電気通信大学情報理工学部
情報・通信工学科情報数理工学コース卒業論文

Partition 制約付き最小化ナップサック問題に対する 3-近似アルゴリズム

平成 30 年 1 月 31 日

情報数理工学コース

学籍番号 1411070

木村優貴

指導教員 村松正和

1　はじめに

離散最適化にはNP困難な問題が多数存在する．NP困難な問題に対して用いられるアプローチは3つある．

1つは多項式時間で終了する保証はないが，最適解を導出するアプローチである．このアプローチについては切除平面法 [1]，分枝限定法 [2] などが研究されている．しかし，このアプローチの問題点は，NP困難な問題に対する多項式時間アルゴリズムは現在知られていないので，問題のサイズによっては現実的な時間内では最適解を求めることができない場合があることである．

もう1つは最適解を求めることを諦め，近似解を現実的な時間で求めるという近似アルゴリズムの設計によるアプローチである ([3]，[4])．このアプローチには2通りある．1つは実行可能な時間内で保証のない近似解を求めるアプローチである．このアプローチについては遺伝的アルゴリズムなどが研究されている．もう1つは多項式時間で保証のある近似解を求めるアプローチである．

本稿ではこのアプローチの中でも代表的手法である主双対法について紹介し，これを用いて近似アルゴリズムの設計を行う．

品物をナップサックに容量を超えることなく詰め込む時に，詰め込む品物の価値の総和を最大化することを目的とした問題をナップサック問題と呼ぶ [2]．ナップサック問題はネットワーク構築 [5] や資源配分問題 [6] など現実問題に対しても多く用いられている．さらに，この問題はNP困難であることが知られている ([1]，[7])．この問題の様々な拡張に対する近似アルゴリズムの研究が広く行われている．

最小化ナップサック問題は，品物をナップサックに価値の総和をある需要以上に保ちつつ，詰め込む品物の重さの総和を最小化することを目的とする問題である [3]．この問題もNP困難な問題であることが知られており，最適解の導出や近似アルゴリズムの設計が広く行われている．

本研究は，この最小化ナップサック問題に対し新たな制約を加え，拡張した問題に対して制度保証のある近似アルゴリズムの提案を行う．本稿の2章では近似アルゴリズム，最小化ナップサック問題などの事前知識について説明し，3章と4章では最小化ナップサック問題に対し，品物の集合に関する制約を加えた問題に対する近似アルゴリズムについて記す．また，本稿の末尾にはそれぞれの近似アルゴリズムの詳細を示した疑似コードを添付する．

2　事前知識

2.1　近似アルゴリズム

近似アルゴリズムとは，最適値に十分に近い目的関数値が得られるような解を求めるアルゴリズムである．近似アルゴリズムの中でも最適化問題の全ての入力に対して最適値の α 倍以内の値を持つ解を返すアルゴリズムをその最適化問題に対する α-近似アルゴリズムと呼ぶ [3]．この時の α を近似率と呼ぶ．最小化問題に対しては $\alpha > 1$ であり，最大化問題に対しては $\alpha < 1$ となる．

2.2　主双対法

近似アルゴリズムの代表的な設計手法として主双対法がある [4]．本節では 0-1 整数計画問題

$$\begin{aligned} \min \quad & \sum_{j=1}^{n} c_j x_j \\ \text{s.t.} \quad & \sum_{j=1}^{n} a_{ij} x_j \geq b_i \quad (i = 1, \ldots, m) \\ & x_j \in \{0, 1\} \qquad (j = 1, \ldots, n) \end{aligned} \tag{2.1}$$

を例にとり，主双対法を説明する．ただし，$(a_{ij}) \in \mathbb{R}^{n \times m}_{++}, \boldsymbol{b} \in \mathbb{R}^{m}_{++}, \boldsymbol{c} \in \mathbb{R}^{n}_{++}$ とする．

主双対法は厳密アルゴリズムに起源を持つ．不等式標準形の線形計画問題

$$\begin{aligned} \min \quad & \boldsymbol{c}^T \boldsymbol{x} \\ \text{s.t.} \quad & A\boldsymbol{x} \geq \boldsymbol{b} \\ & \boldsymbol{0} \leq \boldsymbol{x} \leq \boldsymbol{e} \end{aligned}$$

とその双対問題

$$\begin{aligned} \min \quad & \boldsymbol{b}^T \boldsymbol{y} - \boldsymbol{e}^T \boldsymbol{u} \\ \text{s.t.} \quad & A^T \boldsymbol{y} - \boldsymbol{u} \geq \boldsymbol{c} \\ & \boldsymbol{y} \geq \boldsymbol{0} \\ & \boldsymbol{u} \geq \boldsymbol{0} \end{aligned}$$

のそれぞれの許容解 $\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}, \bar{\boldsymbol{u}}$ が相補性条件

$$\begin{aligned} \bar{\boldsymbol{x}}^T (\boldsymbol{c} - A^T \bar{\boldsymbol{y}} + \bar{\boldsymbol{u}}) = 0 \\ (A\bar{\boldsymbol{x}} - \boldsymbol{b})^T \bar{\boldsymbol{y}} - (\boldsymbol{e} - \bar{\boldsymbol{x}})^T \bar{\boldsymbol{u}} = 0 \end{aligned}$$

を満たしているならばそれらは最適解であるという性質がある [8]．

問題 (2.1) の整数計画問題を線形計画問題への緩和 (LP 緩和と呼ぶ) は

$$\begin{aligned} \min \quad & \sum_{j=1}^{n} c_j x_j \\ \text{s.t.} \quad & \sum_{j=1}^{n} a_{ij} x_j \geq b_i \quad (i = 1, \ldots, m) \\ & 0 \leq x_j \leq 1 \qquad (j = 1, \ldots, n) \end{aligned} \tag{2.2}$$

となる．問題 (2.2) の双対問題は，

$$\begin{aligned} \max \quad & \sum_{i=1}^{m} b_i y_i - \sum_{j=1}^{n} u_j \\ \text{s.t.} \quad & \sum_{i=1}^{m} a_{ij} y_i - u_j \le c_j \quad (j = 1, \ldots, n) \\ & y_i \ge 0 \qquad\qquad\quad (i = 1, \ldots, m) \\ & u_j \ge 0 \qquad\qquad\quad (j = 1, \ldots, n) \end{aligned} \tag{2.3}$$

となる．2 つの問題には以下の関係がある．

補題 2.1 主問題 (2.2) と双対問題 (2.3) はそれぞれ実行可能であるとし，共通の最適値を θ とする．主問題の許容解を $\bar{\boldsymbol{x}}$，双対問題の許容解を $\bar{\boldsymbol{y}}$ とした時，ある $\alpha, \beta > 1$ に対して

$$(j = 1, \ldots, n), \bar{x}_j > 0 \implies \sum_{i=1}^{m} a_{ij} \bar{y}_i - u_j \ge c_j / \alpha \tag{2.4a}$$

$$(i = 1, \ldots, m), \bar{y}_i > 0 \implies \sum_{j=1}^{n} a_{ij} \bar{x}_j \le \beta b_i \tag{2.4b}$$

$$(j = 1, \ldots, n), \bar{u}_j > 0 \implies \bar{x}_j \ge \beta \tag{2.4c}$$

を満たすならば，

$$\sum_{j=1}^{n} c_j \bar{x}_j \le \alpha\beta\theta \tag{2.5}$$

が成立する．

証明 関係 (2.4a) より，

$$\bar{x}_j > 0 \implies \alpha \sum_{i=1}^{m} a_{ij} \bar{y}_i - \alpha u_j \ge c_j \tag{2.6}$$

が得られる．関係 (2.6) と式 (2.4b)，式 (2.4c) を用いると式 (2.5) の左辺は

$$\begin{aligned} \sum_{j=1}^{n} c_j \bar{x}_j &\le \alpha \sum_{j=1}^{n} \sum_{i=1}^{m} a_{ij} \bar{x}_j \bar{y}_i - \alpha \sum_{j=1}^{n} \bar{x}_j \bar{u}_j \\ &\le \alpha\beta \sum_{i=1}^{m} b_i \bar{y}_i - \alpha\beta \sum_{j=1}^{n} u_j \end{aligned}$$

となる．双対問題は最大化問題であることから，

$$\sum_{i=1}^{m} b_i \bar{y}_i - \sum_{j=1}^{n} u_j \le \theta$$

なので，

$$\sum_{j=1}^{n} c_j \bar{x}_j \le \alpha\beta\theta$$

が得られる． □

よって補題 2.1 を満たす $\bar{\boldsymbol{x}}$ から得られる目的関数値は最適値の $\alpha\beta$ 倍以内の値になる．

離散最適化問題に対する主双対法とは整数計画問題を LP 緩和し，その双対問題を考え，そして緩和した問題の相補性条件を緩和することで，その条件を満たす解を得るアルゴリズムを構築することである．

