シングルパスとデザインバランスを考慮したサービスネットワーク設計問題の近傍探索法

A Neighborhood Search for the Single-Path Design-Balanced Service Network Design Problem

片山 直登
流通経済大学 流通情報学部

平成30 年10 月5 日

1 はじめに

ネットワーク設計問題は，物流，輸送通信など幅広い分野に応用がある計画問題である (Magnanti and Wong 1984)．ネットワーク設計問題の中で，多品種の需要を考慮し，アークが固定費用と処理容量をもつ問題は容量制約をもつ固定費付きの多品種フローネットワーク設計問題とよばれ，今日までに多くの研究が行われてきた (Wong 1984, 1985, Minoux 1989, Balakrishnan et al. 1997, Gendron et al. 1997, Crainic 2003, Costa 2005, Yaghini and Rahbar 2012)．この問題は，NP 困難な問題であることが知られている (Magnanti et al. 1986)．

一方，サービスネットワーク設計問題は，ネットワーク設計と共にサービス資源であるアセットの割当を求める問題である (Christiansen et al. 2007, Pedersen et al. 2009)．LTL などの物流ネットワークでは，施設において貨物の積み替え・積み合せが行われて貨物が輸送される．アセットは輸送車両や乗務員などの資源を表したものであり，サービスネットワーク設計問題はこれらのアセットの時間的・空間的なバランスを図るスケジューリング問題となる．一般的なサービスネットワーク設計問題に対しては，Crainic (2000, 2003) が解説を行っている．また，Christiansen et al. (2007) が海運輸送，Cordeau et al. (1998) が鉄道輸送，Crainic and Kim (2007) がインターモーダル輸送を取り扱っている．さらに，Kim et al. (1999)，Armacost et al. (2002) や Barnhart et al. (2002) がマルチモードにおけるエクスプレス輸送におけるアセット管理，Smilowitz et al. (2003) が宅配便におけるアセット管理を取り扱っている．

Pedersen et al. (2009) はサービスネットワーク設計問題にデザインバランス制約の概念を導入し，サービスネットワーク設計問題の基本モデルを提案している．デザインバランス制約は，各アセットが始点からいくつのノードを経由して始点に戻るという計画期間内におけるアセットのデザインバランスを表現している．

これは，アセットをネットワーク上の閉路に対応させ，ノードに出入りするアークの次数が一致するという巡回路制約となる．彼らは，容量制約をもつ固定費付きの多品種フローネットワーク設計問題にデザインバランス制約を追加したモデルを示しており，アド・デリートによる近傍探索法を用いた2段階タブー探索法を示している．

デザインバランス制約をもつモデルに対して，多くの近似解法が開発されている．Bai et al. (2010) はガイド付き局所探索法，Bai et al. (2012) はタブー補助ガイド付き局所探索法を開発している．Chouman and Crainic (2011) は数理計画ソルバーとタブー探索法を組み合せた近似解法を提案している．Vu et al. (2013) は3段階のメタヒューリスティクス解法を示し，このヒューリスティクスでは厳密解法と近傍探索法を併用している．片山直登 (2012) および Katayama (2015) は容量スケーリングと局所分枝法を組み合わせた解法を示している．Chouman and Crainic (2015) は下界値，変数固定法と学習メカニズムを用いた切除平面メタヒューリスティクス，Crainic et al. (2016) は列生成法とスロープスケーリングを合わせた近似解法を開発している．近年では，Bai et al. (2018) は k ノード隣接を用いたタブー探索法およびガイド付き局所探索法，片山直登 (2018a) は容量スケーリング法と MIP 近傍探索法を組み合わせた高速解法を提案している．後者は，精度の高い近似解を短時間で生成することに成功している．

サービスネットワーク上で処理される貨物などの需要はそれぞれ発地と着地が与えられている．輸送ネットワーク計画では，需要は発地・着地間の経路上の施設において単一方面に出荷されるのが一般的である．この場合，同一品種の需要は発地・着地間は一つの経路上で輸送されることになる．なお，ここでは同一の始点と終点をもつ需要の集まりを同一品種と定義する．このような同一の品種の始点・終点間の経路が一つで，分割されないフローを非分割フローまたはシングルパスフローとよび，このようなフローを考慮した問題を非分割フローを考慮したネットワーク設計問題またはシングルパスフローを考慮したネットワーク設計問題とよぶ．この問題では，アークのデザイン変数のみならずパスフロー変数やアークフロー変数も 0–1 変数となる．すべての変数が 0–1 変数である大規模な組合せ最適化問題となるため，小規模な問題であっても最適に解くことが困難な問題となる．このデザイン問題に対して，Hewitt et al. (2010) は IP 探索法，Yaghini and Kazemzadeh (2012) はシミュレーテッドアニーリング法，Hewitt et al. (2013) は分枝価格法とガイドつき探索法，片山直登 (2013) は容量スケーリング法と局所分枝法・パス再結合法を開発している．近年では，片山直登 (2018b) が容量スケーリング法と MIP 近傍探索法を組み合わせた高速解法を提案しており，精度の高い近似解を短時間で生成することに成功している．

シングルパスとデザインバランスの二つの条件を同時に考慮する問題をシングルパスデザインバランスサービスネットワーク設計問題 (single-path design-balanced service network design problem; $SPDB$) または非分割フローデザインバランスサービスネットワーク設計問題とよぶ．本研究で扱う $SPDB$ では，多品種の需要をもち，アセットの固定費用と処理容量をもつ問題である容量制約をもつ固

定費付きの多品種フロー問題を対象とし，Pedersen et al. (2009) が提示したデザインバランスを考慮したサービスネットワーク設計問題のモデルにシングルパスフローを考慮したモデル対象とする．この設計問題は，アーク上に配置される資源であるアセットの容量をもつネットワーク上で，ノードにおけるアセットの入出次数のバランスを保ちながら，全体の費用が最小となるような各品種の需要が移動するシングルパス，およびアセットの配置を求める問題である．考慮する費用はアセットの有無に応じて生じる固定費用であるアセット費用と品種のフロー量に関係して生じる変動費用であるフロー費用であり，これらの和を最小化する．$SPDB$ に対する研究は始まったばかりである．Li et al. (2017) が線形局所分枝法と MIP ソルバを組み合わせた LB ヒューリスティクスと，暫定解の線形緩和解の情報を用いた RINS ヒューリスティクスを示している．

