不確定状況下における
ネットワーク上の最適化問題

2000年

島田 文彦

目 次

第1章　序

「意思決定問題」とは，さまざまな情報によって構築され，複数のとり得る選択肢の存在する環境において，ひとつ，もしくはそれ以上の判断基準に従って必要な選択を決定するものを指す．環境が複雑になった場合，情報から解を選択するまでのプロセスは各々のケースで独自のものとなっており，ある程度のパターン化による選択の制限以外は以前の物をそのまま利用することは無意味となるのが普通である．

さらに，その中で，用いる情報と判断の基準を各々数値で表し，選択肢の内の一つを決定する過程を，数学の理論を用いて構築したアルゴリズムによって表すものを「数理計画問題」と呼ぶ．この問題を扱う場合，多くは基となる環境として実際の状況を単純化したモデルを用いることとなる．このモデル化の利点としては，アルゴリズムを通して得られる解の一意性，環境の単純化による解の取得の効率化，そして同一モデルに変換できる複数の問題を一つのアルゴリズムを用いて解くことができると言うことが考えられる．

意思決定問題の内，必要とする情報が，主に複数の要素とそれらの関係上に付加されているものを，ネットワーク上の意思決定問題を呼ぶ．この問題はその構造上，点とそれらを連結する枝の形で表されるグラフまたはネットワークモデルに単純化し，グラフ理論を用いたアルゴリズムにより必要な解を得る事となる．また，これらは基となるネットワークの形状，付加された数値の種類，および意思決定者が必要とする解の種類によってさらに幾つかのパターンが存在し，それらのうちの幾つかについては十分効率的な解が既に提案されている．

現在，多くのモデルと，その解法が提案されているが，そのうち基本的なものの多くはネットワークの確定された構造とその要素，および関係上への数値情報を用いたものである．これは，パターンを単純化することでの解法の効率化とモデルの汎用性を高めることが目的である．しかし，そのために実際の問題での細かな情報の多くを無視する形となっており，現実とのギャップによりモデルでの最適な解と，意思決定者が実際に必要としている解の間にずれが生じる可能性が存在する．

この問題を解決する方法の一つとしては，まず現実問題に近いモデルを構築することが考えられる．しかしながら，単純に必要と思われるデータをすべて付加

した場合，そのモデルの規模に対して爆発的にデータが大きくなり，解くことが非常に困難になることから，最適な方法とは言えない．また，必要となるアルゴリズムも複雑となるために，解くこと自体が不可能になることもありうる．その上，現実問題には数値化すること自体が困難な情報も存在するため，最悪の場合，数値のみを用いたモデルの構築そのものが不可能な問題も存在する．

それらの問題点を回避するための方法の一つとして，新たに付加する情報に制限を設けたものが考えられている．すなわち，問題の複雑化による解法の困難さと実際の問題とのずれによる必要な解からの距離の双方に対する妥協点をとる，という考え方である．また，数値化できない情報を表現する手段としては，その一つとして，不確定値への，確率やファジィ概念を用いた表現の導入が考えられている．それらに対しては，既にいくつかの問題における既存のモデルの拡張としてその解法が提案されている．

本論文では，如何に意思決定者の意に添った，決定者が満足できる解を求めるのか，という考え方を基準に，新たなモデルと，その解法を提案する．実際問題では，適当な解を得る為の判断基準として様々な要因がかかわってくる．しかし，従来のようにそれらを直接データとしてモデルに導入するのではなく，まず意思決定者の判断によって「主要な要因」と「それ以外の要因」に分割する．後者は数値化が困難な情報を含む複数のデータの集合であることが多いため，そのままではモデルに組込みにくい．そこで，意思決定者の判断でそれらの要因の評価を満足度というパラメータに置き換える．ここでいう「満足度」とは,「意思決定者が判断を下す際，各々の要因に対する評価を，実際の判断における重要度等を考慮に入れた上で一つの値にまとめたもの」であるとする．これにより，主要因以外の様々な要因に対する最適化を一つの目標として捕らえることができるようになり，モデルを多目的から二目的へと移行することが可能となる．

本論文の趣旨にのっとって実際の問題をモデル化する際，多くは次のような構造を持っている．まず，ネットワーク上の各要素，およびそれらの関係に付加されている値の内，従来のモデルでも導入されている重要度の高いものと，解の選択基準への寄与が比較的小さいと思われる情報とに分類する．その後，後者を，意思決定者の判断の基にその要素，関係に対する満足度として一括して表現する．そのモデルを，重要度の高い情報による従来の判断基準と，その判断に対する，上記の満足度を元にした総合的な満足度の改善からなる多目的問題としてとらえる．これを解くには，従来の方法を応用した新しい解法を用いる事となる．そして，その問題を解く際，要素とその関係に満足度を付加したネットワークをファジィネットワークとして扱うことで，ファジィ関係とグラフの性質を有効に活用した，より効率的なアルゴリズムを構築することが可能となる．本論文では，従来の問題をこの視点からモデル化し，その解決の基礎となる解を見出す解法を開発する．

本論文の構成は以下のとおりである．

第２章では，本論文の基礎となる手法として，グラフ・ネットワーク，ファジィ概念，およびそれらを組み合わせたファジィネットワークについて概述する．

第３章では，ネットワーク上の最適化問題の一つである従来の最短経路問題を紹介し，新たに満足度の概念を付加したモデルを考え，その効率的解法を提案する．また，現実より得られる距離の値が確定的でない場合も想定し，距離をファジィ数という，ある程度の幅を持たせた数字であらわしたモデルについても考察を行う．ネットワーク上の最短経路問題とは，各点を結ぶ枝の距離が定義されているネットワークに対して，任意に設定した一点から他の点へのグラフに沿ったルートのうち，通過した枝の距離の合計が最も小さくなるものを探し出すものである．これは，意思決定者の「なるべく距離の短いルートを得たい」という意思に従っているためであるが，現実では，ルートを選択する際の判断基準が距離だけであるとは限らない．よって，少なからず影響を与えるであろう距離以外の要因に対し，意思決定者が各枝における満足度の値として表し，「ルート上の距離の合計の最小化」と「各満足度の最小値の最大化」の二目的問題として解くことを考える．具体的な方法としては，まず，満足度が最高の枝のみを用いた部分グラフを作り，そのグラフに対し，従来の最短経路問題と同様のアルゴリズムを用いて任意の二点間の最短経路を求める．次に，満足度が大きい順に枝を部分グラフに追加し，そのつど各点間において最短経路の更新を行う．最終的には，使用した枝の最小値によって複数のルートが得られるが，そのうち最適と思われるものを選択するのは意思決定者に任せることにする．

この新しいモデルとその解法によって，現実のネットワークで最適なルートを選択する問題に於ける，長さ等の第一目的以外の要因も，判断の基準として考慮に入れることが可能となる．例えば，このモデルを活用することで，都市間の道路網というネットワークで，ある都市から別の都市まで短時間で到達可能なルートを求める際，移動に必要な時間以外の，道の保全状態などの要因も考慮に入れた，ある程度短時間で移動でき，尚且つ意思決定者もその他の点で不満のないルートを選び出すことができる．

第４章では，最短経路問題と同様，基本の一つと考えられている最大フロー問題，および，その拡張されたモデルであるシェアリング問題とネットワーク上の最小コストフロー問題について概述し，それら従来のモデルに新たに満足度の概念を付加したものについて考察する．ネットワーク上の最大フロー問題とは，各枝に流せる物資の上限および下限が付加されているネットワークにおいて，予め設定してある供給点（ソース）から需要点（シンク）へフローを流す際，その総量が最大になるようにフローを流すルートを決定する問題である．この問題は物流問題ともいわれ，現実問題におけるネットワーク（道路網，パイプライン等）を

一定の基準のもとに通過する物資の量を，独自の判断基準のもとで調整する問題の基礎となっている．フロー問題の拡張例の一つである最小コストフロー問題は，上記のフロー問題のモデルに新たに「単位量のフローを流す際に必要なコスト」を枝に付加した形のモデルを設定し，そこでシンクへのフローの量とフローを流す際に必要なコストの総量の最適化を目的として持つ，二目的問題として解くものである．また，シェアリング問題は，上記の問題では各一個ずつであったソースとシンクの内シンクを複数設定し，各シンク間で分配されるフローの格差が大きくならないように各枝に流すフローを決定する問題であり，その「格差の最小化」という判断基準のモデルへの変換方法の違いにより，モデル全体，およびそれから得られる解に多少のずれが存在することがある．最大フロー問題において，意思決定者は各枝ごとに流すフローの量を，シンクへ流れる分の総量の最大化という判断基準のもとに決定している．しかし，実際にフローを流す際には，それ以外の要因もかかわってくることが多い．よって，ここでは，満足度の概念を基に多目的問題から二目的問題へのモデルの移行について考え，使用するアークの満足度の最大化とフローの量の調整との二目的問題として解くためのモデルとその解法の一つを提案する．最小コストフロー問題も同様にモデル化を行うことが可能であるが，この問題は，そのモデルの特異性から，使用する枝の満足度に対する閾値を高くすることによって実行可能解を求めることが不可能になる場合が通常のフロー問題と比較して多くなることが考えられる．また，シェアリング問題に対しては，上記の満足度の概念に加えて，各シンクでのフローのバランスの判断にファジィ概念を導入したモデルについても考える．

これらの拡張によって，現実のフロー問題で従来軽視されていたであろう点も考慮に入れた意思の決定が，比較的容易に実行可能となる．そもそも，実際に道路網上で物資を輸送する場合，考慮しなければならないのは，「物資がどれだけ流せるのか」という点だけではなく，「移動中の物資の安全性」「具合の悪い道や長時間の輸送による物資の損壊の回避」等の要因も考慮に入れることが必要であろうから，その意味でも，このモデルを用いた解の構築が有意義であると思われる．

第5章では，フロー問題のバリエーションであるネットワーク上の輸送問題を説明し，供給点の存在が不確定な場合のモデル化，およびその解法の提案を行う．ネットワーク上の輸送問題は，複数ある物資の供給点（主に工場）から幾つかの需要点（主に都市）に物資を輸送する際，いかにして安定した供給を得られるかという判断を求める問題の基礎となっている．これは，二部グラフ（二つの点集合間の関係のみを表したグラフ）の点集合の内，一方を供給点，もう一方を需要点の集合とおき，各供給点から需要点に，各点での条件を満たしながら物資を輸送する際，必要となるコストを最小化することを目的とした問題であり，今までに，幾つかの解法が提案されている．また，各供給点や需要点に存在する限界値

を，確定値からファジィ制限を用いた不確定値に拡張することや，フローの量を整数に制限することによる，より現実に即したモデルも提案されている．ここでは，さらに各供給点毎に，その供給点を使用することに対する満足度を設定し，なるべく満足度が小さいソースからは物資を流さないようにするモデルを構築する．これは，満足度の代わりに供給点に存在可能性を付加し，供給点が使えなくなる可能性を考慮する場合に考えられるモデルと同じ構造をしている．このようなモデルの場合，解を得る為のアルゴリズムとしては主に次の２つの場合があると考えられる．まず，幾つかの供給点が使えなくなった場合（幾つかの供給点からのフローをシャットアウトした場合）に，改めてフローの組み合わせを計算するパターンが挙げられる．これは前出の最短経路問題等でのアルゴリズムの拡張と同様に考えられるものであり，得られる解も，それぞれの段階で最適といえるものが得られる．しかし，実際に供給点をめぐる環境が変化した場合のフローの変更には，多少の手間が必要となる可能性がある．一方，幾つかの供給点が使えなくなった場合でもフローの組み合わせを以前のもので流用し，環境の変化時の手間を省くことを重視したパターンも存在する．この場合，供給点の存在可能性がどの程度まであると確実に安全（もしくは危険）とみなすかによって，確実に残っている（存在しない）供給点の予想が変化し，その分総コストの最小化のみを考える事ができる物資の量が決まるため，実際問題としては，総コストの最小化に関しては前者ほど効率的ではない．

この章での拡張は,「供給点の存在可能性」の導入が主なものであるが，これは，現実問題においては，複数の工場から複数の都市へ物資を分配する場合，その工場からの物資の供給が行われない可能性がある，もしくは意思決定者の判断で特定の工場からの供給をカットする場合の状況の変化を表しているといえる．このような事態になると予測されるときには，流せなくなった物資の代わりを，その都度他の工場からまわしてもらうことでバランスをとる方法と，幾つかの工場からの物資を差し引いてもバランスが取れるように予め設定しておく方法が考えられるが，この章のモデルを用いれば，そのどちらの場合にも対応が可能となっている．

第６章では，ツリーに関する代表的なモデルの一つを紹介し，その問題に満足度を用いた拡張モデルの提案，および，スパニングツリーの定義自体の拡張について考察を行う．ツリー問題の一つである最大スパニングツリー問題は，複数の拠点が存在する領域（複数の都市からなる地域等）において，各拠点間を自由に行き来できるようなネットワークを，なるべくコストをかけないで構築する問題の基礎となっている．また，この問題はあるグラフのスパニングツリー（グラフのすべての点を無駄なく繋ぐ部分グラフ）の内，枝に付加した「使用するために必要なコスト」の和が最小になるようなものを求める問題であり，既に効率的な

解法が存在している．ここでは，まず，上記のモデルに対して，使用する枝の満足度を加味したモデルの構築と解法の提案を行っている．すなわち，実際に幾つかの枝を用いてツリーを作成する際，その枝を使用するかどうかの判断基準として，必要なコストのみでなく，それ以外の様々な要因も加味するのは自然なことである．そのことを考慮に入れた上で，様々な要因を枝ごとの満足度という一つのパラメータの形で付加したモデルを提案するのである．次に，スパニングツリーの定義の一つである，「すべての点を繋げる部分グラフ」「任意の二点を繋ぐルートが二本以上存在しないグラフ」の組み合わせのうち一方を緩和することで，新たに擬ツリーという概念を提案し，さらに擬ツリーが通常のツリーに近い度合いを表すツリー度というパラメータを定義する．さらに，このように定義した擬ツリーを用いて，通常のツリーを利用した問題の拡張を試みる．ここでは，元の問題として，異なる二点からの，部分グラフに沿った形で計算された距離の大きい方の和が最小になるようなスパニングツリーを求めるものを考える．この問題は，通常のモデルでは，ツリーの性質によりそれほど効率的な解が得られない為，その性質を少し緩和することによって，ある程度ツリーとしての形を残しつつより効率的な解を求める，という二目的問題に変換することを可能とするものである．

この章では，スパニングツリーに対する二種類の拡張を考えているが，例えば前者の，最小スパニングツリー問題に，枝の存在可能性の考慮に入れる拡張は，現実問題では電話網設置など，設置の際の費用が主に問題となるような状況で，周りの状況による設置の困難さ等，その他の要因も考慮に入れたより現実的な問題の構築時に必要となる．また，後者はスパニングツリーの概念自体の拡張であるが，これは，現実のツリー構造の構築問題（上の電話網はその一つ）において，「設置もしくは使用するルートの数を最小にする」という前提条件のために，効率的な解がどうしても得られない場合，その前提を少し緩和することで，本来の基準でより効率的な解を得る，という考え方を導入することに対応している．

第7章は，全体のまとめであり，得られた結果，および今後の課題について述べている．

第2章　基本的概念

本章では，準備として本論文を展開するために必要なグラフ理論とファジィ集合，そしてそれらを融合した概念であるファジィグラフの概念について説明し，それらに関連する用語や記号の定義を行う．

2.1　グラフ・ネットワーク

【無向グラフ】

無向グラフ (undirected graph) は，幾つかの点 (node) と，それらのうちの特定の二点を繋ぐ枝 (edge) によって構成される．グラフ G が，点集合 V と枝集合 E から成り立っている場合，$G=(V,E)$ と表される．このとき，点集合と枝集合の要素数を各々 $|V|$，$|E|$ と記す．図 2.1 は無向グラフの例を示したものである．

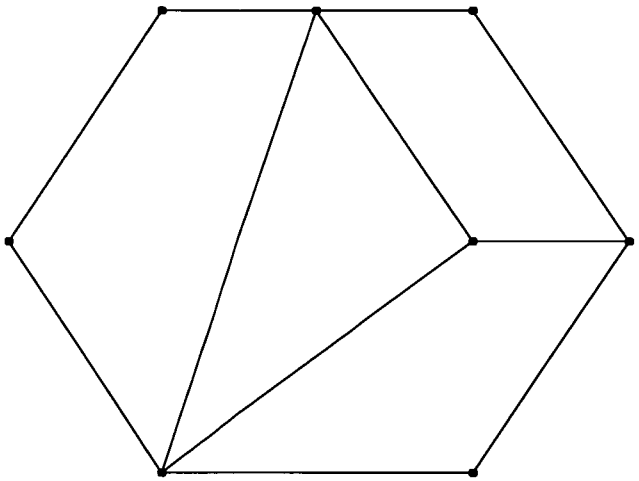

図 2.1: 無向グラフ G

点 u と v （ $u,v \in V$ ）とを繋ぐ枝 e を，$e=(u,v)$ （もしくは $e=(v,u)$ ）で表す．２本の枝 e,f が同じ二点を端点とする別の枝があるとき，この二つの

枝は平行枝 (parallel edges) であるという．また，端点が同じである枝，すなわち $e=(u,u)$ である枝を自己閉路 (self-loop) といい，内部に平行枝も自己閉路も含まないグラフを単純グラフ (simple graoh) と呼ぶ．

単純グラフ $G=(V,E)$ のうち，V の任意の二点を繋ぐ枝が全て枝集合 E に含まれている場合，グラフ G を完全グラフ (complete graph) といい，K_n で表す．

グラフ $G=(V,E)$ において，以下の条件を満たすように点と枝を交互に並べた系列：

$$(v_{i(1)}, e_{i(1)}, v_{i(2)}, e_{i(2)}, \ldots, e_{i(l)}, v_{i(l+1)})$$

$$\text{ただし}, v_{i(k)} \in V(k=1,2,\ldots,l+1), e_{i(k)}=(v_{i(k)}, v_{i(k+1)}) \in E(k=1,2,\ldots,l)$$

を，長さ l の経路 (path) といい，このときの点 $v_{i(1)}$ を始点，$v_{i(l+1)}$ を終点と呼ぶ．また，一つの経路内の点要素が全て異なる（ただし，始点と終点が同一の点であってもかまわないものとする）場合，すなわち，一本の経路が同じ点を二度以上通過しない場合，その経路は単純であるとし，さらに，単純な経路のうち，始点と終点が同一（$v_{i(1)}=v_{i(l+1)}$）のとき，その経路を閉路 (circuit) と呼ぶ．

グラフ $G=(V,E)$ 上の二点 $u,v \in V$ に対し，点 u から v への経路が存在する場合，「点 u から点 v へ到達可能である」という．グラフ G が無向グラフの場合，点 u から点 v へ到達可能であるとき，同時に点 v から点 u へも到達可能であるといえるので，「二点 u，v 間は到達可能である」ということもある．このグラフの V 内の任意の二点間で到達可能であるとき，グラフ G は連結グラフ (connected graph) であると呼ぶ．

グラフ $G=(V,E)$ において，点集合 $V' \subset V$，枝集合 $E' \subset E$ が，

$$(u,v) \in E' \Rightarrow u \in V' \text{且つ} v \in V'$$

を満たす時，言い換えると枝集合 E' 内の全ての枝の端点が，全て点集合 V' に含まれているとき，グラフ $G'=(V',E')$ をグラフ G の部分グラフ (subgraph) という．特に，連結グラフ $G=(V,E)$ に対して，$V'=V$ となる連結部分グラフを全域部分グラフといい，その内枝の本数が最も少ないものを極大木 (spanning tree) という．図 2.2 は無向グラフ G の極大木 T の例を示したものである．

また，極大木の定義としては，「全域部分グラフの内，内部に閉路を含まないグラフ」としても同様の結果が得られる．

非連結グラフ（全ての点が連結しているわけではないグラフ）に対しては，「コンポーネント」という概念がある．非連結グラフに対する連結部分グラフ（点集合 V' 内の任意の二点に関しては到達可能である部分グラフ） G_1 にたいして $G_1 \subset G'$ となるような連結部分グラフが存在しないとき，連結部分グラフ G_1 を

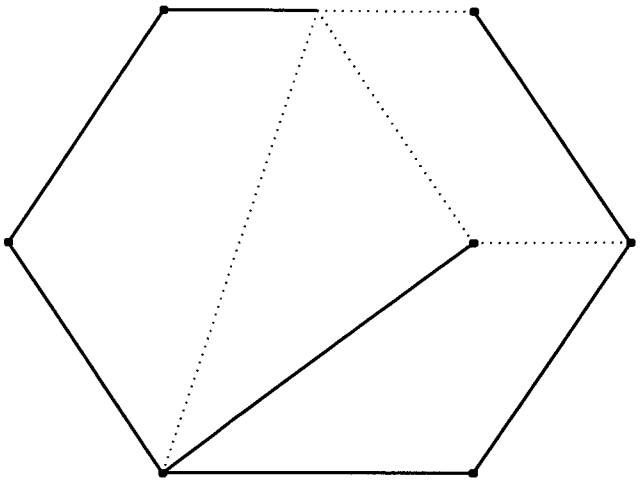

図 2.2: 極大木 T

コンポーネントと呼ぶ．この定義より，非連結グラフ G 内の異なるコンポーネント $G_1, G_2, \ldots, G_k$ は点要素，枝要素の両方の点で互いに共通部分はなく，それらの全ての和は G 全体に等しくなる．すなわち，コンポーネントと元のグラフには，「異なるどの二つのコンポーネント G_i と G_j （ $i \neq j$ ）に対しても $G_i \cap G_j = \emptyset$ が成り立つ」と，「 $G = \bigcup_{I=0}^{k} G_i$ 」の二つの関係がある．

【有向グラフ】

枝に方向性の概念を持たせたものは，枝と区別するためにアーク (弧) と呼ばれる．点集合 V とアーク集合 A からなるグラフ $G = (V, A)$ を有向グラフ (directed graph) と呼ぶが，特に断りがない時には，単にグラフと表記した場合，有向グラフを表すことがほとんどである．図 2.3 は有向グラフの例を示したものである．

点 $u \in V$ を始点， $v \in V$ を終点とするアーク $a \in A$ は $a = (u, v)$ と表されるが，単純グラフの場合でも，無向グラフと違い $a = (u, v)$ と $a' = (v, u)$ は，必ず異なるアークとして表される．また，無向グラフの時と同様に，点とアークが交互に並ぶ系列

$$(v_{i(1)}, a_{i(1)}, v_{i(2)}, a_{i(2)}, \ldots, a_{i(l)}, v_{i(l+1)})$$

ただし, $v_{i(k)} \in V (k = 1, 2, \ldots, l+1), a_{i(k)} = (v_{i(k)}, v_{i(k+1)}) \in A (k = 1, 2, \ldots, l)$

