数理解析研究所講究録
1079巻 1999年 205-214

ファジィネットワーク上のフロー問題
Flow Problems in Fuzzy Network

島田 文彦　　石井 博昭
Fumihiko Shimada　　Hiroaki Ishii

大阪大学大学院工学研究科

1 はじめに

「ネットワークにおける物資の輸送量の調節に関する問題」は典型的なネットワークの問題であり、輸送量の調整の方向によって多くの種類が存在する。それぞれに対してこれまでに様々な解法が考えられてきたが、その内の多くはネットワーク上のフローに対する目的を決める要因として、各アーク、点を通過するフローの量などの数値化が容易な物のみが考えられてきた。しかし、実際にはそれらに加えて、輸送ルートの安全性や通過地点の優先順位など、容易には数値化が可能でない物も多く存在する。加えて、ルートや流量を決める要因が複数になると、それらを全て含んだモデルを構築し、完全に満たす解を求めるアルゴリズムを求めるのは非常に困難となる。このような問題を出来るだけ効率よく解決できるように考えるのが、ここで提案する、ファジィグラフを用いる方法である。

まず、問題内で数値化が容易である要因から重要であると思われるもの、ここではアークを流れる物資の量、及びそのアークでのフローの上限・下限を選び、通常のネットワーク問題での定式化と同様の操作を行う。次に、残りの全ての要因に対して意思決定者が総合的な判断を下し、「総合的な満足度」と言う値を当てはめる。すなわち、複雑になる多目的問題を、この二値に対する非劣解を求める二目的問題に変換する。そして、前者の値をネットワークの各弧に対する値として捉え、後者を各々の存在可能性に対応させる。ここで、ファジィグラフにおいて、ある2点の弧の存在可能性が0であると言うことは、その2点が接続されていないと考える。従って、満足度 μ は $0<\mu\leq 1$ の値を取ることとする。

ネットワークフローに関する問題としては、最大フロー問題と、それを基とした様々な形の応用が存在する。今回は、最大フロー問題と、応用としてのシェアリング問題、及び最少費用流問題について、ファジィネットワークを用いた二目的問題へ拡張し、その説明を行う。フロー問題の応用となる物は、その多くがアルゴリズムの上で最大フロー問題の解を初期実行可能解として持つこととなる為、まず、最大フロー問題の二目的問題への拡張について説明する。拡張の結果、問題は二つの目的を持つこととなるが、これらの目的は多くの場合同時に最適化させることが不可能である為、実際にその解法を考える場合に、目的の内まずどちらを優先させるかによって、二種類のパターンが存在する。一方は、フローを流すアークに対比 s、その存在可能性の下限の最大化を優先させる方法で、これは始めに存在可能性が最大のアークのみを用いた部分グラフを作成し、そのグラフに対して従来の方法で解を求め、その後最初の条件を緩めながらもう一つの目的を満たすように解を更新していくものである。もう一方は逆に従来の目的を優先する方法で、

初めにアークの存在可能性を無視して通常の一目的問題として解き、その後、存在可能性が小さいアークから順にフローを流さないようにしていく、と言う方法である。

今回の最大フロー問題においては、主に前者の方法を用いることとする、これは、フローの下限が設定されていない場合、つまり全てのアークでフローの下限が0である場合には、アーク追加する前の最適解を追加後の実行可能解として用いることが可能となり、計算が効率的になるからである。つまり、ファジィネットワーク上のフロー問題を通常の一目的問題の組合せとして考えた場合、存在可能性がある一定以上のアークからなる部分グラフ（α レベルグラフ）をネットワーク内の異なる存在可能性の数だけ作成し、その各グラフに対して解を求めると言う形になるが、ここでアルゴリズムの効率を考えると、それぞれを完全に独立した問題として解くよりは、一つのグラフでの解が他のグラフでも利用できた方が良い。フロー問題の場合には、定義より、アークの存在可能性のしきい値が大きいグラフの解が小さい方のグラフに流量できるからである。

このようにして求められた複数の解の内、非劣解のみを選び、求める解とする。非劣解の中から最適な解を選択するのは、意思決定者に一任される。ここでは、ある解ベクトル（第一目標の評価値、第二目標の評価値）に関して、その二つの要素において良い値を取るような解が存在しない時に限り、それを非劣解と呼ぶ。