本研究では，シングルパスとデザインバランスを考慮したサービスネットワーク設計問題 $SPDB$ に対して容量スケーリング法と MIP 近傍探索法を組合せた高精度な近似解法を提案する．

2 問題の定式化

ノード集合 N，向きをもつアーク集合 A，ネットワーク上で移動する品種集合 K，アーク (i,j) 上にアセットを配置したときに発生する非負のアセット費用 f_{ij}，アーク (i,j) 上を移動する品種 k の全需要に対するフロー費用 c_{ij}^k，アーク (i,j) 上のアセット容量 C_{ij}，品種 k の始点 O^k と終点 D^k 間を流れる品種 k の需要 d^k が与えられるものとする．

アーク (i,j) 上に配置されたアセットにより移動する品種 k のフローが存在するか否かを表す 0–1 離散変数であるアークフロー変数を x_{ij}^k とする．0–1 離散変数であるアークフロー変数を用いると，各品種の経路が単一であるシングルパスフローを表現することができる．また，アーク (i,j) 上にアセットを配置するとき 1，そうでないとき 0 である 0–1 離散変数であるデザイン変数を y_{ij} とする．デザインバランスは，ノードにおけるデザイン変数の入出の次数の関係式で表現できる．

このとき，$SPDB$ のアークフローによる定式化 $SPDBA$ は次のように表すことができる．

$SPDBA$

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

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

$$\sum_{i\in N_n^+} y_{in} - \sum_{j\in N_n^-} y_{nj} = 0 \quad \forall n \in N \tag{3}$$

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

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

$$x_{ij}^k \in \{0,1\} \quad \forall k \in K, (i,j) \in A \tag{6}$$

$$y_{ij} \in \{0,1\} \quad \forall (i,j) \in A \tag{7}$$

(1) 式は目的関数であり，フロー費用とアセット費用の総和を最小化する．なお，Φ は目的関数の最適値である．(2) 式はアーク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) 式はデザインバランス式であり，ノードに流入するアーク上に配置されるアセットの次数と流出するアセットの次数が一致することを表す．計画期間内で，始点を出たアセットがいくつかのノードを経由して始点に戻るようなアセットの巡回路制約を表している．(4) 式は，アセット容量制約式である．この式は，アーク (i,j) 上にアセットが配置されるときにアーク上を移動するフロー量の合計はアセット容量以下であり，アセットが配置されないときに 0 であることを表す．(5) 式は，アーク上の品種とその需要に関する強制制約式である．この式は，アーク (i,j) 上のアセットが配置されるときに各アーク上を移動する品種 k のフローが流れることが可能であり，アセットが配置されないときには流れることができないことを表す．(6) 式はアークフロー変数の 0–1 条件，(7) 式はデザイン変数の 0–1 条件である．

P^k を品種 k の取りうるパス集合，品種 k がパス p 上を移動するか否かを表す 0–1 離散変数であるパスフロー変数を z_p^k とする．各品種についてパスフロー変数値の合計が 1 であることで，シングルパスフロー条件を表すことができる．また，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 を表す定数を δ_{ij}^p とする．

アーク集合 A，パス集合 P，アーク容量 C が与えられたとき，$SPDB$ のパスフローによる定式化 $SPDBP(A,P,C)$ は次のように表すことができる．

$SPDBP(A,P,C)$

$$\text{最小化} \quad \sum_{(i,j)\in A}\sum_{k\in K} c_{ij}^k \sum_{p\in P^k} \delta_{ij}^p z_p^k + \sum_{(i,j)\in A} f_{ij} y_{ij} \tag{8}$$

$$\text{条件} \quad \sum_{p\in P^k} z_p^k = 1 \quad \forall k \in K \tag{9}$$

$$\sum_{i\in N_n^+} y_{in} - \sum_{j\in N_n^-} y_{nj} = 0 \quad \forall n \in N \tag{10}$$

$$\sum_{k\in K}\sum_{p\in P^k} d^k \delta_{ij}^p z_p^k \le C_{ij} y_{ij} \quad \forall (i,j) \in A \tag{11}$$

$$\sum_{p\in P^k} \delta_{ij}^p z_p^k \le y_{ij} \quad \forall k \in K, (i,j) \in A \tag{12}$$

$$z_p^k \in \{0,1\} \quad \forall p \in P^k, k \in K \tag{13}$$

$$y_{ij} \in \{0,1\} \quad \forall (i,j) \in A \tag{14}$$

(8) 式は目的関数であり，フロー費用とアセット費用の総和を最小化する．(9) 式は，品種 k のパスフロー変数の内の一つが選択される，すなわちシングルパスとなることを表すシングルパスフロー条件式である．(10) 式はデザインバランス式であり，ノードに流入するアーク上に配置されるアセットの次数と流出するアセットの次数が一致することを表す．(11) 式はアセット容量制約式であり，(12) 式は強制制約式である．(13) 式はパスフロー変数の 0–1 条件であり，(14) 式はデザイン変数の 0–1 条件である．

$SPDBP(A,P,C)$ のすべての 0–1 条件を線形緩和した問題を $SPDBPL(A,P,C)$ とする．アーク集合 A，パス集合 P，アーク容量 C に対する線形緩和問題の定式化 $SPDBPL(A,P,C)$ は次のように表される．

$SPDBPL(A,P,C)$

$$\text{最小化} \quad \sum_{(i,j)\in A}\sum_{k\in K} c_{ij}^k \sum_{p\in P^k} \delta_{ij}^p z_p^k + \sum_{(i,j)\in A} f_{ij} y_{ij} \tag{15}$$

$$\text{条件} \quad \sum_{p\in P^k} z_p^k = 1 \quad \forall k \in K \tag{16}$$

$$\sum_{i\in N_n^+} y_{in} - \sum_{j\in N_n^-} y_{nj} = 0 \quad \forall n \in N \tag{17}$$

$$\sum_{k\in K}\sum_{p\in P^k} d^k \delta_{ij}^p z_p^k \le C_{ij} y_{ij} \quad \forall (i,j) \in A \tag{18}$$

$$\sum_{p\in P^k} \delta_{ij}^p z_p^k \le y_{ij} \quad \forall k \in K, (i,j) \in A \tag{19}$$

$$0 \le z_p^k \le 1 \quad \forall p \in P^k, k \in K \tag{20}$$

$$0 \le y_{ij} \le 1 \quad \forall (i,j) \in A \tag{21}$$

3　近似解法

$SPDBA$や$SPDBP(A,P,C)$は，デザインバランス条件とシングルパス条件をもつ容量制約をもつネットワーク設計問題である．サービスネットワーク設計問題や非分割フローネットワーク設計問題の性質と類似していることから，これら2つの問題に対する近似解法である容量スケーリング法とMIP近傍探索法(片山直登 2018a,b)を組み合わせた解法を提案する．