が，長さ l の経路と定義されるが，始点と終点が同一の経路はサイクル (cycle) と表現する．ここで，内部にサイクルが存在しないグラフをアサイクリックグラ

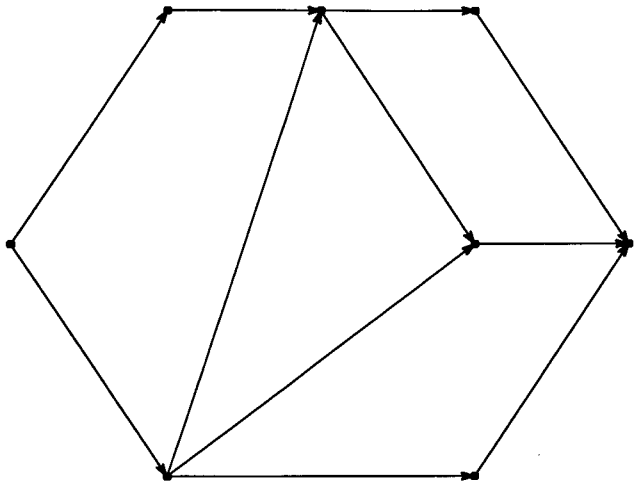

図 2.3: 有向グラフ G

フ (acyclic graph) と呼ぶ．有向グラフにおいて，ある点から他の全ての点に到達可能である，すなわち，ある点を始点とし，他の点を終点とする経路が，ずべての点に対して存在するとき，基準となった始点を根 (root) と呼ぶ．根を有し，アークの方向性を無視した状態でグラフが極大木となるものを，有向グラフにおける有向極大木と呼ぶ．

他に，有向グラフ特有の記号として $\alpha(i) = \{l \mid (l,i) \in A\}$ と $\beta(i) = \{j \mid (i,j) \in A\}$ がある．前者は,「点 i を終点として持つようなアークの，始点の集合」，後者は「点 i を始点として持つようなアークの，終点の集合」をそれぞれ表している．

【ネットワーク】

グラフは基本的に点要素の存在と，各点間の連結構造（枝およびアークの有無や到達可能性など）によって表されるものであり，具体的な各点間の関係などは表現されないことがほとんどである．そこで，そのような関係を表現するために，無向あるいは有向グラフの点および枝（アーク）に何らかの値を付加したものを，ネットワークと呼ぶ．

通常のグラフに対して付加される値は，主なものとしては，距離や通過時に必要なコストなどの，各要素ごとの固有の値や，物資の流量 (flow) が挙げられる．

2.2 ファジィ集合

ファジィ集合とは，要素がそこに含まれるか含まれないかの２値論理によってなりたっている従来の集合を，無限多値論理に対応する形に拡張したものである．従来の集合を表す関数の一つとして特性関数 $m_A(x)$ が挙げられる．これは，ある要素 x が集合 A に含まれるかどうかによって，二種類の値を得る関数となっており，図 2.4 の様に表される：

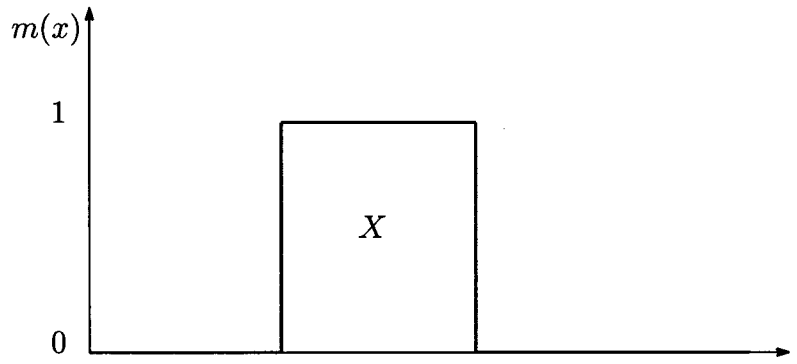

図 2.4: 通常集合の特性関数 $m_A(x)$

$$m_A(x) = \begin{cases} 1, & x \in A \\ 0, & x \notin A \end{cases}$$

この２値関数を拡張し，0 から 1 までのどの値もとり得る様にしたものが，ファジィ集合の帰属度関数 $\mu_A(x)$ となる．図 2.5 はファジィ集合の帰属度関数 $\mu_A(x)$ の例を表している．

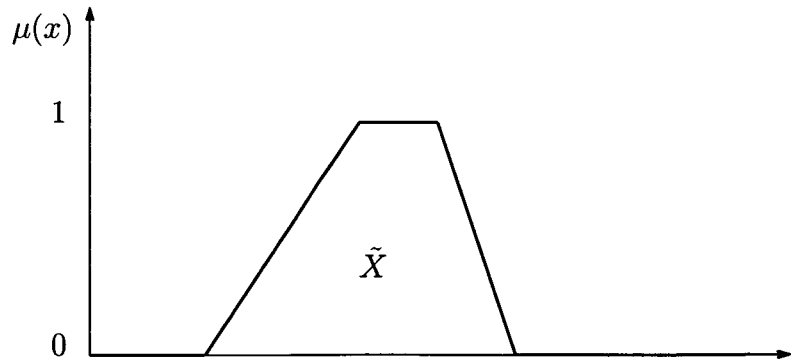

図 2.5: ファジィ集合の帰属度関数 $\mu_A(x)$

このとき．帰属値（帰属度関数の値）が大きいほど，要素 x が集合 A に属する割合が増えることを表している．この概念は表面上は確率によって拡張された集合と似てはいるが，確率による拡張と違い，偶然性に依存するものではない．

【ファジィ集合の基本的演算】

ファジィ集合は従来の集合の拡張であり，その性質にも従来の集合と類似する，もしくはもとの定義から拡張した物を定義することができる．

補集合：全ての要素 x において $\mu_{\overline{A}}(x) = 1 - \mu_A(X)$ となる集合 $\overline{A}$ を，A の補集合とする．

空集合：全ての x に対し，$\mu_A(x) = 0$ となる集合

全集合：全ての x に対し，$\mu_A(x) = 1$ となる集合

また，複数のファジィ集合 A, B 間でも次のような関係を定義することが可能である．

同一性：全ての要素 x において，$\mu_A(x) = \mu_B(x)$ であるとき，且つそのときのみ $A = B$ であるとする．

包含関係：全ての要素 x において，$\mu_A(x) \leq \mu_B(x)$ であるとき，A は B の部分集合であるといい，$A \subseteq B$ と表す．

和集合：あるファジィ集合 C の帰属値が全ての x において $\mu_C(x) = \max\{\mu_A(x), \mu_B(x)\}$ を満たすとき，C は A と B の和集合であるといい，$C = A \cup B$ で表す．

積集合：あるファジィ集合 C の帰属値が全ての x において $\mu_C(x) = \min\{\mu_A(x), \mu_B(x)\}$ を満たすとき，C は A と B の積集合であるといい，$C = A \cap B$ で表す．

De Morgan の法則：$\overline{A \cup B} = \overline{A} \cap \overline{B}$，$\overline{A \cap B} = \overline{A} \cup \overline{B}$

可換則：$A \cup B = B \cup A$，$A \cap B = B \cap A$

結合則：$A \cup (B \cup C) = (A \cup B) \cup C$，$A \cap (B \cap C) = (A \cap B) \cap C$

ただし，従来の集合における全ての性質が保存されるとは限らない．例えば，2値論理によっている従来の集合のみに成り立つ性質として，$A \cap \overline{A} = \emptyset$ （$\emptyset$ は空集合），$A \cup \overline{A} = S$ （S は全集合）が挙げられる．

一つのファジィ集合を，複数の非ファジィ集合（α-レベル集合）の組合せで表すことがある．

台：全ての x において

$$\mu_A^0(x) = \begin{cases} 1, & \mu_A(x) > 0 \\ 0, & \mu_A(x) = 0 \end{cases}$$

となる特性関数 $\mu_A^0(x)$ を持つ非ファジィ集合 A^0 が存在するとき，その閉包を，ファジィ集合 A の台とする．

α-レベル集合：あるファジィ集合 A とパラメータ α （$0 < alpha \leq 1$）に対して，全ての x において

$$\mu_A^0(x) = \begin{cases} 1, & \mu_A(x) \geq \alpha \\ 0, & \mu_A(x) < \alpha \end{cases}$$

となる特性関数 $\mu_A^\alpha(x)$ を持つ非ファジィ集合

ここで，図2.6はファジィ集合 $\tilde{X}$ の α-レベル集合の例を示している．

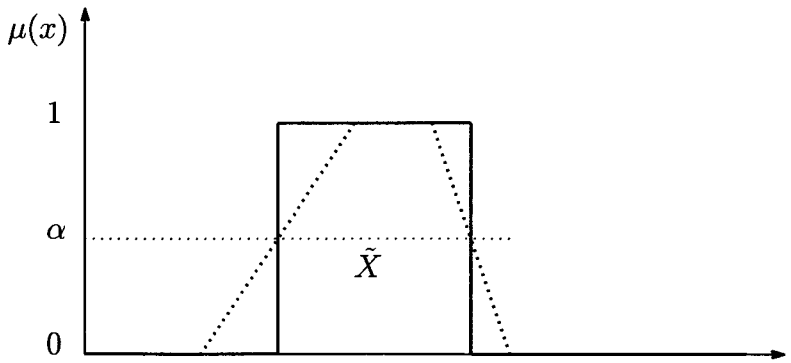

図 2.6: α レベル集合

【ファジィ数】

まず，通常の集合における凸集合の定義を行う．線形空間内の集合 C が凸集合であるとは，C の任意の要素 x_1 と x_2，および 0 から 1 までの任意の値を持

つパラメータ λ に対して，$\lambda x_1 + (1-\lambda)x_2$ がまた C の要素となる場合のことである．ここで，S が線形空間であるとは，その要素 x，y が S に含まれるとき，その線形和 $ax + by$ （a, b はスカラー量）も S に含まれるような空間を指す．

ファジィ数 $\tilde{M}$ は，ここでは次の定義によって帰属度関数 $\mu_{\tilde{M}}(\vec{x})$ が決定付けられる，一次元実数空間 R^1 内のファジィ集合の一つであるとする：

正規性：帰属度関数の最大値が必ず 1 の値をとる （$\max_{\vec{x}} \mu_{\tilde{M}}(\vec{x}) = 1$）
このとき、最大値をとる要素を x_0 とおく

準凹性：R^1 上の任意の要素 x_1, x_2 とパラメータ λ （$0 \leq \lambda \leq 1$）に対して、$\mu_{\tilde{M}}(\lambda x_1 + (1-\lambda)x_2) \geq \min\{\mu_{\tilde{M}}(x_1), \mu_{\tilde{M}}(x_2)\}$ を満たす性質

上半連続性：帰属度関数内で不連続な部分があった場合，その値は左右からの極限のうち大きい方と等しいものとする．つまり次の式が成り立つ：

$$\mu_{\tilde{M}}(x) = \begin{cases} \lim_{x' \to x+0} \mu_{\tilde{M}}(x') & x < x_0 \\ \lim_{x' \to x-0} \mu_{\tilde{M}}(x') & x > x_0 \\ 1 & x = x_0 \end{cases}$$

また，ファジィ数の演算は，通常の演算を拡張することによって得られる．R^1 上の 2 項演算 $\cdot$ は，次のように拡張される．

$$\mu_{\tilde{M} \odot \tilde{N}}(z) = \sup_{z = x \cdot y} \min\{\mu_{\tilde{M}}(x), \mu_{\tilde{N}}(y)\} \qquad x, y, z \in R^1$$

ここで，$\tilde{M}, \tilde{N}$ はファジィ数である．例えば，和 $M + N$ を拡張したものとして，

$$\begin{aligned} \mu_{\tilde{M} \oplus \tilde{N}}(z) &= \sup_{z=x+y} \min\{\mu_{\tilde{M}}(x), \mu_{\tilde{N}}(y)\} \\ &= \sup_{x \in R} \min\{\mu_{\tilde{M}}(x), \mu_{\tilde{N}}(z-x)\} \end{aligned}$$

なる関数 $\mu_{\tilde{M} \oplus \tilde{N}}(x)$ を帰属度関数として持つ拡張和 $\tilde{M} \oplus \tilde{N}$ を考えることができる．

しかし，上の拡張した演算を用いる場合には，値を求める際に非常に複雑な計算が必要となっている．また，意思決定者が通常の数からファジィ数への拡張を行う場合，全ファジィ数を非常に細かく設定することによる計算の複雑化に見合うだけの，満足する解が得られない場合が多い．よって，通常の数のファジィ化を実用的な範囲で簡略化するために，$L-R$ ファジィ数という概念が導入される．

$L-R$ ファジィ数 $\tilde{M}$ は，ある参照関数：

$$F(x) = \begin{cases} L(x) & (-\infty < x \leq 0) \\ R(x) & (0 < x < \infty) \end{cases}$$

（ただし，$L(x), R(x)$ は $[0,\infty)$ において単調減少，$L(-x)=L(x)$，$L(0)=R(0)=1$ とする）を線形変換した

$$\mu_{\tilde{M}}(x)=\begin{cases} L(\frac{m-x}{\alpha}) & (-\infty<x\leq m) \\ R(\frac{x-m}{\beta}) & (m<x<\infty) \end{cases}$$

を帰属度関数とするファジィ集合である．このときの $L-R$ ファジィ数を $\tilde{M}=(m,\alpha,\beta)_{LR}$ で表すとき，同一の参照関数を変形して得られた別の $L-R$ ファジィ数 $\tilde{N}=(n,\gamma,\delta)_{LR}$ との拡張和 $\tilde{M}\oplus\tilde{N}=(m+n,\alpha+\gamma,\beta+\delta)_{LR}$ は，帰属度関数

$$\mu_{\tilde{M}\oplus\tilde{N}}(z)=\begin{cases} L(\frac{m+n-x}{\alpha+\gamma}) & (-\infty<x\leq m+n) \\ R(\frac{x-m-n}{\beta+\delta}) & (m+n<x<\infty) \end{cases}$$

を持つ $L-R$ ファジィ数として表される（[8]）．

また，$L-R$ ファジィ数の特殊な例の一つとして，L-ファジィ数が提案されている．これは，上記の $L-R$ ファジィ数の定義に加え，「任意の要素 x に対して $L(x)=R(x)$」，「$\alpha=\beta$」の二つの性質を加えたものであり，L-ファジィ数 $\tilde{M}=(m,\alpha)_L$ の帰属度関数は

$$\mu_{\tilde{M}}(x)=L(\frac{m-x}{\alpha})$$

となる．$L-R$ ファジィ数同様，$\tilde{M}$ と $\tilde{N}=(n,\gamma)_L$ の和は，$\tilde{M}\oplus\tilde{N}=(m+n,\alpha+\gamma)_L$ のようになる．

2.3 ファジィグラフ・ネットワーク

グラフのファジィ化に関しては，主に二つの方法が考えられる．一つは枝（アーク）集合の拡張，もう一つは点集合のファジィ集合への拡張である．

はじめにアーク集合の拡張について定義する．グラフ $G=(V,A)$ は，点要素 $i\in V$（$i=1,2,\ldots,n$）に対する関係行列 R_G を用いることにより，次のように表される．

$$\begin{aligned} R_G &= \{r_{(i,j)}\} \\ &= \begin{pmatrix} 0 & r_{(1,2)} & r_{(1,3)} & \cdots & r_{(1,n)} \\ r_{(2,1)} & 0 & r_{(2,3)} & \cdots & r_{(2,n)} \\ r_{(3,1)} & r_{(3,2)} & 0 & \cdots & r_{(3,n)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{(n,1)} & r_{(n,2)} & r_{(n,3)} & \cdots & 0 \end{pmatrix} \end{aligned}$$

ただし，

$$r_{(i,j)} = \begin{cases} 1 & (i,j) \in A \\ 0 & (i,j) \notin A \end{cases}$$

であるとする．ここで，関係行列の要素 $r_{(i,j)}$ は，集合 $V \times V$ に属する要素 (i,j) の集合 A に対する特性関数であるため，次のように A をファジィ集合 A' に拡張することが可能である．

$$\begin{aligned} R'_G &= \{r'_{(i,j)}\} \\ &= \begin{pmatrix} 0 & r'_{(1,2)} & r'_{(1,3)} & \cdots & r'_{(1,n)} \\ r'_{(2,1)} & 0 & r'_{(2,3)} & \cdots & r'_{(2,n)} \\ r'_{(3,1)} & r'_{(3,2)} & 0 & \cdots & r'_{(3,n)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r'_{(n,1)} & r'_{(n,2)} & r'_{(n,3)} & \cdots & 0 \end{pmatrix} \end{aligned}$$

ここで $r'_{(i,j)}$ はアーク集合 A' に対する帰属度関数であり，$0 \leq r'_{(i,j)} \leq 1$ とする．この帰属度関数 $r'_{(i,j)}$ は，通常のファジィ集合と同様に，$\mu_A(i,j)$ の形で表される事も多い．また，帰属度関数の定義より，値が大きいほど要素 (i,j) が集合 A' に属する度合いが大きい，すなわち二点 i,j 間にアークが存在する可能性が大きいといえる．

次に，点集合 V の拡張であるが，これは，部分集合 $G' = (V', A')$ における性質の一つ，「$(u,v) \in A'$ ならば，$u \in V'$ 且つ $v \in V'$」と，α-レベル集合の性質より，次の関係が導かれる：

$$\mu_V(u) \geq \max\{\max_{i \in V} \mu_A(i,u), \max_{j \in V} \mu_A(u,j)\}$$

これにより，点に関するファジィ集合も，アークに関するファジィ集合と同様に扱うことが可能となる．

第3章　不確定状況下での最短経路問題

この章では，ネットワーク上の最適化問題の一つであるネットワーク上の最短経路問題と，アーク自体に存在可能性を付した拡張モデルの提案，およびその解法を説明する．

ネットワーク上の最短経路問題は典型的なネットワーク問題の一つであり，これまで様々な解法が考えられてきた．しかし，ほとんどの場合は，ネットワーク上の最適な経路を求める要因として，必要経費や経過時間，２点間の距離等のように数値化できるもののみを考えている．実際の問題では，それらの要因に加え，経路の安全性や個人的な好み等の，通常の方法では数値化しにくい要素についても考慮に入れる必要がある．加えて，実際問題において経路を決定するための要因がそれらのうちの一つだけという事は殆どなく，たいていが複数の要因から総合的な判断を下すことになるため，従来の方法では解を求める際のアルゴリズムが非常に複雑になることが予想される．このような問題をできるだけ効率的に，尚且つ意思決定者が満足する事を目的とするのが，この章で示す，ファジィネットワークを用いた解法である．本論文では，任意の２点間の最短距離を求め，その情報を元に，存在可能性が低いアークを追加した際の最短経路の計算を行う事とする．

3.1　ネットワーク上の最短経路問題

まず，ファジィネットワークのアーク集合のうち，アークの存在可能性が最大のもののみで成り立つ部分グラフ，すなわち閾値が最大のレベル集合からなる非ファジィグラフの最適解を求め，その後，閾値を徐々に下げる過程でグラフに含まれるアークを考慮に入れた，解の更新を行う．

初期解としての最短経路を求める方法として，本論文ではFloyd-Warshall法を用いる．

【Floyd-Warshall法】

これは，$N=(V,A,a)$ における任意の2点 (i,j) 間の距離を求める方法である．まず，次の記号を定義する（ただし，$|V|=n$ とする）．

$u_{ij}^{(m)}$ ：点 m から点 n までを経路の途中に通過しないという条件下での，点 i から点 j までの経路のうち，距離が最短になるものの経路長

Floyd-Warshall法では，ある段階での解を基に通過可能な点を一つ追加した状態での解を求めることによって，最終的に点の通過に対する制限がない状態での最短経路を求める．

$\{u_{ij}^{(m)}\}$ を用いて $\{u_{ij}^{(m+1)}\}$ を求める場合，点 i，j 間の最短距離は，次の二つの場合のうちのどちらかになる：

1. 点 m から点 n までを通らない，という条件下での最短経路：$u_{ij}^{(m)}$
2. 点 m を通り，点 $(m+1)$ から n までを通らない，という条件下での最短経路：$u_{im}^{(m)}+u_{mj}^{(m)}$

また，2点 i，j 間にアークが存在しない場合，その2点間の距離を $a_{ij}=+\infty$，全ての i，m に対して $u_{ii}^{(m)}=0$ とすると，

$$\begin{cases} u_{ij}^{(1)} &= a_{ij} \\ u_{ij}^{(m+1)} &= \min\{u_{ij}^{(m)}, u_{im}^{(m)}+u_{mj}^{(m)}\} \qquad (m=1,2,\ldots,n) \end{cases}$$

という関係式が得られる．これらを計算することにより，2点 i，j 間の最短経路長 $u_{ij}=u_{ij}^{(n+1)}$ が求められる．

《アルゴリズム 3-1》

Step 0 $m=1$ とする

Step 1 全ての $(i,j)\in V\times V$ において，

$$u_{ij}^{(1)}=\begin{cases} a_{ij} & ((i,j)\in A) \\ 0 & (i=j) \\ +\infty & (\text{それ以外}) \end{cases}$$

とする

Step 2 全ての2点間の組合せ (i,j) に対して,

$$u_{ij}^{(m+1)} = \min\{u_{ij}^{(m)}, u_{im}^{(m)} + u_{mj}^{(m)}\}$$

の関係式で値を求める.

Step 3 もし $m < n$ ならば $m = m+1$ とし, **Step 2** へ
さもなければ, $\{u_{ij}^{(n+1)}$ を求める解としてアルゴリズムを終了

3.2 ファジィグラフ上でのモデル化とその解法

アークに存在可能性が付加されているモデルを用いて最短経路問題を解く際, 本論文では, 複数の閾値によって表される複数の非ファジィネットワークに対しての解を求め, 後にそれらの中から意思決定者により最適解を決定する. 各段階での解は, 通常のアルゴリズムを用いて求めることが可能だが, 効率化のため, 前段階での解を基に更新を行うことで解を求める.

まず, 閾値が最大の α-レベルネットワーク, すなわち存在可能性が最大のアークのみで構成された N の部分グラフに対して, Floyd-Warshall 法を用いて, 任意の2点間の最短経路を求める. 次に, 閾値の少し下がった状態での α-レベル集合での解を求めるが, これは, 新たにその集合に含まれるアーク (s,t) を加え, そのアークを通過してもよいという条件に変更された時に, 最短経路がどう変化するかを調べることに対応している. すなわち, 新しい条件下での最短経路は, 前段階での最適解と同様, 新たに加えたアークを通過しない経路と, そのアークを通過する条件下での最短経路のどちらになるかを計算により決定し, 新たな最適解とするものである. この更新作業を各閾値に移行する際に付加するアークの数だけ繰り返すことにより, その段階での解を求める.

この方法で最適解が必ず求められることを証明する. それには, アーク (s,t) を追加した際の i,j 間の最短経路長 u'_{ij} が, u_{ij} と $u_{is} + a_{st} + u_{tj}$ のどちらかで表されることを示せばよい.

