容量制約をもつネットワークデザイン問題の貪欲解法

片山　直登

1　はじめに

容量制約をもつネットワークデザイン問題（capacitated network design problem, *CND*）は，ノードおよび容量をもつアーク候補からなるネットワークと，ネットワーク上を流れる多品種の需要量が与えられたときに，ネットワークのデザイン費用とルーティング費用の合計が最小化となるアークの選択と各品種のフローの経路を求める問題である。*CND* は，通信ネットワークの設計，交通ネットワークの設計や輸送・配送ネットワークの設計など，幅広い分野で応用されている問題である。

ネットワーク上のノードは，通信ネットワークの中継機やコンピュータ，交通ネットワークの交差点・交通需要発生点・吸収点，輸送・配送ネットワークの配送センター・消費地などを表す。アークは，通信回線，道路や輸送経路などのノード間をつなぐものを表す。アークの容量は，通信回線の容量，交通容量や輸送能力など，単位時間当たりのアークのフロー処理能力を表す。

CND は，ネットワークデザイン問題の中でも最も難しい問題の一つとして知られている[1)]。*CND* の線形緩和問題は，多品種フロー問題（multi－commodity flow problem, *MCF*）となる。この *MCF* は線形緩和問題であるため，理論的には容易に解くことができるが，線形計画問題の中でも比較的難しい問題のひとつである。また，ノード30の完全グラフ上の問題では品種数が約400種，変数が約40万個を含む *MCF* となり，*MCF* の解法を用いても短時間で解くことは容易ではない。さらに，近似解法では繰り返し解の実行可能性を判断する必要があり，この際，タイトな *MCF* を繰り返し解くことが必要がある。一方，ノード数が少ない問題であっても，最適解において容量の残余分が発生すると，線形緩和問題と元問題との間の双対ギャップが非常に大きくなる場合がある。したがって，一般的な組合せ最適化問題で有効である線形緩和を用いた分枝限定法や，

シミュレーテッドアニーリング法，遺伝的アルゴリズムやタブーサーチ法など繰返しタイプのメタ解法を活用しにくい問題となっている。

1980年代になってから，*CND* に対する研究が数多く行われるようになってきている。片山－春日井[19]はカット制約を用いた双対上昇法を用いた解法を示し，片山－春日井[18]はフロー迂回条件を用いた妥当不等式を提案している。Magnanti－Mirchandani 1[4]は，*CND* の派生問題であるネットワークローディング問題に対する3分割不等式およびアーク剰余容量不等式を示し，これらを用いた数値解析を行っている。Bienstock－Gunluk[3]は多面体構造の解析を行い，Gunluk[11]はそれらを利用したBranch－Cut法を提案している。

一方，Lagrange緩和などを用いた緩和法を用いた解法が提案されている。片山－春日井[19]は，双対上昇法との比較の中で，フロー保存条件Lagrange緩和法を使用している。Gendron－Crainic[7][8][9]およびGendron－Crainic－Frangioni[10]は，連続緩和，フロー保存条件Lagrange緩和，容量制約Lagrange緩和などとそれらの解法を示し，数値実験を行っている。また，Crainic－Frangioni－Gendron[4]は，劣勾配法にBundle法を用いた解法を示している。Herrmann－Ioannou－Minis[12]は，容量をもたないネットワークデザインに対するBalakrishnan－Magnanti－Wongの双対上昇法[2]を拡張した解法を示しているが，この解法による下界値が真の下界値にならない例がGendron[6]により示されている。

また，*CND* に対する近似解法として，次のような研究が行われている。Gendron－Crainic[8][9]は多品種フロー問題の解法であるresoure decomposition法を利用した貪欲解法を示している。Crainic－Gendreau－Farvolden[5]は，シンプレックス法の基底解変更にタブーサーチ法を適用した近似解法を示しており，優れた解を導出している。Holmberg－Yuan[13]はLagrange緩和法と近似的な分枝限定法を用いた近似解法を示している。

一方，Minouxタイプの貪欲解法が，容量制約をもたないネットワークデザイン問題[15][16]や予算制約をもつネットワークデザイン問題[17]に適用され，高速かつ精度の高い解法であることが示されている。

