《論　文》

非分割フローを考慮した容量制約をもつネットワーク設計問題

片　山　直　登

1　はじめに

ネットワーク設計問題は，ネットワーク上の施設・設備であるアークにかかる固定的な費用とものの移動にかかる変動的な費用を考慮して，施設・設備などに対応するアークやノードを適切に選択することによりネットワークを形成し，かつ異なる始点と終点をもつ複数の荷物，商品やデータなどの移動経路であるパスを決める問題である．この問題は，輸送，ロジスティクス，通信や生産システムなどに幅広い応用分野をもつネットワークの構造を設計する問題であり，一般的なネットワーク設計問題はNP−困難な問題であることが知られている（Magnanti and Wong 1984）．

容量制約をもつネットワーク設計問題はアークまたはノードに容量をもつ問題であり，ネットワーク設計問題の中でも最適解や良い近似解を求めることが困難な問題である．容量制約をもつネットワーク設計問題に関しては，今日まで数多くの研究が行われている．ネットワーク設計問題とその関連問題に関するサーベイとして，Magnanti and Wong（1984），Wong（1984, 1985），Minoux（1989），Balakrishnan et al.（1997），Gendron et al.（1997），Crainic（2003），Costa（2005），片山直登（2008）およびYaghini and Rahbar（2012）などがある．

通信ネットワーク上では，同一の始点と終点をもつパケットなどのフローは必ずしも単一の経路上で送信されるとは限らず，トラフィックの状態に応じて複数の経路上で分割されて送信されることが一般的である．一方，ロジスティクスやサプライチェーンのネットワーク上では，同一の発地と着地をもつ荷物が複数に分割され，別々の経路で輸送されることはなく，単一の経路上で輸送されるのが一般的である．このような分割されないフローを非分割フローとよび，この非分割フローを考慮した問題を非分割フローを考慮した容量制約をもつネットワーク設計問題（*UCND*：Unsplittable Capacitated

Network Design Problem）とよぶ．この問題では，アークのデザイン変数のみならず，パスやアークフロー変数も0–1 離散変数となり，すべての変数が離散変数である困難な組合せ最適化問題となる．ネットワーク設計問題においてデザイン変数を固定した問題は，分割フローを許す問題では多品種フロー問題，すなわち線形計画問題となるため，比較的容易に解くことができる．一方，非分割フローを考慮した問題では，デザイン変数を固定した問題でさえ 0–1 変数をもつ組合せ最適化問題となるため，最適に解くことが困難となる．

非分割フローを考慮した容量制約をもつネットワーク設計問題に対する研究は最近始まったばかりであり，それほど多くはない．従来の研究としては，Yaghiniand Kazemzadeh（2012）のシミュレーテッドアニーリング法を用いた研究，Hewitt et al.（2012）の IP 探索法および分枝価格法とガイドつき探索法を用いた研究がある．

本研究では，非分割フローを考慮した容量制約をもつ設計問題に対して，容量スケーリング法と局所探索法およびパス再結合法を組合せた近似解法を提案する．

2　問題の定式化

はじめに，*UCND* の前提条件，使用する記号および *UCND* の定義を示す．続いて，アークフローによる定式化，およびパスフローによる定式化を示す．

2．1　前提条件，記号および問題の定義

UCND の前提条件を示す．

・ノード集合が与えられている．
・向きをもつアーク集合が与えられている．
・アークには，非負のデザイン費用が与えられている．
・複数の品種からなる品種集合が与えられている．
・アークには，品種ごとの全需要に対する非負のフロー費用が与えられている．
・アークには，単位期間当たりの処理量の上限であるアーク容量が与えられている．
・各品種ごとの需要が与えられている．
・各品種の需要は，始点から終点までの 1 つのパス上を移動する．

UCND の定式化で使用する記号の定義を示す．

・N：ノード集合
・A：アーク集合
・K：品種集合
・N_n^+：ノード n を終点とするアークの終点であるノード集合
・N_n^-：ノード n を始点とするアークの始点であるノード集合

・P^k：品種 k の取り得るパス集合
・c_{ij}^k：アーク (i, j) 上における品種 k の全需要に対する非負のフロー費用
・f_{ij}：アーク (i, j) の非負のデザイン費用
・C_{ij}：アーク (i, j) のアーク容量
・d^k：品種 k の需要
・δ_{ij}^p：パス p にアーク (i, j) が含まれるとき 1，そうでないとき 0 を表す定数
・O^k：品種 k の始点
・D^k：品種 k の終点
・x_{ij}^k：品種 k のフローがアーク (i, j) 上に含まれるとき 1，そうでないとき 0 であるアークフロー変数；0-1 変数
・z_p^k：品種 k のフローがパス p 上を移動するとき 1，そうでないとき 0 であるパスフロー変数；0-1 変数
・y_{ij}：アーク (i, j) を選択するとき 1，そうでないとき 0 であるデザイン変数；0-1 変数

次に，*UCND* の定義を示す．

定義 2．1（*UCND*） デザイン費用 f，フロー費用 c，アーク容量 C をもつ向きをもつアーク集合 A が与えられ，ノード集合 N および品種の需要 d をもつ品種集合 K が与えられている．このとき，フロー費用とデザイン費用の合計を最小にするアーク集合 A' $(\subseteq A)$，およびアーク容量を満足する非分割アークフロー x または非分割パスフロー z を求めよ．

2．2　アークフローによる定式化

UCND のアークフローによる定式化を *UCNDA* とする．*UCNDA* は，次のようになる．

$(UCNDA)$

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

subject to

$$\sum_{i\in N_n^+} x_{in}^k - \sum_{j\in N_n^-} x_{nj}^k = \begin{cases} -1 & if\ n = O^k \\ 1 & if\ n = D^k \\ 0 & otherwise \end{cases} \quad \forall n \in N,\ k \in K \tag{2}$$