2 下限条件の存在しない二目的最大フロー問題

まず、下限条件が存在しない、つまり、全てのアークにおいてフローの下限が0になっている問題についての解法を説明する。これは、「全てのアークにおいて流量が0」の状態が初期実行可能解となる為、後述する下限条件のある問題より容易に解くことが可能である。

2.1 定式化

ノード集合 $V=\{1,2,\ldots,n\}$ （中に供給点 s 、需要点 t を含む）、アーク集合 $A=\{(i,j)\mid i,j\in V\}$ で表されるグラフを $G=\{V,A\}$ とする。この時、グラフの各アーク (i,j) に、容量 $c(i,j)$ と存在可能性 $\mu(i,j)$ が割り当てる。

ここで、ネットワーク $N=\{V,A,c,\mu\}$ にフローを流すことを考える。アーク (i,j) におけるフローの量を $f(i,j)$ とした時、定義より次のような条件を満たしている必要がある：

$$\begin{array}{ll} 0\le f(i,j)\le c(i,j) & \text{For all}\quad (i,j)\in A \\ \sum_{i\in\alpha(j)} f(i,j)=\sum_{l\in\beta(j)} f(i,j) & \text{For all}\quad j\in V-\{s,t\} \end{array}$$

但し、$\alpha(j)$ を、j を終点とするアークの始点の集合、$\beta(j)$ を、j を始点とするアークの終点の集合とする。

これより、二目的最大フロー問題 **BMFP** を提案する。

BMFP:

$$\begin{array}{ll} \text{Minimize} & \sum_{i\in\alpha(t)} c(i,j)f(i,j) \\ \text{Maximize} & \min_{(i,j)\in A_F} \mu(i,j) \\ \text{s.t.} & 0\le f(i,j)\le c(i,j) \\ & \qquad \text{For All}\quad (i,j)\in A \\ & \sum_{i\in\alpha(j)} f(i,j)=\sum_{l\in\beta(j)} f(i,j) \\ & \qquad \text{For all}\quad j\in V-\{s,t\} \end{array}$$

但し、$f(i,j)>0$ であるアーク (i,j) の集合を A_F と置く。

2.2 解法

ここでは、初めに使用するアークの存在可能性の下限の最大化を優先して行い、その後、その解を総フローが最大になる方向に更新していく。

まず、新たに必要となる Notation について説明する。

$\{\mu^l\}$ ：アークの異なる存在可能性の値を降順に並べたリスト。
$1 \geq \mu^0 > \mu^1 > \cdots > \mu^{k-1} > \mu^k = 0$ である。

A^l ：しきい値 μ^l を持つ A のα-レベル集合。
$A^l = \{(i,j) \mid (i,j) \in A, \mu(i,j) \geq \mu^l\}$

G^l ：しきい値 μ^l を持つグラフ。$G^l = \{V, A^l\}$

W^l ：グラフ G^l における総フローの最適値。

この概念を基に、次のような流れを持つアルゴリズムを提案する：

《全体の流れ》

Step 1 存在可能性が最大のアークのみを用いてアーク集合 A^0 を求め、これにより部分グラフ $G^0 = \{V, A^0\}$ を作成する。

Step 2 グラフ G^l に対して最大フロー問題を解き、その解を W^l とおく。
この時点では一目的問題であるから、アルゴリズムは従来のものを使用。
また、その際の初期実行可能解として、W^l をそのまま利用することが可能である。

Step 3 G^l に、まだ含まれていないアークの内、存在可能性が最大の物を加え、グラフ G^{l+1} とする。

Step 4 もしも追加するアークが存在しなければ、アルゴリズムを終了。
そうでなければ、l の値を 1 増やして **Step 2** へ。

最終的には、解の組 (W^l, μ^l) の中から非劣解を選択することとなる。

2.3 シェアリング問題への応用

シェアリング問題は「輸送・配分問題」を解くネットワーク計画問題の一モデルであり、問題の性質上大元の情報に曖昧さが含まれる事も多い。そのために、曖昧な情報を考慮した解法も多く考えられてきた。ここでは「経路の安全性」を導入したシェアリング問題の拡張モデルを提案する。これは、はじめに経路の存在可能性が最高のアークのみを用いてシェアリング問題を解き、使用するアークの存在可能性の下限を下げながらその解を更新していく方法である。最大フローを求める方法としてプリフロー-プッシュ算法を用いる。

