぶことと，δ-Fat-Path に沿うフローをそのような道がなくなるまで増加させることである．RFP のアルゴリズムの入力は，ゲイン無しのネットワーク G と誤差パラメタ ξ である．まず，ゲインを丸めて，そのネットワークを H，そこでのフローを h とする．Δ は超過食い違い量の上限で，フロー $|h|$ と $OPT(H)$ の間の差である．δ-Fat Path がなくなると，δ は $\frac{1}{2}$ に減らされて，Δ も $\frac{1}{2}$ に減少し，新しいフェーズにいく．この減法により RFP は Cancel Cycles で H_h において残余的なフロー生成閉路を消す．Fat Augmentations のサブルーチンは Fatness パラメ

Input : no-gain network G,error parameter $0 < \xi < 1$

Output : 2ξ-optimal flow g

Set base $b = (1+\xi)^{\frac{1}{n}}$ and round gains in network G to powers of b.

Let H be the resulting network.

Initialize $\Delta \leftarrow \Delta_0$ and $h \leftarrow 0$

repeat

$(h', \mu) \leftarrow$ CANCEL CYCLES (H_h)

$h \leftarrow h + h'$

$h' \leftarrow$ FAT AUGMENTATIONS $(H_{h,\mu}, \frac{\Delta}{(2m)})$

$h \leftarrow h + h'$

$\Delta \leftarrow \frac{\Delta}{2}$

until $\boldsymbol{\Delta \leq \xi \mathbf{OPT}(\mathbf{H})}$

$g \leftarrow$ interpretation of h in network G

図 3.11: RFP(G, ξ)

タ $\delta = \frac{\Delta}{(2m)}$ で呼ばれて，このあと，超過食い違い量は高々$m\delta = \frac{\Delta}{2}$ となり，Δ は減少する．δ-Fat Augmentation は点の残余超過を 0 にするかもしくは，少なくとも δ だけフローの値を増やすので，Δ-フェーズにつき高々$n + \frac{\Delta}{\delta} = n + 2m$ の増加がある．図 3.11 にアルゴリズムをのせる．

3.4.3 Recursive Rounded Fat Path (RRFP)

RFP は ξ- 最適フローを多項式時間で計算する．しかしながら，最適なフローを計算するためには，もとの Fat Path アルゴリズムより長い時間が必要になる．そこで繰り返しを使うことで元のものより速く，近似最適解，及び最適解を計算できる．RRFP は繰り返しごとに，再びネットワークを丸めて，そこでフロー生成閉路を消す．

4　計算量の評価

この章では，各アルゴリズムの計算量の評価を行う．計算量の定義は O( ) で行うが，この論文では $f \log^{O(1)} m$ を $\tilde{O}(f)$ と定義する．以下では Tardos-Wayne の論文 [6] で定理としてあげられている計算量である．

定理 4.1: $b > 1$ として，もしすべてのコストが整数で $-C$ 以上ならば CANCEL CYCLES のアルゴリズムは $\tilde{O}(mn \log C)$ 時間かかる．

定理 4.2: もしゲインが 1 と B の間の整数の比で与えられれば，CANCEL CYCLES のアルゴリズムは $\tilde{O}(mn^2 \log B)$ 時間かかる．

補題 4.1: B^{-4m}- 最適フローが与えられれば，最適フローは $\tilde{O}(mn^2 \log B)$ 時間で計算できる．

(Truemper の algorithm について)

定理 4.3: RT はゲイン無しのネットワークにおいて，ξ-最適フローを $\tilde{O}(mn^3 \xi^{-1} \log B)$ 時間で計算できる．(これは系 (3.1) をつかって示される．)

定理 4.4: IRT は ξ - 最適フローを $\tilde{O}(mn^3 \log B \log \xi^{-1})$ 時間で計算し，最適フローは $\tilde{O}(m^2 n^3 \log^2 B)$ 時間かかる．

論文 [6] の元版では IRT は最適フローを計算するのに $\tilde{O}(m^2 n^3 \log B + m^2 n^2 \log^2 B)$ 時間かかることが証明されている．

(Preflow – Push について)

定理 4.5: RPP は ξ-最適フローを $\tilde{O}(mn^3 \xi^{-1} \log B)$ 時間で計算する．

定理 4.6: IRPP は ξ- 最適フローを $\tilde{O}(mn^3 \log B \log \xi^{-1})$ 時間で計算し，最適フローは $\tilde{O}(m^2 n^3 \log^2 B)$ 時間かかる．

(Rounded Fat Path について)