2.3 最小化ナップサック問題

最小化ナップサック問題は品物の集合 $V = \{1, \ldots, n\}$ を用いて，

$$\min \sum_{j \in V} c_j x_j \tag{2.7a}$$

$$\text{s.t.} \sum_{j \in V} a_j x_j \geq b \tag{2.7b}$$

$$x_j \in \{0, 1\} \qquad \forall j \in V \tag{2.7c}$$

と表現される．この時 $\boldsymbol{a}, b, \boldsymbol{c}$ はそれぞれ $\boldsymbol{a}, \boldsymbol{c} \in \mathbb{Z}^n_{++}, b \in \mathbb{Z}_{++}$ である．式 (2.7a) は選択した品物の重さの総和を最小化することを目的としている．式 (2.7b) は選択した品物の価値の総和を需要 b 以上に保つということを意味し，ナップサック制約と呼ばれる．式 (2.7c) はバイナリ制約と呼ばれる．

2.3.1 LP 緩和

本節では最小化ナップサック問題の LP 緩和とその双対問題を紹介する．

一般に整数計画問題の最適値とその LP 緩和の最適値は異なる．Carr ら [9] は最小化ナップサック問題において品物の部分集合 $A \subseteq V$ に対して新しい制約

$$\sum_{j \in V \setminus A} a_j(A) x_j \geq b(A) \tag{2.8}$$

を導入した．ただし，

$$a_j(A) = \min\{a_j, b(A)\} \qquad \forall j \in V \setminus A \tag{2.9a}$$

$$b(A) = \max\{0, b - \sum_{j \in A} a_j\} \tag{2.9b}$$

である．この制約は A に含まれている品物は全て選択したとし，残りの品物で需要 $b - \sum_{j \in A} a_j$ を満たすように品物を選択する制約である．

さらに Carr らは最小化ナップサック問題に対し，制約 (2.8) を品物の全ての部分集合 $A \subseteq V$ それぞれに対して導入し以下の LP 緩和問題を考えた:

$$\begin{aligned} \min \quad & \sum_{j \in V} c_j x_j \\ \text{s.t.} \quad & \sum_{j \in V \setminus A} a_j(A) x_j \geq b(A) \quad \forall A \subseteq V \\ & x_j \geq 0 \qquad\qquad\qquad\quad \forall j \in V \end{aligned} \tag{2.10}$$

問題 (2.10) の双対問題は

$$\begin{aligned} \max \quad & \sum_{A\subseteq V} b(A)y(A) & & \\ \text{s.t.} \quad & \sum_{A\subseteq V: j\notin A} a_j(A)y(A) \le c_j & \forall j \in V & \\ & y(A) \ge 0 & \forall A \subseteq V & \end{aligned} \tag{2.11}$$

となる [3]．ここで A は任意の V の部分集合を取りうるので変数 $y(A)$ の個数は $2^{|V|}$ 個になる．

補題 2.2 問題 (2.7) の許容解は LP 緩和した問題 (2.10) の許容解でもある．

証明 問題 (2.7) の許容解を $\bar{\boldsymbol{x}}$ とし，$\bar{S} = \{j \in V | x_j = 1\}$ とする．任意の $A \subseteq V$ に対して，

$$\begin{aligned} \sum_{j\in V\setminus A} a_j(A)\bar{x}_j - b(A) &= \sum_{j\in V\setminus A} \min\{a_j, b(A)\}\bar{x}_j - b(A) \\ &= \sum_{j\in \bar{S}\setminus A} \min\{a_j, b(A)\} - b(A) \end{aligned}$$

となる．

$b(A) = 0$ の時，

$$\sum_{j\in \bar{S}\setminus A} \min\{a_j, b(A)\} - b(A) = 0$$

となるので，問題 (2.7) の許容解は LP 緩和した問題 (2.10) の許容解である．

次に，$b(A) > 0$ の時について考える．この時，

$$\sum_{j\in \bar{S}\setminus A} \min\{a_j, b(A)\} - b(A) = \sum_{j\in \bar{S}\setminus A} \min\{a_j, b(A)\} - b(A)$$

となる．ここである $j \in \bar{S} \setminus A$ に対し，$a_j \ge b(A)$ のとき，

$$\sum_{j\in \bar{S}\setminus A} \min\{a_j, b(A)\} - b(A) \ge b(A) - b(A) = 0$$

となる．また，任意の $j \in \bar{S} \setminus A$ に対し，$a_j < b(A)$ のとき，

$$\begin{aligned} \sum_{j\in \bar{S}\setminus A} \min\{a_j, b(A)\} - b(A) &= \sum_{j\in \bar{S}\setminus A} a_j - b + \sum_{j\in A} a_j \\ &= \sum_{j\in \bar{S}\cup A} a_j - b \\ &\ge \sum_{j\in \bar{S}\cup A} a_j\bar{x}_j - b \\ &= \sum_{j\in V} a_j\bar{x}_j - b \ge 0 \end{aligned}$$

となる．したがって $b(A) > 0$ の場合でも問題 (2.7) の許容解は LP 緩和した問題 (2.10) の許容解である． □

したがって補題 2.2 より，問題 (2.7) の最適値を θ^*_{carr}，問題 (2.10) の最適値を θ_{carr} とすると，

$$\theta^*_{carr} \geq \theta_{carr}$$

が成り立つ．

2.3.2 主双対法

以上で示した最小化ナップサック問題を LP 緩和した問題 (2.10) とその双対問題 (2.11) を用いて主双対法によるアルゴリズムの構築が研究されている [3]．その準備として，以下の補題を示す．

補題 2.3 主問題 (2.10) の許容解を $\bar{\boldsymbol{x}}$ とし，双対問題 (2.11) の許容解を $\bar{\boldsymbol{y}}$ とする．問題 (2.10) と問題 (2.11) の共通の最適値を θ_{mkp} とする．もし

$$\forall j \in V, \bar{x}_j > 0 \implies \sum_{A \subseteq V : j \notin A} a_j(A)\bar{y}(A) = c_j \tag{2.12a}$$

$$\forall A \subseteq V, \bar{y}(A) > 0 \implies \sum_{j \in V \setminus A} a(A)_j \bar{x}_j \leq 2b(A) \tag{2.12b}$$

が満たされるならば，

$$\sum_{j \in V} c_j \bar{x}_j \leq 2\theta_{mkp} \tag{2.13}$$

が成り立つ．

証明 式 (2.12a) と式 (2.12b) を用いると式 (2.13) の左辺は補題 2.1 より

$$\sum_{j \in V} c_j \bar{x}_j \leq 2 \sum_{A \subseteq V} b(A)\bar{y}(A)$$

となる．この時，双対問題は最大化問題なので，

$$\sum_{A \subseteq V} b(A)\bar{y}(A) \leq \theta_{mkp}$$

が成り立つので，

$$\sum_{j \in V} c_j \bar{x}_j \leq 2\theta_{mkp}$$

が得られる． □

系 2.1 問題 (2.10) の 0 と 1 のみからなる許容解を $\hat{\boldsymbol{x}}$，問題 (2.11) の許容解を $\hat{\boldsymbol{y}}$ とする．これらが補題 2.3 の条件を満たすとする．この時，問題 (2.7) の最適値を θ^*_{mkp} とすると．

$$\sum_{j \in V} c_j \hat{x}_j \leq 2\theta^*_{mkp}$$

が成り立つ．

2.3.3 最小化ナップサック問題の 2-近似アルゴリズム [3]

系 2.1 を満たす解 $\tilde{\boldsymbol{x}}$ と $\tilde{\boldsymbol{y}}$ を得る最小化ナップサック問題の 2-近似アルゴリズムをここに示す.

Algorithm1

Input: V: 品物の集合，$\boldsymbol{a}$: 品物の価値のベクトル，b: ナップサック内に保ちたい価値，
$\boldsymbol{c}$: 品物の重さのベクトル

Output: $\tilde{\boldsymbol{x}}$: 主問題の解，$\tilde{\boldsymbol{y}}$: 双対問題の解

Step0: $\boldsymbol{x} = \boldsymbol{0}, \boldsymbol{y} = \boldsymbol{0}$ を初期解として与える．また，$S = \emptyset$，$\bar{b} = b$，$\bar{c}_j = c_j (\forall j \in V)$ を初期値として与える.

Step1: $\bar{b} \leq 0$ ならば $\tilde{\boldsymbol{x}} = \boldsymbol{x}, \tilde{\boldsymbol{y}} = \boldsymbol{y}$ とし，アルゴリズムを停止する．そうでないならば，
$s = \underset{j \in V \setminus S}{\arg\min} \left\{ \dfrac{\bar{c}_j}{a_j(S)} \right\}$ を計算する.

Step2: $y(S) = \dfrac{\bar{c}_s}{a_s(S)}, x_s = 1$，$S = S \cup \{s\}$，$\bar{b} = b - a_s$，$\bar{c}_j = \bar{c}_j - a_j(S)y(S) (\forall j \in V \setminus S)$
とし，Step1 の初めに戻る.