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

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

$$x_{ij}^k \in \{0,1\} \quad \forall k \in K, (i,j) \in A \tag{5}$$

$$y_{ij} \in \{0,1\} \quad \forall (i,j) \in A \tag{6}$$

(1)式は目的関数であり，フロー費用とデザイン費用の総和を最小化する．(2)式はフロー保存式であり，ノードに流入するフロー変数値と流出するフロー変数値の差が，品種 k の始点であれば -1，終点であれば 1，その他のノードであれば 0 であることを表す．この式は，各品種について，必ず始点から終点まで需要が移動することを保証する．(3)式は，容量制約式である．アーク (i, j) が選択されるとき，左辺はアーク上を移動するフロー量の合計であり，これが右辺のアーク容量以下であることを表す．また，アークが選択されないときはフロー量の合計が 0 であることを表す．(4)式は，アーク (i, j) における品種 k に関する強制制約式である．この式は，アーク (i, j) が選択されるときには品種 k のアークフロー変数値が最大で 1 となり，アーク (i, j) が選択されないときには 0 となることを表す．(5)式はアークフロー変数の 0–1 条件，(6)式はデザイン変数の 0–1 条件である．

UCNDA は，$|A||K|$ 個のアークフロー変数，$|A|$ 個のデザイン変数，および $|N||K| + |A| + |A||K|$ 本の制約式をもつ問題となる．

2．3　パスフローによる定式化

UCND のパスフローによる定式化 *UCNDP* を示す．

$(UCNDP)$

$$\min \sum_{(i,j)\in A} \sum_{k\in K} c_{ij}^k \sum_{p\in P^k} \delta_{ij}^p z_p^k + \sum_{(i,j)\in A} f_{ij} y_{ij} \tag{7}$$

subject to

$$\sum_{p\in P^k} z_p^k = 1 \quad \forall k \in K \tag{8}$$

$$\sum_{k\in K} \sum_{p\in P^k} \delta_{ij}^p d^k z_p^k \le C_{ij} y_{ij} \quad \forall (i,j) \in A \tag{9}$$

$$\sum_{p\in P^k} \delta_{ij}^p z_p^k \le y_{ij} \quad \forall k \in K, (i,j) \in A \tag{10}$$

$$z_p^k \in \{0,1\} \quad \forall p \in P^k, k \in K \tag{11}$$

$$y_{ij} \in \{0,1\} \quad \forall (i,j) \in A \tag{12}$$

(7)式は目的関数であり，フロー費用とデザイン費用の総和を最小化する．ここで，$\sum_{p\in P^k}\delta_{ij}^p z_p^k$ は x_{ij}^k に一致する．(8)式は，品種 k のパスフロー変数値の合計が 1，すなわち単一のパス上のみにフローが流れることを表す．(9)式は，アーク (i, j) が選択されるときはアーク上を移動するフロー量の合計がアーク容量以下であり，アークが選択されないときは 0 であることを表す容量制約式である．(10)式は，アーク (i, j) における品種 k に関する強制制約式である．(11)式はパスフロー変数の 0–1 条件，(12)式はデザイン変数の 0–1 条件である．

UCNDP は，$\sum_{k\in K}|P^k|$ 個である指数個のパスフロー変数，$|A|$ 個のデザイン変数と $|K|+|A|+|A||K|$ 本の制約式をもつ問題となる．0–1 変数が指数個と膨大なものとなるので，小規模な問題であってもこの定式化を直接解くことは困難である．実際には，逐次，必要なパスフロー変数を生成して問題を解く列生成法が用いられる．この列生成法をうまく適用すれば，アークフローによる定式化の場合よりも，陽的に使用する変数の数を抑えることができる．

アークフロー変数およびパスフロー変数が 0–1 変数であること以外，基本的には容量制約をもつネットワーク設計問題と同一である．そこで，容量制約をもつネットワーク設計問題に対する近似解法である容量スケーリング法と局所分枝法を適用し，さらにパス再結合法を併用する．

3　容量スケーリング法

3．1　線形緩和問題と容量スケーリング

容量スケーリング法は，線形緩和問題の解をもとに，アーク容量を変更して繰り返し線形緩和問題を解き，0–1 変数解を導く近似解法である．列生成法は限定主問題を解き，適時，被約費用が負となる変数を生成する方法である．また，行生成法は，列生成法により生成された変数に対する有効な制約式を逐次追加する方法である．これらの方法については，容量制約をもつネットワーク設計問題に対する片山直登（2010）の研究に詳細が記載されており，アークフロー変数およびパスフロー変数が 0–1 変数であることを除けば同一の解法である．

UCNDP において，デザイン変数およびパスフロー変数の 0–1 条件を 0 から 1 の連続数に緩和した線形緩和問題 LR を考える．

(LR)

$$\min \sum_{(i,j)\in A}\sum_{k\in K} c_{ij}^k \sum_{p\in P^k}\delta_{ij}^p z_p^k + \sum_{(i,j)\in A} f_{ij}y_{ij} \tag{13}$$

subject to

$$\sum_{p\in P^k} z_p^k = 1 \quad \forall k \in K \tag{14}$$

$$\sum_{k\in K}\sum_{p\in P^k} \delta_{ij}^p d^k z_p^k \le C_{ij} y_{ij} \quad \forall (i,j) \in A \tag{15}$$

$$\sum_{p\in P^k} \delta_{ij}^p z_p^k \le y_{ij} \quad \forall k \in K, (i,j) \in A \tag{16}$$

$$0 \le z_p^k \le 1 \quad \forall p \in P^k, k \in K \tag{17}$$

$$0 \le y_{ij} \le 1 \quad \forall (i,j) \in A \tag{18}$$

LR において，繰り返し回数 l のときのアーク（i, j）上のアーク容量を C_{ij}^l とした問題を $LR(C^l)$とし，繰り返し毎にアーク容量を変化させる．LR の l 回目の繰り返しにおける $LR(C^l)$は，次のようになる．

