ネットワークデザイン問題の近似解法

片山　直登　流通経済大学　流通情報学部

1　はじめに

ネットワークデザイン問題は，通信網設計，輸送・配送計画，交通ネットワーク計画や下水道計画問題などに現れるネットワーク問題の基本的かつ重要な問題の一つである．この問題は，ノードとアークの集合，アークのための費用，アーク上のフロー費用，およびネットワーク上を移動するものの量が与えられたときに，ネットワークを構成するアーク費用とものを流すためのフロー費用を考慮して，適切なネットワークとフローの形態を選定する問題である．ネットワークの形態を決める問題には例えば最小木問題やシュタイナー木問題などがあり，フローや経路を決める問題には最短路問題や最小費用流問題などがあるが，ネットワークデザイン問題はこれらの問題を同時に考慮する問題と考えることができる．

アークは通信回線，車両，道路区間，幹線管渠などの地点間に固定的な費用が必要なものに対応する．フローはデータ，荷物，交通，下水など地点間を移動するものに対応する．また，ノードは交換機，端末，ターミナル，顧客，交差点，セントロイド，処理場などフローが発生・集中する地点や中継点に対応する．

問題の対象や形態によって，フロー費用はフロー量の線形，凸，凹関数などに分類される．凸関数の場合は混雑による遅れ，凹関数は規模の経済性による単位費用の減少に対応している．線形関数は，フローに関する費用がフロー量に比例することを意味しているが，アーク費用とフロー費用を併せると固定費付きの関数となり，凹関数の近似と捕らえることもできる．

フロー費用が線形であり，アーク費用を目的関数に含み，容量制約のない最も基本的な問題は，容量制約のないネットワークデザイン問題 (Uncapacitated Network Design Problem:問題 UND) とよばれている．本研究では，容量制約のないネットワークデザイン問題に対する貪欲解法とタブーサーチ法を用いたメタヒューリスティック解法の 2 種類の方法を提案し，数値実験による従来の解法との比較，およびパラメータの検討を行うことを目的とする．

2　従来の研究

容量制約のないネットワークデザイン問題は，数多くの最適解法，近似解法および解析などの研究がなされている．最適解法には分枝限定法 [9] やベンダーズ分解原理 [10] があり，下界値の解法には双対上昇法 [2] やラグランジュ緩和法 [8] があり，NP 完全性の証明 [7] や切除平面の解析 [1] がなされている．また，サーベイ [11][13][5] や文献目録集 [**?**] がある．

近似解法では，数多くの貪欲解法が提案されている．Scott[14] および Dionne and Florian[4] は予算制約をもつネットワークデザイン問題に対して，アークを削除・付加したときの目的関数値の変化量を指標としたアド・デリート型の近似解法を提案した．これらの解法は，そのまま問題 UND にも適用が可能である．Billheimer and Gray のデリート法 [3] は，各アークの両端間について，予め第二最小費用路からアークを除去した場合の目的関数値の変化量を近似的に計算し，この値が最大値のアークから順次削除していく方法である．Minoux のデリート法 [12] は，初期のアーク削除時の目的関数値の変化量のデータをリストに保持し，削除対象時に再計算した変化量がリスト中で最大の場合のみ削除する方法である．Los and Lardinois のデリート法 [9] は，最初にアークを除去した場合の目的関数値の変化量を厳密に求め，単位アーク費用当たりの変化量を基準にアークを除去していく方法である．

一方，近年，得られた近似解をさらに改善するメタヒューリスティック解法であるタブーサーチ法，シミュレーテッドアニーリング法や遺伝的アルゴリズムなどの研究が行われている．タブーサーチ法は Glover[6] が提案した手法であり，多くの問題に適用されており，久保が詳しい解説と種々の問題への適用を行っている．