定理 **4.7**:
RFPはゲイン無しのネットワークにおいて，2ξ - 最適フローを $\tilde{\mathrm{O}}((m^2+mn\log(\xi^{-1}\log B))\log\xi^{-1})$ 時間で計算する．

定理 **4.8**: RRFPは ξ - 最適フローをゲイン無しのネットワークにおいて，$\tilde{\mathrm{O}}(m(m+n\log\log B)\log\xi^{-1})$ 時間で計算する．もしネットワークが残余的なフロー生成閉路をもてば，さらに $\tilde{\mathrm{O}}(mn^2\log B)$ 時間必要になる（これは，定理 4.2 による）．また，$\tilde{\mathrm{O}}(m^2(m+n\log\log B)\log B)$ 時間で最適フローを計算する．

5 問題の一般化

この章では，一般化最大フロー問題をさらに拡張した問題をいくつか考えていく．

5.1 手数料を含む場合

一般化最大フロー問題をより，一般的に考えるために，ここでは手数料を含んだ場合を考える．現在，銀行などでキャッシュサービスを受ける時に，必要とされる手数料について考える．またそれが，含まれることにより，どのような最大フロー問題に帰着されるか，を考えよう．

5.1.1 問題の設定及びモデル化

まず，簡単な例から考えてみよう．2 点 v,w があり，それを結ぶ枝 $a=(v,w)$ があったとしよう．そして，そこを流れるフローを考える．まず，始点 v に入ってきたフローは $\varphi(v)$ として出ていくとする．すると終点 w に入る時には $\varphi(w)=\gamma(a)\varphi(v)$ となるのは明らかである．(図 5.12) これは全ての枝についていえる．

次に，これに付加的な要素（ここでは，負で手数料になる）が加えられた場合を考える．この関数を $\alpha : E \to \mathbf{R}_+$ とする．$\alpha(a)$ がその枝 a に関する付加量になる．この考えを図 5.12 と合わせて考えて，定式化してみると，始点 v のフロー $\varphi(v)$ が，$\gamma(a)\varphi(v)-\alpha(a)$ と

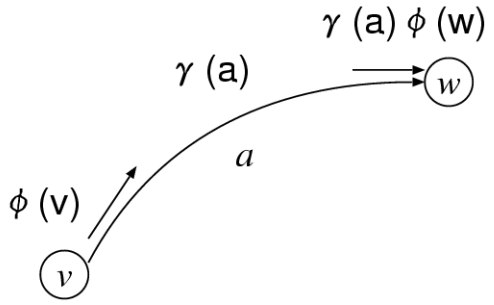

図 5.12: 枝の始点と終点でのフロー

なって，終点 w にとどく（図 5.13）．

$$\varphi(w) = \gamma(a)\varphi(v) - \alpha(a) \tag{5.9}$$

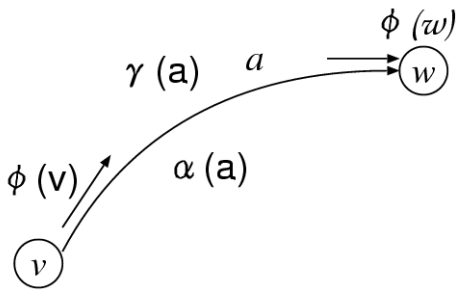

図 5.13: 枝間で手数料が加わった場合．

ただし，この場合，$\gamma(a)\varphi(v) \leq \alpha(a)$ となる時は，手数料がフローの量を上回っているので徴収できないことになるので $\gamma(a)\varphi(v) > \alpha(a)$ になるまで 0 とする（図 5.14）．

今までは 1 本の枝で考えてきたが，今度は道で考える．図 5.15 の例で考えてみよう．各枝に示したパラメタはゲインで数字が手数料とする．x は最初のフローとする．するとシンクへ最終的に流れ込む量は

$$\begin{aligned} &\{(ax-3)b-4\}c-2 \\ &= abcx - 3bc - 4c - 2 \end{aligned}$$

となる．つまり，最初のフローには全てのゲインがかかり，そして 1 本目の枝の手数料に 2 本目の枝からのゲインがかかる．それ以降の手数料も次の枝からのゲインがかかる．こ