本研究では，*CND* に対して貪欲的なMinouxタイプの近似解法を提案し，この解法を用いた数値実験を行い，提案した近似解法の有効性を検討する。

2　*CND* の定式化

ノード集合を N，アーク候補集合を A，品種の集合を K と表す。ここで扱う品種は $N \times N$ 上で定義される。アーク (i,j) を設置するか否かを表すデザイン変数を y_{ij} とし，アーク (i,j) 上の品種 k のフロー量を表すフロー変数を x_{ij}^k，アーク (i,j) の容量を C_{ij} とする。アーク (i,j) のデザイン費用を f_{ij}，アーク (i,j) 上の品種 k の単位フロー量当たりの

ルーティング費用を c_{ij}^k とする，品種 k の始点を O^k，終点を D^k，OD フロー量を d^k とする。また，$N^-(n)=\{j\in N|(j,n)\in A\}$, $N^+(n)=\{j\in N|(n,j)\in A\}$ とする。

このとき，*CND* の定式化は，次のように表される。

最小化
$$\sum_{(i,j)\in A}\sum_{k\in K}c_{ij}^k x_{ij}^k+\sum_{(i,j)\in A}f_{ij}\,y_{ij} \tag{1}$$

条件
$$\sum_{i\in N^-(n)}x_{in}^k-\sum_{j\in N^+(n)}x_{nj}^k=\begin{cases}-d^k & if\ \ n=O^k\\ d^k & if\ \ n=D^k \qquad for\ \ all\ \ n\in N,\ k\in K\\ 0 & otherwise\end{cases} \tag{2}$$

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

$$0\le x_{ij}^k\le d^k y_{ij}\quad for\ \ all\ \ k\in K,\ (i,j)\in A \tag{4}$$

$$y_{ij}\in\{0,\ 1\}\quad for\ \ all\ \ (i,j)\in A \tag{5}$$

(1)式は，ルーティング費用とデザイン費用の総和の最小化を表す。(2)式はフロー保存式とよばれる式であり，ノード n における流入量と流出量の差が，ノード n が品種 k の始点 O^k であれば $-d^k$，終点 D^k であれば d^k，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。これは，ネットワーク上のアークを経由して，与えられたすべての品種のフローが始点から終点に到着することを表している。(3)式の左辺はアーク (i,j) 上のフロー量の合計であり，右辺はアーク (i,j) が存在するときに C_{ij}，存在しないとき0となる。これは，アーク (i,j) が存在するときにのみフローの存在を許し，かつそのフロー量は容量以下であることを表している。(4)式は，品種 k のアーク (i,j) 上のフロー量がアーク (i,j) 存在するときに最大 d^k で最小0であり，アークが存在しないときに0であることを表している。(5)式はデザイン変数が0−1離散変数であることを示している。

CND には，2種類の変数，デザイン変数 $\boldsymbol{y}$ とフロー変数 $\boldsymbol{x}$ が含まれている。そこで，デザイン変数 $\boldsymbol{y}$ を $\overline{\boldsymbol{y}}$ に固定し，$\overline{y}_{ij}=1$ からなるアーク集合を $\overline{A}$ とする。このとき，$\boldsymbol{y}$ を $\overline{\boldsymbol{y}}$ に固定した *CND* は，フロー変数 $\boldsymbol{x}$ からなる $\overline{A}$ 上の *MCF* として表すことができる。そこで，アーク集合 A の部分集合 $\overline{A}$ に対する *MCF* の最適目的関数値とデザイン費用の和を $\phi(\overline{A})$ とおく。このとき，*CND* は，$\phi(\overline{A})$ を最小化するように，アーク集合 A から部分集合 $\overline{A}$ を選択する2段階の問題と見なすことができる。

$$\text{最小化}_{\ \forall\overline{A}\in A}\quad \phi(\overline{A}) \tag{6}$$

条件

$$\phi(\overline{A})=\text{最小化}\sum_{(i,j)\in\overline{A}}\sum_{k\in K}c_{ij}^k x_{ij}^k+\sum_{(i,j)\in\overline{A}}f_{ij} \tag{7}$$