また、シェアリング問題は、目標の基準により、Min-max シェアリング、Max-min シェアリング、そして最適シェアリング等が存在する。ここでは、Min-max シェアリング問題を解くアルゴリズムの一つを拡張する。

・定式化

ノード集合 $V_0=\{1,2,\cdots,n\}$、アーク集合 $E_0\subseteq\{(i,j)\mid i\neq j,(i,j)\in V_0\times V_0\}$ で表されるネットワークを $G_0(V_0,E_0)$ とする。ここで、E_0 の各要素 (i,j) に容量 $c(i,j)$ 及び存在可能性 $0\leq\mu(i,j)\leq 1$ が割り当てられている。V_0 には、供給点の集合 $S=\{s_1,s_2,\cdots,s_k\}$ と需要点の集合 $T=\{t_1,t_2,\cdots,t_l\}$ が含まれる。また、アーク (i,j) を流れるフローを $f(i,j)$、t_j に流れ込むフローを f_j とする。

以上のことより、次のような二目的 Min-max シェアリング問題 **BMSP** を提案する。

BMSP:

$$\begin{aligned}
&\text{Minimize} \quad \max_{t_j\in T} f_j/w_jv\\
&\text{Maximize} \quad \min_{(i,j)\in E_F}\mu(i,j)\\
&\text{s.t.} \qquad \textstyle\sum_{i\in V_0-j} f(i,j)=\sum_{i\in V_0-j} f(j,i) \quad (\,j\in V\cap\overline{S}\cap\overline{T}\,)\\
&\qquad\quad v=\textstyle\sum_{t_j\in T} f_j\\
&\qquad\quad 0\leq f(i,j)\leq c(i,j)
\end{aligned}$$

但し、$w_j>0$ をシンク t_j の重要度とし、$f(i,j)>0$ であるアーク (i,j) の集合を E_F と置く。

目的関数が「Minimize $\max f_j/w_j$」ではなく「Minimize $\max f_j/w_jv^*$」となっているのは、前者では、総フローを減少させる事で最終的には 0 まで第一目的関数を減少させる事が可能になるためである。

・解法

まず、シェアリング問題を最大フロー問題として解く事を考える。

はじめに、$V=V_0\cup\{s,t\}$、$E=E_0\cup\{(s,s_i)\mid i=1,2,\cdots,k\}\cup\{(t_j,t)\mid j=1,2,\cdots,l\}$ なる拡張ネットワーク $G(V,E)$ を考える。ここで、スーパーソース s を各ソース s_i にフローを流すソース、スーパーシンク t を各シンク t_j からのフローを受けるシンクとする。また、$\alpha(j)=\{(i,j)\mid(i,j)\in E\}$、$\beta(i)=\{(i,j)\mid(i,j)\in E\}$、$c(s,s_i)=\sum x(s,s_i)$、$\mu(s,s_i)=\mu(t_j,t)=1$ ($i=1,2,\cdots,k$、$j=1,2,\cdots,l$) とする。

・プリフロー-プッシュ法

最大フロー問題をプリフローの概念を用いて解く。

プリフロー-プッシュ算法は、プリフロー $x(i.j)$、距離 $d(i)$、残存量 $e(i)$ のパラメータを使用する方法である。

フローを拡張したものとして、プリフローを定義する。

フロー $f(i,j)$:

$$\begin{cases} 0\leq f(i,j)\leq c(i,j)\\ \sum_{(j,i)\in\alpha(i)} f(j,i)-\sum_{(i,j)\in\beta(i)} f(i,j)=0 \qquad (i\notin S\cup T)\end{cases}$$

プリフロー $x(i,j)$:

$$\begin{cases} 0\leq x(i,j)\leq c(i,j)\\ \sum_{(j,i)\in\alpha(i)} x(j,i)-\sum_{(i,j)\in\beta(i)} x(i,j)\geq 0 \qquad (i\notin S\cup T)\end{cases}$$

このような拡張を行うことで、フロー問題をアーク毎に分割して捉えることが容易となった。

また、問題を解くために、次の定義を行う。

残存量 $e(i)$：ノード i から更にシンクに流す必要のあるプリフローの量

$$e(i) = \sum_{(j,i)\in\alpha(i)} x(j,i) - \sum_{(i,j)\in\beta(i)} x(i,j) \qquad (i \notin S \cup T)$$