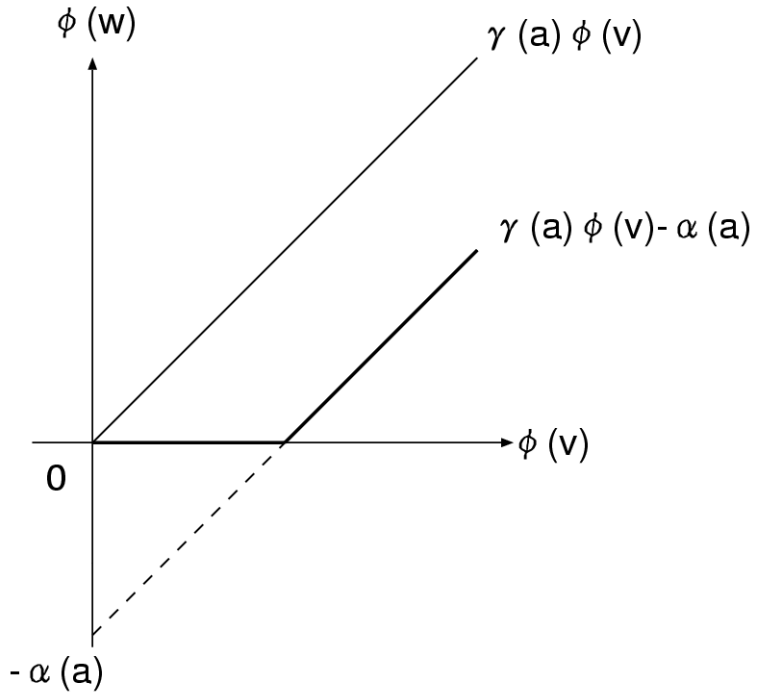

図 5.14: $\gamma(a)\varphi(v) \leq \alpha(a)$ の時の扱い.

れを，一般形で書いてみると，小文字のパラメタがゲイン，大文字が手数量とする．φ は最初のフローの量とする．

$$(abc\cdots xyz)\varphi - (abc\cdots xy)A - (abc\cdots x)B - \cdots (abc)W - (ab)X - (a)Y - Z \quad (5.10)$$

ただし，道は a から z の 26 本で構成されているとした．

結局，道を通しての量が（5.10）によって分かった．これを残余グラフをつかって何本かの道で最大にすればよい，と考える．今の場合容量は無視したが，入れると制約条件がふえる．

5.2 ゲインが凹関数で与えられる場合の最大フロー問題

この章では，ゲインが凹関数で与えられる場合について考える．これは，流量によってゲインが上に凸な形に変化するということである (図 5.16)．$\varphi(w)$, $\varphi(v)$ はそれぞれ枝 (v, w)

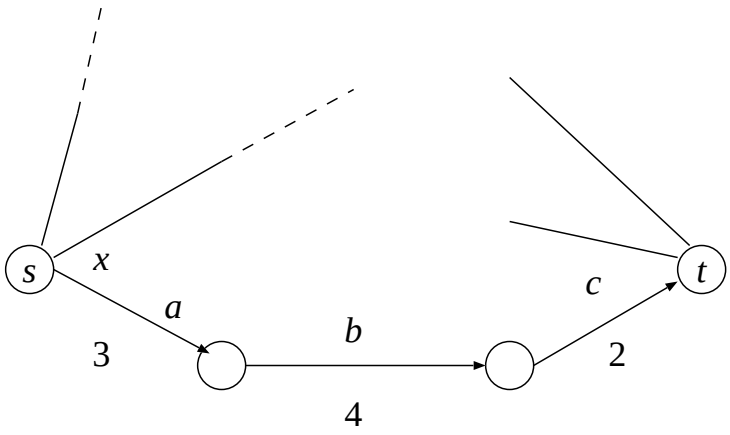

図 5.15: 道で考えた，手数料

の終点に入る流量，始点から出る流量とする．0 から a まではゲイン γ_1 で流れる．a から b まで γ_2 で流れる（流量としては $b-a$）．γ_3 も同様である．

次に，このような枝をどう考えるかであるが，1 本で考えるのではなく，そのゲインの異なる傾きの数の本数の枝を考える (図 5.17)．v から w への枝が γ_1 から γ_3 まで 3 本ある．それぞれには γ_1, γ_2, γ_3 のゲインがかかっている．容量はそれぞれ a, $b-a$, $c-b$ になる．このように考えると，ゲインの大きい枝から順番に容量制約を満たしながらフローを流すと，最大フローが得られることが分かる．