3.1　容量スケーリング法

$SPDBA$や$SPDBP(A,P,C)$はアークとフローに対する多くの0–1変数を含む最適化問題であり，小規模な問題であっても最適解または適切な近似解を算出することが困難である．そこで，容量スケーリング法を用いて，対象となるアークから適切な近似解に含まれるであろうアークを選別する．

容量スケーリング法は，線形緩和問題を解き，デザイン変数解の値とスケーリングパラメータに従ってアーク(アセット)容量を変化させ，0また1のデザイン変数解を導出するものである．容量スケーリングでは，少ない繰り返し回数で多くのデザイン変数が0に収束することが知られている．そこで，アーク集合Aに対して，0に収束しないデザイン変数の数が終了判定数α以下になるまで容量スケーリング法を適用し，0に収束しないデザイン変数のみを選定する．この処理により，わずかな計算量で多くの適切な近似解に含まれないであろうデザイン変数を除外することができ，効率的に問題規模を縮小することが可能となる．

本研究では，パスフローによる定式化である$SPDBP(A,P,C)$の線形緩和問題$SPDBPL(A,P,C)$に容量スケーリング法を適用する．適宜，パスフロー変数を列生成法により生成し，必要な強制制約式を行生成法により生成する．容量スケーリング法のアルゴリズムをAlgorithm1に示す．なお，多くのデザイン変数は0または1となるが，フロー変数の離散性は考慮することはできない．容量スケーリング法の詳細は，Katayama et al. (2009)を参照のこと．

容量スケーリング法により得られたデザイン変数を$\hat{y}$とする．$\hat{y}$はすべて0または1に収束しているとは限らず，デザインバランス条件を満足することは保証されない．さらに，線形緩和問題の解であるので，パスフロー変数もすべて離散値をとることはあり得ない．そこで，容量スケーリング法により得られたデザイン解をもとにシングルパス条件とデザインバランス条件を同時に満足する$SPDB$の実行可能解を導出する．

容量スケーリング法により得られたデザイン変数解の中で，0に収束していない$\hat{y}$に対応するアークからなる集合を$\hat{A}$とする．また，初期パス集合と容量スケーリング法を解く際に列生成法により生成されたパス集合の和集合を$\bar{P}$とする．

3.2 デザインバランス条件

はじめに，シングルパス条件は考慮せずにデザインバランス条件のみを満足するデザイン解を導出する．$SPDBP(A,P,C)$ において，フロー変数のみを線形緩和した問題を $SPDBPLZ(A,P,C)$ とおく．

容量スケーリング法から得られたデザイン変数解 $\hat{y}$ が，すべて0または1であれば，$SPDBPLZ(A,P,C)$ のデザインバランス条件を満足する．しかし，いくつかのデザイン変数解が小数値である場合，0–1条件のもとでデザインバランス条件を満足はしないため，デザイン変数解 $\hat{y}$ は $SPDBPLZ(A,P,C)$ の実行可能解とは限らない．そこで，$\hat{y}_{ij} > 0$ であるアセットを $y_{ij} = 1$ と固定した，アーク集合 $\hat{A}$ に関するデザインバランス問題 $ABL(\hat{A})$ を解くことによって，デザインバランス式を満足するデザイン解を求める．

$ABL(\hat{A})$

$$\text{最小化} \quad \sum_{(i,j)\in A\backslash\hat{A}} f_{ij}y_{ij} \tag{22}$$

$$\text{条件} \quad \sum_{i\in N_n^+} y_{in} - \sum_{j\in N_n^-} y_{nj} = 0 \quad \forall n \in N \tag{23}$$

$$y_{ij} = 1 \quad \forall (i,j) \in \hat{A} \tag{24}$$

$$y_{ij} \in \{0,1\} \quad \forall (i,j) \in A\backslash\hat{A} \tag{25}$$

(22) 式は，デザインバランス式を満足するために新たに付加されるアークに対応するアセット費用の合計である．(24) 式は，$\hat{A}$ に含まれるアークのデザイン変数を1に固定する式である．

$ABL(\hat{A})$ を解くことによって，$SPDBPLZ(A,P,C)$ のデザインバランス式を満足し，追加的なアセット費用が最小となるデザイン解を求めることができる．$ABL(\hat{A})$ の最適解が求めることができれば，この解を $\bar{y}$ とおく．容量スケーリング法により得られフロー解はアセット容量を満足することから，$\bar{y}$ は $SPDBPLZ(A,P,C)$ の実行可能解となる．$\bar{y}_{ij} = 1$ であるデザイン変数に対応するアーク集合を $\bar{A}$ とおく．続いて，計算時間 T の制限のもとで $SPDBPLZ(\bar{A},\bar{P},C)$ を MIP ソルバーを用いて解いて得られた解を $\tilde{y}$ とおき，$\tilde{y}_{ij} = 1$ であるアーク集合を $\tilde{A}$ とおく．

なお，$ABL(\hat{A})$ が実行不可能である場合，アークを付加するだけではデザインバランス条件を満足できないことになる．このときは，計算時間 T の制限のもとで $\bar{A}$ を A に置き換えた問題 $SPDBPLZ(A,\bar{P},C)$ を解いて得られた解を $\tilde{y}$ とおき，$\tilde{y}_{ij} = 1$ であるアーク集合を $\tilde{A}$ とおく．