新しい経路が上記の二つのどちらとも異なると仮定した場合, 経路がアーク (s,t) を通過するかによって次の二通りのパターンが存在する:

(a) 値 u'_{ij} を与える経路がアーク (s,t) を通過しない

この場合は, アーク (s,t) を追加する前のネットワークにおいても, u'_{ij} が最短経路でなければならないこれより $u'_{ij} = u_{ij}$ となり, 矛盾が生じる.

(b) 値 u'_{ij} を与える経路がアーク (s,t) を通過する

定義よりアーク u'_{ij} は u'_{is}, a_{st}, u'_{tj} の3本の経路に分割することが可能である．ここで，最適性の原理（全ての段階において最適な選択肢が存在する場合，その群の一部，すなわちある二つの段階にはさまれた部分選択肢群は，その他の部位の選択に関係なく，必ず最適な決定となっている）より，u'_{is} および u'_{tj} はそれぞれ (i,s), (t,j) 間の最短経路である．これより，$u'_{is}=u_{is}$ および $u'_{tj}=u_{tj}$ となり，矛盾が生じる．

よって，元の仮定が正しくないことから，上記の方法で必ず各段階での最適解が求められることが示された．

《アルゴリズム 3-2》

Step 0 $m=1$, $L=1$ とする

Step 1 N 上の全てのアークの存在可能性をソートし，$1=\mu^0>\mu^1>\mu^2>\ldots>\mu^k>0$ で表す．ここで，k はこのような異なる値の μ_{ij} の数である．

Step 2 全ての $(i,j)\in V\times V$ において，

$$u_{ij}^{(1)}=\begin{cases} a_{ij} & (\mu_{ij}=1\text{となるアーク}(i,j)\text{が存在する}) \\ 0 & (i=j) \\ +\infty & (\text{それ以外}) \end{cases}$$

とする

Step 3 全ての2点間の組合せ (i,j) に対して，

$$u_{ij}^{(m+1)}=\min\{u_{ij}^{(m)},u_{im}^{(m)}+u_{mj}^{(m)}\}$$

の関係式で値を求める．

Step 4 もし $m<n$ ならば $m=m+1$ とし，**Step 3** へ戻る

Step 5 $\mu_{st}=\mu^l$ で，まだネットワークに含まれて居ないアーク (i,j) を任意に選び，ネットワークに加える．

Step 6 もし $u_{st}\le a_{st}$ ならば **Step 3** へ戻る

Step 7 $u_{st}=a_{st}$ とする

Step 8 全ての点の組 $i,j(\neq s,t)$ に対して，$u_{ij}=\min\{u_{ij},u_{is}+a_{st}+u_{tj}\}$ とする

Step 9 もし，存在可能性が μ^l に等しいアークのうちまだネットワークに追加されていないものが存在するのであれば，**Step 5** へ戻る

Step 10 全ての2点の組 $(i,j) \in V \times V$ において， $u_{ij}^l = u_{ij}$ とする

Step 11 もし $l < k$ ならば $l = l+1$ とし，**Step 5** へ戻る
さもなければ， $\{u_{ij}^l\}$ を求める解としてアルゴリズムを終了させる

このアルゴリズムでは，一本の枝をネットワークに追加するごとに $2(n-1)^2$ 回の加法，および $(n-1)^2$ 回の比較を行う．よって，最終的に m 本の枝を元のネットワークに追加するとすると，計算量は $O(mn^2)$ となる．

【距離がファジィ数で表されるネットワークでの解法】

アーク (i,j) に付加された距離 a_{ij} がL-ファジィ数である場合の解法を述べる．ファジィ数は非ファジィ数（通常の数）の拡張であるので，殆どの場合において非ファジィ数と同様の振る舞いを示す．ただ，従来のファジィ数間の大小関係 $\preceq$ は半順序性しか持っておらず，全ての2数の組合せにおいて大小関係が求められるわけではないので，拡張和に一定の条件を加えることで全順序性を持たせる必要がある．本論文では，全順序性を持った（条件付の）拡張和として λ-ordering $\geq_\lambda$ を用いる（[10]）．

λ-ordering $\geq_\lambda$ の定義は次のようになっている：

二つの L-ファジィ数 $\tilde{M} = (m,\alpha)_L$ と $\tilde{N} = (n,\beta)_L$ そしてパラメータ λ （ $0 \leq \lambda \leq 1$ ）に対し，その λ-ordering の元での大小関係は，

$$\tilde{M} \leq_\lambda \tilde{N} \iff \begin{cases} (\mathrm{a}) x_0|\alpha-\beta| \leq n-m \\ \text{or} \\ (\mathrm{b}) \lambda x_0|\alpha-\beta| \leq n-m < x_0|\alpha-\beta| \\ \text{or} \\ (\mathrm{c}) |n-m| < \lambda x_0|\alpha-\beta| \quad \text{かつ} \quad \beta > \alpha \end{cases}$$

によって決定する．ここで， $x_0 = \inf\{x > 0 \mid L(x) = 0\}$ とする．上記の関係式は，各々

1. $\tilde{N}$ のグラフが $\tilde{M}$ と1点以下で交差している状態
   この時，中心値 (m,n) が大きい方が大きい数である．
2. $\tilde{M}$ と $\tilde{N}$ がある程度重なっている状態
   どの程度までここに含まれるかは， λ の値によって決まる．
   この時，中心値 (m,n) が大きい方が大きい数である．

3. $\tilde{M}$ と $\tilde{N}$ がかなり重なっている状態
この時は，数の広がり (α, β) が大きい方が大きい数である．

に対応している．

このように，全ての場合において，$\tilde{M}$ と $\tilde{N}$ の順序関係が求められるので，L-ファジィ数は λ-ordering に対して全順序性をもつと言える．

λ-ordering を用いて，演算 min を拡張する．拡張された min である λ-Min を，次のように定義する．

λ-Min$_{1\leq j\leq m}\tilde{M}_j$：$m$ 個の L-ファジィ数 $\tilde{M}_1, \tilde{M}_2, \cdots, \tilde{M}_m$ に対して，各々の大小関係を λ-ordering を用いて求め，一番小さい数を値として返す．

また，λ-ordering の定義より，拡張和に対する単調性：

$$\tilde{A} \leq_\lambda \tilde{B} \Longrightarrow \tilde{A} \oplus \tilde{C} \leq_\lambda \tilde{B} \oplus \tilde{C}$$

が成り立つ.. ここで，$\tilde{A}$, $\tilde{B}$, $\tilde{C}$ は L-ファジィ数とする．

[証明]

λ-ordering の定義より，$\tilde{A} = (a, \alpha)_L$, $\tilde{B} = (b, \beta)_L$, $\tilde{C} = (c, \gamma)_L$ とすると，

$$\tilde{A} \leq_\lambda \tilde{B} \Leftrightarrow \begin{cases} \lambda x_0\alpha + a \leq \lambda x_0\beta + b & (\beta \leq \alpha) \\ \lambda x_0\alpha + a < \lambda x_0\beta + b & (\beta = \alpha) \end{cases}$$

が成り立つ．$\beta + \gamma < \alpha + \gamma \quad (\beta < \alpha)$ の場合，

$$\begin{array}{rcl} \lambda x_0\alpha + a + (\lambda x_0\gamma + c) & \leq & \lambda x_0\beta + b + (\lambda x_0\gamma + c) \\ \Longleftrightarrow \quad \lambda x_0(\alpha + \gamma) + a + c & \leq & \lambda x_0(\beta + \gamma) + b + c \end{array}$$

となる．同様に $\beta + \gamma = \alpha + \gamma \quad (\beta = \alpha)$ の場合は，

$$\lambda x_0(\alpha + \gamma) + a + c < \lambda x_0(\beta + \gamma) + b + c$$

となる．よって，$\tilde{A} \oplus \tilde{C} = (a + c, \alpha + \gamma)_L$，$\tilde{B} \oplus \tilde{C} = (b + c, \beta + \gamma)_L$ であるので，

$$\tilde{A} \leq_\lambda \tilde{B} \Rightarrow \tilde{A} \oplus \tilde{C} \leq_\lambda \tilde{B} \oplus \tilde{C}$$

∎

よって，Floyd-Warshall 法及びアークの追加に対する更新方法が，次の様に拡張される ([22]).

[拡張された Floyd-Warshall 法]

Step 0 $m=1$ とし，適当な $\lambda \in [0,1]$ を決定する．

Step 1 全ての $(i,j) \in V \times V$ において，

$$\tilde{u}_{ij}^{(1)} = \begin{cases} \tilde{a}_{ij} & (\text{アーク } (i,j) \text{ が存在する}) \\ 0 & (\text{i=j}) \\ +\infty & (\text{それ以外}) \end{cases}$$

とする

Step 2 全ての (i,j) において，

$$\tilde{u}_{ij}^{(m+1)} = \lambda\text{-Min}\{\tilde{u}_{ij}^{(m)}, \tilde{u}_{im}^{(m)} \oplus \tilde{u}_{mj}^{(m)}\}$$

とする

Step 3 もし，$m < n$ ならば，$m = m+1$ とし，**Step 2** へ戻る
そうでなければ，$\tilde{u}_{ij}^{(n+1)}$ を求める値とし，終了

[アークの追加による更新方法]

Step 0 最短経路長行列 $\{\tilde{u}_{ij}\}$ を求める

Step 1 アーク (l,m) をネットワークに追加する．

Step 2 もし，$\tilde{u}_{lm} \leq_\lambda \tilde{a}_{lm}$ ならば，更新する必要がない為，ここで終了．そうでなければ，$\tilde{u}_{lm} = \tilde{a}_{lm}$ とする

Step 3 全ての $i,j(\neq l,m)$ において，

$$\tilde{u}_{ij} = \lambda\text{-Min}\{\tilde{u}_{ij}, \tilde{u}_{il} \oplus \tilde{a}_{lm} \oplus \tilde{u}_{mj}\}$$

とした後終了

λ の値によっては，最短経路が変わることがある．例えば，

$$L(x) = \begin{cases} 1-x & (0 \leq x \leq 1) \\ 0 & (x > 1) \end{cases}$$

のとき $(x_0 = 1)$，$\tilde{M} = (10,3)_L$ と $\tilde{N} = (11,1)_L$ を比較したとする．図 3.1 は，この二つのファジィ数の関係を図示したものである．

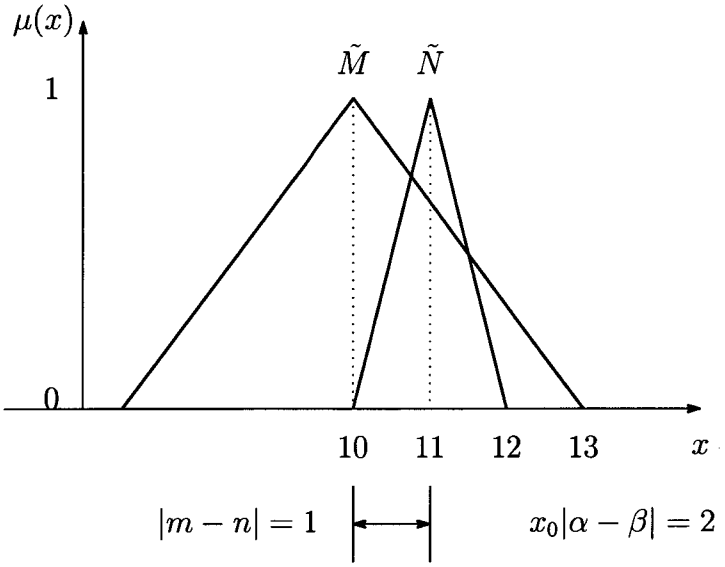

図 3.1: 二つのファジィ数の関係

この時，$\lambda = 0.5$ を境に，

$$\lambda\text{-Min}\{\tilde{M}, \tilde{N}\} = \begin{cases} \tilde{M} & (\lambda > 0.5) \\ \tilde{N} & (\lambda \leq 0.5) \end{cases}$$

となる．

一般的に，λ が小さくなるにつれ，比較に対する中心値の大小の重要性が増し，$\lambda = 0$ の場合はどの比較も，全て中心値の比較のみで行われる．

$$\tilde{M} \leq_0 \tilde{N} \Leftrightarrow m \leq n$$

また，$\lambda = 1$ の場合は，$\leq_\lambda$ は次の形になる．

$$\tilde{M} \leq_1 \tilde{N} \Leftrightarrow \begin{cases} x_0|\alpha-\beta| \leq n-m \\ \text{or} \\ |n-m| < x_0|\alpha-\beta| \quad \text{and} \quad \beta > \alpha \end{cases}$$

第4章　不確定状況下でのフロー問題

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拡張が存在する．本論文は，最大フロー問題と，その拡張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の一般化を行う．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，二目的のうち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し，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，という方法である．

本論文においては，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，すなわち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．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形になる．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る方が良い．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用できるからである．

このようにして求められた複数の解のうち，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，ある解に対する評価値のベクトル（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．

4.1 不確定状況下での最大フロー問題

下限条件が存在しない，すなわち，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述べる．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．

4.1.1 ネットワーク上の最大フロー問題

ノード集合 $V = \{1, 2, \ldots, n\}$ （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 $c(i,j)$ が割り当てる．

ここで，4.1で表すようなネットワーク $N = \{V, A, c\}$ のソース s からシンク t にフローを流すことを考える．図4.1は，対象とするネットワークの例を示したものである．ちなみに，各アークの脇にある数字は，アークに流せるフローの上限を表すものとする．

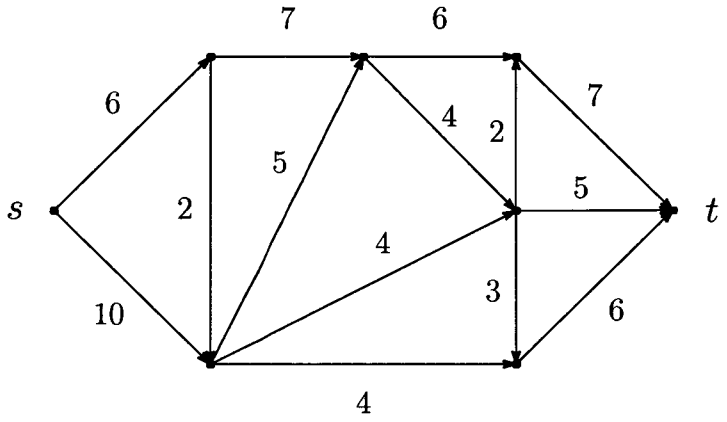

図 4.1: ネットワーク N の例

アーク (i,j) におけるフローの量を $f(i,j)$ とした時，次のような条件を満たす事が必要とする：

$$0 \leq f(i,j) \leq c(i,j) \qquad \text{For all} \quad (i,j) \in A$$
$$\sum_{l \in \alpha(i)} f(l,i) = \sum_{j \in \beta(i)} f(i,j) \qquad \text{For all} \quad j \in V - \{s,t\}$$

ここで，前者は「どのアークにおいても，その枝を流れるフローはアーク毎の上限 c と下限 (0) によって制限される」こと，後者は「供給点と需要点を除く全ての点において，各点に流れ込むフローと各点から流れ出るフローの量は常に等し

い」ことを表している．ただし，$\alpha(j)$ 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，次の形の最大フロー問題 **MFP** を提案する．

BMFP:

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

【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．

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 $f(i,j)$ ：

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

プリフロー $x(i,j)$ ：

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

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，次の定義を用いる．

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

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

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

《アルゴリズム 4-1》

Step 0 A に含まれる全アーク (i,j) に対して $x(i,j)$ =0，また，V に含まれる全ノード i に対して $e(i) = 0$ とする.

Step 1 $d(t) = 0, U = \{t\}, l = 1$ とする.

Step 2 $U' = \{i \mid i \notin U, \beta(i) \in U\}$ とする.

Step 3 U' に含まれるノード i の全てに $d(i) = l$ とラベル付けする.

Step 4 $U = U \cup U'$ とし，$l = l + 1$ とする．

Step 5 もしも s にラベルがついていなければ，**Step 3** に戻る．

Step 6 $d(s) = n$ とする．

Step 7 $i' = \beta(s)$ となる全ての点 i' に対し，$x(s, i') = c(s, i')$ 及び $e(i') = x(s, i')$ とする．

Step 8 $\Delta = 2^g$ とする．但し，$\log(\max e(i)) \leq g < 1 + \log(\max e(i))$ とする．

Step 9 この時点での残余ネットワーク $\{q_{ij}\}$ を求める．

Step 10 $I(\Delta) = \{i \mid e(i) \geq \frac{\Delta}{2}\}$ とおく．

Step 11 もし $I(\Delta) = \phi$ ならば，$\Delta = \frac{\Delta}{2}$ として **Step 9** に戻る．

Step 12 $i_m(\Delta) = \{i' \mid d(i') = \min_{i \in I(\delta)} d(i)\}$ となる集合の中からノードを選択し，$i_{\min}$ とする．

Step 13 $\{(i_m, j') \mid j' = \beta_r(i_m), d(i_m) = d(j') + 1\}$ とする．

Step 14 もし $\{(i_m, j')\} = \phi$ ならば，$d(i_m) = \min\{d(j) + 1 \mid j \in \beta(i_m)\}$，とした後 **Step 13** に戻る．

Step 15 $\{(i_m, j')\}$ の中から，(i'', j'') を選択する．

Step 16 $\delta = \min\{e(i''), q(i'', j''), \Delta - e(j'')\}$ とする．

Step 17 プリフロー $x(i'', j'')$ を $x(i'', j'') = x(i'', j'') + \delta$, $e(i'') = e(i'') - \delta$ と置き，残存量 $e(i'')$ を $e(j'') = e(j'') + \delta$ とする．

Step 18 $\{q_{ij}\}$ を更新する（$q(i'', j'') = q(i'', j'') - \delta$, $q(j'', i'') = q(j'', i'') + \delta$）．

Step 19 もし $\sum_{i \in V_0} e(i) = 0$ ならば $\{f(i, j)\} = \{x(i, j)\}$ としてアルゴリズムを終了．そうでなければ **Step 9** に戻る．

（注）

アルゴリズム中の「選択する」では，元の集合の要素であれば，どれを選んでもアルゴリズムが成り立つ．よって，このアルゴリズムでは集合内の最初にある要素を選択することとする．

4.1.2 ファジィグラフ上でのモデル化とその解法

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．

【解法】

この解法は，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，新たに幾つかの記号について定義する..

$\{\mu^l\}$ ：アークの異なる存在可能性 $\mu(i,j)$ の値を降順に並べたリスト．
$1\ge\mu^0>\mu^1>\cdots>\mu^{k-1}>\mu^k=0$ であり，k は異なる $\mu(i,j)$ の数とする．

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

$G^l=\{V,A^l\}$ ：しきい値 μ^l を持つグラフ．

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：

《アルゴリズム 4-2》

Step 0 $j=1$, $P=\phi$, $A^0=\phi$ とする.

Step 1 $\mu(i,j)$ の異なる値をソートし, μ^l を作成する.

Step 2 アークの集合 A^j を作る : $A^j=A^{j-1}\cup\{(i',j') \mid \mu(i',j')=\mu^j,(i',j',)\in A\}$

Step 3 A^j を元にネットワーク $N^j=\{V,A^j\}$ を作成する.

Step 4 アルゴリズム 4-1 を用いて, 最大フロー $\{f^1(i',j')\}$ を得る.

Step 6 もし $j=k$ であればアルゴリズムを終了. そうでなければ, $j=j+1$ として **Step 2** に戻る.

(注)

Step 4 で, アルゴリズム 3 中に, サブルーチンとしてアルゴリズム 4-3 を呼び出している. この時, 使用するネットワーク $N=N^j$ は次の様に定義される :

$$c_i(i',j')=\begin{cases} c_i^j(i',j') & (\mu(i',j')\geq\mu^j) \\ 0 & (\mu(i',j')<\mu^j). \end{cases}$$

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.

＜計算複雑性＞

本論文におけるこのアルゴリズムの計算複雑性は次のようにして求められる.

まず, プリフロー-プッシュ法を用いて最大フローを算出するには, $O(n^2m)$ のオーダーの計算が必要となる ([24]). 本論文では各段階での解の取得のため, プリフロー-プッシュ法を異なるアークの存在可能性の数 $|\{\mu^l\}|\leq m$ だけ繰り返す. よって, 最悪の場合, 必要な計算量は $O(n^2m^2)$ となる.