$(L(C^l))$

$$\min \sum_{(i,j)\in A}\sum_{k\in K} c_{ij}^k \sum_{p\in P^k} \delta_{ij}^p z_p^k + \sum_{(i,j)\in A} f_{ij} y_{ij} \tag{19}$$

subject to

$$\sum_{p\in P^k} z_p^k = 1 \quad \forall k \in K \tag{20}$$

$$\sum_{k\in K}\sum_{p\in P^k} \delta_{ij}^p d^k z_p^k \le C_{ij}^l y_{ij} \quad \forall (i,j) \in A \tag{21}$$

$$\sum_{p\in P^k} \delta_{ij}^p z_p^k \le y_{ij} \quad \forall k \in K, (i,j) \in A \tag{22}$$

$$0 \le z_p^k \le 1 \quad \forall p \in P^k, k \in K \tag{23}$$

$$0 \le y_{ij} \le \frac{C_{ij}}{C_{ij}^l} \quad \forall (i,j) \in A \tag{24}$$

ここでは，デザイン変数の上限値である 1 を変更し，$y_{ij}^s \le C_{ij}/C_{ij}^l$に変更している．

パラメータ λ $(0 < \lambda < 1)$，$LR(C^{l-1})$におけるアーク(i, j)上の総フロー量 $\tilde{X}_{ij}$ およびアーク(i, j)のアーク容量 C_{ij}^{l-1}を用いて，$LR(C^l)$におけるアーク容量を次のように更新する．

$$C_{ij}^{l} := \lambda \tilde{X}_{ij} + (1-\lambda) C_{ij}^{l-1} \quad \forall (i,j) \in A \tag{25}$$

ここで，$LR(C^{l-1})$における実行可能なパスフロー変数を$\tilde{z}$とすると，$\tilde{X}_{ij}$は次式となる．

$$\tilde{X}_{ij} = \sum_{k \in K} \sum_{p \in \bar{P}^k} \delta_{ij}^{p} d^k \tilde{z}_p^k \quad \forall (i,j) \in A \tag{26}$$

一方，⑯式があるために，最適解において必ずしも$\tilde{y}_{ij} = \tilde{X}_{ij}/C_{ij}^{l}$が成り立たず，$C_{ij}^{l}\tilde{y}_{ij} \geq \tilde{X}_{ij}$となり，$C_{ij}^{l}\tilde{y}_{ij}$は$\tilde{X}_{ij}$の上限値となる．そこで，$LR(C^{l-1})$におけるアーク$(i, j)$上のデザイン変数$\tilde{y}_{ij}$を用いて，$LR(C^{l})$におけるアーク容量を次のように更新することもできる．

$$C_{ij}^{l} := \lambda C_{ij}^{l-1} \tilde{y}_{ij} + (1-\lambda) C_{ij}^{l-1} = \{1 - \lambda(1-\tilde{y}_{ij})\} C_{ij}^{l-1} \quad \forall (i,j) \in A \tag{27}$$

容量スケーリング法のアルゴリズムを Algorithm1に示す．ここで，ITE_{min}は容量スケーリング法の最小繰り返し回数，ITE_{max}は容量スケーリング法の最大繰り返し回数であり，UB は上界値である．ここで，Column Generation and Row Generation は列生成と行生成法，Restricted Branch and Bound は限定分枝限定法，Path Relinking はパス再結合法であり，Local Branching は局所分枝法である．限定分枝限定法により実行可能解が見つかるか，繰り返し回数が上限を超えたときに，アルゴリズムが終了する。

3．2　限定主問題

$LR(C^{l})$には非常に多くのパスフロー変数が含まれるため，直接解くことは困難である．そこで，あらかじめすべてのパスフロー変数を含む問題を対象とするのではなく，逐次，必要なパスフロー変数を生成し，問題に追加していく．生成するパスフロー変数が単体法の列に相当することから，このような方法を列生成法とよぶ．

一方，$LR(C^{l})$には，非常に多くの強制制約式である㉒式が含まれている．しかし，列生成により生成されたパスフロー変数が含まれる強制制約式はそれほど多くなく，生成されたパスフロー変数が左辺に含まれていない強制制約式は不要なものとなる．そこで，生成したパスフロー変数が初めて左辺に含まれる強制制約式を逐次生成し，問題に追加する．生成する制約式が単体法の行に相当することから，このような方法を行生成法とよぶ．なお，適時，線形緩和解を除外するような有効な強制制約式のみを加えることも可能であり，このような方法を切除平面法をよぶ．

Algorithm 1: Capacity Scaling

Set λ, ITE_{min}, ITE_{max};
$C^1 := C$; $UB := \infty$; $l := 1$;
repeat
  Solve $LR(C^l)$ by Column Generation and Row Generation;
  Get the design solution $\tilde{y}$ and the total flow solution $\tilde{X}$ of $LR(C^l)$;
  Restricted Branch and Bound;
  Path Relinking;
  $l := l + 1$;
  for $(i, j) \in A$ **do**
    $C^l_{ij} := \lambda \tilde{X}_{ij} + (1 - \lambda) C^{l-1}_{ij}$ or $C^l_{ij} := \{1 - \lambda(1 - \tilde{y}_{ij})\} C^{l-1}_{ij}$;
  end
until *If* $l \geq ITE_{min}$ *and* $UB \neq \infty$, *or* $l \geq ITE_{max}$
Path Relinking;
Local Branching;