デザインバランス条件を満足する解の探索のアルゴリズムを Algorithm2 に示す．

3.3 シングルパス条件

続いて，3.2 節で選定されたアーク集合をもとに，デザインバランス条件とシングルパス条件を同時に満足する解を探索する．

前節で求めたデザイン変数解 $\tilde{y}$ を用いて，$SPDBPLZ(A,P,C)$ のフロー変数解を求めた場合は，多くの場合，得られたフロー変数解は小数値を取るため，$SPDBP(A,P,C)$ の実行可能解とはならない．そこで，$\tilde{y}_{ij}=1$ であるデザイン変数解に対応するアーク集合 $\tilde{A}$ から始め，フロー変数の 0–1 条件のもとで，アーク集合 $\tilde{A}$ に $\tilde{A}$ に含まれていないアークを順次付加していき，$SPDBP(A,P,C)$ の実行可能なアーク集合を選定する．これによって最適解に含まれる可能性の高いアークを選別し，すべての制約条件を満足する解を探索することができる．

ここでは，アークフローを用いた定式化 $SPDBA$ をもとに展開する．はじめに，$SPDBA$ においてデザイン変数とフロー変数に対応する (6) 式および (7) 式を線形緩和した問題 $SPDBP(A,P,C)$ に $\tilde{A}$ に含まれるアークに対するデザイン変数を 1 に固定する次の条件を付加した問題を $SPDBALF(\tilde{A})$ とし，この問題を解く．

$$y_{ij}=1 \quad \forall(i,j)\in\tilde{A} \tag{26}$$

$SPDBALF(\tilde{A})$ は線形計画問題であるので，容易に解くことができる．得られた $SPDBALF(\tilde{A})$ の最適なデザイン解を $\hat{y}$ とおく．

アーク集合 $\tilde{A}$ に含まれるアーク (i,j) については 1 に固定されているため，$\hat{y}_{ij}=1$ となる．フロー変数の 0–1 条件のもとで実行可能解を含むセット集合を得るために，$\tilde{A}$ に含まれていない $\hat{y}_{ij}>0$ であるアークを $\tilde{A}$ に付加していくことを考える．まず，$\hat{y}_{ij}$ の降順に $\tilde{A}$ に含まれるアークをソートする．$\hat{y}_{ij}$ が 1 に近いアークは $SPDBPA$ の実行可能解に含まれる可能性が高く，0 に近いアークは実行可能解に含まれる可能性が低いと考えることができる．

続いて，$SPDBA$ のデザイン変数の 0–1 条件である (7) 式のみを線形緩和し，アーク集合 $\tilde{A}$ に対する (26) 式と $\tilde{A}$ に含まれないすべてのアークに対応するデザイン変数を 0 に固定した (27) 式を付加した問題を $SPDBALY(\tilde{A})$ とする．

$$y_{ij}=0 \quad \forall(i,j)\in A\backslash\tilde{A} \tag{27}$$

$SPDBALY(\tilde{A})$ を MIP ソルバーを用いて解く．しかし，この問題には多数の 0-1 変数であるアークフロー変数が含まれるため，最適に解くことは困難である．そこで，最適に解くのではなく，実行可能解が存在するか否かのみを判定する．1 つの実行可能解を探索するのみであるので，相対的に短い時間で判定することができる．

適当な整数を H とし，$\tilde{A}$ に含まれていないアーク集合で，デザイン変数 $\hat{y}_{ij}$ の値の降順に上位の H 本のアークを選択し，対応するアーク集合 $\tilde{A}$ に付加して，$\tilde{A}$ に対する $SPDBALY(\tilde{A})$ を解く．$\tilde{A}$ に対して，実行可能解が存在すると判定できた場合は，手順を終了する．そうでない場合は，$\hat{y}_{ij}>0$ であるすべてのアークを付加するまで，順次，次の H 本のアークを選択して，同様な手順を繰り返す．以上の操作によって，$SPDBA$ の実行可能なデザイン変数に対応するアーク集合を特定することができる．

3.4 限定分枝限定法と実行可能解の算出

$SPDBP(A,P,C)$ の実行可能解を算出することを考える．$\tilde{A}$ は $SPDBP(A,P,C)$ の実行可能なアーク集合を含んでいる．そこで，列生成法にて容量スケーリング法を解いた際に生成されたパス集合を $\bar{P}$ とし，設定した計算時間 T のもとで，実行可能なデザイン解を含むアーク集合 $\tilde{A}$ と $\bar{P}$ に限定された $SPDBP(\tilde{A},\bar{P},C)$ を MIP ソルバーを用いて解く．この問題はアーク集合とパス集合が限定されているため，比較的容易に解を求めることができる．この方法を限定分枝限定法とよぶ．

設定した計算時間 T 以内で実行可能解解を算出できた場合は，この解を $\tilde{y}$ とする．T 以内で実行可能解が算出できない場合は，$\tilde{A}$ に含まれるすべての集合を $y_{ij}=1$ とした $SPDBA$ を直接解き，実行可能解が得られた場合は，得られた解を $\tilde{y}$ とする．それでも実行可能解が求まらない場合は，前節の $\hat{y}$ を切り上げたものを $\tilde{y}$ とする．

シングルパス条件を満足する解の探索と限定分枝限定法のアルゴリズムを Algorithm3 に示す．

3.5 MIP 近傍探索法

前節で求めた近似解 $\tilde{y}$ を初期の暫定解とし，$SPDBA$ において暫定解の近傍を MIP ソルバーを用いて探索していく．

適当な初期解から探索範囲を限定する条件を付加した問題を MIP ソルバーを用いて解く解法として局所分枝法 (Fischetti and Lodi 2003) が知られており，多くのネットワーク設計問題に適用されている．本研究では，別のタイプの探索範囲を限定する条件を付加する MIP 近傍探索法を適用する．始めに，$SPDBA$ に次の制約式を追加する．

$$\sum_{(i,j)\in A|\tilde{y}_{ij}=1} y_{ij} \leq L-1 \tag{28}$$

ここで，L は暫定解 $\tilde{y}$ において $\tilde{y}_{ij}=1$ であるデザイン変数の数である．(28) 式は，暫定解で $\tilde{y}_{ij}=1$ に対応するアークから少なくとも 1 つのアークはネットワークから取り除くことを表しており，実行可能領域から暫定解を排除することができる．(28) 式が等式である場合はデリート型の近似解法になる．しかし，1 つのアークのみを削除するとデザインバランス式が成立しないため，不等号としている．