アルゴリズムの疑似コードを付録 A に示す.

補題 2.4 Algorithm1 より得られた解は補題 2.3 の条件 (2.12a) と条件 (2.12b) を満たす.

証明 Algorithm1 は $\boldsymbol{x} = \boldsymbol{0}, \boldsymbol{y} = \boldsymbol{0}$ から始まり，$x_j = 1$ となるのはアルゴリズム中で $y(S)$ の値を変更した後に S の更新を行うことに注意すると，

$$y(S) = \frac{c_j - \sum_{A \subseteq S \setminus \{k\}} a_j(A)y(A)}{a_j(S)} \tag{2.14}$$

の時のみである．ただし，k は $x_j = 1$ となる前の反復で 0 から 1 へと更新された変数とする．この時，式 (2.14) を c_j について書くと，

$$\begin{aligned} c_j &= \sum_{A \subseteq S \setminus \{k\}} a_j(A)y(A) + a_j(S)y(S) \\ &= \sum_{A \subseteq S : k \notin A} a_j(A)y(A) \end{aligned}$$

であり，この時，任意の $B \not\subseteq S$ については $y(B) = 0$ である．したがって，$\tilde{x}_j > 0$ ならば

$$\sum_{A \subseteq V : j \notin A} a_j(A)\tilde{y}(A) = c_j$$

を満たすことがわかる.

アルゴリズムより得られた $\tilde{\boldsymbol{x}}$ を用いて $\tilde{S} = \{j \in V | \tilde{x}_j = 1\}$ とする．また，$\tilde{x}_l$ を Algorithm1 の最後に 0 から 1 へと更新された変数とする．ここで $\tilde{y}(A) > 0$ である A について，Algorithm1 では x_s と $y(S)$ を更新したのちに $S = S \cup \{s\}$ とするので

$$A \subseteq \tilde{S} \setminus \{l\}$$

とすることができる．また，$\tilde{x}_l = 1$ とする直前ではナップサック制約を満たしていないので，

$$\sum_{j \in \tilde{S} \setminus \{l\}} a_j < b \tag{2.15}$$

である．式 (2.9a) より $a_j(A) \leq a_j$ なので，$\tilde{y}(A) > 0$ である A について，

$$\sum_{j \in (\tilde{S} \setminus \{l\}) \setminus A} a_j(A) \leq \sum_{j \in (\tilde{S} \setminus \{l\}) \setminus A} a_j = \sum_{j \in \tilde{S} \setminus \{l\}} a_j - \sum_{j \in A} a_j$$

であり，式 (2.9b) より $b(A) \geq b - \sum_{j \in A} a_j$ なので，これと式 (2.15) から，

$$\sum_{j \in \tilde{S} \setminus \{l\}} a_j - \sum_{j \in A} a_j < b - \sum_{j \in A} a_j \leq b(A)$$

であるので，

$$\sum_{j \in (\tilde{S} \setminus \{l\}) \setminus A} a_j(A) \leq b(A)$$

となることがわかる．また，式 (2.9a) より，$a_j(A) \leq b(A)$ なので，

$$\sum_{j \in V \setminus A} a_j(A) \tilde{x}_j = \sum_{j \in \tilde{S} \setminus A} a_j(A) = \sum_{j \in (\tilde{S} \setminus \{l\}) \setminus A} a_j(A) + a_l(A) \leq 2b(A)$$

となる．したがって，Algorithm1 は条件 (2.12b) を満たす．

以上より，Algorithm1 より得られた解は補題 2.3 の条件 (2.12a) と条件 (2.12b) を満たす． □

補題 2.5 問題 (2.7) が実行可能ならば，$\boldsymbol{x} = (1, \ldots, 1)$ は主問題 (2.10) の許容解である．

証明 問題 (2.7) が実行可能であるための条件は

$$\sum_{j \in V} a_j \geq b$$

である．したがって，$\boldsymbol{x} = (1, \ldots, 1)$ の時

$$\sum_{j \in V} a_j x_j \geq b$$

を満たす．また，(2.9b) より，$b(A) \geq b - \sum_{j \in A} a_j$ なので，任意の $A \subseteq V$ に対して $\boldsymbol{x} = (1, \ldots, 1)$ の時，

$$\sum_{j \in V \setminus A} a_j(A) x_j \geq b(A)$$

を満たす． □

補題 2.6 Algorithm1 から得られた $\tilde{\boldsymbol{x}}$ は主問題 (2.10) の 0 と 1 からなる許容解であり，$\tilde{\boldsymbol{y}}$ は双対問題 (2.11) の許容解である．

証明　補題 2.5 より，問題 (2.7) が実行可能ならば，$\boldsymbol{x} = (1, \ldots, 1)$ は主問題 (2.10) の許容解である．つまり，品物を全て選択するような解も許容解となる．また，Algorithm1 は $\boldsymbol{x} = \mathbf{0}$ から始まり，$\bar{b} \leq 0$ となるまである x_j を 0 から 1 へと変更し，反復する．したがってこのアルゴリズムは元問題 (2.7) のナップサック制約を満たすように x_j を 0 から 1 へと変更している．

さらに,，この解が主問題 (2.10) の解であるかについて考える．$\tilde{S} = \{j \in V | \tilde{x}_j = 1\}$ とする．$A \subseteq V, \tilde{S} \cap A \neq \emptyset$ であるような A について考える．この時,

$$\begin{aligned}
\sum_{j \in V \setminus A} a_j(A)\tilde{x}_j - b(A) &\geq \sum_{j \in V \setminus A} a_j(A)\tilde{x}_j - b + \sum_{j \in A} a_j \\
&= \sum_{j \in V \setminus A} a_j(A)\tilde{x}_j - b + \sum_{j \in \tilde{S} \cap A} a_j + \sum_{j \in A \setminus \tilde{S}} a_j \\
&\geq \sum_{j \in \tilde{S}} a_j - b \\
&\geq 0
\end{aligned}$$

より，主問題 (2.10) の許容解である．さらに，$A \subseteq V, \tilde{S} \cap A = \emptyset$ であるような A について考える．

$$\begin{aligned}
\sum_{j \in V \setminus A} a_j(A)\tilde{x}_j - b(A) &\geq \sum_{j \in V \setminus A} a_j(A)\tilde{x}_j - b + \sum_{j \in A} a_j \\
&\geq \sum_{j \in \tilde{S}} a_j - b \\
&\geq 0
\end{aligned}$$

より，これもまた主問題 (2.10) の許容解である．

したがって，$\tilde{\boldsymbol{x}}$ は主問題 (2.10) の 0 と 1 のみからなる許容解である．

一方，$\boldsymbol{y} = \mathbf{0}$ は双対問題の許容解である．Algorithm1 はこれを初期値とし，補題 2.4 で示したように，アルゴリズム全体を通して双対問題の実行可能性を維持している．したがって $\tilde{\boldsymbol{y}}$ は双対問題 (2.11) の許容解である．　□

定理 2.1　Algorithm1 は最小化ナップサック問題に対する 2-近似アルゴリズムである．

証明　Algorithm1 から得られる $\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}$ は補題 2.4 を満たしている．

この時，最小化ナップサック問題 (2.7) の最適値を θ^* とおくと，目的関数値は系 2.1 より，

$$\sum_{j \in V} c_j \tilde{x}_j \leq 2\theta^*$$

を満たす．また最小化問題の性質から，

$$\theta^* \leq \sum_{j \in V} c_j \tilde{x}_j$$

が成り立つ．したがって，Algorithm1 より得られる解からなる目的関数値は最適値の 2 倍以内の値になる．　□

3 Partition 制約付き最小化ナップサック問題

最小化ナップサック問題における V を用いた集合 $\mathcal{P}$ を

$$\begin{aligned}&\mathcal{P} = \{P_1, \ldots, P_m\}\\ &P_j \subseteq V, |P_j| \geq 2 \qquad \forall j \in \{1, \ldots, m\}\\ &P_i \cap P_j = \emptyset \qquad \forall i, j \in \{1, \ldots, m\}(i \neq j)\end{aligned}$$

と定義し，以下の問題を考える:

$$\min \sum_{j \in V} c_j x_j \tag{3.1a}$$

$$\text{s.t.} \sum_{j \in V} a_j x_j \geq b \tag{3.1b}$$

$$\sum_{j \in P} x_j \geq 1 \qquad \forall P \in \mathcal{P} \tag{3.1c}$$

$$x_j \in \{0, 1\} \qquad \forall j \in V \tag{3.1d}$$

制約 (3.1c) は任意の $P \in \mathcal{P}$ に含まれる品物のうち少なくとも一つは選択することを意味する．この制約を Partition 制約と呼ぶことにする．問題 (3.1) を Partition 制約付き最小化ナップサック問題 (Minumum Knapsack Problem with Partition Constraints:MKPPC) と呼ぶことにする．