条件

$$\sum_{i\in\overline{N}^-(n)} x_{in}^k - \sum_{j\in\overline{N}^+(n)} x_{nj}^k = \begin{cases} -d^k & if\ \ n=O^k \\ d^k & if\ \ n=D^k \\ 0 & otherwise \end{cases} \quad for\ all\ \ n\in N,\ k\in K \tag{8}$$

$$\sum_{k\in K} x_{ij}^k \leq C_{ij} \quad for\ all\ \ (i,j)\in\overline{A} \tag{9}$$

$$0\leq x_{ij}^k \leq d^k \quad for\ all\ \ k\in K,\ (i,j)\in\overline{A} \tag{10}$$

ここで，$\overline{N}^-(n)=\{j\in N|(j,n)\in\overline{A}\}$，$\overline{N}^+(n)=\{j\in N|(n,j)\in\overline{A}\}$ とする。あるアーク集合 $\overline{A}$ に対して $\phi(\overline{A})$ の値を定めることは，*MCF* の最適値を求める，すなわち *MCF* を解くことを意味している，後節の近似解法では，この定式化を利用する。

3　Minoux タイプの貪欲解法

一般的なネットワークデザインに対する基本的な解法として，アーク集合 A からアークを順々に取り除いていくデリートタイプの貪欲解法が挙げられる。そこで，*CND* に対するデリートタイプの貪欲解法の手順を示しておく。

デリートタイプの食欲解法

ステップ1　$\boldsymbol{A}$ を初期解 $\overline{A}$ とし，$\phi(\overline{A})$ を求める。$l:=1$ とする。

ステップ2　すべての $(i,j)\in\overline{A}$ に対して $\phi(\overline{A}\backslash\{(i,j)\})$ を求め，総費用の減少量 $\phi_{ij}^l:=\phi(\overline{A})-\phi(\overline{A}\backslash\{(i,j)\})$ を求める。$\overline{A}\backslash\{(i,j)\}$ が実行不可能であれば，$\phi_{ij}^l:=-\infty$ とする。

ステップ3　$\phi_{ij}^l>0$ であるアーク (i,j) が存在しなければ，終了する。

ステップ4　ϕ_{ij}^l が最大であるアーク (i^*,j^*) を求め，$\overline{A}:=\overline{A}\backslash\{(i^*,j^*)\}$, $l:=l+1$ として，ステップ2へ戻る。

この解法では，削除アークを求めるために，すべての $(i,j)\in\overline{A}$ に対して $\overline{A}\backslash\{(i,j)\}$ 上のネットワークにおいて *MCF* を解き，ϕ の値を定める必要がある。さらに，アークを削除するたびに，この計算を行う必要がある。したがって，全体として $O(|A|^2)$ 回，*MCF* を解く必要があり，計算量は膨大なものとなる。

一方，容量制約をもたないネットワークデザインに対して，ステップ2における目的関数値 ϕ を繰り返しのたびに厳密に求めるのではなく，近似値 $\tilde{\phi}$ を用いる方法が提案されており，その一つに Minoux タイプの貪欲解法がある。これは，繰り返しのたびに目的関数値の減少量を厳密に評価するのではなく，通常の場合は前回の目的関数値の減少量の評価値をそのまま利用する。そして，アークが実際の削除の対象となった場合に限り，この減少量の評価値を近似的または厳密に再計算し，この値が依然アークの中で

最大であればこのアークを削除し，そうでなければこのアークの評価値の更新だけを行う方法である。したがって，目的関数値の評価回数を大幅に減少することができるため，容量制約をもたないネットワークデザイン問題では効率的な解法となっている。