次に，$SPDBA$ に次の M 近傍を与える制約式を追加する．

$$\sum_{(i,j)\in A|\tilde{y}_{ij}=1} y_{ij} \geq L-M \tag{29}$$

(29) 式は，暫定解で選択されたアークから高々 M 個のアークをネットワークから取り除くことを表す．なお，M は正の整数で近傍の範囲である．M が大きければ，条件を付加した $SPDBA$ の実行可能領域は広くなるため，良い解を算出できる可能性があるが計算時間内で算出できない可能性が高くなる．一方，M が小さけれ

ば実行可能領域は狭くなるため，相対的に短時間で実行可能解を探索できる可能性が高まることになる。

また，現在までの最良の上界値をUBとおき，次の式も追加する．

$$\Phi < UB \tag{30}$$

(28) 式と (30) 式により探索済みの解および暫定解を排除し，解の循環を防ぐことができる．なお，現在までの最良値よりも良い上界値が存在しなければ，問題は実行不可能となる．また，(28) 式と (29) 式から分かるように，付加するアークの数には制限がない．

計算に制限時間Tを設け，MIP ソルバーを用いて$SPDBA$に 3 本の制約式を付加した問題を解く。実行可能解が得られた場合は，改善された解が探索されたことになり，得られた解を新たな暫定解$\tilde{y}$とする．続いて，追加した 3 本の制約式を削除して，更新された暫定解に対応した 3 本の制約式を追加し，近傍探索を繰り返す．実行不可能であることが判明した場合は，M近傍において暫定解よりも良い解が無いと判断できたことになり，探索を終了する．

一方，計算時間内に暫定解より良い解を算出することができず暫定解が更新されない場合は，計算時間内で実行不可能とは判断できないが実行可能解も算出できていない状態である．そこで，$M := \lfloor M/\beta \rfloor$として$M$を減少させ，探索範囲を縮小する．ここで，$\beta(> 1)$は$M$の変更基準である．これにより，計算時間内で実行可能解を算出できるまたは実行不可能と判断できる可能性が高まることになる．$M > 0$である間，同様の探索を繰り返す．

このような探索法を MIP 近傍探索法とよぶ．MIP 近傍探索法のアルゴリズムを Algorithm4 に示す．

4 数値実験

容量制約をもつネットワーク設計問題で用いられるベンチマーク問題である C 問題 (Pedersen et al. 2009) の内，$SPDB$で用いられている 31 問に対して，数値実験を行った．

数値実験で使用した設定した機器等は以下の通りである．

- 使用 OS および言語：UBUNTU 17.04，C++
- 最適化ソルバー：Gurobi 8.0
- CPU AMD Ryzen7 1800X 3.6GHz 8Cores，RAM 16GByte
- 使用コア数：容量スケーリング 1 コア，MIP 近傍探索 8 コア

また，数値実験で使用した設定したパラメータは以下の通りである．

- 容量スケーリングパラメータλ：0〜0.25
- 容量スケーリングの終了判定アーク数α：200
- 容量スケーリングの最小繰返し回数ITE_{min}：100

- 容量スケーリングの最大繰返し回数 ITE_{max}：250
- アーク追加数 H：10
- 近傍 M の変更基準 β：2
- MIP ソルバー計算時間の制限時間 T と近傍 M の組合せ:(50 秒,20)，(100 秒,30)

近似解の誤差を算出するために，MIP ソルバーである Gurobi により，定式化 $SPDBA$ を 30 時間解き，下界値を算出した．同時に上界値 (GRB30) も算出した．

C 問題に対しては Li et al. (2017) が LB ヒューリスティクス (LBH) と RINS ヒューリスティクス (RINSH) と 2 つを組み合わせた解法 (RINSH+LBH) による結果を公表している．これに加え，本研究である容量スケーリング・MIP 近傍探索法 (CSMIP50, CSMIP100)，およびパラメータチューニングをした容量スケーリング・MIP 近傍探索法 (CSMIPB50，CSMIPB100) の結果を示す．λ は同一ノード数のインスタンスごと表 1 のように設定した．CSMIPB では，問題ごとにスケーリングパラメータ λ を変化させた中の最良値を採用した．なお，誤差 (Gaps) は「(各解法の上界値－下界値)/下界値」とし，平均誤差はこれらの平均値である．

表 2 に C 問題に対する上界値の平均誤差を示す．従来の研究では，LBH は平均誤差 3.56%，RINSH は 2.65%，RINSH+LBH は 2.90%であった．なお，Gurobi による平均誤差は 0.98%であった．一方，本研究である容量スケーリング・MIP 近傍探索法では，CSMIP50 が平均誤差 1.39%であり，CSMIP100 が平均誤差 1.29%であった．また，パラメータをチューニングした容量スケーリング・MIP 近傍探索法では，CSMIPB50 が 1.12%であり，CSMIPB100 が 1.03%であった．容量スケーリング・MIP 近傍探索法は，RINSH よりも CSMIP50 では 1.26%，CSMIP100 では 1.36%優れており，概ね半分の誤差であった．また，チューニングした容量スケーリング・MIP 近傍探索法は，RINSH よりも CSMIPB50 では 1.53%優れ，CSMIPB100 では 1.62%優れていた．

表 3 に，個別問題の上界値を示す．なお，LB は下界値 (L) または最適値 (O) であり，太字は最適値，斜体文字は最良値を表している．LRH，RINSH および RINSH+LRH では 10 問の最適値を算出している．一方，CSMIP50 および CSMIP100 では 10 問の最適値を算出し，パラメータをチューニングした CSMIPB50 では 13 問，CSMIPB100 では 14 問の最適値を算出している．なお，GRB30 では 16 問の最適値を算出している．また，LRH，RINSH，RINSH+LRH および CSMIP50 では，最適値を除く最良値を算出できていない．一方，CSMIP100 と CSMIPB50 では 2 問，CSMIPB100 と GRB30 では 7 問の最適値を除く最良値を算出している．

表 4 および表 5 に，C 問題に対する平均計算時間と個々の問題に対する計算時間を示す．従来の研究の計算時間は，論文に掲載しているものであり，使用しているコンピュータが異なっているため，計算時間を直接比較することはできない．なお，LBH，RINSH および RINSH+LBH では，CPU が PentiumD(2GHz) であるコンピュータを使用している．