3.1 MKPPC の分割

MKPPC の近似アルゴリズムを構築するために 2 つの部分問題

$$\begin{aligned}\min \quad & \sum_{j \in V} c_j x_j\\ \text{s.t.} \quad & \sum_{j \in P} x_j \geq 1 \quad \forall P \in \mathcal{P}\\ & x_j \in \{0, 1\} \quad \forall j \in V\end{aligned} \tag{3.2}$$

と，

$$\begin{aligned}\min \quad & \sum_{j \in V} c_j x_j\\ \text{s.t.} \quad & \sum_{j \in V} a_j x_j \geq b\\ & x_j \in \{0, 1\} \quad \forall j \in V\end{aligned} \tag{3.3}$$

を考える．

問題 (3.2) は Partition 制約 (3.1c) を持った問題であり，問題 (3.3) はナップサック制約 (3.1b) を持った問題である．前章により，問題 (3.3) に対する 2-近似アルゴリズムが存在することが分かっている．

3.2 問題 (3.2) に対する厳密解法

問題 (3.2) は入力の条件として任意の $\mathcal{P}$ の要素は互いに素であることを仮定しているので，容易に最適解を導出することができる．その厳密解法の詳細をここに示す．

Algorithm2

Input: V: 品物の集合，$\mathcal{P}$，$\boldsymbol{c}$: 品物の重さのベクトル

Output: $\tilde{\boldsymbol{x}}$: 問題 (3.2) の解

Step0: $\boldsymbol{x} = \mathbf{0}$ を初期解として与える．$\bar{\mathcal{P}} = \mathcal{P}$ を初期値として与える．

Step1: $P \in \bar{\mathcal{P}}$ を 1 つ選択し，$\bar{\mathcal{P}} = \bar{\mathcal{P}} \setminus \{P\}$ とする．

Step2: $s = \underset{j \in P}{\arg\min}\, c_j$ を計算し，$x_s = 1$ とする．

Step3: $\bar{\mathcal{P}} = \emptyset$ ならば $\tilde{\boldsymbol{x}} = \boldsymbol{x}$ とし，アルゴリズムを終了する．そうでないならば Step1 へ戻る．

アルゴリズムの疑似コードを付録 A に示す．

3.3 MKPPC の 3-近似アルゴリズム

MKPPC の 3-近似アルゴリズムを以下のように提案する．

Algorithm3

Input: V: 品物の集合，$\mathcal{P}$，$\boldsymbol{a}$: 品物の価値のベクトル，b: ナップサック内に保ちたい価値，$\boldsymbol{c}$: 品物の重さのベクトル

Output: $\tilde{S}$: 選択した品物の集合

Step0: $S_1 = S_2 = \emptyset$ とする．

Step1: 問題 (3.2) を満たすように Algorithm2 を適用．$S_1 = \{j \mid x_j = 1\}$ とする．また，$\bar{b} = b - \sum_{j \in S_1} a_j$ とする．

※ Step1 の結果がナップサック制約を満たしているなら Step3 へ

Step2: 残った品物 $V' = V \setminus S_1$ と $\bar{b}$ で問題 (3.3) を構成．Algorithm1 を適用．$S_2 = \{j \mid x_j = 1\}$ とする．

Step3: $\tilde{S} = S_1 \cup S_2$ とし，アルゴリズムを停止する．

アルゴリズムの疑似コードを付録 A に示す．

補題 3.1 Algorithm3 より得られた解は MKPPC の 0 と 1 のみからなる許容解である．さらに MKPCC の最適値を θ とした時，$\sum_{j \in \tilde{S}} c_j \leq 3\theta$ を満たす．

証明　Algorithm3 より得られた解は問題 (3.2) と問題 (3.3) 両方の制約を満たす．また，アルゴリズムは $\boldsymbol{x} = \mathbf{0}$ から始まり，Step1 で Partition 制約を満たすように $\mathcal{P}$ について Algorithm2 を用いて x_s を 0 から 1 へと変更している．Step1 の反復が終了した段階で $\boldsymbol{x}$ は少なくとも Partition 制約を満たしている．Step1 での変更でナップサック制約が満たされない場合は Step2 でナップサック制約を満たすように Algorithm1 を用いて x_s を 0 から 1 へと変更している．Step2 の反復が終了した状態で $b - \sum_{j \in V} a_j x_j \leq 0$ となっているため，ナップサック制約を満たしていることがわかる．したがって以上より，この解は MKPPC の 0 と 1 のみからなる許容解である．

さらに

$$\sum_{j \in V} c_j x_j = \sum_{j \in S} c_j = \sum_{j \in S_1} c_j + \sum_{j \in S_2} c_j$$

と変形できる．問題 (3.2) の最適値を θ_1，問題 (3.3) の最適値を θ_2 とおくと，問題 (3.1) は問題 (3.2) にナップサック制約を加えたものなので，$\theta_1 \leq \theta$．同様にして $\theta_2 \leq \theta$ なので，

$$\begin{aligned} \sum_{j \in S_1} c_j + \sum_{j \in S_2} c_j &\leq \theta_1 + 2\theta_2 \\ &\leq \theta + 2\theta \\ &= 3\theta \end{aligned}$$

となる．したがって $\sum_{j \in V} c_j x_j \leq 3\theta$ を満たす．　□

補題 3.2　Algorithm3 の計算量は $p = \max_{P \in \mathcal{P}} |P|$ とおくと，$\mathcal{O}(p|\mathcal{P}| + |V|^2)$ である．

証明　　Step1 の 1 回の反復での計算は $\arg\min_{j \in P} c_j$ のみなので，その計算量は高々$\mathcal{O}(p)$ となる．また，反復回数は $\mathcal{O}(|\mathcal{P}|)$ なので，Step1 全体の計算量は高々$\mathcal{O}(p|\mathcal{P}|)$ であることがわかる．

次に，Step2 の 1 回の反復での計算は 2 種類ある．$a_j(S)$ の計算量は高々$\mathcal{O}(|V|)$ である．また，$\arg\min_{j \in V \setminus S} \left\{ \frac{\bar{c}_j}{a_j(S)} \right\}$ の計算量も高々$\mathcal{O}(|V|)$ である．さらに，Step2 の反復回数は高々$|V|$ 回である．よって Step2 全体の計算量は $\mathcal{O}(|V|^2)$ となる．

以上より，Algorithm3 の計算量は $\mathcal{O}(p|\mathcal{P}| + |V|^2)$ である．　□

定理 3.1　Algorithm3 は MKPPC に対する 3-近似アルゴリズムである．

4　Forcing Hypergraph 付き最小化ナップサック問題

本章では MKPPC を一般化した問題とその問題を解く近似アルゴリズムについて記す．

4.1　ハイパーグラフ

V を頂点集合，$\mathcal{E} \subseteq 2^V$ を辺集合とした $H = (V, \mathcal{E})$ をハイパーグラフと呼ぶ [10]．

ハイパーグラフはグラフの概念の一般化となっている．もし，あるハイパーグラフの任意の $E \in \mathcal{E}$ について $|E| = 2$ ならばそのハイパーグラフは通常の無向グラフに等しい．

4.2 Forcing Hypergraph 付き最小化ナップサック問題

Forcing Hypergraph $H = (V, \mathcal{E})$ を以下のように定義する: $V = \{1, \ldots, n\}$，任意の $E \in \mathcal{E}$ について $|E| \geq 2$.

以下の問題を考える:

$$\min \sum_{j \in V} c_j x_j \tag{4.1a}$$

$$\text{s.t.} \sum_{j \in V} a_j x_j \geq b \tag{4.1b}$$

$$\sum_{j \in E} x_j \geq 1 \qquad \forall E \in \mathcal{E} \tag{4.1c}$$

$$x_j \in \{0, 1\} \qquad \forall j \in V \tag{4.1d}$$

制約 (4.1c) は任意の $E \in \mathcal{E}$ に含まれる頂点のうちどれか一つは必ず選択しなければならないということを意味する．この制約を Forcing 制約と呼ぶことにする．問題 (4.1) を Forcing Hypergraph 付き最小化ナップサック問題 (Minimum Knapsack Problem with Forcing Hypergraph:MKPFH) と呼ぶことにする．

任意の 2 つの要素 $E_1, E_2 \in \mathcal{E} (E_1 \neq E_2)$ が互いに素である時，MKPFH は MKPPC に等しい．

4.3 LP 緩和

主双対法を用いるために問題 (4.1) の LP 緩和を考える．そのために，最小化ナップサック問題と同様に任意の品物の部分集合 $A \subseteq V$ に対して新しい制約

$$\sum_{j \in V \setminus A} a_j(A) x_j \geq b(A) \tag{4.2}$$

を導入する．ただし，

$$a_j(A) = \min\{a_j, b(A)\} \qquad \forall j \in V \setminus A \tag{4.3a}$$

$$b(A) = \max\{0, b - \sum_{j \in A} a_j\} \tag{4.3b}$$

とする．MKPFH に対し，最小化ナップサック問題の LP 緩和と同様に制約 (4.2) を品物の全ての部分集合 $A \subseteq V$ それぞれに対して導入し以下の LP 緩和問題を考えた:

$$\begin{aligned} \min \quad & \sum_{j \in V} c_j x_j \\ \text{s.t.} \quad & \sum_{j \in V \setminus A} a_j(A) x_j \geq b(A) \quad \forall A \subseteq V \\ & \sum_{j \in E} x_j \geq 1 \qquad \forall E \in \mathcal{E} \\ & x_j \geq 0 \qquad \forall j \in V \end{aligned} \tag{4.4}$$

さらに問題 (4.4) の双対問題は

$$\begin{aligned} \max \quad & \sum_{A\subseteq V} b(A)y(A) + \sum_{E\in\mathcal{E}} z_e & & \\ \text{s.t.} \quad & \sum_{A\subseteq V: j\notin A} a_j(A)y(A) + \sum_{E\in\mathcal{E}: j\in E} z_E \le c_j & & \forall j \in V \\ & y(A) \ge 0 & & \forall A \subseteq V \\ & z_E \ge 0 & & \forall E \in \mathcal{E} \end{aligned} \tag{4.5}$$

と表される.