3 問題 UND の定義

ノード集合 N，無向アーク集合 A，品種集合 K が与えられているものとする．品種 k の始点を O_k，終点を D_k とする．ここでは，同じ始点・終点をもつ場合を 1 品種とし，異なる始点・終点をもつ場合は別の品種と区別する．アーク (i,j) を使用する場合に 1，そうでない場合 0 であるアーク変数を x_{ij} とし，アーク (i,j) を使用するときに発生するアーク費用を a_{ij} とする．アーク (i,j) 上をノード i から j に品種 k が流れる量をあらわすフロー変数を y_{ij}^k とし，アーク (i,j) 上を品種 k が 1 単位流れるとき発生する費用であるフロー費用を c_{ij}^k とする．また，各品種の輸送量は 1 単位に変換されているものとする．このとき，問題 UND は次のように定式化される．

UND :

$$minimize \qquad \sum_{(i,j)\in A}\sum_{k\in K}(c_{ij}^k \cdot y_{ij}^k + c_{ji}^k \cdot y_{ji}^k) + \sum_{(i,j)\in A} a_{ij} \cdot x_{ij} \tag{1}$$

subject to

$$\sum_{i\in N} y_{in}^k - \sum_{j\in N} y_{nj}^k = \begin{cases} 1 & n = D_k \\ -1 & n = O_k, \quad n \in N, k \in K \\ 0 & \text{その他} \end{cases} \tag{2}$$

$$y_{ij}^k \le x_{ij},\ y_{ji}^k \le x_{ij}, \quad k \in K, (i,j) \in A \tag{3}$$

$$x_{ij} \in \{0,1\}, \quad (i,j) \in A \tag{4}$$

$$y_{ij}^k \ge 0,\ y_{ji}^k \ge 0, \quad k \in K, (i,j) \in A \tag{5}$$

(1) 式は，アーク上を流れるフロー費用の合計とアーク費用の合計の和である総費用を最小化することを表している．(2) 式は各品種のフローが必ず始点から終点へ流れるこ

とを表すフロー保存制約である．この制約の左辺の第一項はノードに入ってくるフローの合計，第二項はノードから出るフローの合計を表しており，始点であれば第一項と第二項の差が -1，終点であれば 1，その他の中継ノードであれば 0 となることを表している．(3) 式は，アークが存在するときにのみアーク上を容量に関係なくフローが流れることができることを表している．(4) 式はアーク変数が 0–1 変数であることを表し，(5) 式はフロー変数が非負であることを表している．

問題 UND の解の実行可能性を判定することは，比較的容易である．アーク変数が実行可能であるためには，すべての品種の起点・終点間が連結していれば良い．特に，すべてのノードが起点または終点を兼ねていれば，ネットワークが連結していれば良いことになる．フロー変数が実行可能であることは，このネットワーク上の起点・終点間でフローが連結していることである．ある実行可能なアーク変数が与えられたときのフロー変数の最適解は，フロー費用をアークの長さとしてすべての起点・終点間の最短路問題を解くことによって決定することができる．

4 貪欲解法

4.1 貪欲解法の性質と概要

問題 UND は，0–1 変数を含む混合整数計画問題である．このため，ノード数が 10 程度の小規模な問題であれば，汎用パッケージで最適解が得られる可能性がある．しかし，ネットワークが完全グラフの場合，ノード数が 30 程度であっても，0–1 変数であるアーク変数が 400 を超え，フロー変数は 30 万を超えてしまう．このため，一般的な汎用パッケージでは解くことが困難となる．また，最適解法であるベンダース分解原理を利用した場合でも，ノード数が 30，アーク数が 90 程度までは最適解を求めることができるが，アーク数が増加した場合，最適解を求めることは困難となる．したがって，最適解を求める代わりに，良い実行可能解である近似解を求める解法が重要となってくる．