平均誤差の最も小さい Gurobi では 30 時間の制限を付けているため，最適解を求められない場合は 108000 秒となり，平均計算時間は 57652 秒で 16 時間を越えている．また，LBH の平均計算時間は 404 秒，RINSH の平均計算時間は 426 秒，

RINSH+LBH の平均計算時間は 430 秒である．なお，これらでは 600 秒の計算時間の上限を設定している．一方，本研究である容量スケーリング・MIP 近傍探索法の平均計算時間は，CSMIP50 では 349 秒，CSMIP100 では 658 秒であった．また，パラメータをチューニングした容量スケーリング・MIP 近傍探索法の平均計算時間は，CSMIPB50 では 344 秒，CSMIPB100 では 634 秒であった．なお，後者は，実際にはパラメータ選定のための事前の計算時間が必要である．使用しているコンピュータが異っているとしても，従来の研究と同程度の計算時間で，高精度が解を算出できていることが分かる．

5 おわりに

本研究では，本研究では，シングルパスとデザインバランスを考慮したサービスネットワーク設計問題に対して容量スケーリング法と MIP 近傍探索法を組合せた高速な近似解法を提案した．これは，容量スケーリング法の解をもとに，シングルパス条件とデザインバランス条件を満足するアーク集合を選定し，限定分枝限定法により実行可能解を求め，この解に対して MIP 近傍探索法を適用する解法である．また，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．従来の解法よりも精度の高い近似解を算出することができた．

本研究は科学研究費基盤研究 C(課題番号 17K01268) による成果の一部である．

参考文献

Armacost, A. P., C. Barnhart, K. A. Ware. 2002. Composite variable formulations for express shipment service network design. *Transportation Science* **36** 1–20.

Bai, R., G. Kendal, J. Li. 2010. An efficient guided local search approach for service network design problem with asset balancing. *ICLSIM* **1** 110–115.

Bai, R., G. Kendal, R. Qu, J. Atkin. 2012. Tabu assisted guided local search approaches for freight service network design. *Information Science* **189**(15) 266–281.

Bai, R., J.R. Woodward, N. Subramanian, J. Cartlidge. 2018. Optimisation of transportation service network using k-node large neighbourhood search. *Computers & Operations Research* **89** 193–205.

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.

Barnhart, C., N. Krishnan, D. Kim. 2002. Network design for express shipment delivery. *Computational Optimization and Applications* **21** 239–262.

Chouman, M., T. G. Crainic. 2011. MIP-based tabu search for service network design with design-balanced requirements. Tech. Rep. CIRRELT-2011-68, Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation, Université de Montréal.

Chouman, M., T. G. Crainic. 2015. Cutting-plane matheuristic for service network design with design-balanced requirements. *Transportation Science* **49**(1) 99–113.

Christiansen, M., K. Fagerholt, B. Nygreen, D. Ronen. 2007. Maritime transportation. C. Barnhart, G. Laporte, eds., *Handbooks in Operations Research and Management Science*, vol. 14. North Holland, 189284.

Cordeau, J-F, P. Toth, D. Vigo. 1998. A survey of optimization models for train routing and scheduling. *Transportation Science* **32** 380–404.

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. 2000. Service network design in freight transportation. *European Journal of Operational Research* **122**(2) 272–288.

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

Crainic, T. G., K. H. Kim. 2007. Intermodal transportation. C. Barnhart, G. Laporte, eds., *Transportation: Handbooks in Operations Research and Management Science*. North-Holland, 467–537.

Crainic, T.G., M. Hewitt, M. Toulouse, D.M. Vu. 2016. Service network design with resource constraints. *Transportation Science* **50**(4) 1380–1393.

Fischetti, M. M., A. Lodi. 2003. Local branching. *Mathematical Programming* **98**(1-3) 23–47.

Gendron, B., T. G. Crainic, A. Frangioni. 1997. Multicommodity capacitated network design. Tech. Rep. CIRRELT-98-14, Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation, Université de Montréal.

Hewitt, M., G. L. Nemhauser, M. W. P. Savelsbergh. 2010. Combining exact and heuristics approaches for the capacitated fixed charge network flow problem. *Journal on Computing* **22** 314–325.

Hewitt, M., G. L. 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**(2) 302–316.

Katayama, N. 2015. A combined matheuristics for service network design problem. *The International Federation of Logistics and SCM Systems* **8** 11–20.

Katayama, N., M. Z. Chen, M. Kubo. 2009. A capacity scaling procedure for the multi-commodity capacitated network design problem. *Journal of Computational and Applied Mathematics* **232**(2) 90–101.

Kim, D., C. Barnhart, G. Ware, G. Reinhardt. 1999. Multimodel express package delivery : a service network design application. *Transportation Science* **33** 291–407.

Li, X., K. Wei, Y.P. Aneja, P. Tian, Y. Cui. 2017. Matheuristics for the single-path design-balanced service network design problem. *Computers & Operations Research* **77** 141–153.

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

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.

Pedersen, M. B., T. G. Crainic, O. B. G. Madsen. 2009. Models and tabu search metaheuristics for service network design with asset-balance requirements. *Transportation Science* **43** 158–177.

Smilowitz, K. R., A. Atamtúrk, C. F. Daganzo. 2003. Deferred item and vehicle routing within integrated networks. *Transportation Research Part E* **39**(4) 305–323.

Vu, D. M., T. G. Crainic, M. Toulous. 2013. A three-stage matheuristic for the capacitated multi-commodity fixed-cost network design with design-balance constraints. *Journall of Heuristics* **19** 757–795.

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**(2) 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. 2012. アセットバランスを考慮したサービスネットワーク設計問題. 流通経済大学流通情報学部紀要 **17**(1) 29–50.

片山直登. 2013. 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. 流通経済大学流通情報学部紀要 **20**(1) 1–19.

片山直登. 2018a. アセットバランスを考慮したサービスネットワーク設計問題の MIP 近傍探索法. 流通経済大学流通情報学部紀要 **22**(2) 1–16.

片山直登. 2018b. 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の MIP 近傍探索法. 流通経済大学流通情報学部紀要 **22**(2) 17–31.