4.4 主双対法

式 (4.4) と式 (4.5) に主双対法における満たすべき条件を考える．そのために以下の補題を示す. 以下では $k = \max_{E\in\mathcal{E}} |E|$ とする.

補題 4.1 主問題 (4.4) の許容解を $\bar{\boldsymbol{x}}$ とし, 双対問題 (4.5) の許容解を $(\bar{\boldsymbol{y}}, \bar{\boldsymbol{z}})$ とする．問題 (4.4) と問題 (4.5) の共通の最適値を θ とする．もし,

$$\forall j \in V, \bar{x}_j > 0 \implies \sum_{A\subseteq V: j\notin A} a_j(A)\bar{y}(A) + \sum_{E\in\mathcal{E}: j\in E} \bar{z}_E = c_j \tag{4.6a}$$

$$\forall E \in \mathcal{E}, \bar{z}_E > 0 \implies \sum_{j\in E} \bar{x}_j \le k \tag{4.6b}$$

$$\forall A \subseteq V, \bar{y}(A) > 0 \implies \sum_{j\in V\setminus A} a_j(A)\bar{x}_j \le kb(A) \tag{4.6c}$$

が満たされるならば,

$$\sum_{j\in V} c_j\bar{x}_j \le k\theta \tag{4.7}$$

が成り立つ.

証明 条件 (4.6) を用いると，式 (4.7) の左辺は

$$\begin{aligned} \sum_{j\in V} c_j\bar{x}_j &= \sum_{j\in V} \left\{ \sum_{A\subseteq V: j\notin A} a_j(A)\bar{y}(A) + \sum_{E\in\mathcal{E}: j\in E} \bar{z}_E \right\} \bar{x}_j \\ &= \sum_{j\in V} \sum_{A\subseteq V: j\notin A} a_j(A)\bar{y}(A)\bar{x}_j + \sum_{j\in V} \sum_{E\in\mathcal{E}: j\in E} \bar{z}_E\bar{x}_j \\ &= \sum_{A\subseteq V} \sum_{j\in V\setminus A} a_j(A)\bar{y}(A)\bar{x}_j + \sum_{E\in\mathcal{E}} \sum_{j\in E} \bar{z}_E\bar{x}_j \\ &\le k \sum_{A\subseteq V} b(A)\bar{y}(A) + k \sum_{E\in\mathcal{E}} \bar{z}_E \\ &= k \left\{ \sum_{A\subseteq V} b(A)\bar{y}(A) + \sum_{E\in\mathcal{E}} \bar{z}_E \right\} \end{aligned}$$

となる．この時双対問題は最大化問題なのでその性質より，

$$\sum_{A\subseteq V} b(A)\bar{y}(A) + \sum_{E\in\mathcal{E}} \bar{z}_E \leq \theta$$

が成り立つので，

$$\sum_{j\in V} c_j \bar{x}_j \leq k\theta$$

が得られる． □

系 4.1 問題 (4.4) の 0 と 1 のみからなる許容解を $\hat{\boldsymbol{x}}$，問題 (4.5) の許容解を $(\hat{\boldsymbol{y}}, \hat{\boldsymbol{z}})$ とする．これらが補題 4.1 の条件 (4.6) を満たすとする．この時，問題 (4.1) の最適値を θ^* とすると．

$$\sum_{j\in V} c_j \hat{x}_j \leq k\theta^*$$

が成り立つ．

4.5 MKPFH に対する k-近似アルゴリズム

系 4.1 を満たす解 $\tilde{\boldsymbol{x}}$ と $(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}})$ を得る MKPFH の k-近似アルゴリズムをここに示す．

Algorithm4

Input: $H = (V, \mathcal{E})$(V: 品物の集合，$\mathcal{E}$: 品物のクラスタの集合)，$\boldsymbol{a}$: 品物の価値のベクトル，b: ナップサック内に保ちたい価値，$\boldsymbol{c}$: 品物の重さのベクトル

Output: $\tilde{\boldsymbol{x}}$: 主問題の許容解，$(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}})$: 双対問題の許容解

Step0: $\boldsymbol{x} = \boldsymbol{0}$，$(\boldsymbol{y}, \boldsymbol{z}) = (\boldsymbol{0}, \boldsymbol{0})$ を初期解として与える．また，$S = \emptyset$，$\bar{\mathcal{E}} = \mathcal{E}$，$\bar{b} = b$，$\bar{c}_j = c_j (\forall j \in V)$ を初期値として与える．

Step1: $\bar{\mathcal{E}} = \emptyset$ ならば Step3 へ進む．そうでないならば $\min_{E\in\bar{\mathcal{E}}} |E|$ となる E を 1 つ選択し，$\bar{E}$ とする．$\bar{\mathcal{E}} = \bar{\mathcal{E}} \setminus \{\bar{E}\}$ とする．

Step2: $\sum_{j\in E} x_j \geq 1$ ならば Step1 に戻る。そうでないならば $s = \arg\min_{j\in E}\{\bar{c}_j\}$ を計算し，$z_E = \bar{c}_s$，$x_s = 1$ する．さらに，$S = S \setminus \{s\}$，$\bar{b} = b - a_s$，$\bar{c}_j = \bar{c}_j - z_E (\forall j \in E)$ とし，Step1 に戻る．

Step3: $\bar{b} \leq 0$ ならば $\tilde{\boldsymbol{x}} = \boldsymbol{x}$，$(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}}) = (\boldsymbol{y}, \boldsymbol{z})$ とし，アルゴリズムを停止する．そうでないならば，Step4 へ．

Step4: $s = \arg\min_{j\in V\setminus S} \left\{ \dfrac{\bar{c}_j}{a_j(S)} \right\}$ を計算し，$y(S) = \dfrac{\bar{c}_s}{a_s(S)}$，$x_s = 1$ とする．さらに，$S = S \setminus \{s\}$，$\bar{b} = b - a_s$，$\bar{c}_j = \bar{c}_j - a_j y(S) (\forall j \in V \setminus S)$ とし，Step3 の初めに戻る．

アルゴリズムの疑似コードを付録 A に記す．

補題 4.2 Algorithm4 より得られた解は補題 4.1 の条件 (4.6) を満たす.

証明 Algorithm4 は $\boldsymbol{x} = \mathbf{0}, (\boldsymbol{y}, \boldsymbol{z}) = (\mathbf{0}, \mathbf{0})$ から始まり，Step1 と Step2 で $x_j = 1$ となるのは Forcing 制約を満たすように

$$z_E = c_j - \sum_{B \in \mathcal{E} \setminus \{E\} : j \in B} z_B$$

とした時のみである．Step1 と Step2 でナップサック制約が満たされたとすると，$\tilde{x}_j > 0$ ならば

$$\sum_{E \in \mathcal{E} : j \in E} \tilde{z}_E = c_j$$

を満たすことがわかる.

次に Step1 と Step2 でナップサック制約が満たされない場合を考える．Step3 と Step4 で $x_j = 1$ となるのはナップサック制約を満たすように

$$y(S) = \frac{c_j - \sum_{A \subseteq S \setminus \{k\} : j \notin A} a_j(A) y(A) - \sum_{E \in \mathcal{E} : j \in E} z_E}{a_j(S)}$$

とした時のみである．ただし，k は $x_j = 1$ となる前の反復で 0 から 1 へと更新された変数とする．アルゴリズム中で $y(S)$ の値を変更した後に S の更新を行うことに注意する．この時，式 (2.14) を c_j について書くと，

$$\begin{aligned} c_j &= a_j(S) y(S) + \sum_{A \subseteq S \setminus \{k\}} a_j(A) y(A) + \sum_{E \in \mathcal{E} : j \in E} z_E \\ &= \sum_{A \subseteq S} a_j(A) y(A) + \sum_{E \in \mathcal{E} : j \in E} z_E \end{aligned}$$

となる．この時，任意の $B \not\subseteq S$ について $y(B) = 0$ である．したがって，$\tilde{x}_j > 0$ であるならば

$$\sum_{A \subseteq V : j \notin A} a_j(A) \tilde{y}(A) + \sum_{E \in \mathcal{E} : j \in E} \tilde{z}_E = c_j$$

を満たすことがわかる．以上より，Algorithm4 は条件 (4.6a) を満たすことがわかる.