4.2 不確定状況下でのシェアリング問題

シェアリング問題はネットワーク計画問題としての「輸送・配分問題」の一モデルである. ここでは「経路の安全性」等を導入したシェアリング問題の拡張モデルを提案する. 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

新していく解法を提案し，最大フローを求める方法としてはプリフロー-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を拡張して，不確定状況下でのシェアリング問題の解法を提案する．

4.2.1 シェアリング問題

ノード集合 $V_0 = \{1, 2, \cdots, n\}$，アーク集合 $A_0 \subseteq \{(i,j) \mid i \neq j, (i,j) \in V_0 \times V_0\}$ で表されるネットワークを $G_0(V_0, A_0)$ とする．ここで，A_0 の各要素 (i,j) に容量 $c(i,j)$ が割り当てられているものとする．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 Min-max シェアリング問題 **MSP** が考えられる．

MSP:

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

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 A_F で示す．

目的関数が「Minimize $\max f_j/w_j$」ではなく「Minimize $\max(f_j/w_j)v$」となっているのは，後者では，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．

はじめに，$V = V_0 \cup \{s,t\}$，$A = A_0 \cup \{(s,s_i) \mid i = 1,2,\cdots,k\} \cup \{(t_j,t) \mid j = 1,2,\cdots,l\}$ なる拡張ネットワーク $G(V,A)$ 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，$\alpha(j) = \{(i,j) \mid (i,j) \in A\}$，$\beta(i) = \{(i,j) \mid (i,j) \in A\}$，$c(s,s_i) = \sum x(s,s_i)$，$\mu(s,s_i) = \mu(t_j,t) = 1$ （$i = 1,2,\cdots,k$，$j = 1,2,\cdots,l$）とする．

【Min-maxシェアリング問題】

ここでは，プリフロー-プッシュ法を用いたアルゴリズムをシェアリング問題に拡張する．

まず，プリフロー-プッシュ法を用いて最大フロー $\{f_0(i,j)\}$ を求め，D_1 の値を容量 $c(j,t)$ の上限とする．その後，フロー $f(i,j)$ を，目標に沿うように修正する．この時総フローが最初に求めた最大フロー問題のものより小さかった場合，目的の条件を多少緩め，フローを調整する．この事を，最終的に $\sum_{j=1}^{l} f(i,j) = \sum_{j=1}^{l} f_0(i,j)$ となるまで繰り返す．

ここで，いくつかの記号の定義を行う．

N_i : i 回目の更新の結果得られるネットワーク

D_i : 容量 $c_i(i,t)$ の上限

$f_i(i',j')$: i 回目の更新時の，N_i 上にあるノード i' から j' へのフローで，下の条件を満たすもの：

$$\begin{cases} \sum_{j\in\alpha(i')} f_i(j',i') - \sum_{j\in\beta(i')} f_i(i',j') = 0, (i \notin \{s,t\}) \\ \sum_{j\in\alpha(s)} f_i(j',s) - \sum_{j\in\beta(s)} f_i(s,j') \leq 0 \\ \sum_{j\in\alpha(t)} f_i(j',t) - \sum_{j\in\beta(t)} f_i(t,j') = f_i(j') \end{cases}$$

$c(i',j')$ アーク (i',j') の容量

$f_i(j)$ i 回目の更新時，シンク t_j に流れ込むフローの総量

v_i : i 回目の更新時，ネットワーク上の総フロー

v^* : 最初に求めた最大フローの解に対応する総フロー

X_i : i 回目の更新時の最小カット集合

$e_i(s_l)$: i 回目の更新時，ソース s_l に存在する初期残存量
前回の更新終了時，s_l から s に戻したプリフローの値と同じ値を持つ．

$\overline{D}$: シェアリング問題の解となる，容量の上限

《アルゴリズム 4-3》

Step 0 $i=1$ と置き，その後全シンクに対して $c(t_j,t)=\infty$ とする．

Step 1 最大フロー $\{f(i',j')\}$，総フロー v^*，そして残余量 $\{q_{i'j'}\}$ をアルゴリズム1を用いて求める．

Step 2 $D_1 = v^*/(\sum_{t_j \in T} w_j)$ 及び $e_i(s_i) = c(s, s_i)$ とする．

Step 3 $i = i + 1$ とする．

Step 4 $c_i(t_j, t) = D_i w_j$, $c_i(i', j') = c(i', j'), (i', j' \notin T)$ とする．

Step 5 A 上の各アーク (i', j') に対して $x(i', j') = f_i(i', j')$ とし，V に含まれる全ノードに対して $e(i') = 0$ とする．

Step 6 各ソース s_i に対して $c(s, s_i) = \sum_{j' \in \beta(s_i)}, c(s_i, j')$ とする．

Step 7 $d(t) = 0$, $U = \{t\}$, $l = 1$ とする．

Step 8 $U' = \{i' \mid i' \notin U, \beta(i') \in U\}$ とする．

Step 9 U' に含まれるノード i' の全てに $d(i') = l$ とラベル付けする．

Step 10 $U = U \cup U'$ とし，$l = l + 1$ とする．

Step 11 もしも s にラベルがついていなければ，**Step 9** に戻る．

Step 12 $d(s) = n$ とする．

Step 13 全ソースに対し，$e(s_i) = x(s, s_i)$ とする．

Step 14 $\Delta = 2^n$ とする．ここで，$\log(\max e(i)) \leq n < 1 + \log(\max e(i))$ とする．

Step 15 残余ネットワーク $\{q_{ij}\}$ を求める．

Step 16 $I(\Delta) = \{i' \mid e(i') \geq \frac{\Delta}{2}\}$ とする．

Step 17 もし $I(\Delta) = \phi$ ならば，$\Delta = \frac{\Delta}{2}$ として **Step 15** に戻る．

Step 18 $i'_m(\Delta) = \{i'' \mid d(i'') = \min_{i' \in I(\delta)} d(i')\}$ で求められる集合の中から，その最初の要素を選択し，これを $i'_{\min}$ とする．

Step 19 $\{(i'_m, j') \mid j' = \beta_r(i'_m), d(i'_m) = d(j') + 1\}$ とする．

Step 20 もし $\{(i'_m, j')\} = \phi$ ならば，$d(i'_m) = \min\{d'(j) + 1 \mid j \in \beta(i'_m)\}$ として **Step 18** に戻る．

Step 21 集合 $\{(i'_m, j')\}$ の最初の要素を (i'', j'') と置く．

Step 22 $\delta = \min\{e(i''), q(i'', j''), \Delta - e(j'')\}$ とする.

Step 23 $x(i'', j'') = x(i'', j'') + \delta$, $e(i'') = e(i'') - \delta$, 及び $e(j'') = e(j'') + \delta$ とする.

Step 24 もし $i'' = s_i$ and $j'' = s$ ならば, $e_{i+1}(s_i) = e_{i+1}(s_i) + \delta$ とする.

Step 25 $\{q_{i'j'}\}$ を更新する ($q(i'', j'') = q(i'', j'') - \delta$, $q(j'', i'') = q(j'', i'') + \delta$) .

Step 26 もし $\sum_{i' \in V_0} e(i') = 0$ ならば, $\{f_i(i', j')\} = \{x(i', j')\}$ とする. そうでなければ **Step 10** に戻る.

Step 27 $v_i = \sum_{t_j \in T} f_i(t_j)$ とする.

Step 28 もし $v^* = v_i$ ならば, D_i を求める解としてアルゴリズムを終了 .

Step 29 X_i を求め, $T_i = T \cap X_i$ とする.

Step 30 $D_{i+1} = (v^* - v_i)/(\sum_{t_j \in T_i} w_j) + D_i$ とする.

Step 31 **Step 3** に戻る.

4.2.2 ファジィグラフ上でのモデル化とその解法

上記のアルゴリズム 2 を, 本来の目的と使用したアークの存在可能性との二目的問題を解くアルゴリズムに拡張する.

ノード集合 $V_0 = \{1, 2, \cdots, n\}$, アーク集合 $A_0 \subseteq \{(i, j) \mid i \neq j, (i, j) \in V_0 \times V_0\}$ で表されるネットワークを $G_0(V_0, A_0)$ とする. ここで, A_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:

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

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 A_F で示す．

【解法】

最初に，アークに付加された存在可能性 $\mu(i,j)$ を大きい順にソートし，$\{\mu^l\}$ で示す．それから，ネットワーク N の部分ネットワークとして，存在可能性が N の中で最大のアークのみで構成された初期ネットワーク N^0 を作成する．このネットワーク上のシェアリング問題を解き，その後，次に存在可能性が大きいアークを追加して，解を更新する．これを全てのアークが追加されるまで繰り返し，最終的に各存在可能性に対応した非劣解の組を得る．

ここで，さらに次の記号についての定義を行う．

$\overline{D^j}$: $\{f^j(i',j')\}$ に対する各シンクでの重み付き容量の上限

v^j : N^j に流れる最大フロー $\{fl^j(i',j')\}$ の総フロー

P : 問題 **BMSP** における Pareto 最適解（非劣解）の集合

《アルゴリズム 4-4》

Step 0 $j = 1$, $P = \phi$, $A^0 = \phi$ とする．

Step 1 $\mu(i,j)$ の異なる値をソートし，μ^l を作成する．

Step 2 アークの集合 A^j を作る：$A^j = A^{j-1} \cup \{(i',j') \mid \mu(i',j') = \mu^j, (i',j',) \in A\}$

Step 3 A^j を元にネットワーク $N^j = \{V, A^j\}$ を作成する．

Step 4 アルゴリズム 4-3 を用いて，最大フロー $\{f^1(i',j')\}$ を得る．

Step 5 もし全ての $j' = 1, 2, \ldots, j-1$ に対して $\overline{D^j}/v^j \geq \overline{D^{j'}}/v^{j'}$ ならば，**Step 6** に進む．そうでなければ，集合 P に $(\{f^0(i',j')\}, \mu^0)$ を加える．

Step 6 もし $j = k$ であればアルゴリズムを終了そうでなければ，$j = j+1$ として **Step 2** に戻る．

（注）

Step 4 で，アルゴリズム 3 中に，サブルーチンとしてアルゴリズム 4-3 を呼び出している．この時，使用するネットワーク $N = N^j$ はコストのみが次の様に変更されている：

$$c_i(i',j') = \begin{cases} c_i^j(i',j') & (\mu(i',j') \geq \mu^j) \\ 0 & (\mu(i',j') < \mu^j). \end{cases}$$

＜計算複雑性＞

本論文におけるこのアルゴリズムの計算複雑性は次のようにして求められる．まず，初期実行可能解を求める際には，シェアリング問題を解くアルゴリズムと同量の計算を行う必要がある．この計算複雑性は，$O(|T|\, mf(n,m))$ で表される ([3])．ここで，$mf(n,m)$ は最大フロー問題を解くアルゴリズムの計算複雑性（本論文では $mf(n,m) = O(n^2m)$ ）を表すものとする．その後，解を更新する際に，上記のアルゴリズムと同量（オーダー $O(|T|\, mf(n,m))$ ）の計算を行う．但し，各段階での初期実行可能解は容易に求まるため，実際の計算量は一度目よりも少ない．これより，このアルゴリズム全体での計算複雑性は，$O(|T|n^2m^2)$ となる．

4.3 不確定状況下での最小コストフロー問題

本論文では，アークにおけるフローの上限および下限と総コストの最小化という，従来の判断基準以外の要因に対して意思決定者が満足度を決定し，総コストの最小化と，使用するアークの満足度の下限の最大化の二目的問題として解く．

まず，使用するアークの存在可能性の下限が最大になるように，閾値を最大にし，元のファジィネットワークから存在可能性が最大のアークのみを抜き出した非ファジィネットワークを作成する．このネットワークに対して通常の最小コストフロー問題として解き，その解を基に，使用するアークの存在可能性の下限が小さくなる方向に更新を行う．

4.3.1 ネットワーク上の最小コストフロー問題

ここでは，フローを特定の供給点から需要点への流れではなく，どのノードにおいても流れの生成・消滅のない循環流として考える．

【定式化】

まず，問題を定式化する際に必要な幾つかの記号の定義の追加を行う．

あるグラフ $G=(V,A)$ の各々のアーク $(i,j)\in A$ には，コスト $c(i,j)$ の他に通過可能なフローの上限 $\overline{u}(i,j)$ と下限 $\underline{u}(i,j)$ が付加されているものとする．ここで，$0\leq \underline{u}(i,j)\leq \overline{u}(i,j)$ である．

また，アーク (i,j) を流れるフロー $f(i,j)$ が，各ノードにおいて $\sum_{j=\beta(i)} f(i,j)-\sum_{l=\alpha(i)} f(l,i)=0$ を満たすとき，フローベクトル $f\in R^A$ を循環流と呼ぶ．

もしソース s からシンク t への循環型でないフローが存在する最小コストフロー問題を本論文での解法を用いて解く場合には，需要点から供給点にアーク (t,s) を追加し，そのコストおよびフローの上限，下限を，それぞれ $c(t,s)=0$，$\overline{u}(t,s)=\min\{\sum_{i\in\alpha(t)}\overline{u}(i,t),\sum_{j\in\beta(s)}\overline{u}(s,j)\}$，$\underline{u}(t,s)=0$ とすることで，循環流に置き換えることが可能となる．

これらの定義より，次のような最小コストフロー問題（MCFP）が考えられる．

MCFP:

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

ただし，コスト $c(i,j)$ は全て整数であるとする．

【下限つき最大フロー問題】

最小コストフロー問題の初期実行可能解として，コストを無視した状態での最大フロー問題の解を用いる．前提としてアーク毎のフローの容量に上限と下限を持たせている為，初期実行可能解を求める際に，ネットワークをアークの容量に下限のないもの（N'）に変換する必要がある．まず，新たにソース s'，およびシンク t' を加え，元の点とのアークを A に追加する．次にアークの容量を次のよう

に変更する.

$$
\begin{aligned}
\overline{u}'(i,j) &= \overline{u}(i,j) - \underline{u}(i,j) \\
\overline{u}'(s',j) &= \begin{cases} \sum_{j\in\beta(i)} \underline{u}(i,j) - \sum_{l\in\alpha(i)} \underline{u}(l,i) \\ \qquad (\sum_{j\in\beta(i)} \underline{u}(i,j) - \sum_{l\in\alpha(i)} \underline{u}(l,i) > 0) \\ 0 \qquad \text{(otherwise)} \end{cases} \\
\overline{u}'(i,t') &= \begin{cases} \sum_{l\in\alpha(i)} \underline{u}(l,i) - \sum_{j\in\beta(i)} \underline{u}(i,j) \\ \qquad (\sum_{l\in\alpha(i)} \underline{u}(l,i) - \sum_{j\in\beta(i)} \underline{u}(i,j) > 0) \\ 0 \qquad \text{(otherwise)} \end{cases}
\end{aligned}
$$

このネットワークは容量の下限か常に0であるため，全てのアークにおいてフローを0としたものがこのネットワークでの初期実行可能解として成り立つ．その解に対して通常の最大フロー問題の解法を適応することで．最大フローを求めることが可能となる．また，元のネットワークのフロー $f(i,j)$ は，この解 $f'(i,j)$ より， $f(i,j) = f'(i,j) + \underline{u}(i,j)$ として求めることができる.

【Out-of-Kilter法】

Out-of-Kilter法は，初期フローの流れているネットワークに対して，フローが確定できるアークを検索してフローを固定し，その分のずれ，すなわちアークが連結している二つの点でのフローの発生および消滅を隣接するアークの流量を調整することによって解消するという方法である.

最大フロー問題におけるプリフロープッシュ法と同様，この解法ではプリフロー，残存量，残余ネットワーク等を用いる必要がある．これらの記号については既に説明がなされているが，定義がそのときのものと多少異なるため，改めて記号の定義を行う.

プリフロー $x(i,j)$ ：

$$
\begin{cases} \underline{u}(i,j) \le x(i,j) \le \overline{u}(i,j) \\ \sum_{(j,i)\in\alpha(i)} x(j,i) - \sum_{(i,j)\in\beta(i)} x(i,j) \text{ は必ずしも0ではない} \qquad (i \in V) \end{cases}
$$

残存量 $e(i)$ **：** 点 i でのプリフローの発生（吸収）量

$$
e(i) = \sum_{(j,i)\in\alpha(i)} x(j,i) - \sum_{(i,j)\in\beta(i)} x(i,j) \qquad (i \in V)
$$

ネットワーク全体では $\sum_{i\in V} r(i) = 0$ が成り立っている.

残余量 $q(i,j)$ **：** 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

$$q(i,j) = \begin{cases} \overline{u}(i,j) - f(i,j) & (i,j) \in A \\ f(j,i) - \underline{u}(i,j) & (j,i) \in A \\ 0 & \text{otherwise} \end{cases}$$

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

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

残余ネットワーク N' **：** グラフ G' の各アークに残余量 $q(i,j)$ を付与したネットワーク

フローの定義より，全ての点で $e(i) = 0$ を満たすとき，プリフローは循環流と同等になる．

また，Out-of-Kilter 法の独自の記号として，プライス関数 $p(i)$ と修正コスト $c_p(i,j)$ がある．

プライス関数 $p(i)$ ： 各点のポテンシャル

修正コスト $c_p(i,j)$ ： アーク (i,j) の流量を決定するために必要なパラメータで，$c_p(i,j) = c(i,j) + p(i) - p(j)$ で表される．

これらの記号を用いて，Out-of-Kilter 法についての説明を行う．

この方法は，Kilter 条件と呼ばれる関係式に確実に従うアークのフローから値の固定を行い，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 $\{p(i)\}$ を決定する方法である（[16]）．Kilter 条件そのものは，次の定理（[37]）によって求められる．

（定理 3-1）（[16]）

循環流 $\{f(i,j)\}$ 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，$c_p(i,j) < 0$ 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 $\{p(i)\}$ 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ベクトルを見つけることであるが，直接求めることは困難である．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狭めていく方法をとる．すなわち，各アークにおけるフローのうち，Kilter条件の制約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，Kilter条件を緩和したものとして，ϵ-最適という関係が有る．

（定理 3-2）（[16]）

> 循環流 $\{f(i,j)\}$ があるプライス関数 $\{p(i)\}$ に対して 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 $f(i,j) = \overline{u}(i,j)$，
> > $c_p(i,j) > \epsilon$ ならば $f(i,j) = \underline{u}(i,j)$
>
> となることである．

また，ある ϵ において x が ϵ-最適であり，且つ $\epsilon' < \epsilon$ であるどのような ϵ' に対しても ϵ'-最適でない時，そのプリフロー x は ϵ-tight であるという．

アルゴリズムの関係上 $\epsilon = 0$ とならないこともあるが，$\epsilon < 1/n$ （n はネット江ワーク上の点要素の数）となった時の ϵ-最適な循環流は，最小コストフローに等しいことが証明されている．

フローの選択の幅を狭める方法として，本論文ではフロー $\{f(i,j)\}$ と誤差定数 ϵ を固定した状態でプライス関数 $\{p(i)\}$ を決定し，その後，さらに誤差定数が小さくなるように，プライス関数を固定した状態でフローを変更することを考える．すなわち，まずプライス関数を調整することで「ϵ-最適な循環流」を作成し，そこから，フローが ϵ'-tight となるような ϵ' （$\rightarrow$「ϵ'-tight な循環流」）を求める．さらに，ϵ' の値を半減させ，ϵ-最適条件に無関係にフローが決定できる領域 $-\epsilon \leq c_p(i,j) \leq \epsilon$ から修正コストが外れたアークに対しては，$f(i,j) = \overline{u}(i,j)$，もしくは $f(i,j) = \underline{u}(i,j)$ として修正を行う（$\rightarrow$「$\epsilon'/2$-最適なプリフロー」）．その後残存量を調整し，$\epsilon'/2$-最適な循環流の形にする．この過程を繰返し，最終的に $\epsilon < 1/n$ となった時点のフローが最適解となる．

上記のルーチンの成立には，次の定理に表れる性質がかかわってくる：

（定理 4-3）

> 循環流 $\{f(i,j)\}$ が最小コストフローであることの必要十分条件は，
> $\{f(i,j)\}$ を流した状態での残余ネットワーク N' 内に，コストの和
> $\sum_{(i,j)\in C} c(i,j)$ が負になるサイクル C が存在しないことである．

ここで，$c^{(\epsilon)}(i,j) = c(i,j) + \epsilon$ とすると，上記の定理を次のように拡張することが可能となる：

（定理 4-4）

循環流 $\{f(i,j)\}$ が最小コストフローであることの必要十分条件は，$\{f(i,j)\}$ を流した状態での残余ネットワーク N' 内に，コストの和 $\sum_{(i,j)\in C} c^{(\epsilon)}(i,j)$ が負になるサイクル C が存在しないことである．

[プライス関数の決定]

固定されたフロー $\{f(i,j)\}$ と $\epsilon > 0$ から，フローを ϵ-最適とするようなプライス関数 $\{p(i)\}$ を求める副問題を，本論文では，後述する最小スパニングツリー問題の解法を用いて解くこととする．最小スパニングツリーを求めるネットワークは次の形になる．

$$\begin{aligned} N_S &= \{V_s, a_s, c_s\} \\ V_S &= \{s\} \cup V \\ A_S &= \{(s,j) \mid i \in V\} \\ &\quad \cup \{(i,j) \mid (i,j) \in A\} \cup \{(i,j) \mid (j,i) \in A\} \end{aligned}$$

$$c_S(i,j) = \begin{cases} 0 & (i = s) \\ c^{(\epsilon)}(i,j) & ((i,j) \in A) \\ -c^{(\epsilon)}(i,j) & ((j,i) \in A) \end{cases}$$

ここで求めたツリーを T_ϵ とすると，ルート s から点 i への距離 $d(i)$ には，$d(\mathrm{parent}(i)) = c(\mathrm{parent}(i), i) + \epsilon + d(i)$ の関係が有る．ただし，$\mathrm{parent}(i)$ は，ツリー上で点 i を終点に持つ唯一のアークの始点を表している．全ての点において，距離 $d(i)$ とプライス関数 $p(i)$ は等しい値を持っている．

[誤差定数の決定]

固定されたフローとプライス関数から，フローが ϵ-tight となるような ϵ を求めるには，残余ネットワークの最小サイクル平均，すなわちネットワーク中の全てのサイクル C に対して，コストの平均 $\sum_{(i,j)\in C} c(i,j)/k$ の最小値を用いる（[25]）．このときもとめた最小サイクルの平均を $m(N', c)$ とすると，得られる ϵ の値は $\epsilon = m(N', c)$ となる．

具体的な手順は次のようになっている：

まず，基準となる点 i_0 を決定し，そこから $F_k(i)$ （基準点から点 i までの長さ k の経路のうち，経路の全アークのコストの合計の最小値と比較する．また，長さ k の経路が存在しない場合， $F_k(i)=\infty$ とする．ここで， $\{F_k(i)\}$ に対する次の関係式から，全ての k， i が求まる：

$$F_k(i)=\min_{(i,j)\in A'}\{F_{k-1}(u)+c'(i,j)\} \qquad (k=1,2,\ldots n-1)$$

但し $F_0(i_0)=0$， i_0 以外の全ての点 i で $F_0(i)=\infty$ とする．
これより，次の計算でグラフ N' の最小サイクル平均が求まる：

$$m(N',c)=\min_{i\in V}\max_{(l,i)\in A'}\left\{\frac{F_n(i)-F_k(i)}{n-k}\right\}$$

4.3.2 ファジィグラフ上でのモデル化とその解法

本論文では，存在可能性が最大のアークのみを持った，すなわち最大の閾値によってファジィネットワーク N から求められた非ファジィネットワークに対して，前述した方法で最小コストフロー問題を解きその後，閾値を下げることにより新たに求められるネットワークに対して，前回までの解を基に効率的に解を求める．

ネットワークのファジィネットワークへの拡張として，ネットワークの各アークに，新たに存在可能性 $\mu(i,j)$ の概念を加える．

また，ファジィグラフにおいて，閾値 μ^l によって決定付けられる非ファジィグラフを $G^l=\{V,A^l\}$，とする．ここで， A^l は，閾値 μ^l 以上の存在可能性をもつアークのみで構成された集合である（ $A^l=\{(i,j)\mid (i,j)\in A\mathbf{and}\mu(i,j)\geq\mu^l\}$ ）．グラフ G^l に値を付加したものをネットワーク N^l で表し，そのネットワークに対する最適解を $\{f^l(i,j)\}$ とする．

以上の決定の下で，二目的最小コストフロー問題 **BMCFP** を考える．

BMCFP:

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

但し， $f(i,j)>0$ であるアーク (i,j) の集合を A_F と置き， A 上の各アークにおけるコスト $c(i,j)$ は整数であるとする．

《アルゴリズム 4-5》

Step 0 $l=-1$ とし，全ての点 i において $p(i)=0$ とする．

Step 1 もし閾値が既に最低であれば，アルゴリズムを終了．そうでなければ $l=l+1$ として次のステップへ．

Step 2 グラフ G^l を作成し，対応した値を付加して N^l とする．

Step 3 もし $l=0$ 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 もし実行可能な循環流が存在しないのであれば **Step 1** へ戻る．

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 $\epsilon=0$ ならば，全てのアークの組 (i,j) に対して $f^l(i,j)=x(i,j)$ とし，**Step 1** へ戻る．

Step 7 x が ϵ-最適となるようなプライス関数 $\{p(i)\}$ を求める（[プライス関数の決定]）．

Step 8 x が ϵ-tight となるような ϵ を求める（[誤差定数の決定] より）．

Step 9 $\epsilon=\epsilon/2$ とする．

Step 10 全てのアーク (i,j) に対して $c_p(i,j)=c(i,j)+p(i)-p(j)$ を計算する．

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

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

Step 13 全ての点 i で，$e(i)=\sum_{j\in\beta(i)}x(i,j)-\sum_{l\in\alpha(i)}x(l,i)$ とする．

Step 14 もし全ての点で $e(i)=0$ ならば，**Step 6** へ戻る．

Step 15 プリフロー $\{x(i,j)\}$ にたいする残余ネットワーク N' を作成する．

Step 16 $e(i)>0$ である点 i に対して，もし 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，$\delta=\min(e(i),q(i,j))$ とおいて，$e(i)-e(i)+\delta$，$e(j)=e(j)-\delta$ とする．そうでなければ，**Step 18** へ進む．

Step 17 もし $(i,j) \in A$ 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 $e(i) > 0$ である点 i に対して, もし **Step 14** にみられるようなアークが存在しなければ, $p(i) = \max_{(i,j)\in A'}(p(j) - c'(i,j) - \epsilon)$ とする.