問題 UND に対する近似解法の設計の難しさの一つは，アーク変数が確定したとしても，最適なフロー変数を決定し，近似解の目的関数値を評価するためには，すべての品種の起点・終点間の最短路を解く必要があり，$O(|N|^3)$ の計算量を必要とすることである．一般に，近似解法では，アークの削除や付加を数多く繰り返し，そのたびにフロー変数を決定し，目的関数値を計算する必要がある．このため，全体として膨大な計算量となってしまう．アークを削除や付加したネットワーク上の最短路問題は，Murchland 法 [15] を用いることによって実際の計算時間は大幅に短縮できるが，それでもアーク付加では $O(|N|^2)$，アーク削除では $O(|N|^3)$ の計算量を必要とする．このため，従来の研究では，フロー変数をヒューリスティックに決定するなど，計算量を軽減するための工夫を行っている．

基本的な近似解法として貪欲解法 (greedy method) がある．貪欲解法は，何らかの基準によって変数を確定していき，試行錯誤を繰返さない方法であり，一般に計算量を抑えることができる．問題 UND では，アークを削除していくデリート法とアークを付加していくアド法が知られている．一方，近傍探索を行い，最も良い局所解を探索していく局所探索法 (local search method) がある．貪欲解法も局所探索法の一つであるが，局所探索法はたとえばアークの入替えなどを行う方法である．しかし，前述のようにフロー変数の決定に時間がかかることから，問題 UND に適用した例はほとんど見られない．

4.2 Minoux のデリート法

Minoux のデリート法は，アークを削除したときに，アークの両端点間の最短路上にフローを流し代えることによってフロー変数と目的関数値の減少量を算出し，この減少量の大きい順にアークの削除を検討する．さらに，アークが削除の対象となるときに目的関数値の減少量を再評価し，この値が未削除のアークの中で最大であれば削除し，そうでなければリストの適切な位置にアークを挿入する方法である．

・Minoux のデリート法

[1] すべてのアーク変数を 1 としたネットワークを初期解とし，フロー費用をアークの長さとした各品種の起点と終点間の最短路を求め，この路上に流したフローをフロー変数の解とし，目的関数値を計算する．

[2] すべてのアークについて，このアーク変数を 0(削除) とした場合に，フロー費用をアークの長さとしたアークの両端点間の最短路上に，このアーク上に現在流れているフローを流し代えたものをフロー変数の解とし，目的関数値の減少量を計算する．

[3] 目的関数値の減少量の大きい順に，アークと目的関数値の減少量をリストに格納する．

[4] リスト内の減少量が最大のアークを対象アークとする．減少量が正値であるアークがなければ，終了．

[5] 対象アーク変数を 0 とした場合のアークの両端点間の最短路上に，対象アーク上に流れているフローを流し代えたものをフロー変数の解とし，目的関数値の減少量を再計算する．減少量がリスト中で最大でなければ，対象アークをリストの適切な位置に再格納し，[4] へ．

[6] 対象アークをネットワークとリストから削除し，[4] へ．

Minoux のデリート法の計算量は $O(|N|^6)$ であるが，平均的には $O(|N|^4)$ であり，計算量も少なく，得られる近似解も比較的優れたものである．

4.3 Minoux のデリート法の修正

Minoux のデリート法のステップ [2][5] では，削除したアークの両端点間の最短路上に，このアーク上のフローを流し代えたものをフロー変数の解としている．しかし実際には，削除したアーク上のフローのすべてがアークの両端点間を経由して，両端点間の最短路上を流れる場合よりも，このアークの両端点を経由しない経路に流れるフロー解の方が目的関数値が小さくなる場合が発生する．したがって，ステップ [2][5] で求められた目的関数値の減少量は，この時点でのアーク変数に対する最適値ではなく，上界値となる．

このように，目的関数値の減少量を上界値として評価しているのは，計算量の増加を防ぐためである．しかし，近年ではコンピュータは高性能化し，この評価を厳密に行って

も，十分大規模な問題を解くことが可能であると考えられる．このため，2つ厳密性を加えたアルゴリズムを提案する．多くの部分が共通であるので，Minoux アルゴリズムとの相違個所のみを記述する．修正アルゴリズム 1 では Minoux アルゴリズムのステップ [5] を [5'] に，修正アルゴリズム 2 ではステップ [2] を [2']，ステップ [5] を [5'] に変更している．