そこで，*CND* に対して，Minoux タイプの貪欲解法を適用する。*CND* に対する Minoux 法の手順を示す。

Minoux タイプの貪欲解法

ステップ 1　A を初期解 $\overline{A}$ とし，$\tilde{\phi}(\overline{A})$ を求める。$l:=1$ とする。

ステップ 2　すべての $(i,j)\in\overline{A}$ に対して $\tilde{\phi}(\overline{A}\backslash\{(i,j)\})$ を求め，目的関数値の減少量の評価値 $\varphi_{ij}^{l}:=\tilde{\phi}(\overline{A})-\tilde{\phi}(\overline{A}\backslash\{(i,j)\})$ を求める。$\overline{A}\backslash\{(i,j)\}$ が実行不可能であれば，$\varphi_{ij}^{l}:=-\infty$ とする。

ステップ 3　φ_{ij}^{l} の降順に，該当するアークと減少量の評価値をリストに格納する。

ステップ 4　$\varphi_{ij}^{l}>0$ であるアーク (i,j) が存在しなければ，終了する。

ステップ 5　リスト内にある φ_{ij}^{l} が最大であるアーク (i^*,j^*) を求め，この減少量の評価値 $\tilde{\phi}(\overline{A})-\tilde{\phi}(\overline{A}\backslash\{(i^*,j^*)\})$ を再計算し，$\varphi_{i^*j^*}^{l}:=\tilde{\phi}(\overline{A})-\tilde{\phi}(\overline{A}\backslash\{(i^*,j^*)\})$ として更新する。

ステップ 6　依然として，φ_{ij}^{l} がリスト内で最大であればステップ 8 へ行く。そうでなければ，$\varphi_{i^*j^*}^{l}$ の降順にしたがって，このアーク (i^*,j^*) をリスト内の適切な位置に移動する。

ステップ 7　リストに含まれるすべてのアーク (i,j) に対して $\varphi_{ij}^{l+1}:=\varphi_{ij}^{l}$ とし，$l:=l+1$ としてステップ 4 へ戻る。

ステップ 8　$\overline{A}:=\overline{A}\backslash\{(i^*,j^*)\}$ とし，リストからアーク (i^*,j^*) と $\varphi_{i^*j^*}^{l}$ を削除する。$l:=l+1$ としてステップ 4 へ戻る。

バックワード法とは異なり，この Minoux 法では最適値 ϕ ではなく，最適または近似的な評価値 $\tilde{\phi}$ を使用している。この解法では，ステップ 2 における初期計算を除けば，アーク (i,j) が実際に削除の対象となる場合に限り目的関数値の減少量の評価値を計算するだけである。このため，*MCF* を解く（評価値を求める）回数は最悪の場合には $O(|A|^2)$ であるが，平均的には $O(|A|)$ となる。

評価値 $\tilde{\phi}$ をどのように計算するかによって，次のようないくつかのバリエーションが考えられる。

・評価値 $\tilde{\phi}$ を ϕ の厳密な最適値とする。

・ステップ 2 における $\tilde{\phi}(\overline{A})$ を $\phi(\overline{A})$ の近似値とし，それ以外を最適値 $\phi(\overline{A})$ とする。

・すべての $\tilde{\phi}(\overline{A})$ を $\phi(\overline{A})$ の近似値とする。

容量制約をもたないネットワークに対する Minoux 法では，現在の解 $\overline{A}$ においてノード i，j 間の最小フロー路（最短路）問題を解き，アーク (i,j) 上のフローをこの最小費

用路に流し代えたフロー解を用いて，評価値 $\tilde{\phi}(\overline{A}\backslash\{(i,j)\})$ を算出している。この考えを *CND* にも，適用することが可能である。一方，一般に *MCF* は，resoure decomposition 法や Lagrange 緩和法を用いて解くことになる。これらの解法は，上界値や下界値を更新していく解法である。そこで，これらの計算を途中で打ち切り，この時点の上界値や下界値を評価値に用いることができる。

$\tilde{\phi}(\overline{A})$ の近似値をまとめる次のようになる。

- $\overline{A}$ 上のフロー解におけるアークの残余容量をアーク容量とした $\overline{A}$ からなるネットワーク上で，ノード i，j 間の1品種最小ルーティング費用問題を解き，$\overline{A}$ におけるアーク (i,j) 上のフローをこの最小費用路に流し代えたフロー解を用いて，算出する。
- *MCF* の計算を途中で打ち切り，下界値を近似値とする，
- *MCF* の計算を途中で打ち切り，上界値を近似値とする。