品種 k の適当なパスの部分集合 $\bar{P} = (\bar{P}^k)$ および適当な強制制約式の部分集合 $A\bar{K}(\subset A \times K)$が求められているものとする．このとき，パス集合が $\bar{P}$，強制制約式が $A\bar{K}$ に限定されている次のような限定主問題 $LRR(C^l, \bar{P}, A\bar{K})$ を考える．

$(LRR(C^l, \bar{P}, A\bar{K}))$

$$\min \quad \sum_{(i,j)\in A} \sum_{k\in K} c^k_{ij} \sum_{p\in \bar{P}^k} \delta^p_{ij} z^k_p + \sum_{(i,j)\in A} f_{ij} y_{ij} \tag{28}$$

subject to

$$\sum_{p\in \bar{P}^k} z^k_p = 1 \quad \forall k \in K \tag{29}$$

$$\sum_{k\in K} \sum_{p\in \bar{P}^k} \delta^p_{ij} d^k z^k_p \leq C^l_{ij} y_{ij} \quad \forall (i,j) \in A \tag{30}$$

$$\sum_{p\in \bar{P}^k} \delta^p_{ij} z^k_p \leq y_{ij} \quad \forall (k,i,j) \in A\bar{K} \tag{31}$$

$$0 \leq z^k_p \leq 1 \quad p \in \bar{P}^k, k \in K \tag{32}$$

$$0 \leq y_{ij} \leq \frac{C_{ij}}{C^l_{ij}} \quad \forall (i,j) \in A \tag{33}$$

(31)式はアーク $(i,\ j)$ を通る品種 k のパスフロー変数が生成されているときのみ存在する強制制約式となる．

この問題は線形計画問題であるため，$\bar{P}$ や $A\bar{K}$の要素数が少なければ，汎用の最適化ソルバーを用いて解くことができる．

3．3　列生成法と行生成法

$LRR(C^l, \bar{P}, \bar{AK})$はパスフロー変数が限定された問題であるため，最適解を求めるためには，逐次，基底に入るであろう新たなパスフロー変数を生成しなければならない．そのために，価格付け問題とよばれる問題を解き，被約費用が負であるパスフロー変数を求める．パスフロー変数とそれに対応するパスを$\bar{P}^k$に加え，再度$LRR(C^l, \bar{P}, \bar{AK})$を解き直す．この操作を被約費用が負である変数がなくなるまで繰り返す．被約費用が負である変数がなければ，$LR(C^l)$の最適解が得られたことになる．

(29)式に対する双対変数をs，(30)，(31)式に対する非負の双対変数をu，wとする．これらの値は，$LRR(C^l, \bar{P}, \bar{AK})$を最適化ソルバーにより最適に解くことにより求めることができる．

パスフロー変数 z に関する被約費用は，

$$\sum_{(i,j)\in A} \delta_{ij}^p (c_{ij}^k + d^k u_{ij} + w_{ij}^k) - s^k \quad \forall p \in P^k, k \in K \tag{34}$$

である．アーク$(i,\ j)$の長さを$c_{ij}^k + d^k u_{ij} + w_{ij}^k$としたとき，$\sum_{(i,j)\in A} \delta_{ij}^p (c_{ij}^k + d^k u_{ij} + w_{ij}^k)$はパス$p$の長さとなる．また，$s^k$は品種$k$の現在のパス集合$\bar{P}^k$におけるパスの最短距離である．被約費用は「パス$p$の長さ－現在の最短距離」であるので，被約費用が負である変数を見つけることは現在の最短距離よりも短いパスを見つけることになる．

s^kは定数項として扱えるので，被約費用が負であるパスフロー変数を見つけるには，品種kに対して，(34)式の第一項を最小化するパスpを見つければ良い．したがって，$LRR(C^l, \bar{P}, \bar{AK})$における品種$k$に関する価格付け問題は，次のような問題$PR^k$に帰着される．

(PR^k)

$$\min \sum_{p\in P^k} \Big\{ \sum_{(i,j)\in A_{\bar{P}^k}} \delta_{ij}^p (c_{ij}^k + d^k u_{ij} + w_{ij}^k) + \sum_{(i,j)\in A\setminus A_{\bar{P}^k}} \delta_{ij}^p (c_{ij}^k + d^k u_{ij}) - s^k \Big\} z_p^k \tag{35}$$

subject to

$$\sum_{p\in P^k} z_p^k = 1 \tag{36}$$

$$0 \le z_p^k \le 1 \quad \forall p \in P^k \tag{37}$$

ここで，$A_{\bar{P}^k}$は$\bar{P}^k$に含まれるアーク集合である。また，品種k，アーク$(i,\ j)$の需要に関する強制制約式である(22)式が生成されていない，すなわち$\bar{AK}$に含まれない制約式に対するw_{ij}^kは0とし，(35)式には含めていない．

s^kは定数であるので，PR^kはアーク$(i,\ j)$の長さが非負の$c_{ij}^k + d^k u_{ij} + w_{ij}^k$または

$c_{ij}^k + d^k u_{ij}$とした品種kに対する始点・終点間の最短路問題に帰着され，この問題はダイクストラ法により容易に解くことができる．

Algorithm 2: Column Generation and Row Generation

```
Set P̄ and ĀK := φ;
repeat
    Solve LRR(C^l, P̄, ĀK) by an MIP solver;
    Get the optimal dual solution (s, u, w) of LRR(C^l, P̄, ĀK);
    for k ∈ K do
        Solve PR^k;
        Get the shortest path p^k and the shortest path length t^k;
        if t^k < s^k then
            p^k is added to P̄^k and generate path variable z^k_{p^k}
            for (i, j) ∈ A do
                if (i, j, k) ∉ ĀK and δ^{p^k}_{ij} = 1 then (i, j, k) is add to ĀK;
            end
    end
until no path variable is generated
```