・修正アルゴリズム 1

[5'] 対象アーク変数を 0 とした場合に，各品種の起点と終点間の最短路を求め，最短路上の各品種のフローをフロー変数の解とし，目的関数値の減少量を厳密に計算する．減少量がリスト中で最大でなければ，対象アークをリストの適切な位置に再格納し，[4] へ．

・修正アルゴリズム 2

[2'] すべてのアークについて，このアーク変数を 0 とした場合，フロー費用をアークの長さとした各品種の起点と終点間の最短路を求め，最短路上の各品種のフローをフロー変数の解とし，目的関数値の減少量を厳密に計算する．

[5'] 対象アーク変数を 0 とした場合に，各品種の起点と終点間の最短路を求め，最短路上の各品種のフローをフロー変数の解とし，目的関数値の減少量を厳密に計算する．減少量がリスト中で最大でなければ，対象アークをリストの適切な位置に再格納し，[4] へ．

修正アルゴリズム 1 は，ステップ [5] においてフロー変数の最適解を厳密に計算する修正を行ったものである．また，修正アルゴリズム 2 は，さらにステップ [2] において同様な修正を行ったものである．修正アルゴリズム 1 の計算量は，$O(|N|^3 + |N|^2 \times |N|^2 + |N|^2 \log |N|^2 + |N|^2 \times |N|^3 \times |N|^2) = O(|N|^7)$ であるが，平均的には $O(|N|^3 + |N|^2 \times |N|^2 + |N|^2 \log |N|^2 + |N|^2 \times |N|^3) = O(|N|^5)$ である．また，修正アルゴリズム 2 の計算量も $O(|N|^7)$，平均的には $O(|N|^5)$ である．

デリート法の中では，Scott・Dionne のデリート法が厳密な意味での first 改善 (または best 改善) 法である．このため，良い近似解を得ることが出来るが，計算量は平均的にも $O(|N|^7)$ である．一方，修正アルゴリズム 1 および 2 も厳密な意味での first 改善法であるため，Scott・Dionne のデリート法と同程度の解が得られ，計算時間は大幅に短縮できることが期待される．

5 貪欲解法の数値実験

5.1 数値実験の条件

Scott・Dionne のデリート法 (Scott 法)，Minoux のデリート法 (Minoux 法) と提案した修正アルゴリズム 1(修正法) を数値実験によって比較する．100×100 の平面上にランダムにノードを発生し，ノード間のユークリッド距離を整数化した値をアーク費用する．フロー費用はすべての品種で同一とし，アーク費用に比例するものとする．また，すべての 2 ノード対間にアークが使用可能とし，すべての 2 ノード対の間を品種が流れるものとする．ノード数は 10 から最大 100 までとし，同一ノード数では 10 組のデータを使用

表 1: 問題の規模

ノード数	アーク変数	品種数	フロー変数
10	45	45	4050
20	190	190	72200
30	435	435	378450
40	780	780	1216800
50	1225	1225	3001250
60	1770	1770	6265800
70	2415	2415	11664450
80	3160	3160	19971200
90	4005	4005	32080050
100	4950	4950	49005000

表 2: 目的関数値と計算時間の比較：費用比 5

ノード数	Scott 法		Minoux 法		修正法	
	目的関数値	計算時間 (s)	目的関数値	計算時間 (s)	目的関数値	計算時間 (s)
10	1.000	0.1	1.009	0.0	1.000	0.0
20	1.000	1.4	1.014	0.1	1.002	0.1
30	1.000	18.9	1.014	0.4	1.001	0.4
40	1.000	113.0	1.016	1.1	1.002	1.3
50	1.000	449.5	1.016	2.7	1.001	2.9
60	1.000	1390.9	1.018	5.7	1.001	6.1
70	1.000	3732.4	1.020	11.1	1.001	12.0
80	1.000	8731.4	1.020	19.5	1.001	21.1
90	1.000	18076.6	1.022	31.4	1.001	34.4
100	1.000	33045.7	1.022	49.8	1.001	52.0