次に条件 (4.6b) についてだが，Algorithm4 中で任意の $j \in V$ に対して x_j は常に 0 か 1 しかとらない．したがって $k = \max_{E \in \mathcal{E}} |E|$ であることから Algorithm4 は常に条件 (2.4b) を満たしている.

最後に条件 (4.6c) だが，これについては Algorithm4 が Step3 で直ちに停止したかどうかの 2 つの場合について考える.

まず停止した時，つまり Step4 を 1 度も実行しなかった場合を考える．この時は Step4 で $y(A)$ が操作されないので任意の $A \subseteq V$ に対して $y(A) = 0$ となる．したがってこの時 Algorithm4 は条件 (4.6c) を満たす.

次に Step4 を 1 回以上実行した場合について考える．アルゴリズムから得られた $\tilde{\boldsymbol{x}}$ を用いて $\tilde{S} = \{j \in V | \tilde{x}_j = 1\}$ とする．また，$\tilde{x}_l$ を Algorithm4 の最後に 0 から 1 へと更新された変数とする．ここで $\tilde{y}(A) > 0$ である A について，Algorithm4 では x_s と $y(S)$ を更新したのちに $S = S \cup \{s\}$ とするので

$$A \subseteq \tilde{S} \setminus \{l\}$$

とすることができる．また，$\tilde{x}_l = 1$ とする直前ではナップサック制約を満たしていないので，

$$\sum_{j \in \tilde{S} \setminus \{l\}} a_j < b \tag{4.8}$$

である．式 (4.3a) より $\tilde{y}(A) > 0$ である A について，

$$\sum_{j \in (\tilde{S} \setminus \{l\}) \setminus A} a_j(A) \leq \sum_{j \in (\tilde{S} \setminus \{l\}) \setminus A} a_j = \sum_{j \in \tilde{S} \setminus \{l\}} a_j - \sum_{j \in A} a_j$$

であり，式 (4.3b) と式 (4.8) から，

$$\sum_{j \in \tilde{S} \setminus \{l\}} a_j - \sum_{j \in A} a_j < b - \sum_{j \in A} a_j \leq b(A)$$

であるので，

$$\sum_{j \in (\tilde{S} \setminus \{l\}) \setminus A} a_j(A) \leq b(A)$$

となることがわかる．また，式 (4.3a) より，$a_j(A) \leq b(A)$ なので，

$$\sum_{j \in V \setminus A} a_j(A)\tilde{x}_j = \sum_{j \in \tilde{S} \setminus A} a_j(A) = \sum_{j \in (\tilde{S} \setminus \{l\}) \setminus A} a_j(A) + a_l(A) \leq 2b(A)$$

となることがわかる．ここで，入力として与えるハイパーグラフの条件より $k \geq 2$ なので，

$$\sum_{j \in V \setminus A} a_j(A)\tilde{x}_j \leq kb(A)$$

となる．したがって，Step3 で直ちに停止しない場合でも Algorithm4 は条件 (4.6c) を満たす．

以上より，Algorithm4 より得られた解は系 4.1 を満たす． □

補題 4.3 Algorithm4 より得られる $\tilde{\boldsymbol{x}}$ は主問題 (4.4) の 0 と 1 のみからなる許容解であり，$(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}})$ は双対問題 (4.5) の許容解である．

証明 問題 (4.1) が実行可能ならば，$\boldsymbol{x} = (1, \ldots, 1)$ は主問題 (4.4) の許容解である．Algorithm4 は $\boldsymbol{x} = \boldsymbol{0}$ から始まり，Step1 で Forcing Hypergraph から辺を一つ選択し，Step2 でその辺に含まれる品物のうち少なくとも 1 つは選択している．したがって Step1 と Step2 で Forcing 制約を満たすように x_j を 0 から 1 へと変更していることがわかる．また，Step3 と Step4 では $\bar{b} = 0$ となるまで品物を選択していることから，ナップサック制約を満たすように x_j を 0 から 1 へと変更していることがわかる．したがって，$\tilde{\boldsymbol{x}}$ は主問題 (4.4) の 0 と 1 のみからなる許容解である．

さらに，$(\boldsymbol{y}, \boldsymbol{z}) = (\boldsymbol{0}, \boldsymbol{0})$ は双対問題の許容解である．Algorithm4 はこれを初期値とし，補題 4.2 で示したようにアルゴリズム全体を通して双対問題の実行可能性を維持している．したがって $(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}})$ は双対問題 (4.5) の許容解である． □

補題 4.4 Algorithm4 の計算量は $\mathcal{O}(k|\mathcal{E}| + |V|^2)$ である．

証明　Step2 の 1 回の反復での計算量は高々$\mathcal{O}(k)$ となる．また，反復回数は高々$\mathcal{O}(|\mathcal{E}|)$ なので，Step1 と Step2 全体の計算量は $\mathcal{O}(k|\mathcal{E}|)$ であることがわかる．

$a_j(S)$ と $s = \arg\min_{j\in V\setminus S}\left\{\frac{\bar{c}_j}{a_j(S)}\right\}$ の計算量は高々$\mathcal{O}(|V|)$ である．さらに，Step4 の反復回数は高々$|V|$ 回である．よって Step4 全体の計算量は $\mathcal{O}(|V|^2)$ となる．

以上より，Algorithm4 の計算量は $\mathcal{O}(k|\mathcal{E}| + |V|^2)$ である．　□

定理 4.1 Algorithm4 は MKPFH に対する k-近似アルゴリズムである．

証明　Algorithm4 から得られる $\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}$ は補題 4.1 を満たしている．

この時，問題 (4.1) の最適値を θ^*_{mkpfh} とおくと，目的関数値は系 4.1 より，

$$\sum_{j\in V} c_j\tilde{x}_j \leq k\theta^*_{mkpfh}$$

を満たす．また，最小化問題の性質から，

$$\theta^*_{mkpfh} \leq \sum_{j\in V} c_j\tilde{x}_j$$

が成り立つ．したがって，Algorithm4 より得られる解からなる目的関数値は最適値の k 倍以内の値になる．　□

5　数値実験

提案アルゴリズムを C++で実装したものと，数理計画ソルバー gurobi を比較する数値実験をそれぞれ MKPPC と MKPFH に対して行なった．実験環境は次の表 1 の通りである．

表 1: 実行環境

開発言語	ソルバー	CPU	メモリ	OS
C++14	gurobi7.0.2	4GHz Intel Core i7	16GB	macOS 10.12.5

また，全実験に共通する各入力パラメータの値は

$$\begin{aligned} a_j &\in [1, 20] && (\forall j \in V) \\ b &\in \left[\frac{4}{5}\sum_{j\in V} a_j, \sum_{j\in V} a_j\right] \\ c_j &\in [1, 20] && (\forall j \in V) \end{aligned}$$

とした．

5.1 MKPPC に対する 3-近似アルゴリズム

5.1.1 実験 1

実験1ではMKPPCにおける $\mathcal{P}$ を V の等分割により与える．分割の詳細としては，乱数で $[2, \frac{|V|}{2}]$ から $|V|$ をちょうど割りきれる整数 m を選択した．m と V を入力として Algorithm A を用いて $\mathcal{P}$ を決定した．

Algorithm A $\mathcal{P}$ の決定 (等分割)

Input: V, m

Output: $\mathcal{P}$

$\mathcal{P} \leftarrow \emptyset$
$i \leftarrow 1$
while $i < |V| + 1$ **do**
    $j \leftarrow 0$
    $P \leftarrow \emptyset$
    while $j < m$ **do**
        $P \leftarrow P \cup \{i\}$
        $i \leftarrow i + 1$
        $j \leftarrow j + 1$
    end while
    $\mathcal{P} \leftarrow \mathcal{P} \cup \{P\}$
end while

$|V| = 1000$ から $|V| = 5000$ まで，頂点数を 100 ずつ増加させ，各頂点数について問題を 20 問生成し，gurobi と Algorithm3 との実行時間の平均を取り，その比較を行った．実験結果を図 1 にまとめた．