活性頂点：残存量 $e(i)$ が正の値を持つ ノード i

距離ラベル $d(i)$ ：各々のアークの長さを 1 と置いた場合のシンク（スーパーシンク）からノード i までの距離の下限

残余量 q_{ij}：ネットワーク上にフローが流れている状態で、各々のアークに更にどれだけの量のフローを増加、減少させ得るかを表す量

$$q_{ij} = \begin{cases} c(i,j) - f(i,j) & (i,j) \in A \\ f(j,i) & (j,i) \in A \\ 0 & \text{otherwise} \end{cases}$$

残余グラフ G'：ネットワーク上のフローを増加、減少可能なアークに対し、それぞれに対応する二方向のアークを配置したグラフ

$$\begin{aligned} G' &= \{V, A'\} \\ A' &= \{(i,j) \mid (i,j) \quad f(i,j) < c(i,j)\} \\ &\quad \cup \{(j,i) \mid (i,j) \in E \quad f(i,j) > 0\} \end{aligned}$$

残余ネットワーク N'：グラフ G' の各アークに残余量 q_{ij} を付与したネットワーク

残存量の上限 Δ：全残存量 $e(i)$ の上限で、 $\Delta = 2^g$ の値を取る (g は整数)
但し、 $\log(\max e(i)) \leq g < 1 + \log(\max e(i))$ とする

$\alpha_r(i)$：残余ネットワーク N' における $\alpha(i)$

$\beta_r(i)$：残余ネットワーク N' における $\beta(i)$

飽和プッシュ：フローを流す事によって、元のノードの残存量が 0 になるプッシュ
ノード i から j に量 δ のプリフローが押し出される際、$e(i) = \delta$ である

非飽和プッシュ：フローを流す事によっても、元のノードの残存量が 0 とならないプッシュ
ノード i から j に量 δ のプリフローが押し出される際、$e(i) > \delta$ である

これらの定義の基、上の条件式を満たしながら、 $d(i)$ の大きい点から小さい点へと枝ごとにプリフローを流し、最終的に全ての $e(i) = 0$ になるように調整していく。

・シェアリング問題

プリフロー-プッシュ法を拡張することによる、次の様なシェアリング問題のアルゴリズムを提案する：

Step 1 $A^0 = \{(i,j) \in A \mid \mu(i,j) = 1\}$ 、$c(t_j, t) = \infty$ とし、 (V, A^0) での最大フロー問題を解く。

Step 2 この時点のフローの総量 F^0 に対して、BSP の第一の目標が最大になるように f_j を求め、その値を各シンクからの容量 $c(t_j,t)$ とする。

Step 3 $x(t_j,t) < c(t_j,t)$ となるシンクで $x(t_j,t) = c(t_j,t)$ とし、残りの容量を他のシンクに均等にまわす。

Step 4 もし $x(t_j,t) > c(t_j,t)$ となるシンクが存在すれば、$e(t_j) = x(t_j,t) - c(t_j,t)$ 、 $x(t_j,t) = c(t_j,t)$ として再びプリフロープッシュ算法を用いる。

Step 5 $\sum x(t_j,t) = F^0$ となればアルゴリズムを終了する。そうでなければ、 **Step 3** に戻る。

・二目的問題への拡張

次に、存在可能性が大きい順に弧を追加し、最大フローを更新して行く。この時、前回のフロー、容量を最初の実行可能解とでき、効率化が図れる。

この様にして複数の解 $(W,\mu) =$ （ f_j/w_j の最小値，フローが存在している弧の存在可能性の下限）が求まるので、意思決定者は各自必要とする解を採用すればよい。

《全体の流れ》

Step 1 N 上の全ての弧の存在可能性 $0 < \mu_{ij} \leq$ をソートし、$1 \geq \mu^0 > \mu^1 > \cdots > \mu^k > 0$ とする。ここで、 k は異なる値の μ_{ij} の数である。また、 $m = 0$ とする。

Step 2 T' 、 S' を求める。もしもアーク $\{(i,j) \mid \mu_{ij} = \mu^m\}$ が全て「 $i \notin S'$**and** $j \notin T'$ 」「 $i \in S'$**and** $j \in S'$ 」、若しくは「 $i \in T'$**and** $j \in T'$ 」ならば、最大フローが変化しない。