MCF には多くの解法が提案されているが，ここでは容量制約 Lagrange 緩和法などのような下界値を算出解法を利用する。この解法では，繰り返しの毎に下界値が上昇していく。そこで，$\overline{A}$ における *MCF* の計算中の下界値を $LOW(\overline{A})$ とおく。このとき，

$$\tilde{\phi}(\overline{A}) \leq LOW(\overline{A}\backslash\{(i,j)\}) \tag{11}$$

となれば，

$$\tilde{\phi}(\overline{A})-\tilde{\phi}(\overline{A}\backslash\{(i,j)\}) \leq \tilde{\phi}(\overline{A})-LOW(\overline{A}\backslash\{(i,j)\}) \leq 0 \tag{12}$$

となるので，目的関数値の減少量は負（増加する）になり，アーク (i,j) は削除の対象から外れ，*MCF* の計算を打ち切ることができる。

一方，ネットワークの規模が大きい場合や密な場合には，アーク候補集合 A が大きくなるため計算量が増大する，そこで，あらかじめ何らかの基準でアーク候補を適当なアーク集合 $\tilde{A}$ に絞っておけば，計算精度は低下するが計算量を抑えることができる。例えば，$|A| \leq a|N|$（a は正の定数）とすれば，*MCF* を平均的に解く回数は $O(|A|)$ から $O(|N|)$ にすることができる。基準の例として，次のようなものが挙げられる。

表1：問題の規模

ノード数	アーク変数	品種数	フロー変数
5	10	10	200
10	45	45	4050
15	105	105	22050
20	190	190	72200
25	300	300	180000
30	435	435	378450
35	595	595	708050

表２：貪欲解法による目的関数値の誤差(％)と計算時間（D/R 比＝５）

ノード数	貪欲解法１		貪欲解法２		貪欲解法３	
	誤差(％)	計算時間(S)	誤差(％)	計算時間(S)	誤差(％)	計算時間(S)
5	9.92	0.1	9.92	0.0	9.92	0.1
10	4.00	1.9	4.00	1.9	4.00	1.9
15	3.12	25.2	3.12	25.1	3.12	32.0
20	3.45	143.3	3.45	143.3	5.00	195.1
25	4.10	563.6	3.87	558.4	4.90	708.8
30	4.46	1611.0	4.25	1578.3	10.41	1707.5
35	4.61	4852.9	4.56	5153.0	12.09	3903.9

・アーク費用の安い順に $a|N|$ 本をアーク候補 $\tilde{A}$ とする。

・Lagrange 緩和法を用いて *CND* の下界値を算出し，その緩和問題の目的関数における各アークの係数の小さい順に $a|N|$ 本をアーク候補 $\tilde{A}$ とする。

しかしながら，a の大きさによって $\tilde{A}$ 自体が実行不可能解となる場合がある。これを防ぐためには，はじめに $\tilde{A}$ の実行可能性を判定し，実行不可能解であれば a の値を増加させるなどが必要となる。

4　数値実験

提案した Minoux タイプの貪欲解法を用いて，数値実験を行った。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デザイン費用とする。ルーティング費用はすべての品種で同一とし，デザイン費用に比例するものとする。すべての２ノード対間のアークを候補とする。また，すべての２ノード対間に品種が流れるものとし，すべての需要量を１とする。また，アーク容量はすべて $|N|$ とする。

ノード数は10から35まで，アークのデザイン費用/アークのルーティング費用（D/R 比）を５から20まで変化させ，同一ノード数，D/R 比に対して10組のデータを使用する。使用計算機は IBMPC 互換機（CPU Pentium1.7GHz，メモリ256Mb，OS Windows 2000)，使用言語は Microsoft Fortran Power Station Ver.4である。なお，近似解の誤差を定めるために，Lagrange 緩和法[8]を用いて下界値を算出する。

表１にノード10から35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。

表 3：貪欲解法による目的関数値の誤差(%)と計算時間（D/R 比＝10)