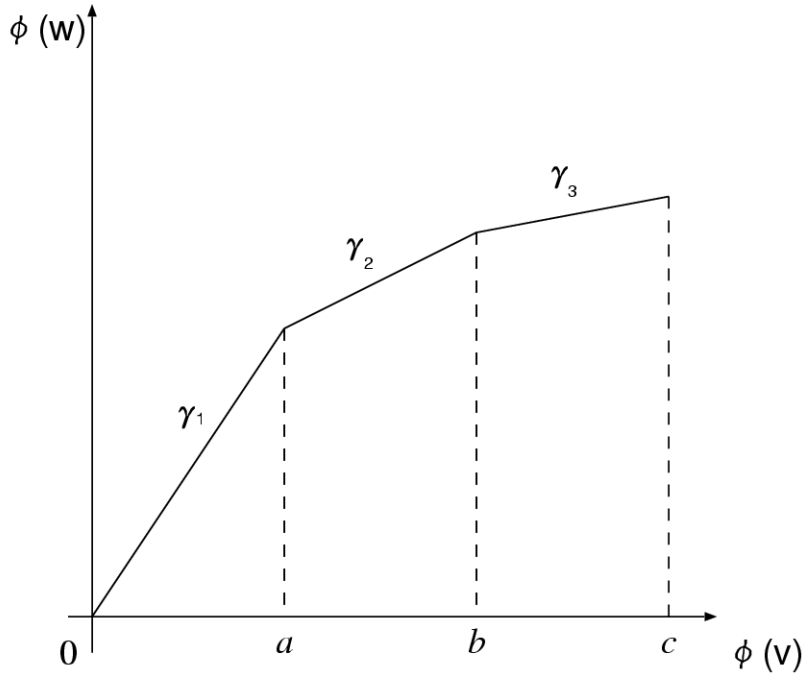

図 5.16: 凹なゲイン

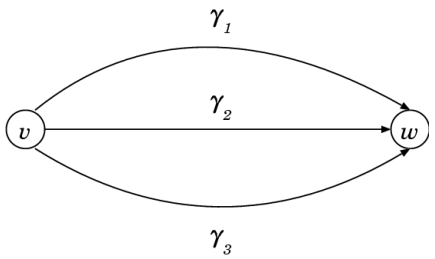

図 5.17: 凹なゲインを持つ枝の等価な取り扱い方.

6　結論

今現在の社会における一般化最大フロー問題の重要性が，解法に関するアルゴリズムのさらなる，多様化，高速化がもとめられてくる．Truemper のアルゴリズム，Preflow-Push のアルゴリズム Fat Path のアルゴリズムを基本として考えていきたい．

さらに，私が今回，提示した手数料を含む一般化最大フロー問題及び凹関数で与えられるゲインを持つ最大フロー問題に関しても今の段階では問題の設定，モデル化までで終わっているが，これからの研究課題として，これを実際に解くアルゴリズムの開発になってくると思う．さらに，もっと社会的に役にたつような問題の研究もあげられるだろう．

謝辞

今回この論文を作成するにあたって，御指導頂きました藤重悟教授に厚く感謝の意を表します．また本研究全般にわたってお世話していただいた岩田覚助教授にも深く感謝いたします．最後に論文作成にあたって，さまざまな面で手助けしていただきました，佐藤さん，新さん，都築さんに感謝します．ありがとうございました．

参考文献

[1] A. V. Goldberg, S. A. Plotkin, and É. Tardos: Combinatorial algorithms for the generalized circulation problem, *Mathematics of Operation Research* **16** (1991) 351–379.

[2] A. V. Goldberg and R. E. Tajan: Finding minimum-cost circulations by canceling negative cycles, *Journal of the ACM* **36** (1989) 388–397.

[3] A. V. Goldberg and R. E. Tarjan: Solving minimum cost flow problems by successive approximation, *Mathematics of Operations Research* **15** (1990) 430–466.

[4] K. Onaga: Dynamic programming of optimum flows in lossy communication nets,*IEEE Trans. Circuit Theory* **13** (1966) 308–327.

[5] T. Radzik: Faster algorithms for the generalized network flow problem, *Mathematics of Operations Research, to appear* (1993).

[6] É. Tardos and K. D. Wayne: Simple maximum flow algorithms in lossy networks, *LNCS* **1412** Springer-Verlag, (1998) 310–324.

[7] K. Truemper: On max flows with gains and pure min-cost flows, *SIAM J. Appl.Math.* **32** (1977) 450-456.

[8] P. Tseng and D. P. Bertsekas: An ϵ-relaxation method for separable convex cost generalized network flow problems, *LNCS* **1084** Springer-Verlag, 85–93.

[9] K. D. Wayne: Generalized maximum flow algorithms(1999).