Step 3 $A^m = \{(i,j) \mid \mu_{ij} \geq \mu^m\}$ と置き、非ファジィネットワーク $\{V, A^m, \{c8i,j)\}\}$ の最大フロー $\{f^m(i,j)\}$ を求める。ここで、 $m \leq 2$ の場合は $\{f^{m-1}(i,j)\}$ を初期解として用いる。

Step 4 シェアリング問題を解き、その値を D^m と置く。もしも $D^m > D^{m-1}$ ならば、(D^m, μ^m) を非劣解として残す。$m = k$ ならば終了。そうでなければ m を一つ増やして **Step 3** に戻る。

S' 、（ T' ）は、次のように求める：

Step 1 $S' = S'_0 = \{s\}$ 、（ $T' = T'_0 = \{t\}$ ）、 $m = 1$

Step 2 $S'_m = \{j \mid (i,j) \in A, i \in S'_{m-1}, r_{ij}\}$ （ $T'_m = \{i \mid (i,j) \in A, j \in T'_{m-1}, r_{ij} > 0\}$ ）

Step 3 $S' = S' \cup S'_m$ （ $T' = T' \cup T'_m$ ）

Step 4 もしも S'_m （ T'_m ）が空集合ならば終了。そうでなければ m を一つ増やして **Step 2** に戻る。

3 下限条件の存在する二目的最大フロー問題

次に、下限条件が存在する、つまり、幾つかのアークにおいてフローの下限が 0 になっていない問題についての解法を説明する。

3.1 定式化

ノード集合 $V=\{1,2,\ldots,n\}$ （中に供給点 s 、需要点 t を含む）、アーク集合 $A=\{(i,j)\mid i,j\in V\}$ で表されるグラフを $G=\{V,A\}$ とする。この時、グラフの各アーク (i,j) に、フローの上限 $\overline{u}(i,j)$ 、下限 $\underline{u}(i,j)$ と存在可能性 $\mu(i,j)$ が割り当てる。

ここで、ネットワーク $N=\{V,A,\overline{u},\underline{u},\mu\}$ にフローを流すことを考える。アーク (i,j) におけるフローの量を $f(i,j)$ とした時、定義より次のような条件を満たしている必要がある：

$$\begin{array}{ll}\underline{u}(i,j)\le f(i,j)\le\overline{u}(i,j) & \text{For all}\quad (i,j)\in A\\ \sum_{i\in\alpha(j)}f(i,j)=\sum_{l\in\beta(j)}f(i,j) & \text{For all}\quad j\in V-\{s,t\}\end{array}$$

但し、$\alpha(j)$ を、j を終点とするアークの始点の集合、$\beta(j)$ を、j を始点とするアークの終点の集合とする。

これより、二目的最大フロー問題 **BMFP2** を提案する。

BMFP2:

$$\begin{array}{ll}\text{Minimize} & \sum_{i\in\alpha(t)}c(i,j)f(i,j)\\ \text{Maximize} & \min_{(i,j)\in A_F}\mu(i,j)\\ \text{s.t.} & \underline{u}(i,j)\le f(i,j)\le\overline{u}(i,j)\\ & \qquad\text{For All}\quad (i,j)\in A\\ & \sum_{i\in\alpha(j)}f(i,j)=\sum_{l\in\beta(j)}f(i,j)\\ & \qquad\text{For all}\quad j\in V-\{s,t\}\end{array}$$

但し、$f(i,j)>0$ であるアーク (i,j) の集合を A_F と置く。

3.2 解法

ここでは、初めに使用するアークの存在可能性の下限の最大化を優先して行い、その後、その解を総フローが最大になる方向に更新していく。

下限条件の無い最大フロー問題で定めた概念を基に、次のような流れを持つアルゴリズムを提案する：

《全体の流れ》

Step 1 存在可能性が最大のアークのみを用いてアーク集合 A^0 を求め、これにより部分グラフ $G^0=\{V,A^0\}$ を作成する。

Step 2 初期実行可能解を求める。

Step 3 グラフ G^l に対して最大フロー問題を解き、その解を W^l とおく。
この時点では一目的問題であるから、アルゴリズムは従来のものを使用。
また、その際の初期実行可能解として、W^l をそのまま利用すること通常不可能であるが、最大フロー問題の解法としてプリフロー-プッシュ法を用いている場合は、その一部を流用可能である。