この最短路問題の最短パスをp^k，最短パスの長さをt^kとする．このとき，$t^k - s^k < 0$であれば，品種kにおける被約費用が負であるパスフロー変数が見つかったことになり，p^kが生成すべきパス，パスに対応するパスフロー変数$z_{p^k}^k$が生成すべき変数となる．また，すべての品種kについて$t^k - s^k \geq 0$であれば，被約費用が負である変数が存在しないため，$LR(C^l)$が最適に解けたことになる．続いて，生成したp^k上のアークに関して，要素(i,j,k)を$\bar{AK}$に追加する．

容量スケーリング法では，容量を変更して繰り返し線形緩和問題を最適化ソルバー最適により解く．このため，$LR(C^l)$では，$LR(C^{l-1})$までに生成したパス集合を初期集合として利用できる．列生成法と行生成法のアルゴリズムを Algorithm2 に示す．

4　近似解法

容量スケーリング法により生成されたデザイン解をもとにした限定分枝限定法，パス再結合法および局所分枝法の 3 種類の近似解法を使用する．

4．1　限定分枝限定法

容量スケーリング法により，$LR(C^l)$のすべてのデザイン変数が 0 または 1 に収束し，かつそれらの値にデザイン変数を固定した問題において実行可能な非分割フローが求まれば，*UCNDP* の実行可能解が求まることになる．しかしながら，容量スケーリング法では，必ずしもすべてのデザイン変数が 0 または 1 に収束する保証はなく，また多くの

繰り返しが必要となる場合も多い．そこで，大半のデザイン変数が収束したとしても，非分割フローを最適に求めることも困難である．

Algorithm 3: Restricted Branch and Bound

Set ϵ, B, B_{min}, ΔB, α and T;
if $B \geq B_{min}$ **then**
  for $(i,j) \in A$ **do**
    if $\tilde{y}_{ij} < \epsilon$ **then** $y_{ij} = 0$;
    else if $\tilde{y}_{ij} > 1 - \epsilon$ **then** $y_{ij} = 1$;
    else y_{ij} is free;
  end
  if *the number of free variables of* y *is less than* B **then**
    Solve $UCNDA$ associated with the restricted design variable y and adding $Z_{UCNDA} < UB \times (1 + \alpha)$ as a constraint by an MIP solver within T;
    Get the objective function value Z_{UCNDA} and the design solution $\bar{y}$ of $UCNDA$;
    $B := \max\{B_{min}, B - \Delta B\}$;
    if $Z_{UCNDA} < UB$ **then**
      $UB := Z_{UCNDA}$ and $\hat{y} := \bar{y}$;

そこで，収束していないデザイン変数およびフロー変数のみを変数とし，それ以外を固定した限定分枝限定法を適用する．適当な判定基準値を設定し，収束していないデザイン変数の数がこの判定基準値以下となったときに，*UCNDA* に対して分枝限定法を適用する．*UCNDP* ではなく，*UCNDA* に対して分枝限定法を適用するのは，最適解に含まれるパスフロー変数を生成しておくことが困難なためである．このような限定分枝限定法を適用した後，判定基準値を一定値だけ減じ，限定分枝限定法を繰り返す．このように変数を限定しても *UCNDA* の最適解を求めることは困難であるので，適当な計算時間の上限を設定する．さらに，現在の最良の目的関数値である上界値の一定倍以下になるという条件を制約式として加えておき，解法の収束を限定操作を強化する．目的関数値が最良の上界値未満である制約を追加すれば計算時間を短縮できるが，後に示すパス再結合法のための目標値を算出するために，目的関数値が最良の上界値よりも $1 + \alpha$ 倍までという制約を追加する．

限定分枝限定法のアルゴリズムを Algorithm3に示す．ここで，ϵ は限定分枝限定法における閾値，Bは限定分枝限定法の適用基準の初期値，ΔBは限定分枝限定法の適用基準の減少量，B_{min}は限定分枝限定法の適用基準の最小値，α は限定分枝限定法の上限パラメータであり，Tは計算時間の上限である．

Algorithm 4: Path Relinking

Set T;
Get current best design solution $\hat{y}$, and current design solution $\bar{y}$ by Restricted Branch and Bound or Local Branching;
if *New current design solution* $\tilde{y}$ *is found* **then**
  for $(i,j) \in A$ **do**
    if $\hat{y}_{ij} = \bar{y}_{ij}$ **then** $y_{ij} = \hat{y}_{ij}$;
    else y_{ij} is free;
  end
  Solve $UCNDA$ associated with the restricted design variable y by an MIP solver within T;
  Get the objective function value Z_{UCNDA} and the design solution $\tilde{y}$ of $UCNDA$;
  if $Z_{UCNDA} < UB$ **then**
    $UB := Z_{UCNDA}$ and $\hat{y} := \tilde{y}$;

4．2　パス再結合法

パス再結合法は，解集合から初期解と目標解の2つの解を選択し，これら2つの解の中間的な解を探索する近似解法である．ネットワーク設計問題に対しては，Ghamlouche et al.（2004）がパス再結合法を用いた解法を示している．

本研究では，現在までの最良解を初期解とし，限定分枝限定法により得られた実行可能解を目標解に設定する．これらの解において，2つのデザイン変数の値が一致すればその値に固定し，そうでなければ自由変数とした問題を作成し，最適化ソルバー最適により解くことにする．現在までの最良解を $\hat{y}$，限定分枝限定法により得られた実行可能解を $\bar{y}$ とする．このときアーク (i, j) に対して，初期解と目標解が一致，すなわち $\hat{y}_{ij} = \bar{y}_{ij}$ であれば，このデザイン変数 y_{ij} を $\hat{y}_{ij}$ に固定する．$\hat{y}_{ij} \neq \bar{y}_{ij}$ であれば，このデザイン変数を 0-1 の自由変数とする．適当な計算時間の上限 T のもとで，この制限をもつ *UCANDA* を解く．パス再結合法のアルゴリズムを Algorithm4 に示す．