する．使用計算機は IBM PC 互換機，使用言語は Microsoft Fortan Power Station である．また，複数の計算機を使用しているため，Pentium100MHz 機相当の計算時間に換算している．

表 1 にノード 10 から 100 までの問題の変数規模を示す．ノード数自体は少数でも，決定すべき変数は大規模であることがわかる．また，設定した実験条件では，修正アルゴリズム 1 と修正アルゴリズム 2 は一致する．

5.2 計算結果

アーク費用／フロー費用=5 の場合，各解法によって得られた最良の近似解の平均目的関数値と 1 問当りの平均計算時間を表 2 に示す．ただし，目的関数値は Scott 法によって得られた値を 1 とし，Scott 法との比率で表している．同様に，アーク費用／フロー費用=10 の結果を表 3 に，アーク費用／フロー費用=15 の結果を表 4 に示す．

Scott 法で求められた目的関数値が最も優れているが，いずれの費用比の場合でも計算時間がかかり，ノード数が 100 では 1 問当たり約 10 時間程度となり，実用的ではないことがわかった．Minoux 法はノード数が 100 でも 1 問当たり平均 50 秒以内で求めることができるが，費用比が増加するに従い，Scoott 法との目的関数値の差が増大し，最大で 5 %となった．一方，本研究で提案した修正法では，計算時間はいずれの費用比でも Minoux 法よりも 10 %程度長いだけであり，目的関数値は Scoott 法に比べ最大でもわずか 0.3 %程度大きいだけであった．したがって，提案した修正法によって，短い計算時間で，良い近

表 3: 目的関数値と計算時間の比較：費用比 10

ノード数	Scott 法		Minoux 法		修正法	
	目的関数値	計算時間 (s)	目的関数値	計算時間 (s)	目的関数値	計算時間 (s)
10	1.000	0.1	1.012	0.0	0.999	0.0
20	1.000	1.6	1.024	0.1	1.000	0.1
30	1.000	20.0	1.023	0.4	1.002	0.5
40	1.000	119.9	1.030	1.1	1.002	1.2
50	1.000	489.2	1.036	2.6	1.002	2.9
60	1.000	1525.5	1.033	5.6	1.001	6.1
70	1.000	4148.4	1.036	10.7	1.001	12.0
80	1.000	9196.0	1.037	18.9	1.002	20.6
90	1.000	20081.6	1.039	30.3	1.001	34.8
100	1.000	37459.9	1.039	48.5	1.002	54.7

表 4: 目的関数値と計算時間の比較：費用比 15

ノード数	Scott 法		Minoux 法		修正法	
	目的関数値	計算時間 (s)	目的関数値	計算時間 (s)	目的関数値	計算時間 (s)
10	1.000	0.1	1.004	0.0	1.000	0.0
20	1.000	1.8	1.029	0.1	1.002	0.1
30	1.000	21.2	1.039	0.4	1.003	0.4
40	1.000	124.1	1.048	1.1	1.003	1.3
50	1.000	482.5	1.048	2.5	1.000	3.0
60	1.000	1540.2	1.049	5.4	1.003	6.4
70	1.000	4181.9	1.050	10.2	1.003	11.6
80	1.000	9366.0	1.045	18.2	1.003	20.7
90	1.000	20380.7	1.050	30.1	1.002	34.1
100	1.000	38376.0	1.050	46.8	1.002	52.8

似解が得られることが明らかになった．

6 タブーサーチ法を用いた解法

6.1 タブーサーチ法

問題 UND のタブーサーチ法を用いた解法を提案する．タブーサーチ法はタブーリストを用いて，探索済み以外の近傍解を探索していく方法である．タブーリストには 2 種類が知られており，探索済解のキューリストを保持しリスト内の解の探索を禁止するタブーリストタイプと，変数の変更時点を保持し一定回数この変数の変更を禁止する短期メモリタイプがある．ここで，タブーサーチ法の一般的なアルゴリズムを示す．