Algorithm 1: Capacity Scaling

INPUT: A(arc set), C(capacity), $\bar{P}$(initial feasible path set), α(termination parameter), λ(smoothing parameter), ITE_{min}(minimum iteration of capacity scaling), ITE_{max}(maximum iteration of capacity scaling);
OUTPUT: $\hat{y}$(current solution), $\hat{A}$(arc set, $\hat{y} > 0$), $\bar{P}$(current feasible path set);

$l \leftarrow 0$;
$C^1 \leftarrow C$;
repeat
  $l \leftarrow l + 1$;
  Solve $SPDBPL(A, \bar{P}, C^l)$;
  Get the solution $\hat{y}$ of $SPDBPL(A, \bar{P}, C^l)$;
  Add paths to $\bar{P}$ by Column Genaration;
  $n \leftarrow 0$;
  for $(i, j) \in A$ **do**
    $C_{ij}^{l+1} \leftarrow \lambda C_{ij}^l \hat{y}_{ij} + (1 - \lambda) C_{ij}^l$;
    if $\hat{y}_{ij} > 0$ **then**
      $n \leftarrow n + 1$
    end
  end
until $l \geq ITE_{min}$ *and* $n \leq \alpha$, *or* $l \geq ITE_{max}$;
$\hat{A} \leftarrow \varnothing$;
for $(i, j) \in A$ **do**
  if $\hat{y}_{ij} > 0$ **then**
    $\hat{A} \leftarrow \hat{A} \cup \{(i, j)\}$;
  end
end

Algorithm 2: Design Balance

INPUT: C(capacity), $\hat{A}$(arc set, $\hat{y} > 0$), $\bar{P}$(current feasible path set), T(computation time);
OUTPUT: $\tilde{A}$(arc set, $\tilde{y} = 1$);

Solve $ABL(\hat{A})$;
if *the feasible solution* $\bar{y}$ *of* $ABL(\hat{A})$ *is found* **then**
    $\bar{A} \leftarrow \varnothing$;
    for $(i, j) \in A$ **do**
        if $\bar{y}_{ij} = 1$ **then**
            $\bar{A} \leftarrow \bar{A} \cup \{(i, j)\}$
        end
    end
else
    $\bar{A} \leftarrow A$;
end
Solve $SPDBPLZ(\bar{A}, \bar{P}, C)$ within time T;
if *the feasible solution* y' *of* $SPDBPLZ(\bar{A}, \bar{P}, C)$ *is found* **then**
    $\tilde{y} \leftarrow y'$;
else
    Solve $SPDBPLZ(A, \bar{P}, C)$ within time T;
    Get the solution y' of $SPDBPLZ(A, \bar{P}, C)$;
    $\tilde{y} \leftarrow y'$;
end
$\tilde{A} \leftarrow \varnothing$;
for $(i, j) \in A$ **do**
    if $\tilde{y}_{ij} = 1$ **then**
        $\tilde{A} \leftarrow \tilde{A} \cup \{(i, j)\}$
    end
end

Algorithm 3: Single Path Flow and Resticted Branch-and-Bound

INPUT: $\tilde{A}$(arc set, $\hat{y}=1$), H(number of added arcs), T(computation time);
OUTPUT: $\tilde{y}$(feasible solution), UB(upper bound);

Solve $SPDBALF(\tilde{A})$ with equations (26) and get the solution $\hat{y}$;
Sort the the solution $\hat{y}_{ij}$,$\forall(i,j) \in A\backslash\tilde{A}$ in descending order;
Set $1,\cdots,|A\backslash\tilde{A}|$ to the arc indexs in ascending order of $\hat{y}$;
for $l=1$ *to* $|A\backslash\tilde{A}|$ **do**
    if $\hat{y}_l=0$ **then**
        break;
    end
    $\tilde{A} \leftarrow \tilde{A} \cup arc_l$;
    if l Mod $H=0$ **then**
        Solve $SPDBALY(\tilde{A})$ within time T;
        if *a feasible solution of* $SPDBALY(\tilde{A})$ *is found* **then**
            break;
        end
    end
end
Solve $SPDBP(\tilde{A},\bar{P},C)$ within time T;
if *the feasible solution* y' *of* $SPDBP(\tilde{A},\bar{P},C)$ *is found* **then**
    Get the upper bound UB of $SPDBP(\tilde{A},\bar{P},C)$;
    $\tilde{y} \leftarrow y'$; Get the upper bound UB of $SPDBP(\tilde{A},\bar{P},C)$;
else
    Add equations (26) to $SPDBA$ associated with $\tilde{A}$;
    Solve $SPDBA$ within time T;
    if *the solution* y' *of* $SPDBA$ *is found* **then**
        $\tilde{y} \leftarrow y'$;
        Get the upper bound UB of $SPDBA$;
    else
        $\tilde{y} \leftarrow \lfloor \hat{y} \rfloor$;
        $UB \leftarrow \infty$;
    end
end

Algorithm 4: MIP Neighborhood Search

INPUT: $\tilde{y}$(current solution), UB(upper bound), β(decrease parameter), M(neighborhood parameter), T(computation time);
OUTPUT: $\tilde{y}$(feasible solution), UB(upper bound);

repeat
  Add equations (28), (29) and (30) to $SPDBA$ associated with the current solution $\tilde{y}$;
  Solve $SPDBA$ within time T;
  if *$SPDBA$ has no feasible solution* **then**
    break;
  else
    if *the solution y' of $SPDBA$ is found* **then**
      Get the upper bound UB_{neigh} of $SPDBA$;
      $\tilde{y} \leftarrow y'$;
      $UB \leftarrow UB_{neigh}$;
    else
      $M \leftarrow \lfloor M/\beta \rfloor$;
    end
  end
  Delete equations (28), (29) and (30) from $SPDBA$;
until $M = 0$;

表 1: Smoothing Parameter λ

Arcs	230	520	700
CSMIP50	0.004	0.011	0.015
CSMIP100	0.006	0.0025	0.015

表 2: Average Gaps for C-Category Problems(%)