Step 19 もし全ての点で $e(i) = 0$ が成り立つのであれば, **Step 6** へ戻る.

＜計算複雑性＞

このアルゴリズムのオーダーは, 次のように求められる.

従来の, 最小サイクル平均の探索を軸にした最小コストフロー問題のアルゴリズムは, $O(nm \log n\bar{c})$ ($\bar{c}$ はネットワーク全体でのコストの最大値) のオーダーで解を求めることが可能である ([16]). まず, 最小サイクル平均を求める際に $O(nm)$ または $O(nm \log n\bar{c})$ 回の計算が必要となる. 加えて, 各段階で求められる最小サイクル平均は必ず非増加であり, m 回の探索の後に $1 - (1/n)$ 以下になること, 最大値と最小値の差が $\bar{c} - 1/n$ であることから, 上記のオーダーが求められる.

本論文でのアルゴリズムでは, まずはじめに従来のアルゴリズムで問題を解き, その後初期実行解を求める部分を省略した改良版アルゴリズムをもちいる. この部分を異なる存在可能性の数 $|\mu'| - 1 < m$ だけ繰り返すことによって非劣解への更新を行う形をとっている. しかし, 初期実行解を求めるアルゴリズムは最大フロー問題の変形であるため, そのオーダーは上記のものと比べて小さい. よって, アルゴリズム全体でのオーダーは $O(nm^2 \log nbarc)$ となる.

第5章　不確定状況下での輸送問題

輸送問題とは、供給上限の存在する複数の供給点から、必要な物資の量が決められた複数の需要点に、輸送時の総コストが最小になるように物資を分配する問題である。その問題の性質上、条件内の境界値が確定値では表せないことも多い。その為に、従来のモデルにより現実的な新たな概念を付加したモデル、及びその解法が多く提案されてきた。

本論文では，従来の輸送問題，およびそこからの派生である整数ファジィ輸送問題のモデルならびにその解法についての概述と，供給点に存在可能性を付加し，それを供給点から物資を流す量の判断基準のひとつとした新たなモデルを提案する．ここで述べるモデルは，実際に供給点が使用できなくなった，もしくは使用しないほうがよいと意思決定者が判断した際，フローの流れの変化をどう捉えるかにより，大きく分けて二つのパターンが考えられる．一つは，供給点のほうに変化が起きる毎に全体のフローを調整し，常にベストの状態でフローを流すことを目的とするモデル，もう一つは，毎回の調整の手間を省くために，はじめの段階である程度コストを犠牲にし，各供給点からのフローを点単位でバランスよく分配する事を目的とするモデルである．

5.1　輸送問題

供給点集合 $S = \{s_1, s_2, \ldots, s_m\}$ と需要点集合 $T = \{t_1, t_2, \ldots, t_n\}$，そして各供給点と需要点を繋ぐアークの集合 $A = \{(s_i, t_j) \mid s_i \in S, t_j \in T\}$ からなる完全二部グラフを $BG = (S, T; A)$ とする．図5.1は二部グラフの例を表すものとする．

ここで，供給点 s_i から需要点 t_j にフロー $f(s_i, t_j)$ を流すモデルを構築するが，その際，各アークにフローを単位量流す毎にそのアーク固有のコスト $c(i, j)$ がかかるとする．また，s_i から供給されるフローの総量を $f_{s_i} = \sum_{j=1}^{n} f(s_i, t_j)$，$t_j$ に流れ込むフローの総量を $f_{t_j} = \sum_{i=1}^{m} f(s_i, t_j)$ とする．さらに，各供給点 s_i には供給可能上限 $f^*_{s_i}$，需要点 t_j には必要需要下限 $f^*_{t_i}$ が設定され，各点でのフローの総量は常にその制限を受けるものとする．先程の二部グラフ BG に対する輸送問題は，図5.2のように示される．

このモデルでは $\sum_{i=1}^{m} f^*_{s_i} = \sum_{j=1}^{n} f^*_{t_j}$ という関係が成り立っているが，実際

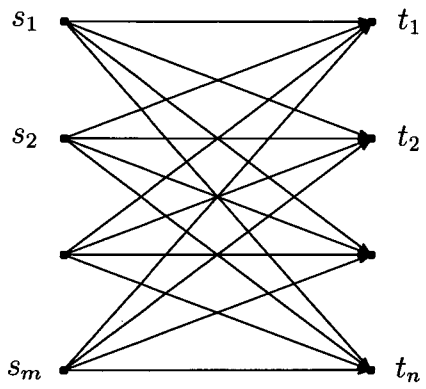

図 5.1: 二部グラフ BG

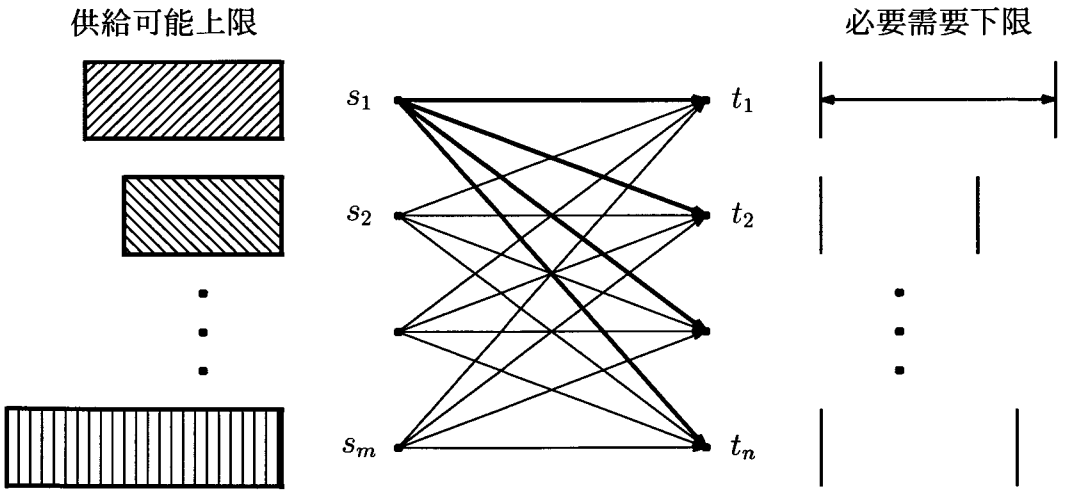

図 5.2: 輸送問題の一例

の問題での制限 $\sum_{i=1}^{m} f_{s_i}^* \geq \sum_{j=1}^{n} f_{t_j}^*$ に対しても，ダミー需要点 t_{n+1}（ $f_{t_{n+1}}^* = \sum_{i=1}^{m} f_{s_i}^* - \sum_{j=1}^{n} f_{t_j}^*$，全ての i に対して $c(s_i, t_{n+1}) = 0$ ）を設けることで，本論文でのモデルに変換が可能である．

これより，全体での輸送コストの最小化を目的とした輸送問題 **TP:**が考えられる．

TP:

$$
\begin{array}{ll}
\text{Maximize} & C^* = \sum_{i=1}^{m} \sum_{j=1}^{n+1} c(s_i, t_j) f(s_i, t_j) \\
\text{s.t.} & \sum_{t_j \in T^*} f(s_i, t_j) = f_{s_i}^* \quad {}^{\forall} i = 1, 2, \ldots, m \\
& \sum_{s_i \in S} f(s_i, t_j) = f_{t_i}^* \quad {}^{\forall} j = 1, 2, \ldots, n+1 \\
& f(s_i, t_j) \geq 0 \qquad {}^{\forall}(s_i, t_j) \in S \times T^* \\
& (i = 1, 2, \ldots m, \quad j = 1, 2, \ldots, n+1)
\end{array}
$$

ここで，$\{f_{s_i}^*\}$ と $\{f_{t_j}^*\}$ は共に整数，$T^* = T \cup \{t_{n+1}\}$，すなわち，基となる需要点集合にダミーの需要点を加えたものとする．

【TP の解法】

上記の輸送問題は，線形計画問題になっているが，その構造上，線形計画問題を解くためのアルゴリズムでは少々非効率的である．この問題に対する効率的な解法としては，既に「北西隅ルール」と「飛び石法」の組合せによる解法が存在する．

北西隅ルールは初期実行可能解を求める方法の一つである．これは，各アークにおけるフローの上限が両端点のフロー制限によってのみ決まること，および供給点の限界の和と需要点の必要量の和が等しいため，必ず全ての点で限界量に等しい分が送られること，という特殊な状況ゆえに成り立つ方法である．まず，行を供給点，列を需要点で表した行列表現では左上（北西）の隅にあたる (s_1, t_1) に，限界量（両端点の制限のうち小さいほう）ちょうどの量のフローを流す．次に，もしもその時点で供給点 s_1 からのフローがそのアークを通じて全て流れた場合，需要点 t_j に対してまだ足りない分は，次の供給点 s_2 からアーク (s_2, t_1) を通じて流すことになる．逆に需要点 t_1 の方必要な量が全て送られてきた場合には，s_1 からの余ったフローは，アーク (s_1, t_2) を通って需要点 t_2 に流れることになる．同様の行程を繰り返し，全ての供給点からの限界までのフローが全需要点に流れた時点でのフローが初期実行可能解となる．

飛び石法は，初期実行可能解を基に解を逐次更新していき，最終的に最適解を求める方法である．まず，基底変数（ここでは，正のフローが流れているアークの流量） $f(s_i, t_j)$ に対するコスト $c(s_i, t_j)$ を双対変数 u_i, v_j （ $i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, n$ ）の和として，$c(s_i, t_j) = u_i + v_j$ のように表す．この変数は，元の基

底変数との個数の差より自由度が1余るため，任意の変数，ここでは u_1 の値を固定（$u_1 = 0$）することにより，コストとの関係から全ての値を求めることができる．この双対変数の値から，非基底変数（直接は解の値に影響を与えない状況にある変数，ここでは，フローの流れていないアークを指す）に対応するコストを修正する．すなわち，全ての点の組 (i, j) に対して $c'(s_i, t_j) = c(s_i, t_j) - u_i - v_j$ の形で修正コストを求める．次に，$\min\{c'(s_i, t_j) \mid i = 1, 2, \ldots, m, j = 1, 2, \ldots, n\} = c'(s_p, t_q)$ の式によって，修正コストの最小値を持つアーク (s_p, t_q) を求める．もし $c'(s_p, t_q) < 0$ であれば，アーク (s_p, t_q) に他のアークからフローを移す事によって総コストを減少させることができるが，全ての修正コストが正または0であればこれ以上コストを減少させるようにフローの組を変更させることができないため，その時点での解を最適解とする．

しかし，各アークにおいてフローは必ず非負でなければならないので，$f(s_p, t_q)$ の値を増加させるためには，ある h, k で $f(s_p, t_h)$，$f(s_k, t_q)$，$f(s_k, t_h)$ が基底変数であるものを探し出し，それらの中だけでフローを調整する必要がある．ここで，$\theta = \min\{f(s_p, t_h), f(s_k, t_q)\}$ と置くと，$f(s_p, t_h)$ と $f(s_k, t_q)$ のどちらかが $f(s_p, t_q)$ の代わりに非基底変数となる．

その後，$f(s_p, t_q) = f(s_p, t_q) + \theta$，$f(s_p, t_h) = f(s_p, t_h) - \theta$，$f(s_k, t_q) = f(s_k, t_q) - \theta$，$f(s_k, t_h) = f(s_k, t_h) + \theta$ とすることでフローの更新を行い，その基底変数の組合せを基にさらに修正コストを計算する．これを値の更新が不可能な状態になるまで繰り返すことにより，最適解を求めることができる．

【必要需要下限の整数ファジィ制限への拡張】

TPでは，供給点や需要点に対する制限は，供給可能な量（上限）や必要な物資の量（下限）を確定値で示していたが，現実問題では，そのようにはっきりした値をとることは少ない．特に工場から各都市への分配という問題のモデル化の場合，供給点に比べて需要点の制限はいっそう曖昧になることが多くなる．また，工場からの輸送の際，理論上最適だが大型輸送機関を用いて運ぶには中途半端な量を運ぶよりは，一見少々無駄に見えてもまとめて運んだほうが結果的に効率的であることが多いため，需要点の制限として「必ず整数値の物資を必要とする」を組み込むほうが実際的である．よって本論文では，TPに対して，需要点におけるフロー制限を確定値から満足度関数を用いたファジィ制限にし，さらにそのとり得る値を整数値のみとするモデルを考えた．ここで，TPの目的である「総コストの最小化」を決められた予算内でフローを流すという条件の一つに止めておき，その代わりにその予算内で以下に需要点にバランスよく分配でき，各需要点でなるべく不満が出ないようにするかを，このモデルの主目的とする．

上記のモデルを構築するために必要な記号を，既出のものに加えて幾つか定義しておく必要がある．

需要点での制限であるが，TP での確定値から，次のような関係式によって決定付けられる満足度関数へと緩和させる：

$$\mu_{t_j}(f_{t_j}) = \begin{cases} 0 & f_{t_j} \leq d_{t_j} \\ \frac{f_{t_j}-d_{t_j}}{e_{t_j}-d_{t_j}} \equiv F_{t_j}(f_{t_j}) & d_{t_j} < f_{t_j} < e_{t_j} \\ 1 & f_{t_j} \geq e_{t_j} \end{cases} .$$

ここで，f_{s_i} と f_{t_j} は整数であるとする．

この満足度関数の値が大きいほど，送られてきた物資の量に対する需要点側の制限をよく満たしていると言えるので．ネットワーク全体での目的は，各需要点での満足度の最小値を最大化する，すなわち一番物資の量に不満を持っている需要点の不満を最小限に抑えることとなる．

このことより，整数ファジィ輸送問題 （IFTP）は次のように表される：

IFTP:

$$\begin{array}{ll} \text{Maximize} & \min_j\{\mu_{t_j}(f_{t_j})\} \\ \text{s.t.} & \sum_{t_j \in T} f(s_i,t_j) = f_{s_i} \quad {}^\forall i = 1,2,\ldots,m \\ & \sum_{s_i \in S} f(s_i,t_j) = f_{t_i} \quad {}^\forall j = 1,2,\ldots,n \\ & \sum_{i=1}^{m}\sum_{j=1}^{n} c(s_i,t_j) f(s_i,t_j) \leq \bar{C} \\ & f(s_i,t_j) \geq 0 \qquad {}^\forall (s_i,t_j) \in S \times T \\ & f_{s_i}, f_{t_j}\text{: integer} \\ & (i = 1,2,\ldots m, \quad j = 1,2,\ldots,n) \end{array}$$

ここで，$\bar{C} > 0$ は総コスト $C = \sum_{i=1}^{m}\sum_{j=1}^{n} c(s_i,t_j) f(s_i,t_j)$ の予算とする．

[解法]

この問題の最適解を得る為に，次のような段階を踏むこととする．

まず，予算条件を無視した状態で，最適解を計算する．送られる物資の総量は変化しないため，最適値はその物資を全ての需要点の満足度が等しくなるように分配するときに得られる（完全割り当て：Perfect Assignment）．次に，そのときの各需要点での下限が整数値になるように値の調整を行い，その後かかったコストが予算をオーバーしていないかの判定を行う．もしも予算を超えていなければ，その時点での解が最適解となる．そうでない場合には，今度は満足度条件のほうを緩和して，総コストが下がるように調整していかなければならない．

[Perfect Assignment]

予算および整数条件を無視した状態で，はじめに満足度の最小値が最大になるようなフローの組み合わせを求める．

全需要点の満足度が全て同じ値 α であると仮定すると，輸送問題の定義より次の関係式が成り立つ：

$$\begin{aligned} \sum_{s_i \in S} f_{s_i} &= \sum_{t_j \in T} f_{t_j} \\ &= \sum_{j=1}^{n} \{\alpha d_{t_j} + (1-\alpha) e_{t_j}\} \end{aligned}$$

この式より，全ての満足度が等しい場合の満足度 $\bar{\alpha}$ が求まる：

$$\begin{aligned} \bar{\alpha} &= \frac{\sum_{i=1}^{m} f_{s_i} - \sum_{j=1}^{n} e_{t_j}}{\sum_{j=1}^{n} (d_{t_j} - e_{t_j})}, \\ f_{t_j} &= F_{t_j}^{-1}(\bar{\alpha}) \end{aligned}$$

この値を満足度関数の逆関数に代入することにより，予算・整数両条件を無視した状態での各需要点における必要な物資の量が求まる．

[Adjustment of the flow]

次に，先程得られた解に対し，各需要点での整数条件を満たすように調整を行う．はじめに，各需要点での制限が全て整数かどうかを判定し，もしも一つでも整数では表されないものが存在する場合には，全ての制限を $\tilde{f}_{t_j} = \lceil f_{t_j} \rceil$, ($j = 1, \ldots, n$) のように整数化する．ここで，$\lceil f_{t_j} \rceil$ は，f_{t_j} より小さくない最小の整数を表す．この時点では，必要量と供給可能量との間にギャップ（その大きさを k とする）ができる．このとき，ギャップは $k = \sum_{j=1}^{n} \lceil f_{t_j} \rceil - \sum_{i=1}^{m} f_{s_i}$ で表される正の整数である．

条件を満たすにはギャップを埋める必要があるため，需要点での満足度を犠牲にしても必要輸送量を減少させる必要があるが，その際，「満足度の最小値が最大になる」事を満たした上で，すなわち，ある需要点の満足度を減少させることによる全体の満足度の減少を最小限に抑えられるように，必要量を減らす需要点を選択する必要がある．

ここで，最初の状態から必要量の下限を l 下げた場合の需要点 t_j の満足度を $W_{j,l}$ とし，この時点で全組み合わせについて求めておくと，繰り返し満足度を求める際に効率的である．まず全ての組合せに対して $W_{j,l}$ を求め，それを満足度が大きい順に並べ替え，$1 \geq \mu^{(1)} > \mu^{(2)} > \cdots > \mu^{(r)} > 0$ と $\mu^{(l)}$ の形で示す．後は，満足度が大きい順に k 個の W を選び，それに対応する需要点の下限を順に下げていくことで，条件に見合った解を得ることができる．なお，k 番目が複数あっ

た場合は，そのうちのどれを選んでも解自体には変化がないので，意思決定者によって任意に選ぶことができるが，供給点の限界との関係により，ここは，同率 k 位の W に対応する需要点の下限は全て削るものとする．

[TP の組合せへの変換]

上記の手順により，暫定的に，各需要点 t_j に必要な物資の量の下限が確定値 $f^*_{t_j}$ として定まるため，その値を基に，IFTP を TP として一度解く．ただし，この時点で $\sum_{i=1}^{m} f_{s_i} > \sum_{j=1}^{n} f^*_{t_j}$ となっている場合は，前述の通りダミー需要点を設置することで $\sum_{i=1}^{m} f_{s_i} = \sum_{j=1}^{n+1} f^*_{t_j}$ の形に変換し，通常の TP として解くこととする．

その後，ここで得られた解 $C^* = \sum_{i=1}^{m} \sum_{j=1}^{n+1} c(s_i, t_j) f(s_i, t_j)$ と予算 $\bar{C}$ を比較し，$C^* \leq \bar{C}$ ならばこの時点の解を最適解とする．そうでなければ，先程の $W_{j,l}$ を系列を基に，さらに需要点での必要な物資量の下限を削らなければならない．

この手順により，最終的に，予算条件と整数条件を満たした上での最適解が得られる．

5.2 ファジィグラフ上でのモデル化とその解法

5.2.1 供給点の増減によってフローを変更する場合

ここでは，供給点が存在しなくなる，もしくは意思決定者の決定により物資の供給を断ることになった場合を想定し，考えられる残存供給点の組合せに対する物資の流れを予め求めておくことを主眼に置いたモデルを考える．

まず，このモデルを構築するために必要な記号を，既出のものに加えて幾つか定義しておく必要がある．

各供給点の存在可能性 μ_{s_i} を供給点 s_i に付加し，扱うグラフをファジィグラフへと拡張する．また，このモデルでは，供給点の存在可能性が μ_{s_1} から降順に並んでいるとしても一般性が保たれている．

BIFTP:

$$
\begin{array}{ll}
\text{Maximize} & \min_j\{\mu_{t_j}(f_{t_j})\} \\
\text{Maximize} & \min_{s_i \in S_F} \mu_{s_i} \\
\text{s.t.} & \sum_{t_j \in T} f(s_i, t_j) = f_{s_i} \quad {}^\forall i \in S_F \\
& \sum_{s_i \in S} f(s_i, t_j) = f_{t_i} \quad {}^\forall j = 1, 2, \ldots, n \\
& \sum_{i=1}^{m} \sum_{j=1}^{n} c(s_i, t_j) f(s_i, t_j) \leq \bar{C} \\
& f(s_i, t_j) \geq 0 \qquad {}^\forall (s_i, t_j) \in S \times T \\
& f_{t_j} \text{: integer} \\
& (j = 1, 2, \ldots, n)
\end{array}
$$

ここで，$\bar{C} > 0$ は問題の総コスト $C = \sum_{i=1}^{m} \sum_{j=1}^{n} c(s_i, t_j) f(s_i, t_j)$ の予算，S_F を，正のフローを供給する供給点の集合とする．また，f_{s_i} は整数値を取るものとする．

[解法]

本論文では，複数の閾値によって上記のファジィネットワークを複数の非ファジィネットワークとして表し，各段階に対してIFTPの解法を適応することによって，複数の閾値に対応する複数の解を得る方法を取る．

まず，存在可能性が最大の供給点のみを含む，すなわち閾値が最大の状態での部分グラフを求め，その非ファジィネットワークにおける最適解を，IFTPを用いて得る．次にネットワークの閾値を下げ，新たに存在可能性が比較的大きい供給点を含むネットワークを構築する．この解も先程と同様に求めるが，2度目以降では，初期実行可能解を前段階での最適解とし，そこから解を更新することとなる．

最終的には，全ての閾値に対する解を求めた状態でアルゴリズムが終了する．ただし，複数の解から適当なものを選択することは，意思決定者の意思に任せる．

《アルゴリズム 5-1》

Step 0 $L = 0$ とする．

Step 1 供給点の存在可能性の異なる値を降順で並べ替え，$1 \geq \mu^0 > \mu^1 > \mu^2 > \ldots > \mu^{L_s} > 0$ と表す．ここで，L_s は供給点の存在可能性の異なる値の数とする．

Step 2 $S^0 = \{s_i \mid \mu_{s_i} = \mu^0\}$ とする．

Step 3 $\bar{\alpha} = \frac{\sum_{s_i \in S^0} f_{s_i} - \sum_{j=1}^{n} e_{t_j}}{\sum_{j=1}^{n} (d_{t_j} - e_{t_j})}$ とする．

Step 4 $f_{t_j} = c_j(1-\bar{\alpha}) + d_j\bar{\alpha}$ とする.