・タブーサーチ法のアルゴリズム

[1] 初期解を求め，試行解とする．長期メモリと短期メモリをクリアする．

[2] 短期メモリ (またはタブーリスト) によって限定された近傍を探索し，目的関数値と長期メモリから，次の試行解を求める．

[3] 長期メモリと短期メモリを更新する．

[4] 一定回数繰返せば終了，そうでなければ [2] へ．

タブーサーチ法では，初期解，タブーリスト，短・長期メモリ．近傍の定義，および繰返し回数など種々の設計やパラメータの設定が重要である．

6.2 近傍の定義

問題 UND では，フロー変数の決定に計算量を必要とする．タブーサーチ法では，限定された近傍の中で最良と思われる試行解を繰返し探索する必要があるため，非常に多くのフロー変数を繰返し決定する必要がある．したがって，近傍探索をおいて如何にフロー変数を決定する回数を少なくするかが課題となる．

問題 UND の近傍として，デリート近傍，アド近傍，アーク移動近傍の 3 つを挙げることができる．デリート近傍は 1 アーク変数を 1 から 0(1 本削除) にする近傍，アド近傍は 1 アーク変数を 0 から 1(1 本付加) にする近傍である．アーク移動近傍は 1 アーク変数を 0 とし，他のアーク変数を 0 とするアークを移動する近傍である．アーク移動近傍はこれらの中で最も広範囲の近傍領域を持つため，近傍探索には膨大な計算時間を必要とする．

本研究では，これら 3 つの近傍を併用するが，アーク移動近傍に対しては，次の 2 つの工夫を行う．一般に，特定のアークを移動する場合，目的関数値が減少する移動先は移動元のアークの近傍にあると考えられる．これは，移動先が近傍でないと，アーク移動という要因が及ぼす影響が少なく, たとえ目的関数値が減少するとしてもアーク移動近傍だけはなく，デリート近傍かアド近傍でも目的関数値が減少すると考えられる．したがって，デリート近傍とアド近傍を併用すれば，アーク移動近傍において移動する範囲を近傍に制限しても，それほど解が悪化しないものと考えられる．

アーク移動後の目的関数を評価する場合には，効率的な Murchland 法を適用できない．そこで，アーク移動は，アークの移動ではなく，アークを削除した後に別のアークを付加すると考える．前者の場合，近傍探索の計算量は，移動元のアークの本数 × 移動先のアークの本数 × 目的関数の評価 $= O(|N|^2 \times |N|^2 \times |N|^3) = O(|N|^7)$ となり，平均的には $O(|N| \times |N|^2 \times |N|^3) = O(|N|^6)$ となる．一方，後者では，移動元のアークの本数 ×(削除時の計算量+移動先のアークの本数 × 付加時の計算量) $= O(|N|^2 \times \{|N|^3 + |N|^2 \times |N|^2\}) = O(|N|^6)$ ，平均的には $O(|N|^5)$ となる．さらに，移動先のアーク数を定数個に限定すれば $O(|N|^4)$，平均的には $O(|N|^3)$ となる．

6.3 短期メモリと長期メモリ

短期メモリは局所解の近傍における集中的探索を可能にし，長期メモリは大域的な多様化を可能にしている．本研究では，短期メモリに久保が提案している LSM タイプメモリを使用する．短期メモリに保管する値としては，移動元アーク変数の変更回数，移動先アークの変更回数やその両方などが考えられるが，移動元アーク変数の変更回数とし，移動したアークは一定回数だけ移動先の対象とならないものとする．これは，ネットワークを構成するアークはネットワークを構成しないアークよりも少数と考えられるため，移動先アークを一定回数だけ移動元の対象とならないとした場合，移動元の対象のアーク数が減少し，近傍領域が大幅に減少してしまうと考えるためである．