GRB30	LBH	RINSH	RINSH+LBH	CSMIP50	CSMIP100	CSMIPB50	CSMIPB100
0.98	3.56	2.65	2.90	1.39	1.29	1.12	1.03

表 3: Results for C-Category Problems

Problem	LB	GRB30	LBH	RINSH	RINSH +LBH	CSMIP 50	CSMIP 100	CSMIPB 50	CSMIPB 100
20/230/40/VL	434986^{O}	**434986**	**434986**	**434986**	**434986**	**434986**	**434986**	**434986**	**434986**
20/230/40/VT	405089^{O}	**405089**	**405089**	**405089**	**405089**	**405089**	**405089**	**405089**	**405089**
20/230/40/FT	691642^{O}	**691642**	**691642**	**691642**	**691642**	**691642**	**691642**	**691642**	**691642**
20/230/200/VL	97816 ^{O}	**97816**	101592	100073	101982	98106	98323	97931	97922
20/230/200/FL	139711^{L}	140734	147719	144933	144517	141896	141072	*140379*	*140379*
20/230/200/VT	100948^{O}	**100948**	101963	102411	102518	101077	101056	**100948**	**100948**
20/230/200/FT	138383^{L}	*140712*	153061	143948	147030	141220	140966	140808	140844
20/230/40/VL	439205^{O}	**439205**	**439205**	**439205**	**439205**	**439205**	**439205**	**439205**	**439205**
20/230/40/FL	616525^{O}	**616525**	**616525**	**616525**	**616525**	**616525**	**616525**	**616525**	**616525**
20/230/40/VT	505657^{O}	**505657**	**505657**	**505657**	**505657**	**505657**	**505657**	**505657**	**505657**
20/230/40/FT	656324^{O}	**656324**	**656324**	**656324**	**656324**	**656324**	**656324**	**656324**	**656324**
20/230/200/VL	77309 ^{L}	78776	81524	80067	79510	78987	79104	78654	*78415*
20/230/200/FL	118405^{L}	121473	129103	127335	126433	122060	121779	121587	*121468*
20/230/200/VT	77340 ^{O}	**77340**	79593	77928	78215	77676	77348	77352	**77340**
20/230/200/FT	110688^{L}	*113439*	119619	116549	116696	115469	114924	113777	113813
30/520/100/VL	55363 ^{O}	**55363**	55384	55384	55415	**55363**	55395	**55363**	**55363**
30/520/100/FL	98392 ^{L}	100894	105557	105123	103346	101376	*100425*	100633	*100425*
30/520/100/VT	54961 ^{O}	**54961**	**54961**	**54961**	**54961**	**54961**	**54961**	**54961**	**54961**
30/520/100/FT	102898^{L}	103481	106591	105327	106813	104104	104078	*103471*	103760
30/520/400/VL	114379^{L}	116252	120279	119137	119284	116378	*116155*	116236	*116155*
30/520/400/FL	151311^{L}	*154246*	162427	160749	160837	155942	156019	154768	154343
30/520/400/VT	117093^{L}	*118295*	120483	120196	120802	119191	119276	118725	118703
30/520/400/FT	153811^{L}	158405	164153	162842	163979	160132	159741	160023	*158272*
30/700/100/VL	49039 ^{O}	**49039**	**49039**	**49039**	**49039**	**49039**	**49039**	**49039**	**49039**
30/700/100/FL	61910 ^{O}	**61910**	64141	62738	63372	62524	62524	62373	62195
30/700/100/VT	f48719^{L}	**48719**	**48719**	**48719**	**48719**	48721	**48719**	**48719**	**48719**
30/700/100/FT	58239 ^{O}	**58239**	58616	58391	58460	58405	58454	**58239**	**58239**
30/700/400/VL	98704 ^{L}	*100173*	102663	101780	102460	100518	100384	100384	100211
30/700/400/FL	134332^{L}	*139496*	146169	147604	149579	141494	140846	141197	140846
30/700/400/VT	96734 ^{L}	98785	101927	100818	100933	98966	99307	98703	*98634*
30/700/400/FT	130655^{L}	*134473*	141387	139205	139267	135101	135071	134978	134914

表 4: Average Computation Times for C-Category Problems(%)

GRB30	LBH	RINSH	RINSH+LBH	CSMIP50	CSMIP100	CSMIPB50	CSMIPB100
57652	404	426	430	349	658	344	634

表 5: Computation Times for C-Category Problems(Seconds)

Problem	GRB30	LBH	RINSH	RINSH +LBH	CSMIP 50	CSMIP 100	CSMIPB 50	CSMIPB 100
20/230/40/VL	0	1	1	2	1	1	1	1
20/230/40/VT	0	1	1	2	2	2	2	2
20/230/40/FT	0	1	2	3	2	2	2	2
20/230/200/VL	55070	492	600	600	498	835	498	835
20/230/200/FL	108000	600	600	600	451	863	451	853
20/230/200/VT	20794	600	600	600	404	827	397	827
20/230/200/FT	108000	600	600	600	593	1125	593	1125
20/230/40/VL	0	1	28	2	1	1	1	1
20/230/40/FL	2	1	4	10	4	4	4	4
20/230/40/VT	1	1	2	2	2	2	2	2
20/230/40/FT	0	1	1	3	2	2	2	2
20/230/200/VL	108000	341	600	600	596	877	596	874
20/230/200/FL	108000	600	600	600	448	1124	448	1124
20/230/200/VT	47614	492	600	600	445	921	445	919
20/230/200/FT	108000	341	600	600	548	1023	548	931
30/520/100/VL	565	600	600	600	277	625	206	459
30/520/100/FL	108000	492	600	600	589	895	589	895
30/520/100/VT	28	600	70	116	60	53	40	38
30/520/100/FT	108000	492	600	600	627	1075	627	960
30/520/400/VL	108024	425	600	600	528	1070	528	1070
30/520/400/FL	108000	495	600	600	500	987	500	987
30/520/400/VT	108000	600	600	600	613	1033	613	1027
30/520/400/FT	108000	577	600	600	532	1007	532	1007
30/700/100/VL	97	600	227	238	123	150	111	147
30/700/100/FL	37093	600	600	600	343	638	343	638
30/700/100/VT	160	341	268	367	147	341	126	331
30/700/100/FT	5773	342	600	600	420	817	420	568
30/700/400/VL	108000	497	600	600	509	1096	509	1004
30/700/400/FL	108000	600	600	600	537	1018	537	1018
30/700/400/VT	108000	600	600	600	505	1001	505	1000
30/700/400/FT	108000	600	600	600	503	991	503	991