Step 5 もしも全ての i に対して f_{t_j} が成り立つのならば, **Step 15** へ移動する.

Step 6 全ての j に対して, $f_{t_j} = \lceil f_{t_j} \rceil$ とする.

Step 7 $k = \sum_{j=1}^{n} f_{t_j} - \sum_{s_i \in S^0} f_{s_i}$ とする.

Step 8 全ての j と l の組合せに対して, $W_{j,l}$ の値を求める.

Step 9 求めた $W_{j,l}$ の値を降順で並び替え, $1 \geq \mu^{(1)} > \mu^{(2)} > \cdots > \mu^{(r)} > \mu^{(r+1)} = 0$ として表す.

Step 10 $F = 0, u = 1$ とする.

Step 11 もしも $^{\forall,\exists}\{W_{j,l} \mid W_{k,j} = \mu^{(u)}\}$ ならば, $f_{t_j} = l$ とする.

Step 12 $F = F + \sum_{W_{j,l}=\mu^{(u)}} 1$ とする.

Step 13 $u = u + 1$ とする.

Step 14 もしも $F < k$ ならば **Step 10** に戻る.

Step 15 ダミーへのフローを $f_{t_{n+1}} = \sum_{s_i \in S^0} f_{s_i} - \sum_{j=1}^{n} f_{t_j}$ とし, 全ての i に対して $c(s_i, t_{n+1}) = 0$ とする.

Step 16 全ての i, j に対して $g_{s_i} = f_{s_i}$, $g_{t_j} = f_{t_j}$ と置く.

Step 17 $i = 1, j = 1$ とする.

Step 18 $f(s_i, t_j) = \min\{g_{s_i}, g_{t_j}\}$ とする.

Step 19 $g_{s_i} = g_{s_i} - f(s_i, t_j)$, $g_{t_j} = g_{t_j} - f(s_i, t_j)$ とする.

Step 20 もしも $i = m$ 且つ $j = n+1$ であれば **Step 24** へ移動する.

Step 21 もしも $g_{s_i} = 0$ 且つ $i < m$ ならば $i = i + 1$ とする.

Step 22 もしも $g_{t_j} = 0$ 且つ $j < n+1$ ならば $j = j + 1$ とする.

Step 23 **Step 16** に戻る.

Step 24 この時点での残余ネットワーク $BG' = (S^L, T; A')$ を求める. (ここで, $A' = A \cup \{(t_j, s_i) \mid f(s_i, t_j) > 0\}$ とする)

Step 25 BG' 上の各アーク (t_j, s_i) に対してコスト $c'(t_j, s_i) = -c(s_i, t_j)$ を付加する．アーク (s_i, t_J) に対しては $c'(s_i, t_j) = c(s_i, t_j)$ とする．

Step 26 BG' 上のサーキット $CC = \{s_i, (s_i, t_j), t_j, (t_j, s_{i'}), \ldots, (t_{j'}, s_i), s_i\}$ から，各アークのコスト c' の和が負になるものを一つ選択する．

Step 27 **Step 26** の条件を満たす CC が BG' 上に存在するのであれば，$\Delta = \min_{(t_j, s_i) \in CC} f(s_i, t_j)$ とし，CC 上の各アーク (s_i, t_j) に対しては $f(s_i, t_j) = f(s_i, t_j) + \Delta$，$(t_j, s_i)$ に対しては $f(s_i, t_j) = f(s_i, t_j) - \Delta$ とし，**Step 24** に戻る．

Step 28 **Step 26** の条件を満たす CC が BG' 上に存在しないのであれば，$C^* = \sum_{(s_i, t_j) \in A} c(s_i, t_j) f(s_i, t_j)$ とする．

Step 29 もしも $C^* \leq \bar{C}$ ならば，$\mu_t^L = \min_j \{\mu_{t_j}(f_{t_j})\}$ として **Step 32** へ移動する．

Step 30 もしも $^{\forall} j, ^{\exists} \{W_{j,l} \mid W_{j,l} = \mu^{(u)}\}$ ならば $f_{t_j} = f_{t_j} - 1$, $f_{t_{n+1}} = f_{t_{n+1}} + 1$, $f(s_{i'}, t_j) = f(s_{i'}, t_j) - 1$ そして $f(s_{i'}, t_{n+1}) = f(s_{i'}, t_{n+1}) - 1$ とする．ここで，$s_{i'}$ は，$\{s_i \mid f(s_i, t_j) > 0\}$ である供給点のうち，コスト $c(s_i, t_j)$ が最大のものから任意に一つ選択したものであるとする．

Step 31 **Step 24** に戻る．

Step 32 $(\{f(s_i, t_j)\}; \mu_t^L, \mu^L)$ を非劣解集合 Q に加える．

Step 33 もしも全ての $L_s' = 1, 2, \ldots, L_s - 1$ に対して $\mu_t^L \leq \mu_t^{L_s'}$ ならば，$(\{f(s_i, t_j)\}; \mu_t^L, \mu^L)$ を Q から削除する．

Step 34 もしも $L = L_s$ ならばこのアルゴリズムを終了．そうでなければ $L = L+1$ とする．

Step 35 $S^L = \{s_j \mid \mu(s_i) \geq \mu^L\}$ とする．

Step 36 $\{f'(s_i, t_j)\} = \{f(s_i, t_j)\}$ とする．

Step 37 アーク集合 $A'' = \{(s_{ii}, t_{jj}) \mid s_{ii} \in S^L / S^{L-1},\ f(s_i i, t_{j+1}) < f_{s_i i}$ 且つ $\mu_{t_{jj}}(f_{t_j}) = \min_{t_j \in T} \mu_{t_j}(f_{t_j})\}$ を求める．

Step 38 もしも $A'' = \phi$ ならば，$\mu_t^L = \min_j \{\mu_{t_j}(f_{t_j})\}$ として **Step 32** に戻る．

Step 39 $(s_{i'}, t_{j'}) = \mathrm{argmin}\{c(s_i, t_j)(s_i, t_J) \mid (s_i, t_j) \in A''\}$ となるアーク $(s_{i'}, t_{j'})$ を求める．

Step 40 $f(s_{i'},t_{j'})=f(s_{i'},t_{j'})+1, f(s_{i'},t_{j+1})=f(s_{i'},t_{j+1})-1$ とする.

Step 41 もしも $C^*=\sum c(s_i,t_J)f(s_i,t_J)\leq \bar{C}$ ならば **Step 39** に戻る.

Step 42 この時点での残余ネットワーク $BG'=(S^L,T;A')$ を求める（ここで, $A'=A\cup\{(t_j,s_i)\mid f(s_i,t_j)>0\}$ ）とする.

Step 43 BG' 上の各アーク (t_j,s_i) に対してコスト $c'(t_j,s_i)=-c(s_i,t_j)$ を付加する.

Step 44 BG' 上のサーキット $CC=\{s_i,(s_i,t_j),t_j,(t_j,s_{i'}),\ldots,(t_{j'},s_i),s_i\}$ から, 各アークのコスト c' の和が負になるものを一つ選択する.

Step 45 **Step 44** の条件を満たす CC が BG' 上に存在するのであれば, $\Delta=\min_{(t_j,s_i)\in CC} f(s_i,t_j)$ とし, CC 上の各アーク (s_i,t_j) に対しては $f(s_i,t_j)=f(s_i,t_j)+\Delta$, (t_j,s_i) に対しては $f(s_i,t_j)=f(s_i,t_j)-\Delta$ とし, **Step 42** に戻る.

Step 46 **Step 44** の条件を満たす CC が BG' 上に存在しないのであれば, $C^*=\sum_{(s_i,t_j)\in A} c(s_i,t_j)f(s_i,t_j)$ とする.

Step 47 もしも $C^*\leq\bar{C}$ ならば, $\mu_t^L=\min_j\{\mu_{t_j}(f_{t_j})\}$ として **Step 36** へ進む.

Step 48 $\{f(s_i,t_j)\}=\{f'(s_i,t_j)\}$ とする.

Step 49 この時点での残余ネットワーク $BG'=(S^L,T;A')$ を求める（ここで, $A'=A\cup\{(t_j,s_i)\mid f(s_i,t_j)>0\}$ とする）.

Step 50 BG' 上の各アーク (t_j,s_i) に対してコスト $c'(t_j,s_i)=-c(s_i,t_j)$ を付加する.

Step 51 BG' 上のサーキット $CC=\{s_i,(s_i,t_j),t_j,(t_j,s_{i'}),\ldots,(t_{j'},s_i),s_i\}$ から, 各アークのコスト c' の和が負になるものを一つ選択する.

Step 52 **Step 51** の条件を満たす CC が BG' 上に存在するのであれば, $\Delta=\min_{(t_j,s_i)\in CC} f(s_i,t_j)$ とし, CC 上の各アーク (s_i,t_j) に対しては $f(s_i,t_j)=f(s_i,t_j)+\Delta$, (t_j,s_i) に対しては $f(s_i,t_j)=f(s_i,t_j)-\Delta$ とし, **Step 48** に戻る.

Step 53 **Step 51** の条件を満たす CC が BG' 上に存在しないのであれば, $C^*=\sum_{(s_i,t_j)\in A} c(s_i,t_j)f(s_i,t_j)$ とする.

Step 54 もしも $C^*\leq\bar{C}$ ならば, $\mu_t^L=\min_j\{\mu_{t_j}(f_{t_j})\}$ として **Step 32** へ戻る.

＜計算複雑性＞

このアルゴリズムによって解を求めるために必要な計算量は，次のようにして求められる．

まず，初期実行可能解は整数ファジィ輸送問題（IFTP）の解法と同じアルゴリズムを用いて求めることが可能なため，その計算量は $O(\max\{ct(m,n)\log r, r\log r\})$ である （[50]）．ここで， $ct(m,n)$ は m 個の供給点と n 個の需要点からなる輸送問題（TP）をとくために必要な計算量， r を異なる $\mu^{(l)}$ の総数であるとする．

また，解の更新時に，各段階で「初期実行解が既に求まっている状態でのIFTPの解法」と同等の計算量が必要となる，IFTPの初期実行可能解は $O(ct(m,n))$ で求まるため，初期解の計算の有無にかかわらず，実際のオーダーは $O(\max\{ct(m,n)\log r, r\log r\})$ である．よって，全体でのアルゴリズムの計算複雑性は， $L_s \leq m$ より， $O(\max\{m\, ct(m,n)\log r, mr\log r\})$ となる．

5.2.2 供給点の増減によってフローを変更しない場合

ここでは、需要点における物資の必要量を確定値からファジィ制約に拡張する。加えて、供給点に存在可能性を付加し、それに対して各需要点に物資が供給される可能性を考える。つまり、存在可能性の低い供給点からの物資は供給される可能性が低いとし、それが存在しない場合でも各需要点に供給される物資に極端な偏りが生じないようなフローを予め想定しておくことが必要となる。

本論文では、「需要点における満足度を重視した分配を行う供給点の存在可能性の下限の最大化」と、「一定量のフローが保証される状態での総コストの最小化」の二目的問題を提案し、その解法として、一方のみを完全に満たす解の作成と、他方の条件を満たす方向への更新する方法を提案する。

【定式化】

ここでは，次のような二目的整数ファジィ輸送問題 **BIFTP** を提案する。

BIFTP:

$$
\begin{aligned}
&\text{Minimize} \quad \textstyle\sum_i \sum_j c(s_i,t_j) f(s_i,t_j) \\
&\text{Maximize} \quad \min_{s_i \in S_B} \mu_{s_i} \\
&\text{s.t.} \qquad \textstyle\sum_{t_j \in T} f(s_i,t_j) = f_{s_i} \quad {}^{\forall} i = 1,2,\ldots,m \\
&\qquad\quad \textstyle\sum_{s_i \in S} f(s_i,t_j) = f_{t_i} \quad {}^{\forall} j = 1,2,\ldots,n \\
&\qquad\quad f(s_i,t_j) \geq 0 \qquad {}^{\forall} e_{ij} \in E \\
&\qquad\qquad (i = 1,2,\ldots m, \quad j = 1,2,\ldots,n+1)
\end{aligned}
$$

但し、各需要点の満足度の格差を最小にする、即ちバランス条件を満たすフローを分配する供給点 s_i の集合を S_B とし、問題の第一、第二目標を無視した状態での需要点の最適整数割当の値を f_{t_j} と置く。このときの f_{t_j} および f_{s_i} は，共に整数値を取るものとする.

また，このモデルでは，供給点の存在可能性が μ_{s_1} から降順に並んでいるとしても一般性が保たれている.

《アルゴリズム 5-2》

Step 0 $L=1$ とする.

Step 1 異なる μ_{s_i} の値を昇順に並び替え， $0<\mu^1<\mu^2<\ldots<\mu^{L_s}\leq 1$ と置く. ここで，L_s は供給点の異なる存在可能性の数である.

Step 2 存在可能性が μ^l である供給点を s^l とする（同じ値を持つ供給点が複数存在する場合には， $s^l, s'^l, \ldots$ とする）.

Step 3 各需要点へのフロー f_{t_j} のとり得る全ての値に対して，予め満足度 $\mu_{t_j}(f_{t_j})$（$W_{j,l}$ に対応）を求めておく.

Step 4 S_B に含まれていない供給点のうち，存在可能性が最大のものを S_B に加える.

Step 5 $f=F_{t_j}^{-1}(\alpha)$ を $\alpha=\mu_{t_j}(f)$ の逆関数とし，式 $\sum_{s_i\in S_B} f_{s_i}=\sum_{t_j\in T} F_{t_j}^{-1}(\alpha^L)$ を満たす α^L を求める.

Step 6 各需要点に対して $f_{t_j}^L=F_{t_j}^{-1}(\alpha^L)$ とする.

Step 7 もし，需要点の満足度の下限が 0 であるならば，**Step 3** へ戻る.

Step 8 もし，全ての需要点 f_j で $f_{t_j}^L$ が整数値を持つならば **Step 10** に進む.

Step 9 $f_{t_j}^L$ の中に整数ではないものが存在するならば， $\tilde{f}_{t_j}=\lceil f_{t_j}\rceil$ （$j=1,\ldots,n$），$h=\sum_{t_j\in T}\lceil f_{t_j}\rceil-\sum_{s_i\in S_B} f_{s_i}$ とおき，需要点における満足度の下限の減少量が最小に抑えられるように，物資の必要量の下限値から合計 h だけ減らす.

Step 10 もし，まだ S_B に含まれていない供給点が存在するならば，各需要点において $f_{t_j}^L=\tilde{f}_{t_j}^L$ と代入した後に $L=L+1$ とし，**Step 3** へ戻る.

Step 11 S_B の中で存在可能性が大きい供給点を順に2個選択し，それらをコスト重視の供給点の集合 S_C に移動させ，L の値を2減らす．

Step 12 S_B 内の供給点のみに対応した，各需要点への必要な物資の量 $f_{t_j}^{(L)}$ を求める．

Step 13 供給点 $s_{(m-L)}, s_{(m-L+1)}, \ldots, s_m$ からのフローに対して，飛び石法を用いて総コストの最小化を行う．

Step 14 **Step 13** の解を非劣解の一つとし，Q に加える．

Step 15 $L = L + 1$ とする．

Step 16 もし $S_B \neq \emptyset$ ならば，S_B の要素のうち，存在可能性が最大の供給点を S_C に移動させて **Step 13** へ戻る．そうでなければ，アルゴリズムを終了させる

＜計算複雑性＞

このアルゴリズムの計算複雑性は次のようにして求められる．

初期実行解を求める際に必要な計算量は，IFTP における PA と，その後の整数値への調整にかかる計算量（$O(\max\{ct(m,n), M^2 \log M\})$，([50])）に供給点の数 m を掛け合わせたものである．ここで，$M = m + n$（供給点と需要点の合計）とする．ただし，$\mu_{t_j}(f_{t_j})$ は最初の段階で一度だけ求めることになるため，実際の計算量は $O(\max\{m\, ct(m,n), M^2 \log M\}$ となる．

その後解の更新を行うが，その L 段階目では，$L+1$ 個の供給点と n 個の需要点からなる二部グラフでの TP（初期実行解は既知）を求めるアルゴリズムを用いる．これを $m-1$ 回繰り返すため，全体の計算量は $O(\max\{m\, ct(m,n), M^2 \log M\} + m\, ct(m,n)) \sim O(\max\{m\, ct(m,n), M^2 \log M\})$ となる．

第6章　最小スパニングツリー問題

スパニングツリー問題は，情報網の設置コストの最小化を計るなど，多くの応用を持つ問題である（[15]，[26]）．本論文では，より現実に近いモデルとして，ファジィネットワーク上で，従来の目標であるツリー設置コストの削減および設置する枝の存在可能性（安全性，満足度等も同様）の向上の二目的とした問題と，従来のスパニングツリーの概念自体を拡張し，「スパニングツリーである」ことを条件としたモデルの，意思決定者の判断による緩和を可能にするものの二つについて述べる．

まず，最小スパニングツリー問題のモデルを表す際に必要な幾つかの記号の定義を行う．

n 個の点からなる点集合 $V = \{1, 2, \ldots, n\}$，m 本の無向枝からなる枝集合 $E = \{e\} \subseteq \{(i,j) \mid (i,j) \in V \times V\}$ で表されるグラフを $G = (V, E)$ とする．ここで，G は単純な連結グラフであるとする．また，E の各要素 e に重み（設置コスト，二点間の距離等を表す）$w(e)$ が付加されている．このグラフ G は連結グラフであるから，部分グラフとしてのスパニングツリー $T \subseteq G$ を求めることが可能である．ここで，グラフ G におけるスパニングツリー T とは，

(A) G 上の全ての点を連結する極大グラフ
（V に含まれる任意の2点 (i,j) 間に経路 $p(i,j)$ が必ず存在する）

(B) 内部に閉路を含まない G の部分グラフ
（任意の2点 (i,j) 間にルートの異なる経路が2本以上存在しない）

という条件を満たす G の部分グラフの全てを指す．

これより，次のような最小スパニングツリー問題STPを考えることができる．

STP:

$$\begin{array}{ll} \text{Minimize} & \sum_{e \in T} w(e) \\ \text{s.t.} & T \text{はスパニングツリー} \end{array}$$

【欲張り法】

最小スパニングツリーを求める手段の一つとして,「欲張り法」がある（[4]）. これは，とにかく設置コストの小さい枝から順にツリーの一部に加えていく，という方法である．このアルゴリズムは段階を踏んでツリーを形成することになるが，k 段階目のツリー $T^{(k)}$ に含まれる点の集合を $X^{(k)}$，そのカットセット（連結する点の一方が $X^{(k-1)}$，もう一方が $\bar{X}^{(k-1)} = V - X^{(k-1)}$ に属する枝の集合）を $C^{(k)}$ とする．また，$T^{(k)}$ に属する枝の集合を $F^{(k)}$ とする．

《アルゴリズム 6-1》

Step 0 $k = 1$ とする．

Step 1 V の内の一点 $v^{(0)}$ を任意に選択する．

Step 2 $X^{(0)} = \{v^{(0)}\}$, $F^{(0)} = \emptyset$ とする．

Step 3 $C^{(k)}$ を求める．

Step 4 $C^{(k)}$ の要素のうち，重みが最小の枝 $e^{(k)}$ を求める．

Step 5 $v^{(k)}$ を，$e^{(k)}$ が連結している点のうち，$X^{(k-1)}$ に属さないほうの点とする．

Step 6 $X^{(k)} = X^{(k-1)} \cup v^{(k)}$, $F^{(k)} = F^{(k-1)} \cup e^{(k)}$ とする．

Step 7 もし，$k = n-1$ ならばアルゴリズムを終了．そうでなければ，$k = k+1$ として **Step 3** へ戻る．

このアルゴリズムの結果から，スパニングツリー $T = (V, A^T = F^{(n-1)})$ が得られる．

6.1 ファジィグラフ上でのモデル化とその解法

ファジィネットワーク上の問題にモデルを拡張するため，グラフ G の各枝 e に存在可能性 $\mu(e)$ を付加する．

これより，STP を拡張した二目的最小スパニングツリー問題 BSTP を考えることができる．

BSTP:

Minimize	$\sum_{e\in T} w(e)$
Maximize	$\min_{e\in T} \mu(e)$
s.t.	T はスパニングツリー

【解法】

まず全ての枝を対象に最小スパニングツリー問題を解く．その後，一定の条件のもとでツリーを更新していき，それぞれのツリーを非劣解の一つとする．

最初の段階の解であるツリーは，前述の欲張り法を用いて求める．次に，存在可能性の低い枝から順に変換し，ツリーを更新していく．これは，従来の方法([26])を拡張したものである．

[「k 番目に重みの小さいスパニングツリー」の作成]

ここでは，第 i 段階のツリー T_i，すなわち G に部分グラフとしてのスパニングツリー全体の内で i 番目に総コストが小さいスパニングツリーを，T_1 から T_{i-1} までの，それまでの段階で得られたスパニングツリーに対し，幾つかの枝を削除，追加することによって求める方法について述べる．

まず，その方法を述べるために必要な記号の定義を行う．

μ^l ：枝の存在可能性 $\mu(e)$ の異なる値を昇順に並べ替えたときの，l 番目に小さい値．$0 < \mu^1 < \mu^2 < \ldots < \mu^K \leq 1$ とする．ただし，K は異なる $\mu(e)$ の値の数である．

$T^l(1)$ ：内部に含まれる枝の存在可能性の下限が μ^l という条件の基での最小スパニングツリー．

$T^l(i)$ ：アルゴリズム内で，$T^l(1)$ から $T^{l+1}(1)$ を求める際に生成されるスパニングツリー．

$W^l(i) = \sum_{e\in T^l(i)} w(e)$ ：ツリー $T^l(i)$ の総コスト．$i = 1$ の時には W^l と表記する．

ツリーの変換 $|e, f|$ ：あるスパニングツリー T に対して，それに属する枝の一本 e を削除し，その後分割された二つの部分グラフを，別の枝 f を追加することで先程とは異なるグラフ T' にすること．
ツリーの定義上，T' は必ずスパニングツリーになる．

$IN^l(i)$ ：ツリー $T^l(i)$ を更新する際，削除の対象に入らない枝の集合．

$OUT^l(i)$ ：ツリー $T^l(i)$ を更新する際，追加の対象に入らない枝の集合．

$P^L(i) = \{T^l(k) \mid k > j, \quad IN^l(i) \subset T^l(i), \quad OUT^l(i) \subset E - T^l(i)$ ：$T^l(i)$ が求まった状態で，そこからさらなる更新によって得られるツリーの集合．

$Q^l(i) = \{(e', f'), r\}$ ：$P^l(i)$ で示される更新 $|e, f|$ のうち，総コストの増加量が最も小さい変換．
ここで，各 $f \in E - T^l(i) - OUT^l(i)$．$e \in T^l(i) - IN^l(i)$ に対して，$r = w(f) - w(e)$ となる．

$m^l(i, \mu)$ ：$T^l(i)$ において，その存在可能性が μ である枝の本数．

これらの記号を用いて，アルゴリズムを次のように記述する．

《アルゴリズム 6-2》

Step 0 $l = 1$, $i = 1$, $IN^1(1) = \emptyset$, $OUT^1(i) = \emptyset$ とする．

Step 1 μ^1, $m^1(1, \mu^1)$ を求める．

Step 2 $T^1(1)$ （存在可能性条件を無視した状態での最小スパニングツリー）を，欲張り法を用いて求める．

Step 3 $(T^l(1), \mu^l)$ を非劣解の一つとして記録する．

Step 4 $\mu(e) \leq \mu^l$ となる枝のうち，$T^l(i)$ に含まれないものを $OUT^l(i)$ に追加する．

Step 5 $T^{l+1}(1) = T^l(i)$, $P^{l+1}(1,1) = P^l(i,i)$, $Q^{l+1}(1,1) = Q^l(i,i)$, $IN^{l+1}(1) = IN^l(i)$, $OUT^{l+1}(1) = OUT^l(i)$ とする．

Step 6 $l = l + 1$, $i = 1$ とする．

Step 7 $k = 1, 2, \ldots, i$ となる k に対して $Q^l(k, i)$ を求める．

Step 8 もし，$r(k)$ （$Q^l(k, i)$ における r の値）が有限の値を持つような $Q^l(k, i)$ が存在しないときは，これまでの記録を求める解の集合としてアルゴリズムを終了．

Step 9 各々の k で $W^l(k) + r(k)$ を求め，この値が最小になる k を i^* と置く．また，この時点での $\{(e, f), r\}$ を $\{(e^*, f^*), r^*\}$ とする．

Step 10 $T^l(i+1) = T^l(i^*) - e^* \cup f^*$, $IN^l(i+1) = IN^l(i^*) \cup e^*$, $OUT^l(i+1) = OUT^l(i^*) \cup e^*$ とする.