Step 4 G^l に、まだ含まれていないアークの内、存在可能性が最大の物を加え、グラフ G^{l+1} とする。

Step 5 もしも追加するアークが存在しなければ、アルゴリズムを終了。
そうでなければ、l の値を 1 増やして **Step 3** へ。

最終的には、解の組 (W^l, μ^l) の中から非劣解を選択することとなる。

3.3 最少コストフロー問題への応用

最少コストフロー問題とは、各アークの単位輸送量辺りのコストが付与されているネットワークにおいて、一定量の品物を供給点から需要点に輸送する際、総コストが最小になるようなルートを求める問題をさす。しかし、これが現実問題のモデル化であることから、現実の曖昧さによってずれが生じる恐れが有る。そのために、曖昧な情報を考慮した解法も多く考えられてきた。ここでは「経路の安全性」を導入した最小コストフロー問題の拡張モデルを提案する。これは、はじめに経路の存在可能性が最高のアークのみを用いて最小コストフロー問題を解き、使用するアークの存在可能性の下限を下げながらその解を更新していく方法である。最大フローを求める方法としてプリフロー-プッシュ法を用いる。

・定式化

ノード集合 $V = \{1, 2, \ldots, n\}$ 、アーク集合 $A \subseteq \{(i,j) \mid i \neq j, (i,j) \in V_0 \times V_0\}$ で表されるグラフを $G = (V, A)$ 、ネットワークを $N = (V, A, c, \underline{u}, \overline{u}, \mu)$ とする。ここで、 A の各要素 (i,j) に単位流量辺りのコスト $c(i,j) \geq 0$ 、フローの上限 $\overline{u}(i,j)$ 、下限 $\underline{u}(i,j)$ （ $0 \leq \underline{u}(i,j) \leq \overline{u}(i,j)$ ）、及び存在可能性 $0 \leq \mu(i,j) \leq 1$ が割り当てられている。また、アーク (i,j) を流れるフローを $f(i,j)$ とする。ここで、各ノードにおいて、流れ込むフローを流れ出るフローの量が等しい時、そのフローを循環流と呼ぶ。

以上のことより、次のような二目的最小コストフロー問題 **BMCFP** を提案する。

BMCFP:

$$
\begin{array}{ll}
\text{Minimize} & \sum_{(i,j)\in A} c(i,j) f(i,j) \\
\text{Maximize} & \min_{(i,j)\in A_F} \mu(i,j) \\
\text{s.t.} & \sum_{i\in\alpha(j)} f(i,j) = \sum_{l\in\beta(j)} f(i,j) \\
 & \qquad \text{For all} \quad j \in V - \{s,t\} \\
 & \underline{u}(i,j) \leq f(i,j) \leq \overline{u}(i,j) \\
 & \qquad \text{For All} \quad (i,j) \in A
\end{array}
$$

但し、 $f(i,j) > 0$ であるアーク (i,j) の集合を A_F と置く。

・解法

ここでは、まず使用するアークの存在可能性の下限を最大化することを行い、その後、もう一方の目的について満足できるように更新する。アルゴリズムとしては、通常の最小コストフロー問題の解法の一つである Out-of-kilter 法を拡張することとする。

・**Out-of-kilter 法**

Out-of-kilter 法は、kilter 図と呼ばれる関係図に確実に従うアークのフローから決定していき、それによって全体のフローを制限し、最終的にはフローとそれに対応したパラメータ $p(v)$ を決定する方法である。

kilter 図そのものは、次の定理によって求められる。

（定理１）

循環流 f が最小コストフローであることの必要十分条件は: 全てのアークにおいて、 $c_p(i,j) = c(i,j) + p(i) - p(j) < 0$ ならば $f(i,j) = \overline{u}(i,j)$ 、 $c_p(i,j) > 0$ ならば $f(i,j) = \underline{u}(i,j)$ となるプライス関数 $p(v)$ が存在することである。

最終的な目的はこの関係を満たすプライス関数とフローの組を見つけ出すことだが、直接求めるのは困難である。ここでは、上の関係の条件を緩和したものを考え、その緩和の幅を徐々に小さくしていく方法を取る。つまり、kilter 図の関係から逸脱する分を予め誤差定数の形で組み込んでいるのである。