4．3　局所分枝法

これまでに示した解法から得られた中の最良解に対して，局所分枝法（Fischetti and Lodi 2003）を適用して，より良い解を探索する．局所分枝法は，局所分枝制約によって制約された問題を直接的に解くことによって，現在の解を近傍を効率的に探索する解法である．本研究では，以下のような局所分枝法を利用する。

適当な実行可能解 $\hat{y}$ が求められたときに，近傍パラメータを $M(>0)$ とすると，局所分枝制約は次のようになる．

$$\sum_{(i,j)\in A|\hat{y}_{ij}=1} (1 - y_{ij}) + \sum_{(i,j)\in A|\hat{y}_{ij}=0} y_{ij} \leq M, \tag{38}$$

Algorithm 5: Local Branching

Set M, ΔM and T;
Get current best design solution $\hat{y}$ and current best upper bound UB by Path Relinking;
Add the equation (38) and (39) to $UCNDA$;
repeat
  Solve $UCNDA$ associated with local branching constraints by an MIP solver within T;
  if *a feasible solution of* $UCNDA$ *is found* **then**
    Get the upper bound Z_{UCNDA} and the feasible design solution of $UCNDA$;
    Remove the equation (38) and add the equation (40) to $UCNDA$;
    if $Z_{UCNDA} < UB$ **then**
      $UB := Z_{UCNDA}$ and $\hat{y} := y'$;
      Add the equation (38) and (39) to $UCNDA$;
  else $M := M/\Delta M$;.
until $M < M_{min}$

$$\sum_{(i,j)\in A|\hat{y}_{ij}=1} (1 - y_{ij}) + \sum_{(i,j)\in A|\hat{y}_{ij}=0} y_{ij} > 1. \tag{39}$$

第一の制約は，現在の解 $\hat{y}$ の $M-OPT$ 近傍を含む領域を表す制約である．第二の制約は $\hat{y}$ を除外する制約である．

$\hat{y}$ に対する局所分枝制約である(38)式と(39)式を $UCNDA$ に加えた問題を最適化ソルバーで解く．この問題を最適に解くことは困難な場合があるため，計算時間の上限 T を設定する．もし，現在の解よりも良い実行可能解 y' が見つかれば，$\hat{y} := y'$ として現在の最良解を更新する．次の繰り返しにおいて(38)式を次式で置き換える．

$$\sum_{(i,j)\in A|\hat{y}_{ij}=1} (1 - y_{ij}) + \sum_{(i,j)\in A|\hat{y}_{ij}=0} y_{ij} > M + 1. \tag{40}$$

現在の解よりも良い解が見つからない場合，近傍パラメータ M を減じて，再度局所分枝を実施する．また，局所分枝法により最良解が得られた場合，再度，パス再結合法を行う．局所分枝法のアルゴリズムを Algorithm5 に示す．ここで，M は局所分枝における近傍の初期値，ΔM は近傍の減少量，M_{min} は近傍の最小値である．

5　数値実験

Hewitt et al.（2012）が用いているベンチマーク問題である C 問題に対して，数値実験を行い，従来の研究との比較を行う．このベンチマーク問題は31問であり，これらの問題はフロー変数が 0–1 である以外は容量制約をもつのネットワーク設計問題で用いられるベンチマーク問題と同じインスタンスである．

本研究では，限定分枝限定法のみを行う方法（RBB），限定分枝限定法と局所分枝を行う方法（LBR），およびすべての解法を行う方法（PRL）の結果を示す．

比較する研究は，Yaghini and Kazemzadeh（2012）のシミュレーテッドアニーリング法（SA）と Hewitt et al.（2012）の IP 探索法（IPS）および分枝価格ガイドつき探索法（BPG）である．なお，記載した上界値と計算時間は，各論文に記載されているものである．また，上界値の誤差を算出するために，アークフローによる定式化 *UCNDA* を数理計画ソルバー CPLEX（最大30時間）により解き，下界値または最適値（LB/OPT）を求めている．なお，同時に CPLEX により上界値（CPLEX）も求めている．

本研究の解法で設定した主な条件およびパラメータなどは以下の通りである．

・ 使用 OS および言語：UBUNTU 12.04，GNU C++ compiler
・ 最適化ソルバー：CPLEX 12.4（Parallel Version）
・ 容量の変更方法：$C^l := \{1-\lambda(1-\tilde{y})\}C^{l-1}$
・ スケーリングパラメータ λ：0.025～0.250
・ 容量スケーリング法の最小繰り返し回数 ITE_{min}：100
・ 容量スケーリング法の最大繰り返し回数 ITE_{max}：1000
・ 限定分枝限定法における閾値 ϵ：0.01
・ 限定分枝限定法の適用基準の初期値 B：150
・ 限定分枝限定法の適用基準の減少量 ΔB：5
・ 限定分枝限定法の適用基準の最小値 B_{min}：5
・ 限定分枝限定法の上限パラメータ α：1.02
・ 計算時間の上限 T：2000秒
・ 局所分枝の近傍 M：20
・ 局所分枝の近傍の減少量 ΔM：2
・ 局所分枝の近傍の最小値 M_{min}：2

なお，各論文内で使用しているコンピュータなどは以下の通りである．

・SA：論文に不記載
・IPS，BPG：CPU INTEL Xeon 2.26GHz 8Core，RAM 24GByte
・RBB，LBR，PRL および CPLEX：CPU INTEL i7 2600 3.4GHz 4Core，RAM16GByte