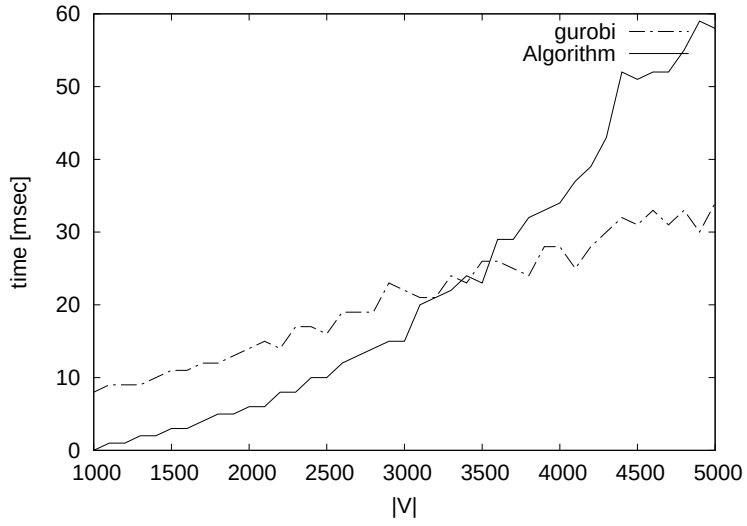

図 1: MKPPC に対する gurobi と Algorithm3 の実行時間の比較 ($\mathcal{P}$: 等分割)

図 1 より，頂点数が増加するにつれて計算時間が二次関数的に増加していることが分かる．この理由としては，補題 3.2 で示した計算量において，頂点数が増加すると $p|\mathcal{P}|$ は $|V|^2$ に対して無視できるほど小さくなるためと考えられる．

gurobi との実行時間の比較としては，$|V| = 3500$ までは Algorithm3 の方が早く終了していることが分かる．

また，gurobi により得られた解を厳密解とし，Algorithm3 により得られる近似解の近似率の確認を生成した 820 問に対して行った．その結果を図 2 にまとめた．

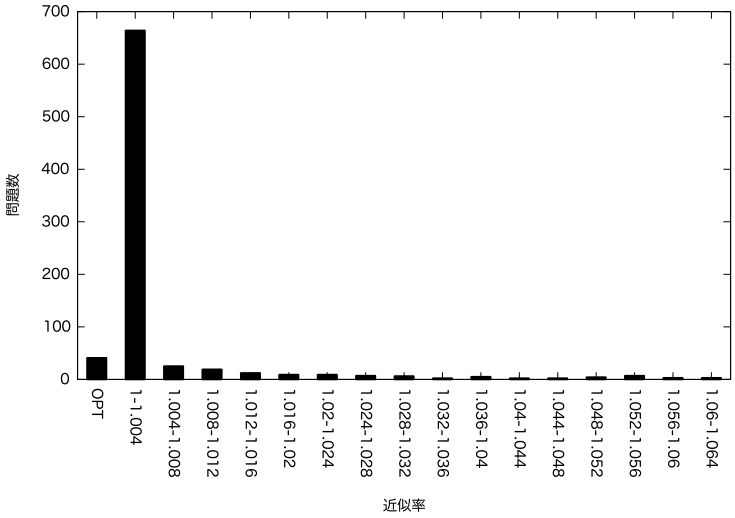

図 2: MKPPC に対する Algorithm3 の近似率 ($\mathcal{P}$: 等分割)

実験結果としては 820 問の問題全てが最適値の 1.064 倍以内の目的関数値が得られていることがわかる．また，多くの問題について最適値の 1.004 倍以内の目的関数値が得られており，高い精度で近似解が得られているのではないかと考えられる．

5.1.2 実験 2

実験 2 では MKPPC における $m = \mathcal{P}$ を V のランダムな分割により与える．分割の詳細としては， まず分割数 $m = |\mathcal{P}|$ を乱数で $[2, \frac{|V|}{4}]$ から整数値で決める．m と V を入力として Algorithm B を用いて $\mathcal{P}$ を決定した．

とすることにより．$\mathcal{P}$ を決定した．

具体的な実験内容は，実験 1 と同様である．実験結果を図 3 にまとめた．

図 3 より，実験 1 と同様に頂点数が増加するにつれて計算時間が二次関数的に増加していることが分かる．

gurobi との実行時間の比較としては，$|V| = 4100$ までは Algorithm3 の方が早く終了していることが分かる．

また，実験 1 と比較すると，gurobi の実行時間は実験 1 の方が早いが，Algorithm3 はどちらの実験でも実行時間に違いがないことが分かる．したがって，提案アルゴリズムでは頂点の分割によりパフォーマンスが左右されないことが確認できた．

Algorithm B $\mathcal{P}$ の決定 (ランダム分割)

Input: V, m

Output: $\mathcal{P}$

$\mathcal{P} \leftarrow \emptyset$
$p \leftarrow \{1\}$
$i \leftarrow 0$
while $i < m$ **do**
    $j \leftarrow [3, |V| - 1]$ の中から一様乱数で整数を 1 つ選択
    if j が任意の p の要素 a に対して $|a - j| > 1$ **then**
        $p \leftarrow p \cup \{j\}$
        $i \leftarrow i + 1$
    end if
end while
while $p \neq \emptyset$ **do**
    $P \leftarrow \emptyset$
    $k \leftarrow \min_{j \in p} j$
    $p \leftarrow p \setminus \{k\}$
    if $p \neq \emptyset$ **then**
        $l \leftarrow \min_{j \in p} j$
    else
        $l \leftarrow |V| + 1$
    end if
    while $k < l$ **do**
        $P \leftarrow P \cup \{k\}$
        $k \leftarrow k + 1$
    end while
    $\mathcal{P} \leftarrow \mathcal{P} \cup \{P\}$
end while

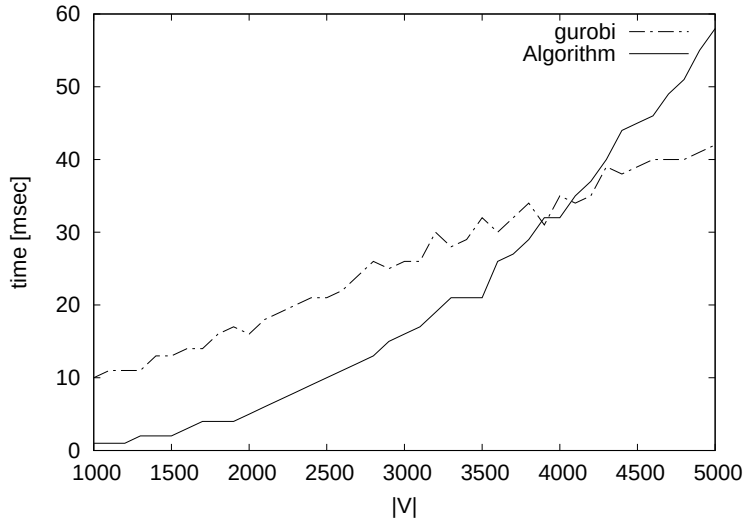

図 3: MKPPC に対する gurobi と Algorithm3 の実行時間の比較 ($\mathcal{P}$: ランダム分割)

また，gurobi により得られた解を厳密解とし，Algorithm3 により得られる近似解の近似率の確認を生成した 820 問に対して行行った．その結果を図 4 にまとめた．

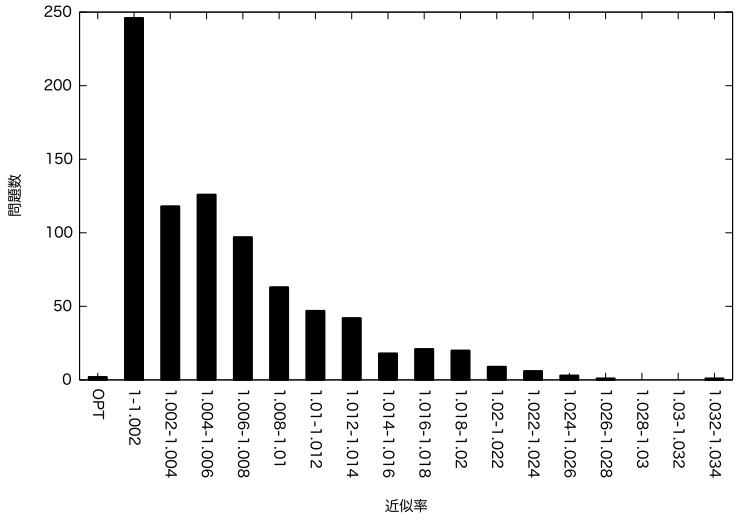

図 4: MKPPC に対する Algorithm3 の近似率 ($\mathcal{P}$: ランダム分割)

実験結果としては 820 問全てについて最適値の 1.034 倍以内の目的関数値が得られたことがわかる．また，多くの問題について最適値の 1.01 倍以内の目的関数値が得られており，実験 1 と同様に高い精度で近似解が得られているのではないかと考えられる．

5.2 MKPFH に対する k-近似アルゴリズム

5.2.1 実験 3

実験 3 では Algorithm4 についての数値実験を行う．

$\mathcal{E}$ については，まず $m = |\mathcal{E}|$ を乱数で $[2, \frac{n}{2}]$ から整数値で決める．m と V を入力として，Algorithm C を用いて $\mathcal{E}$ を決定した．