長期メモリ値には，移動元のアーク変数の変更回数を保持するものとする．近傍探索解の目的関数値は，”本来の目的関数値+移動元のアーク長期メモリ値”として評価し，近傍解の中でこの値の最小値を次の試行解とする．

6.4 初期解とランダムスタート

初期解には，短時間で求められ，かつ良い解が必要である．このため，初期解には貪欲解法で提案した修正法1を用いる．長期メモリは大局的な多様化を可能にしているが，問題UNDで予備数値実験を行ったところ，長期メモリだけでは大局的な探索が上手く出来ていないことが分かった．このため，初期解をランダムに設定してタブサーチ法を繰返すランダムスタートを行う．

しかし，単純に初期解をランダムに設定したのでは，局所最適解を得られるまで時間がかかる．このため，1回目に全アークを対象に修正法1を用いて貪欲解を求め，2回目以降は決められた選択率にしたがってアークをランダムに選択し，この全アークを対象に修正法1を用いて貪欲解を求める．このことによって，短時間に比較的良い多様な初期解を求めることができると考えられる．

7 タブーサーチ法の数値実験

7.1 数値実験の条件

使用データと計算機の実験条件は，貪欲解法と同一とする．ただし，アーク費用／フロー費用=5，アーク移動範囲の制限基準として移動元のアークの端点に隣接するノード間に存在するアークとする．

タブーサーチ法では，多くのパラメータを設定する必要がある．そこで，次のような実験を行った．

(1) 短期メモリの影響(長期メモリなしの場合)

(2) 短期メモリの影響(長期メモリありの場合)

(3) アーク移動範囲を制限する隣接ノード数の影響

(4) 初期解におけるアークの選択率の影響

(5) 貪欲解法との比較

ノード数を30，隣接ノード数を6，短期メモリと長期メモリを200回毎にクリアし，短期メモリの禁止回数をノード数，繰返し回数は全体で2000回を基準とする．(1)(2)では短期メモリの記憶回数を3から60回とし，(3)では隣接ノード数を3から30とする，(4)では100回毎にランダムスタートし，アークの選択率を0.5から1.0とする．

(1)から(4)の結果より，(5)では100回毎にランダムスタートし，アークの選択率を0.9，隣接ノード数を6，短期メモリの禁止回数をノード数とし，ノード数を10から80のネットワークについて計算を行い，Scott法と比較する．

7.2 計算結果

(1)の結果を図1に，(2)の結果を図2に示す．長期メモリを使用しない場合，短期メモリの禁止回数が20回以下では禁止回数の増加とともに目的関数値が減少し，20回以上では安定した解が得られた．一方，長期メモリを使用した場合には，短期メモリの禁止

回数に関係なく安定した解が得られた．このため，長期メモリを使用すれば，禁止回数の設定にそれほど厳密性が必要ないことがわかった．

(3) の目的関数値の結果を図 3 に，平均計算時間を図 4 に示す．隣接ノード数を増加させると近傍探索の範囲が増大するため目的関数値が減少すると考えられたが，隣接ノード数が 5 以上で目的関数値は安定し，隣接ノード数をそれ以上に増加しても余り解は改善されなかった．これは，移動近傍とともに，デリート近傍とアド近傍を併用したことの効果と考えることができる．一方，計算時間は隣接ノード数とともに増加するため，隣接ノード数は 5 から 7 程度が適正値と思われる．

(4) の結果を図 5 に示す．選択率を下げると多様ではあるが悪い初期解を，選択率を上げると類似解ではあるが良い初期解を，選択率 1 では同じ初期解を生成することになる．しかし，結果からは選択率 1 以外では目的関数値に大差は見られず，選択率の影響が余

表 5: タブーサーチ法の計算結果:目的関数値の比較

ノード数	下界値	Scott 法	タブーサーチ法
10	1.000	1.008	1.006
20	1.000	1.011	1.007
30	1.000	1.012	1.009
40	1.000	1.014	1.011
50	1.000	1.015	1.012
60	1.000	1.016	1.012
70	1.000	1.017	1.014
80	1.000	1.017	1.015