Step 11 $i = i+1$ とする.

Step 12 $m^l(i, \mu^l)$ を求める.

Step 13 もし $m^l(i, \mu^l) > 0$ ならば, **Step 6** へ戻る.

Step 14 もし $l = K-1$ ならば, アルゴリズムを終了. そうでなければ, μ^{l+1} を $T^{l+1}(1) = T^l(i)$ に含まれる枝の存在可能性の下限として, **Step 2** へ戻る.

元となったアルゴリズム ([26]) はこのアルゴリズムと違い, ツリーを更新した後に IN および OUT に追加する枝を e でなく f にしている. また, Q を求める際に, 変換可能な枝の組 (e, f) を検索する時に, $\mu(e) = \mu^l$ となるもののみでなく, 全ての枝を対象としている.

本論文のアルゴリズムでは, 上記の制限がなくても更新が終了し, 必要な解が得られることを示す.

本来のアルゴリズムでは, 同じスパニングツリーを複数回作成できないようになっている. これは, $IN^l(i+1) = IN^l(i^*) \cup e^*$, $OUT^l(i+1) = OUT^l(i^*) \cup e^*$ とすることによって, 更新前のツリーから再び同じ枝の組を選択しないように, また, 更新後のツリーから一度取り込んだ枝を外すことのない様に考えられているからである. 拡張される際に削られる更新は,「$T^l(i)$ からはずす枝 e に対して, $\mu(e) > \mu^l$ となるもの」である. このアルゴリズムでは, 一度外された枝はそこから派生する全ての組合せで追加が不可能になるため, T_1 から順に更新を行った場合に後に求めるツリーができるのは, 二段階目で求めるツリー T_3 が表れたとき, すなわち,「$T_1 \to |e_1, f_1| \to T_2 \to |e_2, f_2| \to T_3$ の順で変換し, なおかつ $e_2 = f_1$ である場合」のみである. ただし, $\mu(e_2) = \mu(f_1) > \mu^l$, $\mu(f_2) > \mu^l$, $\mu(e_1) = \mu^l$ であるとする.

一度 T_1 から T_2 を導き出すと, T_1 から T_3 が導き出せなくなる. しかし, 本論文でのアルゴリズムでは, 変更点より T_2 を作成しないため, 集合 $IN^l(i)$ に e_1 が入ることはない. よって, 直接 $T_3 = T_1 - e_1 \cup f_2$ が求まるようになる. これは三段階以上離れているときでも同様である.

加えて, 前に述べた条件付けにより, アルゴリズムはさらに効率的になる. まず, ツリーの更新後, 元のツリー $T^l(i^*)$ は次回からの検索対象から外れる. これは, $IN^l(i^*)$ に存在可能性が μ^l に等しい枝が追加されるため, そのツリーから派生したツリーは, 例外なく存在可能性の下限が μ^l になるからである. 次に, 上記の事を踏まえた上で更新を行った $T^{l+1}(1)$ は, 必ず「枝の存在可能性が μ^{j+1} 以上

であるスパニングツリーの中で総コストが最小のもの」となっている．このことは，次の定理（[15]）によって証明される．

（定理6-1）

T を G における最小スパニングツリー，(e,f) を T からの変換が可能な枝の組のうちコストの差が最小のものとする．このとき，$T-e\cup f$ は，$G-e$ における最小スパニングツリーである．

（証明）

あるスパニングツリー T がグラフ G における最小スパニングツリーであることの必要十分条件は，「G の内部に，$T-e\cup f$ がスパニングツリーとなり，かつ $(f)-w(e)<0$ となる枝の組 (e,f) が存在しない」ことである．これを基に，上の定理の証明を行う．

まず，ツリー $S=T-e\cup f$ が $G-e$ における最小スパニングツリーでないと仮定する．S は最小スパニングツリーではないので，このとき，$S-g\cup h$ が $G-e$ におけるツリーとなり，かつ $w(h)-w(g)<0$ である枝の組 (g,h) が存在する．また，T から e を削除したときにグラフが $U=(V_U,A_U)$ と $W=(V_W,A_W)$ の二つの連結グラフに分割されるとすると，g の連結する点は「U 内と W 内に1点ずつ」か「（ここでは）2点とも U の要素」の二通りとなる．このうち前者は，S 内で U と W を連結する枝は f のみであることから $g=f$ でないと成り立たないが，この場合 h も U と W の両方を連結しなければならない．しかし，$w(f)=w(g)>w(h)$ より，変換 $|e,h|$ の方が $|e,f|$ よりも先に行われるため，題意に反することになる．よって，g の端点は2点とも U に含まれるものとする．ここで，g によって U が X と Y に分割されるものとすると，h は W，X，Y の内のどれか二つを連結する役割を持っている．それは，この3つの部分グラフを連結している枝が，S から g を外した時点では f 一本のみだからである．元の e が W と繋いでいた方のグラフを X とすると，h が連結するグラフは次の3通りのうちの一つとなる．

(X,Y) ：この場合，T の時点で既に変換 $|g,h|$ が可能である．よって T が最小スパニングツリーではなくなるため，題意に反する．

(W,X) ：この場合，f は W と Y を連結する．T の時点で可能な変換は $|e,f|$，$|e,h|$，$|g,f|$．それぞれの枝の関係より，$w(e)<w(f)<w(h)<w(g)$ であるので，$|g,f|$ によって，より総コストの小さいツリーに変換が可能である．よって，T が最小スパニングツリーでなくなるため，題意に反する．

(W,Y) ：この場合，f は W と X を連結する．T の時点で可能な変換は $|e,f|$, $|e,h|$, $|g,h|$. $w(h)-w(g)$ が負の値を持つため，T が最小スパニングツリーでなくなるため，題意に反する．

よって，S を最小スパニングツリーでないと仮定した場合には必ず矛盾が発生する．これにより，$S=T-e\cup f$ は，$G-e$ の最小スパニングツリーである．

元の論文ではこの定理の条件の一つを「(e,f) を T からの変換が可能な枝の組のうちコストの差が最小のもの」としているが，実は「(e,f) を T からの変換が可能な枝の組のうち**ある枝 e に対して**コストの差が最小のもの」でも証明は成り立つ．すなわち，任意の枝 e に対してコストの差が最小になるように f を選択すれば，その値が全ての組合せの中で最小でなくて定理は成り立つのである．

$\mu(e)=\mu^l$ という条件で e を選び，それに対応する適切な f を選択すれば，$T^e\cup f$ は $G-e$ における最小スパニングツリーになっている．この変換を $\mu(e)=\mu^l$ となる全ての枝に対して順に行えば，得られるツリー $T^{l+1}(1)$ は，使用する枝の存在可能性が μ^{l+1} であるスパニングツリーの中で，総コストが最小のものといえる．

これを踏まえた上で，前述したアルゴリズムはさらに効率化が可能である．

《アルゴリズム 6-2》

Step 0 $l=1$, $i=1$, $IN^1(1)=\emptyset$, $OUT^1(i)=\emptyset$ とする．

Step 1 μ^1, $m^1(1,\mu^1)$ を求める．

Step 2 $T^1(1)$ （存在可能性条件を無視した状態での最小スパニングツリー）を，欲張り法を用いて求める．

Step 3 $(T^l(1),\mu^l)$ を非劣解の一つとして記録する．

Step 4 $\mu(e)\le\mu^l$ となる枝のうち，$T^l(i)$ に含まれないものを $OUT^l(i)$ に追加する．

Step 5 $T^{l+1}(1)=T^l(i)$, $P^{l+1}(1,1)=P^l(i,i)$, $Q^{l+1}(1,1)=Q^l(i,i)$, $IN^{l+1}(1)=IN^l(i)$, $OUT^{l+1}(1)=OUT^l(i)$ とする．

Step 6 $l=l+1$, $i=1$ とする．

Step 7 $Q^l(i,i)$ を求める．

Step 8 もし，$r(i)$ が有限の値を持つような $Q^l(i,i)=\{(e,f),r(i)\}$ が存在しないときは，これまでの記録を求める解の集合としてアルゴリズムを終了．

Step 9 $T^l(i+1)=T^l(i)-e\cup f$，$IN^l(i+1)=IN^l(i)\cup e$，$OUT^l(i+1)=OUT^l(i)\cup e$ とする．

Step 10 $i=i+1$ とする．

Step 11 $m^l(i,\mu^l)$ を求める．

Step 12 もし $m^l(i,\mu^l)>0$ ならば，**Step 6** へ戻る．

Step 13 もし $l=K-1$ ならば，アルゴリズムを終了．そうでなければ，μ^{l+1} を $T^{l+1}(1)=T^l(i)$ に含まれる枝の存在可能性の下限として，**Step 2** へ戻る．

＜計算複雑性＞

このアルゴリズムの計算複雑性は，次のように求められる．

まず，初期実行可能解（存在可能性条件を無視した状態での最小スパニングツリー）を求める際に，$O(\min\{n^2, m\log\log n\})$ のオーダーの計算が必要となる．その後解の更新を行うが，各段階の計算量の内，$Q^l(i,i)$ を求めるときのオーダー $O(m)$ が最大となっている．この更新を最大 K （異なる存在可能性の数）解だけ繰り返すため，全体でのオーダーは，$O(Km+\min\{n^2, m\log\log n\})$ となる．

6.2 スパニングツリーの概念の拡張とその応用

スパニングツリーのモデルを扱う問題と，その元となった現実問題とのずれを少しでも補正するための方法の一つとして，スパニングツリーの概念を拡張した「擬ツリー」を定義し，ある実際の問題に対して，その概念を使用したモデルを考える．

スパニングツリーの定義は，(A) 任意の二点間の連結性 と (B) 任意の二点間を繋ぐ経路の唯一性によって成り立っている．この二つの条件のうち，どちらかを緩和することによって，スパニングツリーの概念に近い擬ツリーを定義することにする．どちらを緩和するかによって，擬ツリーの概念は二種類のパターンが考えられるが，どちらの場合に対しても，各々独自の尺度によって，対応するグラフがどれだけツリーの概念に近いかを表すこととする．この尺度により，擬ツリーの概念を従来のモデルに導入することによって拡張されたモデルを，「従来の問題における目標」と「扱う，もしくは解として表れるグラフがツリーに近い度

合い」を目的とした，二目的問題として捕らえることができ，意思決定者は如何にツリーとしての原形を最大限に保ったまま解を目標に近づけるかという考えを具体的に扱うことができる．

また，本論文で定義した擬ツリーの概念を，後述する「二つの根を持つ最短スパニングツリー問題」に対して適応し，より実際の問題に近いモデルの構築を試みる．

[「任意の二点間の連結性」の緩和によるツリー概念の拡張]

グラフ G におけるスパニングツリー T を定義付けている条件のうち，「(A) G 上の全ての点を連結する極大グラフ」を緩和することによる擬ツリーの定義づけを試みる．

第一の条件を緩和することより，この意味での，グラフ $G=(V,E)$ に対する擬ツリー $T'_c=(V,E'_c)\subseteq G$ の定義は次のようになる：

(A') G 上の全ての点を可能な限り連結する極大グラフ
（V に含まれる任意の 2 点 (i,j) 間に経路 $p(i,j)$ が存在する可能性が高い）

(B) 内部に閉路を含まない G の部分グラフ
（任意の 2 点 (i,j) 間にルートの異なる経路が 2 本以上存在しない）

また，擬ツリー T'_c がツリーの概念に近い度合い，すなわち条件 (A') が (A) に近く，任意の二点に経路が存在する度合いをツリー連結度 μ_c で表すとする．ここで条件 (A') より，$\mu_c=$（T'_c の部分グラフの内，枝数が最大のものの枝数）/（G のスパニングツリーの枝数（$|V|-1$））とおく．このパラメータは，T'_c がスパニングツリーのとき 1，全ての点が連結せず，グラフ上に枝がないときに 0 の値をとる．また，擬ツリー T'_c は条件 (B) を常に満たすという前提となっているため，その形態は必ずツリー，もしくはフォレスト（他のツリーと点を共有しない複数のツリーよって成り立つ非連結グラフ）であり，部分グラフとして表れるスパニングでないツリーの規模が大きい方がツリーとしての構造がより近いと考えられることから，この定義は妥当だと思われる．

グラフ G における，あるフォレスト T'_c がどれほどツリーに近いかの度合いは，次のアルゴリズムによって求められる．

《アルゴリズム 6-3》

Step 0 $k=1$，$V_0=\emptyset$ とする．

Step 1 基準点 i_k を，点集合 $V/\bigcup_{l=0}^{k-1} V_l$ の中から任意に選択する．

Step 2 点 i_k を根とする極大木 $T_k = (V_k, E_k)$ を求める．

Step 3 もし $V/\bigcup_{l=0}^{k-1} V_l \neq \emptyset$ ならば，$k = k+1$ として **Step 1** へ戻る．

Step 4 全ての V_k に対して要素数 $|V_k|$ を求め，その最大値 $|V_{\max}| = \max_k |V_k|$ を求める．

Step 5 $\mu_c = (|V_{\max}| - 1)/(|V| - 1)$ とする．

[「二点間の経路の唯一性」の緩和によるツリー概念の拡張]

グラフ G におけるスパニングツリー T を定義付けている条件のうち，「(B) 内部に閉路を含まない G の部分グラフ」を緩和することによる擬ツリーの定義づけを試みる．

第二の条件を緩和することより，この意味での，グラフ $G = (V, E)$ に対する擬ツリー $T'_t = (V, E'_t) \subseteq G$ の定義は次のようになる：

(A) G 上の全ての点を連結する極大グラフ
（ V に含まれる任意の2点 (i, j) 間に経路 $p(i, j)$ が存在する）

(B') 内部に可能な限り閉路を含まない G の部分グラフ
（ 任意の2点 (i, j) 間にルートの異なる経路が2本以上存在しない）

また，擬ツリー T'_t がツリーの概念に近い度合い，すなわち条件 (B') が (B) に近く，任意の二点に経路が一本しか存在しない度合いをツリー連結度 μ_t で表すとする．ここで，条件 (B') より，$\mu_t =$ （ 擬ツリー T'_t の枝のうち，内部のどのサーキットにも属さない枝の数）/（ T'_t の全枝数，$|E'_t|$ ）とおく．このパラメータは，T'_t がスパニングツリーのとき 1，全ての枝がサーキットの一部となっていて次数が1の点が一つも存在しないときに 0 の値をとる．

グラフ G における，ある連結グラフ T'_t がどれほどツリーに近いかの度合いは，次のアルゴリズムによって求められる．

《アルゴリズム 6-4》

Step 0 $k = 0$，$T'' = T'_t$ とする．

Step 1 基準点 i_k を任意に選択する．

Step 2 T'' 上を基準点 i_k から，途中の各点にラベル v_k （ V_k の要素であることを示す）を付与しながら順にたどる．
もし点 i_k からの経路が途中で二手に分かれていた場合，その点を記録し，それぞれのルートに対して同時に経路を形成して行く．

Step 3 もし，たどっていた経路が，分かれたものも含めて，ラベル v_k にぶつからずに全て葉（次数が 1 の点）に到達した場合，言い換えると T'' が内部にサーキットを含まない場合，$\mu_t = |T''|/|T'|$ としてアルゴリズムを終了する．

Step 4 たどった先の点 v'_k に既にラベル v_k が付与されていた場合，そこから両方のルートをさかのぼり，**Step 2** で記録した点のうち最後に共有していた点 v''_k を探す．

Step 5 点 v'_k から 点 v''_k までの二通りのルート $p^1_k(v'_k, v''_k)$ と $p^2_k(v'_k, v''_k)$ から，サーキット C_k を求める．

Step 6 C_k を，一点 c_k に縮合させるように T'' を変形する．

Step 7 一点に縮合することによって T'' 内に自己閉路ができた場合，それを T'' から取り除く．

Step 8 $k = k + 1$ として **Step 2** にもどる．

【二つの根を持つ最短スパニングツリー問題】

擬ツリーの概念を適応する対象として，次のような問題を考える：ネットワーク上の異なる 2 点に，同程度に重要な施設が存在するモデルを考える．この施設から周辺地域に道路を敷設する際，各点においては両施設から近い方が望ましいが，施設を利用するためのルートを敷設する費用を考えると，ルートの組の形状はスパニングツリーである方が好ましい．

よって，移動手段の形状をスパニングツリーにする，という条件のもとで，各点両施設までの距離のうち，どちらか一方でも極端に大きくならないように，すなわち，距離が大きい方の値をできるだけ小さくすることを目的にしたモデルが考えられる．

[定式化]

まず，定義が必要な幾つかの記号について述べる．

$d(i,j)$：二点 i， j 間の距離

N：各枝 (i,j) に距離 $d(i,j)$ が付加されたネットワーク (i,j).

$T=\{V,E'\}\subseteq N$：求める解（スパニングツリー）

s_1,s_2：ネットワークにおける二つの根（2点は N 上の異なる点）

$P(i,j)$：T 上の， i から j への単純経路

$d_1(i)$：T 上の経路 $P(s_1,i)$ の長さ

$d_2(i)$：T 上の経路 $P(s_2,i)$ の長さ

$d(i)=\max\{d_1(i),d_2(i)\}$：点 i での $d_l(i)$ (l=1,2) の最大値

TC' と TT' の定義より， $T(T')$ であるので，次のような値を定義できる：

$\Delta_t(T',(i,j))$：擬ツリー T' に枝 (i,j) を追加することによって新たに TC' に属することになる枝の数
言い換えると， $\Delta_t(T',(i,j))=$ ($T(T')$ 上での二点 i,j 間の経路の長さ) $+1$ である.

これにより，二つの根を持つ最短スパニングツリー問題（TSSTP）を次のように考えることができる.

TSSTP:

Minimize $\sum_{i\in V} d(i)$
s.t. $d(i)=\max\{d_1(i),d_2(i)\}\quad (i\in V)$
$T=\{V,E\}$ はスパニングツリー
$P(i,j)$ は i と j を繋ぐ T 上の経路
$d^1(i)=\sum_{(j,j')\in P(s^1,i)} d(j,j')$
$d^2(i)=\sum_{(j,j')\in P(s^2,i)} d(j,j')$

《アルゴリズム 6-5》

Step 0 $N_0=\emptyset, E_0=\emptyset$ とする.

Step 1 s_1 から s_2 への最短経路 $P_0(s_1,s_2)$ を求める.

Step 2 P_0 の点要素を N_0 に， P_0 の枝要素を E_O に追加する.

Step 3 全ての点 $i\in N_0$ に対して $d(i)$ を求める.

Step 4 $i \in N_0 \mathbf{and} j \notin N_0$ を満たす全ての枝 (i,j) に対して $d_i(j) = d(i) + d(i,j)$ を求める.

Step 5 $d_i(j)$ の値が全ての枝の中で最小となるような枝 (i',j') を一つ選択する.

Step 6 点 j' を N_0 に, (i',j') を E_0 に追加する.

Step 7 もし $N_0 = N$ ならば, $T = (N_0, E_0)$ としてアルゴリズムを終了させる. そうでなければ, $d(j') = d_i(j')$ として **Step 4** に戻る.

＜計算複雑性＞

このアルゴリズムの計算複雑性は, 次のようにして求められる.

まず, 初期実行解として二点間の最短経路を求める際, $O(n^2)$ 回の計算が必要となる. 実際に $d(i)$ を求めるには, 双方向からの各点への最短距離の計算（求められる経路は双方同一のものとなる）と, 各点におけるそれらの値の比較を $|N_0|$ 回繰り返す必要がある. その後, 経路に新たな点を加えることでツリーを構築することになるが, 和 $d_i(j) = d(i) + d(i,j)$ は, アルゴリズム全体の中で, 最大 m (枝の数) 回繰り返すことになる. よって, $d_i(j)$ の比較による (i',j') の導出の際にも, アルゴリズム全体で最大 m 回の比較が必要となる.

よって, 全体では $O(n^2)$ 回の計算が必要となる.

【問題の擬ツリーの概念を用いた拡張】

上記の問題を, 擬ツリーの概念を用いて拡張することを考える. ここで, モデルに組み込む擬ツリーは T_t' （条件 (B) を緩和したもの）とする.

この拡張により, 問題 TSSTP の条件の一つ,「$T = \{V, E\}$ はスパニングツリー」が「$T = \{V, E\}$ は V 上の全ての点を連結する擬ツリー」と緩和される.