ノード数	貪欲解法 1		貪欲解法 2		貪欲解法 3	
	誤差(%)	計算時間 (S)	誤差(%)	計算時間 (S)	誤差(%)	計算時間 (S)
5	15.33	0.0	15.33	0.1	15.33	0.1
10	6.73	2.1	6.73	2.1	6.73	2.1
15	4.84	25.8	4.84	25.8	4.84	31.7
20	5.48	157.5	5.48	157.5	5.49	213.2
25	5.61	615.2	5.71	599.4	6.38	718.2
30	5.27	1643.0	5.06	1612.0	13.25	1680.7
35	6.17	4708.8	5.68	4961.3	17.63	3863.9

表 4：貪欲解法による目的関数値の誤差(%)と計算時間（D/R 比＝20)

ノード数	貪欲解法 1		貪欲解法 2		貪欲解法 3	
	誤差(%)	計算時間 (S)	誤差(%)	計算時間 (S)	誤差(%)	計算時間 (S)
5	13.89	0.1	13.89	0.1	13.89	0.1
10	8.11	2.3	8.11	2.3	8.11	2.3
15	6.78	25.3	6.78	25.2	6.78	31.9
20	7.05	165.5	7.05	165.5	7.40	219.3
25	7.99	625.2	8.36	611.4	8.80	759.61
30	6.97	1710.2	7.17	1666.4	14.21	1695.0
35	7.56	4734.9	7.88	5159.5	22.98	4081.0

数値実験では次の 3 種類の貪欲解法を用いた。

・貪欲解法 1：ステップ 2 では 1 品種最小費用フロー問題を解き，評価値を近似的に求める。ステップ 5 では MCF を解く。アーク候補 $\tilde{A}$ は A とする。

・貪欲解法 2：デザイン費用の安い順に $10|N|$ 本（$a=10$）のアークを候補集合 $\tilde{A}$ とする。$\tilde{A}$ が実行不可能であれば，a を実行可能になるまで 1 づつ増加させる，他は貪欲解法 1 と同じ。

・貪欲解法 3：デザイン費用の安い順に $5|N|$ 本（$a=5$）のアークを候補集合 $\tilde{A}$ とする。$\tilde{A}$ が実行不可能であれば，a を実行可能になるまで 1 づつ増加させる。他は貪欲解法 1 と同じ。

各解法によって得られた近似解の目的関数値の平均誤差と 1 問当りの平均計算時間を表に示す。D/R 比＝ 5 の場合の結果を表 2 に，D/R 比＝10の場合の結果を表 3 に，D/R 比＝20の場合の結果を表 4 に示す。平均誤差は，

100×(近似解の平均目的関数値－Lagrange 緩和法による下界値)/Lagrange 緩和法

による下界値

である。

ノード数5を除くと，貪欲解法1と2では，平均誤差は3～8％程度となった。貪欲解法1と2には，平均誤差に明確な差は見られない。一方，貪欲解法3では，ノード数が多くなると平均誤差が大きくなっている。これは，大半の初期解が実行可能でなくなり，aを調整しても，アークを選択する候補が限られるためである。また，ノード数5～15ではいずれの解法も平均誤差に違いが見られない。これは，$a=5,10$としたため，貪欲解法2,3ともに大半のアークを含むためである。ノード数5の場合，いずれの場合でも誤差が大きい。これは最適解とLagrange緩和解における双対ギャップが大きいためと考えられる。また，D/R比が大きくなると，平均誤差は増加している。これも，双対ギャップの大きさに関係すると考えられる。

ノード数が35を超えると，計算時間は1時間を超え，膨大な計算時間が必要となった。容量制約のないモデルのMinoux法がノード数35で1秒程度で解けることと比べると，計算量の違いが顕著である。貪欲解法1,2,3の間では，ノード数35を除けば計算時間の差はそれほど大きくない。解法の初期の段階では，アークを取り除いたときの*MCF*は容易に解を求めることができる。一方，アーク数が少なくなるとフローが容量に対して飽和状態に近づくため，*MCF*を解くための計算時間が増大する。したがって，計算時間に差が見られないのは，初期にアークを絞ってもほとんど効果が挙がらないことによるものと考えられる。