りないことがわかった．(5) の結果を表 5 に示す．表 5 では，ラグランジュ緩和法 [8] を用いて得られた目的関数の下界値を 1 として，下界値との比率で示す．タブサーチ法では，Scott 法よりも目的関数値が 0.2 %から 0.3 %程度良い解を得ることができたが，その差は比較的小さいものであった．これは表 5 よりわかるように，Scott 法で求められた解の目的関数値の下界値との差が最大でもわずか 1.7 %であるため，解の改善の余地が少ないことも原因の一つと考えられる．しかし，タブサーチ法では，すべての問題で Scott 法よりも同じか良い解を求めることができ，ノード数が 30 以上ではすべての問題で優れた解を算出した．

8 おわりに

本研究では，容量制約のないネットワークデザイン問題に対する 2 種類の近似解法を提案した．Minoux 法に厳密性を加えた修正法は，Scott 法と同程度の目的関数値の解を Minoux 法と同程度の計算時間で求めることができた．また，ノード数が 100，アーク変数が 5000 程度の大規模な問題でも，パーソナルコンピュータ上で 1 分程度で良い近似解を求めることができた．タブーサーチ法では，パラメータの影響を解析することができ，Scott 法に比べ優れた解を求めることができた．得られた解の目的関数値の改善量はわずかであるが，ノード 30 以上では Scott 法よりもすべての問題で優れた解を求めることができた．

参考文献

[1] A. Balakrishnan. LP extreme points and cuts for the fixed–charge network design problem. *Mathematical Programming*, Vol. 39, pp. 263–284, 1987.

[2] A. Balakrishnan, T. L. Magnanti, and R. T. Wong. A dual–ascent procedure for large–scale uncapacitated network design. *Operations Research*, Vol. 37, pp. 716–740, 1989.

[3] J. W. Billheimer and P. Gray. Network design with fixed and variable cost elements. *Transportation Science*, Vol. 7, pp. 49–74, 1973.

[4] R. Dionne and M. Florian. Exact and approximate algorithms for optimal network design. *Networks*, Vol. 9, pp. 37–59, 1979.

[5] B. Gavish. Topological design of telecommunication networks – local access design methods. *Annals of Operations Research*, Vol. 33, pp. 17–71, 1991.

[6] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, 1997.

[7] D. S. Johnson, J. K. Lenstra, and A. H. G. Rinnooy Kan. The complexity of the network design problem. *Networks*, Vol. 8, pp. 279–285, 1978.

[8] 片山直登, 岩田実, 柳下和夫, 三原一郎, 今澤明男. ラグランジュ緩和法を用いた容量制約のないネットワーク計画問題の解法. 土木計画学研究・論文集, Vol. 11, pp. 105–112, 1993.

[9] M. Los and C. Lardinois. Combinatorial programming, statistical optimization and the optimal transportation network problem. *Transportation Research B*, Vol. 16, pp. 89–124, 1982.

[10] T. L. Magnanti, P. Mireault, and R. T. Wong. Tailoring benders decomposition for uncapacitated network design. *Mathematical Programming Study*, Vol. 26, pp. 112–155, 1986.

[11] T. L. Magnanti and R. T. Wong. Network design and transportation planning : Models and algorithms. *Transportation Science*, Vol. 18, pp. 1–55, 1984.

[12] M. Minoux. Multiflots de coût minimal avec fonctions de coût concaves. *Annales des Télécommunications*, Vol. 31, pp. 77–92, 1976.

[13] M. Minoux. Network synthesis and optimum network design problems: Models, solution methods and applications. *Networks*, Vol. 19, pp. 313–360, 1989.

[14] A. J. Scott. The optimal network problem: Some computational procedures. *Transportation Research*, Vol. 3, pp. 201–210, 1969.

[15] P. A. Steenbrink. Transport network optimization in the Dutch integral transportation study. *Transportation Research*, Vol. 8, pp. 11–27, 1974.