加えて, 問題の性格上, 解を表すグラフはスパニングツリーである方が望ましいのであるから, 従来の目的である「ネットワーク全体での, 異なる二点からの距離の min-max」と「擬ツリーがツリーに近い度合い μ_t」の, 二目的問題として考える.

[定式化]

まず, 前述の問題で定義したもの以外で, 定義が必要な幾つかの記号について述べる.

$T' = \{V, E'\} \subseteq N$: 求める解（擬ツリー）

TC' : T' から，内部のどのサーキットにも属していない枝を削除した， T' の部分グラフ

$TT' = T' - TC'$: T' の部分グラフ

C^k : TC' の k 番目のコンポーネント

T^k : TT' の k 番目のコンポーネント

$T(T') = \{V(T'), E(T')\}$: T' 内のコンポーネント C^k を一点 c^k に縮合したグラフ

$V(T') = \{i' \mid i' \in (T' - TC')\} \cup \{c^k, k = 1, 2, \ldots\}$: $T(T')$ の点集合

$E(T') = \{(i', j') \mid (i', j') \in TT', i', j' \notin TC'\} \cup \{(i', c_k) \mid (i', j') \in TT', j' \in C_k\}$: $T(T')$ の枝集合

$\mu_t(T')$: 「擬ツリー T'_t の枝のうち，内部のどのサーキットにも属さない枝の数」を T'_t の全枝数で割った値

$D(T') = \sum_{i \in V} d(i)$: T' 上の全ての点 i における $d(i)$ の総和

PS : 非劣解 $(D(T'), \mu_t(T'))$ の集合

これより，次のような問題が考えられる．

TSSTP:

$$
\begin{aligned}
&\text{Minimize} \quad \textstyle\sum_{i \in V} d(i) \\
&\text{Maximize} \quad \mu_t(T') \\
&\text{s.t.} \qquad d(i) = \max\{d_1(i), d_2(i)\} \quad (i \in V) \\
&\qquad\quad T = \{V, E\} \quad \text{は擬ツリー} \\
&\qquad\quad P(i, j) \text{ は } i \text{ と } j \text{ を繋ぐ } T \text{ 上の経路} \\
&\qquad\quad d^1(i) = \textstyle\sum_{(j,j') \in P(s^1, i)} d(j, j') \\
&\qquad\quad d^2(i) = \textstyle\sum_{(j,j') \in P(s^2, i)} d(j, j')
\end{aligned}
$$

この問題の解を得る為には，まず，$\mu_t = 1$，すなわち使用するグラフを通常のグラフとしたときの解を計算し，次に解のグラフをツリーから徐々に崩してゆきながら，それに対する解を，その直前の解の改良，という形で求めていく．最終的に幾つかの解が求まった後の，その中からの選択は意思決定者に任せることとする．

《アルゴリズム 6-6》

Step 0 初期実行可能解 T_0 を，前述のアルゴリズムを用いて求める．

Step 1 $T(T_0)=t_0$，$l=0$ とする．

Step 2 $(i,j)\in G/T_0$ を満たす枝に対して $\Delta(T_0,(i,j))$ を計算する．

Step 3 $\Delta(T_0,(i,j))$ の値が最小になるような幾つかの枝のうち，$\sum_{i\in V} d(i)$ が最小になるような枝 (i',j') を選択する．

Step 4 もし $D(T_{l+1})<D(T_l)$ ならば，解ベクトル ($D(T_{l+1}),\mu_t(T_{l+1})$) を PS に追加する．

Step 5 $T_{l+1}=T_l\cup(i,j)$ とする．

Step 6 (i,j) を加えることによって発生したサーキットを縮合させ，$T(T_{l+1})$ を求める．

Step 7 $\Delta_t(T_l,(i,j))$ 計算時に考慮に入れた全ての枝 (i,j) に対して $\Delta_t(T_{l+1},(i,j))$ を計算する．

Step 8 $l=l+1$ として **Step 3** へもどる．

＜計算複雑性＞

このアルゴリズムの計算複雑性は次のようにして求められる．

まず，初期実行可能解を求めるのに，$O(n^2)$ だけの計算を行う必要がある．次に，$\Delta_t(T_l,(i,j))$，$\sum_{i\in V_l} d(i)$，およびそれを構成する $T(T_l)$ 上の経路を算出する．ここで，T_0 での算出に $O(n^2m)$ だけの計算がまず必要となる．T_{l-1} での解から T_l の解を導き出す際には，$T(T_{l-1})$ 上の経路を $T(T_l)$ 上のそれに変換するために同様に $O(n^2m)$ だけの計算が必要となる．その後，得られた値のソートに $O(n^2)$，そこから (i',j') を選択する際 $O(1)$ 回の計算を行う．サーキットの融合による $T(T_{l+1})$ の作成には最悪 $O(m)$ 回の計算が必要である．よって，全体では，$O(n^2m^2)$ の計算が必要となっている．

第7章　結言

本論文では，ネットワークモデルを用いた数理計画問題として表すことの可能な意思決定問題に対して，従来の単一目的問題の多くでは切り捨てられてきた，主要判断基準以外の様々な要因を考慮に入れた新たなモデルの構築と，それに対する解法を提案した．

第１章では，ネットワーク問題全般における背景と，その基礎概念について述べた．まず，ネットワーク問題は各々が現実の問題と密接なかかわりを持ち，諸問題の解を得る為の基礎となっていながら，現実との間にできるギャップにより解の適応が困難になる可能性を示唆した．また，それを緩和する手段の一つとして，判断に必要な諸要因の多くを満足度という形で表し，本来複雑な構造と多くの目的をもつ現実問題を，二目的問題として扱うことについて述べた．

第２章では，ネットワーク上の最適化問題と本論文での新たなモデル，およびその解法を説明する際に必要なネットワーク，ファジィ集合そしてそれを組合せたファジィネットワークの概念について述べた．

第３章では，ネットワーク上の最適化問題の一つである従来の最短経路問題を紹介し，そのモデル化では考慮に入れられなかったその他の要因も，意思決定者によって決められたアークごとの満足度の形で判断基準に含む新たなモデルを構築し，その解法の提案を行った．これは満足度を付加したネットワークがファジィネットワークと同様のものとなり，閾値によって複数の通常のグラフに分解することで，各段階での最適解より二目的問題の非劣解を求める方法である．これにより，様々な要因を考慮に入れた，意思決定者を満足させることを目的とした問題を解くことが可能となった．ここでは，予め全ての点の組合せに対して距離を求め，その情報を元に解の更新を行ったが，必要とする解が特定の２点間の距離のみである場合には，より効率的な解法が存在する可能性がある．また，本章では，ファジィネットワークを用いた拡張のほかに，計測されたデータの不確定性の概念より，付加する値を確定値でなくファジィ数で表したモデルも構築した．

第４章では，ネットワーク上の最大フロー問題と，そのバリエーションであるシェアリング問題と最小コストフロー問題を紹介した．また，それぞれの問題に対する主要な判断基準である，物資の輸送量量（および輸送コスト）以外の要因も考慮に入れた新たなモデルの構築とその解法について提案を行った．このモデ

ルを実際の問題に適応することにより，現実の中でのフロー問題の主要な判断基準である輸送量と輸送コスト以外の要因を判断基準の組込み，意思決定者にとっても満足度の高い解を求めることが可能となった．まず，満足度を付加したネットワークに対し，その値のあるアークを何段階かの閾値によって存在・非存在に分け，各閾値毎の部分グラフの作成を行った．次に，閾値が極端に大きい（小さい）部分グラフに対して従来の判断基準のみからなる一目的問題を解き，その解，もしくはそれを加工した解を次に大きい（小さい）閾値のネットワークの初期実行可能解として用い，順に次の段階の解を求めた．最終的には，そのようにして求まった閾値毎の解に対して，意思決定者が独自の判断で最適だと思われる解を選択することとした．今後の課題は，閾値ごとのネットワークとそこに流れるフローのずれに対して，いかに意思決定者が満足できるモデルを構築するかということがある．フローに関する問題の場合，前章の最短経路問題との最大の相違は「使用すると決定したアークに必ず物資が流れるとは限らない」ということである．このため，ファジィグラフを複数の通常のグラフに分割する際，定義の仕方によって，各閾値ごとの解が変わってしまう可能性がある．よって，今後は本論文で紹介したもの以外のフロー問題に対しても，その都度最適と思われる基準を対応させ，モデルを構築する必要があるが，最適な基準を選択するための方法論を確立させる必要がある．

第5章では，上記のネットワーク上のフロー問題のバリエーションの一つであるネットワーク上の輸送問題の紹介と，判断基準として供給点の存在可能性を加えた新たなモデルの構築，およびそのモデルの解法の提案を行った．この章では，満足度をアーク単位でなく各供給点ごとに設定を行ったが，これはアーク，すなわち輸送経路に対する満足度の重要性よりも供給点，すなわち物資の供給元に対する満足度や供給点の存在可能性のそれを重く見る実際の問題を反映したものとなっているからである．また，現実の輸送問題においては，幾つかの供給点からの物資が流れてこない状態でのフローのバランスをとる手段として，供給点の状態に合わせてその都度各供給点からの物資の量を調整する方法と，ある程度流れてこなくなることを見越して予め調整しておく方法があるが，本論文では，そのどちらにも対応できるように，それぞれの場合に対してモデルを構築し，各々異なった形式のアルゴリズムを用いて解く方法を提案した．今後の課題としては，アークの存在可能性についても考慮したモデルの構築と，輸送問題独自の性質を利用した更なる効率化が挙げられる．前者に関しては，ファジィグラフの性質より，一点の存在可能性をそれにつながる全てのアークの存在可能性に転化することで，前章のフロー問題として解くことは可能である．しかし，このモデル独自の構造を利用することによる，より効率的なアルゴリズムが存在する可能性があるので，その解法を構築する研究には意義があると思われる．

第６章では，ネットワーク上の最小スパニングツリー問題の紹介と，アークの設置コスト以外の要因も判断の規準として考えられるように拡張したモデルとその解法の提案，そしてツリーの定義自体の拡張とそれを用いたツリー問題の拡張を行った．前者のモデルを用いることにより，ある地域の拠点全体をカバーするネットワークを構築する際，従来のように各点間を結ぶケーブル等の設置コストのみではなく，実際に設置することに対する危険度や，二点を結ぶことによる利便性の変化等の数値化しにくい要素に対する配慮も容易になった．また，ツリーを拡張したものとして後者の「擬ツリー」を使用することにより，実際の問題検討時の「ツリー条件（解の内部にサークルを作らない，全ての点を完全にカバーする）を問題にならない程度に緩和することで他の条件に見合ったよりよい解を求める」という判断を具体的な形にすることが可能となった．今後の課題としては，前者のモデルに対する，より現実に即したモデルの構築が挙げられる．具体的なものとしては，各枝ごとの設置コストが不確定な値しか得られないことを想定し，その値をファジィ数で表したモデルに，枝の存在可能性を組み合わせた形が実際の問題と照らし合わせた上で妥当であると思われる．また，擬ツリーをモデルの一部として利用しやすくするために，その性質を研究し，全体をはっきりした形で表すことも課題の一つとして考えられる．

従来のモデルから，ファジィネットワークの概念および，概念自体のファジィ化を用いた，本論文のような意思決定者の設定した満足度との二目的問題への拡張に対する研究はまだはじまったばかりである．しかしながら，従来のモデルでは見過ごされてきた，もしくは数値化が難しいために条件として組み込めなかった様々な要因をできるだけ考慮に入れる事が可能で，かつ全体を虱潰しに調べるのではなく効率的に意思決定者の満足する解が得られる本論文のモデルは，ますます情報が複雑化し，膨大となった情報を基に迅速に判断する必要性が大きい現代社会にとっては，計算規模および解の現実とのずれが少ないという点でより合理的である．その意味でも，モデル，およびモデルを構築するための理論の更なる研究がこれからも必要になってくる．実際の問題のモデル構築における意思決定者と作られるモデルの関係，すなわち，最終的には「如何に意思決定者が満足するか」が最終目的となる観点からいえば，本論文のモデルは妥当なものであると言える．今後は，さらにその関係を陽に入れて，発展させていくような研究を行うことが必要となる．

謝辞

本論文は大阪大学大学院工学研究科応用物理学専攻において著者が在学中に行った研究の成果をまとめたものである．

著者がこの分野に興味をもち，研究室を訪ねたとき以来，様々な面で直接，御指導御助言を頂きました本学大学院工学研究科教授 石井博昭先生に厚く御礼申し上げます．

本学大学院工学研究科教授 伊東一良先生， 同助教授 朝日 剛先生，小松 雅治先生，同講師 山本 吉孝先生には本論文作成にあたり細部にわたり御指導頂き，また貴重な御助言を頂きましたことに心より感謝申し上げます．

本学大学院工学研究科講師 齋藤誠慈先生，同助手 都田艶子先生には本研究の遂行，及び本論文の作成において，終始適切な御指導御助言をいただきました． ここに深く感謝申し上げます．

神戸学院大学教授 塩出省吾先生，鹿児島大学助教授 新森修一先生，流通科学大学講師 伊藤健先生，小出武先生，神戸学院大学講師 毛利進太郎先生，広島大学工学部講師 片桐 英樹先生には本学大学院工学研究科在籍時より，本研究の遂行において適切な御指導御助言を頂きました． ここに厚く御礼申し上げます．

学友でもあり良き先輩でもありました石井研究室の皆様には貴重なご意見を頂きました．ここに心より感謝申し上げます．

参考文献

[1] I. Adler & Alan J. Hoffman & Ron Shamir, "Monge and feasibility sequences in general flow problems", *Discrete Applied Mathematics* 44, pp.21-38, 1993

[2] I. Adler & R. Shamir, "Greegily Solvable Transportation Networks and Edg-Guided Vertex Elimination", *Network Optimization Problems*, pp.1-22, 1993

[3] J.R. Brown, "The Sharing Problem", *Operations Research* 27, pp.324-340, 1979

[4] Bernard Carré, *Graphs and Networks*, Oxford Univ. Press, 1979

[5] N. Christfides, *Graph Theory: an algorithm approach*, Academic Press, New York, NY, 1975

[6] R. P. Dirworth "A decomposition theorem for partially ordered sets", *Annals of Mathematics* 51, pp161-166, 1950

[7] Ding-Zhu Du & Pance M. Pardalos, *Network Optimization Problems*, World Scientific, 1992

[8] D. Dubois and H. Prade, *Fuzzy Sets and Systems*, Academic Press, New York, 1980

[9] R. W. Floyd, "Algorithm 97, Shortest path, *Comm. ACM* 5, pp.345, 1962

[10] N. Furukawa, " A Parametric Total Order on Fuzzy Numbers and A fuzzy Shortest Route Problem", *Optimization* 30, pp.367-377, 1994

[11] N. Furukawa, "Parametric Ordering Relations between Fuzzy Numbers and Their Appication to Fuzzy Shortest Path Problem", *Collected Papers from the Thirteenth Fuzzy Systems Symposium*, pp.55-58, 1997

[12] M.Furukawa, "Parametric orders on fuzzy numbers and their roles in fuzzy optimization porlems", *Optimization* 40, pp.171-192, 1997

[13] T. Furuta, "Parametric operator function via Furuta ineqality", *Scienticae Mathematicae*, 1, pp.1-6, 1998

[14] H. N. Gabow, 'A Good Algorithm for Smallest Spanning Trees with a Degree Constraint", *Networks* 8, pp.301-308, 1978

[15] Hoatold N. Gabow, "Two Algorithms for Generating, Weighted Spanning Trees in Order", *SIAM J. COMPUT.* 6, pp.139-150, 1977

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

[17] Horst W. Hamacher, "A note on K best network flows", *Annals of Operations Research* 57, pp.65-72, 1995

[18] M. I. Henig, "The Shortest Path Problem with Two Objective Fancions", *European Journal of Operational Research* 25, pp.281-291, 1985

[19] T. Ichimori, "Optimal Sharing", *Mathematical Programming* 23, pp.341-348, 1982

[20] 石井博昭, "応用代数", 京都コンピュータ学院教科書出版会, 1984

[21] Hiroaki Ishii and Toshio Nishida, "Stochastic Bottleneck Spanning Tree Problem", *Networks* 13, pp.443-449, 1983

[22] 伊藤健, 石井博昭, "可能性測度における組み合わせ最適化問題", 京都大学数理解析研究所講究録 945 「最適化の数理における離散と連続構造」, pp.115-120, 1996

[23] 伊理正夫, 藤重悟, 大山達雄, グラフ・ネットワーク・マトロイド, 産業図書, 1986

[24] Ahuja, R. K., and J. B. Orlin, " A fast and simple algorithm for the maximum flow problem", Working paper 1905-87, Sloan School of Management, M.I.T., Cambridge, MA, to appear in *Operations Research*, 1987

[25] Richard M. Karp, "A Characterization of the Minimum Cycle Means in a Digraph", *Discret Mathematics* 23, pp.309-311, 1978

[26] N. Katoh, T. Ibaraki, and H. Mine, "An Algolrithms for Finding K inimum Spanning Trees", *SIAM J. COMPUT.* 10, pp.247-255, 1981

[27] C. M. Klein, " Fuzzy Shortest Paths", *Fuzzy Sets and Systems* 39, pp.27-41, 1991

[28] E. Lawlwe, *Combinatorial Optimization Networks & Matroids*, Holt, reinehart and Winster, New York, 1976

[29] Lushu. Li, K. K. Lai, "A fuzzy approach to the multiobjective transportation Problem", *Computers & Operations Research* 27, pp.43-57, 2000

[30] 水木雅晴 編, ファジィ集合, 日刊工業新聞社, 1992

[31] Jiři Močkoř, "General Fuzzy Decision Systems", *Acta Mathematica et Informatica Universitatis Ostraviensis Ostraviensis* 5, pp.81-95, 1997

[32] James Munkres, "Algorithm for the assignment and transportation problems", *J. Soc. Indust. Appl. Math.* 5, pp.33-38, 1957

[33] C. V. Negoiță and D. A. Rolescu, *Applications of Fuzzy Sets to Systems Analysis*, Birkhäuser Verlag, Bosel and Strttgart, 1975

[34] 西田俊夫 & 田端吉雄 編, 現代 OR セミナー, 現代数学社, 1995

[35] 西田俊夫 & 竹田英二, ファジィ集合とその応用, 森北出版, 1978

[36] T. Oyama, "Weight of Shortest Path Analyses for the Optimal Location Problem", *Journal of the Operations Research Society of Japan* 43, pp.176-196, 2000

[37] Ford L. R. and Fukerson D. R., *Flows in Networks*, Princeton Univ. Press, 1962

[38] 坂和正敏、石井博昭、西崎一郎, ソフト最適化, 朝倉書店, 1995

[39] 島田文彦, "ファジィネットワーク上のフロー問題", 数理解析研究所講究録 1079 「数理モデルにおける決定理論」, pp.205-214, 1999

[40] F.Shimada and H. Ishii, "Extension of spanning tree and applications", *Scientae Mathematicae Japonicae* (to appear)

[41] F. Shimada and H. Ishii, "Fuzzy Transportation Problem with Existence Possibility of Suppiers", *Proc. of IFORS '99*, IFORS, pp.81, 1999

[42] F. Shimada and H. Ishii, "Transportation Problem with Block Constrains on Fuzzy Network", *Proc. of EFDAN '99 - Fuzzy Decision Analysis*, Fuzzy Logic System GmbH, pp.143-151, 1999

[43] 島田文彦, 石井博昭, "供給点に存在可能性を付加した整数ファジィ輸送問題", 第15回ファジィシステムシンポジウム講演論文集, 日本ファジィ学会, pp.563-564, 1999

[44] 島田文彦, 石井博昭, 伊藤健, "ファジィルートを持つ整数輸送問題", 第14回ファジィシステムシンポジウム講演論文集, 日本ファジィ学会, pp.425-426, 1998

[45] 島田文彦, 石井博昭, 伊藤健, "ファジィネットワーク上での最適化問題", 数理解析研究所講究録 1043 「決定理論とその関連分野」, pp.25-31, 1998

[46] 島田文彦, 石井博昭, 伊藤健, "ファジィネットワーク上の最短経路問題", 日本ファジィ学会誌 10, pp.348-355, 1998

[47] F. Shimada, H. Ishii and T. Itoh, "Spanning Tree Problems on a Fuzzy Network", 1998 Second International Conference on KES'98 proceedings 1, 1998

[48] 島田文彦, 石井博昭, 伊藤健, "ファジィネットワーク上のスパニングツリー問題", 第2回ファジィORミニシンポジウム講演論文集, 日本OR学会関西支部・ファジィOR研究会, pp.21-25, 1997

[49] M. Tada, H. Ishii, T. Nishida and T. Masuda, "Fuzzy sharing problem", *Fuzzy Sets and Systems* 33,pp.303-313, 1989

[50] M. Tada, H. Ishii, "An Integer Fuzzy Transportation Problem" *Computers Math. Applic.* 31, pp.71-87, 1996

[51] 多田実, 西田俊夫, 石井博昭, 増田照雄, "一般化整数シェアリング問題", 電子情報通信学会論文誌 J70-A, pp.239-244, 1987

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

[53] W. T. Tutte, *Connectivity in Graphs*, University of Toronto Press, 1966

[54] Gwo-Hshiung Tzeng, Dušan Teodorović, Ming-Jiu Hwang, "Fuzzy bicriteria multi-index trabsportation problems for coal allocation planning of Taipower", *european Journal of Operational Research* 95,pp.62-72, 1996

[55] S. Warshall, "A Theorem on Boolean Matrics", *J. ACM* 9,pp.11-12, 1962

[56] Uwe Zimmermann, "Negative circuits for flows and submodular flows", *Discrite Applied Mathematics* 36, pp.179-189, 1992

著者発表論文

1. 島田文彦、石井博昭、伊藤健, ファジィネットワーク上の最短経路問題, 日本ファジィ学会誌, Vol.10, No.2, pp.348-355, 1998/4
2. 島田文彦、石井博昭, Extension of spanning tree and applications, Scientae Mathematicae Japonicae, (掲載決定)
3. Fumihiko Shimada, Hiroaki Ishii, Transportation Problem with Existence Possibility of Suppliers, special issue on OPERATIONS RESEARCH(OR) / MANAGEMENT SCIENCE(ME) FROM THEORY TO REAL LIFE IN THE ASIA-PACIFIC REGION within the framework of THE INTERNATIONAL SERIES IN OPERAIONS RESEARCH AND MANAGEMENT SCIENCE Kluwer Academic Publishers (投稿中)
4. Fumihiko Shimada, Hiroaki Ishii, Sharing Problem on a Fuzzy Network, Fuzzy Sets and Systems (投稿中)
5. Fumihiko Shimada, Hiroaki Ishii, Akira Morita, Fuzzy Transportation Problems, special issue of Central European Journal of Operation Research (投稿中)