まず、上の関係を緩和したものとして、 ϵ-最適と言う関係がある：

（定理２）

プリフロー x があるプライス関数である $p(v)$ において ϵ-最適であることの必要十分条件は、全てのアークにおいて、 $c_p(i,j) < -\epsilon$ ならば $x(i,j) = \overline{u}(i,j)$ 、 $c_p(i,j) > \epsilon$ ならば $x(i,j) = \underline{u}(i,j)$ となるプライス関数 $p(v)$ が存在することである。

また、ある ϵ において x が ϵ-最適であり、かつ $\epsilon' < \epsilon$ であるどのような ϵ に対してもϵ-最適でない場合、そのプリフローはϵ-tight であると言う。

《全体の流れ》

Step 0 $l = -1$ 、全ての点において $p(v) = 0$

Step 1 $l = l + 1$

Step 2 グラフ G^l を作成し、対応した値を付加して N' とする

Step 3 もしも $l = 0$ ならば、ネットワーク N^0 における、コストを無視した状態での最大フローを求め、 $\{x(i,j)\}$ とする

Step 4 もしも実行可能な循環流が存在しないならば **Step 1** へ

Step 5 $\epsilon =$ （ N^l におけるアークの容量の上限の最大値）とする

Step 6 もしも $\epsilon = 0$ ならば、 $\{f^l(i,j)\} = \{x(i,j)\}$ として **Step 3** へ

Step 7 x が ϵ-最適となるような $\{p(v)\}$ を求める

Step 8 x が ϵ-tight となるような ϵ を求める

Step 9 $\epsilon = \epsilon/2$

Step 10 $c_p(i,j) = c(i,j) + p(i) - p(j)$

Step 11 $c_p(i,j) < 0$ となる全てのアークに対して $x(i,j) = \overline{u}(i,j)$

Step 12 $c_p(i,j) > 0$ となる全てのアークに対して $x(i,j) = \underline{u}(i,j)$

Step 13 全ての点で $e(i) = \sum(x(i,j) \mid (i,j) \in A) - \sum(x(l,i) \mid (l,i) \in A)$

Step 14 もしも全ての点で $e(v) = 0$ ならば **Step 6** へ

Step 15 残余ネットワーク N' を作成

Step 16 $e(i) > 0$ である点 i に対して、もしも N' 内に i を始点とし、かつ $c_p(i,j) < 0$ となるアーク (i,j) が存在するならば、 $\delta = \min(e(i), q(i,j))$ と置いて $e(i) = e(i) + \delta$ 、$e(j) = e(j) - \delta$ そうでなければ **Step 18** へ

Step 17 もしも $(i,j) \in A$ ならば $x(i,j) = x(i,j) + \delta$ 、 $(j,i) \in A$ ならば $x(j,i) = x(j,i) - \delta$ とする

Step 18 $e(i).$ である点 i に対して、もしも **Step 16** に見られるようなアークが存在しない場合には、$p(i) = \max_{(i,j) \in A'}(p(j) - c'(i,j) - \epsilon)$ とする

Step 19 もしも全ての点で $e(v) = 0$ ならば **Step 6** へ

4 まとめ

今回は、従来の最大フロー問題に対しての、アークの存在可能性の導入による拡張を試み、その解法の内、α-レベル集合によって生成された複数のグラフを用いたものの内の一つを提案した。また、この拡張を、最大フローを基とする幾つかの応用問題に対しても適応した。

今後の課題としては、二つの決定対象に対する認識の違いによって構築されるモデルが変化することに対して、その複数のパターンそれぞれにおける具体的な解法と、解の変化の解明が挙げられる。

参考文献

[1] D. Dubois & H. Prade, "*Fuzzy Sets and Systems*", Academic Press, New York (1980).

[2] M. Tada, " Studies on Fuzzy Combinatorial Optimization", 京都大学工学部博士論文 (1994)

[3] Andrew V. Goldberg and Roberte Tarjan, "Finding Minimum-Cost Circulations by Succesive Approximation", *Mathematics of Operations Research*, 15,pp. 430-466, 1990

[4] Eva Tardos, " A Strongly Polynomial Minimum Cost Circulation Algorithm", *Combinatorica*, 5, pp. 247-255, 1985

[5] Ford L. R. and Fulkerson D. R., *"Floes in Networks"*, Princeton Univ. Press, 1962