5　おわりに

本研究では，容量制約をもつネットワークデザイン問題に対するMinouxタイプの貪欲解法を提案した。数値実験の結果，提案した解法では，最適値（正確には下界値）との差が8％程度以内の差の近似解を算出できることがわかった。しかしながら，ノード数35を超える大規模な*CND*では，大規模な*MCF*を解く回数が増加し，膨大な計算時間が必要となることが分かった。今後は，解の精度を保ちながら，計算時間を大幅に短縮する解法を検討する必要がある。

参考文献

1）A. Balakrishnan, T. Magnanti, P. Mirchandani, (Eds M. Dell'Amico, F. Maffioli, and S. Martello). Network design. Annotated Bibliographies in Combinatorial Optimization, John Wiley & Sons, pp.311－334,1997.

2）A. Balakrishnan, T. Magnanti, and R. Wong. A dual－ascent procedure for large－scale uncapacitated network design. Operations Research, Vol.37, pp.716－740,1989.

3）D. Bienstock and O. Gunluk. Capacitated network design－polyhedral structure and computation. INFORMS Journal on Computing, Vol.8, pp.243－259,1996.

4) T.G. Crainic, A. Frangioni, and B. Gendron. Bundle－based relaxation methods for multicommodity capacitated fixed charge network design problems. Technical Report CRT－96－45, Centre de recherche sur les transports,Universite de Montreal,1998.

5) T.G. Crainic, M. Gendreau, and J.M. Farvolden. A simplex－based tabu search for capacitated network design. INFORMS Journal on Computing, Vol.12, pp.223－236,2000.

6) B. Gendron. A note on "a dual－ascent approach to the fixed－charge capacitated network design problem". Technical Report CRT－99－38, Centre de recherche sur les transports, Universite de Montreal,1999.

7) B. Gendron and T.G. Crainic. Parallel implementations of bounding procedures for multicommodity capacitated network design. Technical Report CRT－94－45, Centre de recherche sur les transports, Universite de Montreal,1994.

8) B. Gendron and T.G. Crainic. Relaxations for multicommodity capacitated network design problems. Technical Report CRT－965, Centre de recherche sur les transports,Universite de Montreal,1994.

9) B. Gendron and T.G. Crainic. Bounding procedures for multicommodity capacitated fixed charge network design problems. Technical Report CRT－96－06, Centre de recherche sur les transports,Universite de Montreal,1996.

10) B. Gendron, T.G. Crainic, and A. Frangioni. Multicommodity capacitated network design. Technical Report CRT－98－14, Centre de recherche sur les transports,Universite de Montreal,1999.

11) O. Gunluk. A branch－and－cut algorithm for capacitated network design problems. Mathematical Programming, Vol.86, pp.17－39,1999.

12) J.W. Herrmann, G. Ioannou, and I. Minis. A dual ascent approach to the fixed－charge capacitated network design problem. European Journal of Operational Research, Vol.95, pp. 476－490,1996.

13) K. Holmberg and D. Yuan. A lagrangian heuristic based branch－and－bound approach for the capacitated network design problem. Operations Reserach, Vol.48, pp.461－481,2000.

14) T. Magnanti, P. Mirchandani, and R. Vachani. The convex hull of two core capacitated network design problems. Mathematical Programming, Vol.60, pp.233－250,1993.

15) M. Minoux. Multifiots de cofit minimal avec fonctions de coilt concaves. Annales des Telecommunications, Vol.31, pp.77－92,1976.

16) 片山直登。ネットワークデザイン問題の近似解法。流通経済大学流通情報学部開校記念論文集，pp.171－191,1997。

17) 片山直登。予算制約をもつネットワークデザイン問題の近似解法。流通経済大学流通情報

学部紀要，Vol.5, pp.29－40,2001。

18）片山直登，春日井博。容量制約付きネットワークデザイン問題の強い妥当不等式。日本経営工学会秋季研究大会予稿集，pp.281－282,1992。

19）片山直登，春日井博。容量制約をもつ多品種流ネットワークデザイン問題。日本経営工学会誌，Vol.44, pp.164－175,1993。