C 問題に対する解法別の上界値の平均誤差を表 1 に示す．なお，SA では31問中の比較的小さな15問の解のみが掲載されている．SA の平均誤差は14.99% と大きく，IP では2.70%，BPG では1.72% であり，最適化ソルバーである CPLEX では1.89% となっている．一方，限定分枝限定法のみの解法である RBB では1.58%，局所分枝法を加えた解法である LBR では1.46%，パス再結合を組み合せた解法である PRL では1.34% であり，いずれの解法においても従来法よりの良い解を求めることができている．C 問題に対する

表１：上界値の平均誤差の比較（%）

CPLEX	SA	IP	BPG	RBB	LBR	PRL
1.89	14.99*	2.70	1.72	1.58	1.46	1.34

*：15 out of 31 instances

表２：平均計算時間の比較（秒）

CPLEX	IP	BPG	RBB	LBR	PRL
62911.0	1800.0	1800.0	5491.2	7566.3	14503.2

平均計算時間を表２に示す．SA では計算時間が未掲載であり，IP と BPG では30分の上限時間を設けている．一方，CLPEX では62911.0秒を要し，RBB では5491.2秒，LBR では7566.3 秒，PRL では14503.2秒を要している．提案した解法では，最良解を算出したパラメータによる計算時間であり，パラメータチューニングを行わない場合，トータルの計算時間はこれらの数倍となる．

表３に，C 問題の個別の問題ごとの最適値 / 下界値，CPLEX および各解法により得られた上界値を示す．“*O*”最適値，“*L*”は下界値であることを表している。また，ボールド体は最適値，イタリック体は最良値を表している。31問の内で最適解を求めることができたのは，SA では０問，IP では９問，BPG では８問である．一方，RBB では２問，LBR では４問と少ないが，PRL では12問である．CPLEX で最適解が求められたすべての問題に対して，PRL でも最適解を求めることができている．また，最適解を除いた最良解を算出したのは，RBB では２問，LBR では15問であり，PRL では19問である．PRL では，すべての問題に対して，最適解または最良解を求めることができている．

表４に，CPLEX および各解法により得られた上界値の誤差を示す．最大誤差は，SA では34.59%, IP では9.92%, BRG では5.57% であり，いずれも最大誤差が５% を超えている．一方, RBB では4.87%, LBR では4.77%, PRL では4.53% であり，すべて５% 以内に収まっている．

表 3 ：個別の問題に対する上界値の比較

Problem	LB/OPT	CPLEX	SA	IP	BRG	RBB	LBR	PRL
20/230/040/V/L	423933.0^O	**423933**		**423933**	**423933**	424075	424075	**423933**
20/230/040/V/T	398870.0^O	**398870**		**398870**	**398870**	400380	400380	**398870**
20/230/040/F/T	668699.0^O	**668699**	670928	**668699**	**668699**	669642	669642	**668699**
20/230/200/V/L	93592.5^L	95187	118785	95695	94843	94644	*94644*	*94644*
20/230/200/F/L	135466.2^L	139570		141700	138476	*138084*	*138084*	*138084*
20/230/200/V/T	98229.0^L	98899	112792	100884	98947	98674	*98610*	*98610*
20/230/200/F/T	133762.1^L	139329	167006	141734	139889	137618	*137594*	*137594*
20/300/040/V/L	430253.0^O	**430253**	433929	**430253**	430314	**430253**	**430253**	**430253**
20/300/040/F/L	597059.0^O	**597059**		**597059**	**597059**	597170	**597059**	**597059**
20/300/040/V/T	501766.0^O	**501766**	511384	**501766**	501889	511019	511019	**501766**
20/300/040/F/T	643395.0^O	**643395**	656412	**643395**	**643395**	651667	644453	**643395**
20/300/200/V/L	74283.8^L	76194	91702	76946	76333	75820	*75802*	*75802*
20/300/200/F/L	112574.7^L	119278	151513	119590	118204	117814	*117617*	*117617*
20/300/200/V/T	75341.6^L	76631	88550	77055	76986	76549	*76357*	*76357*
20/300/200/F/T	105683.4^L	110417		110516	109776	109875	*109385*	*109385*
30/520/100/V/L	54387.0^O	**54387**	57422	54427	**54387**	54432	**54387**	**54387**
30/520/100/F/L	93815.1^L	96176		97199	96277	96357	95947	*95877*
30/520/100/V/T	53812.0^O	**53812**		**53812**	**53812**	53971	53961	**53812**
30/520/100/F/T	98283.3^L	99331		100391	99355	99355	*99204*	*99204*
30/520/400/V/L	112194.9^L	114835	134587	116465	114867	114405	*114295*	*114295*
30/520/400/F/L	147314.5^L	152169		156433	152837	150909	*150965*	*150965*
30/520/400/V/T	114818.7^L	116544		118193	116730	116318	*116306*	*116306*
30/520/400/F/T	150272.4^L	157483		161513	155982	155820	155720	*155641*
30/700/100/V/L	47883.0^O	**47883**	51505	**47883**	**47883**	**47883**	**47883**	**47883**
30/700/100/F/L	60347.3^L	*60384*	68179	61254	60384	60573	*60384*	*60384*
30/700/100/V/T	47670.0^O	**47670**		47736	47686	47766	47733	**47670**
30/700/100/F/T	56686.0^O	**56686**		56931	56809	56942	56934	**56686**
30/700/400/V/L	96945.2^L	99398		100368	98850	98538	*98535*	*98535*
30/700/400/F/L	131078.1^L	141180		144077	138376	137468	137337	*137017*
30/700/400/V/T	94526.0^L	96801		98040	97352	96468	96475	*96366*
30/700/400/F/T	127884.2^L	135693	168070	135512	133759	*132112*	*132112*	*132112*

表 4 ：個別の問題に対する誤差の比較

Problem	CPLEX	SA	IP	BRG	RBB	LBR	PRL
20/230/040/V/L	0.00		0.00	0.00	0.03	0.03	0.00
20/230/040/V/T	0.00		0.00	0.00	0.38	0.38	0.00
20/230/040/F/T	0.00	0.33	0.00	0.00	0.14	0.14	0.00
20/230/200/V/L	1.70	26.92	2.25	1.34	1.12	1.12	1.12
20/230/200/F/L	3.03		4.60	2.22	1.93	1.93	1.93
20/230/200/V/T	0.68	14.83	2.70	0.73	0.45	0.39	0.39
20/230/200/F/T	4.16	24.85	5.96	4.58	2.88	2.86	2.86
20/300/040/V/L	0.00	0.85	0.00	0.01	0.00	0.00	0.00
20/300/040/F/L	0.00		0.00	0.00	0.02	0.00	0.00
20/300/040/V/T	0.00	1.92	0.00	0.02	1.84	1.84	0.00
20/300/040/F/T	0.00	2.02	0.00	0.00	1.29	0.16	0.00
20/300/200/V/L	2.57	23.45	3.58	2.76	2.07	2.04	2.04
20/300/200/F/L	5.95	34.59	6.23	5.00	4.65	4.48	4.48
20/300/200/V/T	1.71	17.53	2.27	2.18	1.60	1.35	1.35
20/300/200/F/T	4.48		4.57	3.87	3.97	3.50	3.50
30/520/100/V/L	0.00	5.58	0.07	0.00	0.08	0.00	0.00
30/520/100/F/L	2.52		3.61	2.62	2.71	2.27	2.20
30/520/100/V/T	0.00		0.00	0.00	0.30	0.28	0.00
30/520/100/F/T	1.07		2.14	1.09	1.09	0.94	0.94
30/520/400/V/L	2.35	19.96	3.81	2.38	1.97	1.87	1.87
30/520/400/F/L	3.30		6.19	3.75	2.44	2.48	2.48
30/520/400/V/T	1.50		2.94	1.66	1.31	1.30	1.30
30/520/400/F/T	4.80		7.48	3.80	3.69	3.63	3.57
30/700/100/V/L	0.00	7.56	0.00	0.00	0.00	0.00	0.00
30/700/100/F/L	0.06	12.98	1.50	0.06	0.37	0.06	0.06
30/700/100/V/T	0.00		0.14	0.03	0.20	0.13	0.00
30/700/100/F/T	0.00		0.43	0.22	0.45	0.44	0.00
30/700/400/V/L	2.53		3.53	1.96	1.64	1.64	1.64
30/700/400/F/L	7.71		9.92	5.57	4.87	4.77	4.53
30/700/400/V/T	2.41		3.72	2.99	2.05	2.06	1.95
30/700/400/F/T	6.11	31.42	5.96	4.59	3.31	3.31	3.31

6　おわりに

本研究では，非分割フローを考慮した容量制約をもつ設計問題に対して，容量スケーリング法，限定分枝限定法，局所探索法およびパス再結合法を組合せた近似解法を提案した．

ベンチマーク問題であるC問題に対して，数値実験を行い，従来の研究との比較を行った．その結果，従来の最良の解法であるガイドつき探索法を用いた解法をよりも平均で，0.14〜0.28% 優れた解を算出することができた．また，提案した解法により，すべての問題に対して最適解または最良解を求めることができた．

本研究は科学研究費基盤研究C（課題番号25350454）による成果の一部である．

参考文献

Balakrishnan, A., T. L. Magnanti, P. Mirchandani. 1997. Network design. M. Dell'Amico, F. Maffioli, S. Martello, eds., *Annotated Bibliographies in Combinatorial Optimization*. John Wiley & Sons, New York, 311–334.

Costa, A. M. 2005. A survey on benders decomposition applied to fixed-charge network design problems. *Computers and Operations Research* **32** 1429–1450.

Crainic, T. G. 2003. Long-haul freight transportation. R. W. Hall, ed., *Handbook of Transportation Science*. Kluwer Academic Publishers, 451–516.

Fischetti, M. Matteo, A. Lodi. 2003. Local branching. *Mathematical Programming* **98** 23–47.

Gendron, B., T. G. Crainic, A. Frangioni. 1997. Multicommodity capacitated network design. Tech. Rep. CIRRELT-98-14, Centre de recherche sur les transports,Université de Montréal.

Ghamlouche, I., T. G. Crainic, M. Gendreau. 2004. Path relinking, cycle-based neighborhoods and capacitated multicommodity network design. *Annals of Operations Research* **131** 109–134.

Hewitt, M., G. Nemhauser, M. W. P. Savelsbergh. 2013. Branch-and-price guided search for integer programs with an application to the multicommodity fixed charge network flow problem. *INFORMS Journal on Computing* **25** 302-316.

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

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

Wong, R. T. 1984. Introduction and recent advances in network design: Models and algorithms. M. Florian, ed., *Transportation Planning Models*. Elsevier Science, North Holland, Amsterdam, 187–225.

Wong, R. T. 1985. Location and network design. M. O'heEigeartaigh, J. Lenstra, A. RinnooyKan, eds., *Combinatorial Optimization Annotated Bibliographies*. John Wiley & Sons, New York, 129–147.

Yaghini, M., M.R.A Kazemzadeh. 2012. A simulated annealing algorithm for unsplittable capacitated network design. *International Journall of Industrial Engineering & Production Research* **23** 91–100.

Yaghini, M., M. Rahbar. 2012. Multicommodity network design problem in rail freight transportation planning. *Procedia - Social and Behavioral Sciences* **43** 728–739.

片山直登．2008. ネットワーク設計問題．朝倉書店 .

片山直登．2010. 多品種を考慮したロジスティクスネットワーク設計問題の数理的解法に関する研究．博士論文，流通経済大学 .