Algorithm C $\mathcal{E}$ の決定

Input: V, m
Output: $\mathcal{E}$

```
1: E ← ∅
2: i ← 0
3: while i < m do
4:     E ← ∅
5:     for j ∈ V do
6:         k ← [0, 1] から一様乱数で値を 1 つ選択
7:         if k < 0.3 then
8:             E ← E ∪ {j}
9:         end if
10:    end for
11:    E ← E ∪ {E}
12:    i ← i + 1
13: end while
```

$|V| = 1000$ から $|V| = 10000$ まで，頂点数を 100 ずつ増加させ，各頂点数について問題を 20 問生成し，gurobi と Algorithm4 との実行時間の平均を取り，その比較を行った．実験結果を図 5 にまとめた．

MKPFH は集合被覆問題 [1] を部分問題として持つ．この問題も NP 困難であることが知られている，したがって gurobi は厳密解を得られているが，MKPFH を計算機によって効率的に厳密解を求めることは難しいため頂点数が増加するにつれて実行時間が大幅に増加している．よって Algorithm4 により，現実的な時間で近似解を求めることができていることがわかる．

また，図 6 は Algorithm4 のみについて実験結果をプロットした図である．

これより，Algorithm4 においても実験 1,2 と同様に $|V|$ が増加すると Algorithm4 の実行時間は二次関数的に増加していることが分かる．この理由としては，補題 4.4 で示した計算量において，頂点数が増加すると $k|\mathcal{E}|$ は $|V|^2$ に対して無視できるほど小さくなるためと考えられる．

また，gurobi により得られた解を厳密解とし，Algorithm4 により得られる近似解の近似率の確認を生成した 1820 問について行った．その結果を図 7 にまとめた．

実験結果としては，1820 問全てについて最適値の 1.013 倍以内の目的関数値が得られていることがわかる．また，多くの問題について最適値の 1.004 倍以内の目的関数値が得られており，非常に高い精度で近似解が得られているのではないかと考えられる．

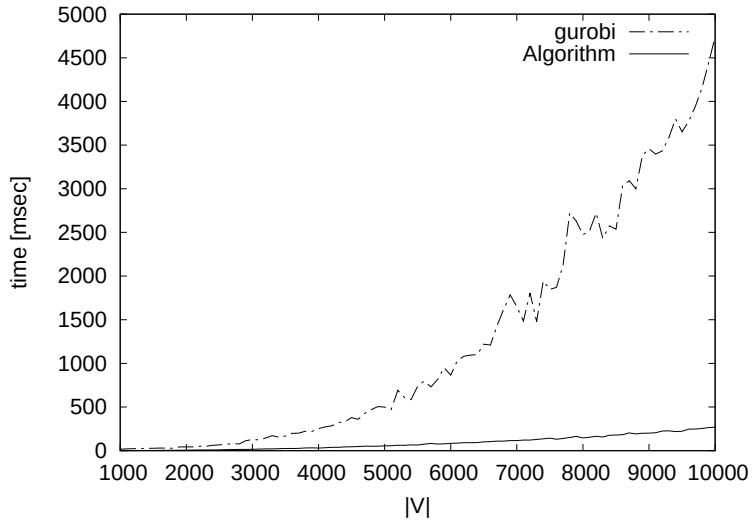

図 5: MKPFH に対する gurobi と Algorithm4 の実行時間の比較

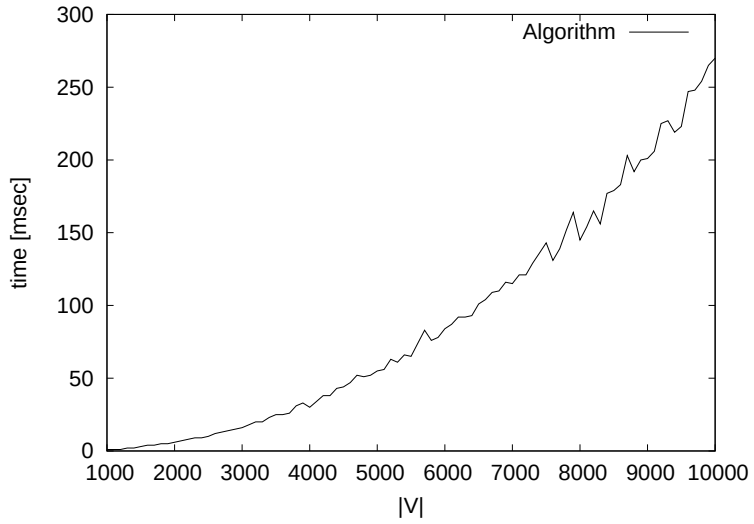

図 6: Algorithm4 の実行時間

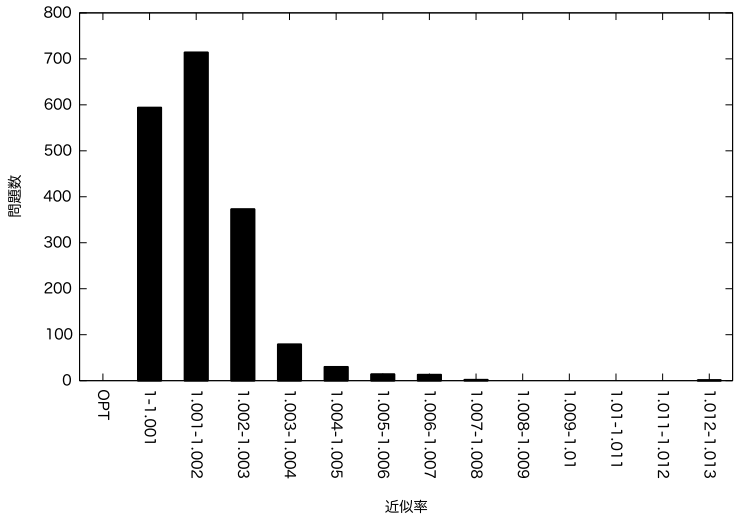

図 7: MKPFH に対する Algorithm4 の近似率

6 終わりに

本研究では最小化ナップサック問題に集合から少なくとも 1 つは選択するという制約を加えた問題についての近似アルゴリズムの提案を行った，これらの問題は商品などの選択において重要な制約となると思われる．

MKPFH に関しては集合カバー問題を部分問題に持つため，現状より良い近似率にすることは難しいと考えられる．しかし，MKPPC については解析により近似率を 2 に近づけることは可能でないかと考えている．したがってこれは今後の課題としたい．

また MKPPC と MKPFH の中間の問題として，最小化ナップサック問題に Partition 制約を緩和し，高々 1 つの重なりを許す制約を加えた問題を考えることもできる．この問題については主双対法を用いることで高々1 つの重なりを許す集合被覆問題の 2-近似アルゴリズムを考えることができる．したがって MKPPC の 3-近似アルゴリズムの設計と同様にして，最小化ナップサック問題に対する 2-近似アルゴリズムと合わせて 4-近似アルゴリズムを考えることができるのではないかと思われる．

今回の研究では，最小化ナップサック問題を中心として進めてきたが，今後の研究ではさらに別の問題を基軸とした近似アルゴリズムの設計を行っていきたい．

7 謝辞

本研究を進めるにあたり，ご指導を頂いた指導教員の村松正和教授に深謝の意を表する．また，助言をいただきました岡本吉央教授，高橋里司助教と村松研究室，高橋研究室の皆様にも感謝の意を表する．

参考文献

[1] B. コルテ，J. フィーゲン 著，浅野孝夫，浅野泰仁，小野孝男，平田富夫 訳: 組合せ最適化，シュプリンガージャパン (2009)

[2] 日本オペレーションズ・リサーチ学会 編: OR 用語辞典，日科技連 (2000)

[3] David P. Williamson，David B. Shmoys 著，浅野孝夫 訳: 近似アルゴリズムデザイン，共立出版 (2015)

[4] V. V. ヴァジラーニ 著，浅野孝夫 訳: 近似アルゴリズム，シュプリンガージャパン (2002)

[5] 山崎論，小市俊悟，鈴木敦夫: 災害時の代替経路の確保を考慮した道路ネットワーク構築法，日本オペレーションズ・リサーチ学会和文論文誌 Vol. 56 (2013)，pp. 31-52

[6] 一森哲男，森口聡子: フィードバックのある資源配分問題，日本オペレーションズ・リサーチ学会和文論文誌 Vol. 48 (2005)，pp. 1-11

[7] Michael R. Garey，David S. Jphnson: COMPUTERS AND INTRACTABILITY A Guide to the Theory of NP-Completeness，Freedman (1979)

[8] 田村明久，村松正和: 最適化法，共立出版 (2002)

[9] Robert D. Carr，Lisa K. Fleischer，Vitus J. Leung，Cynthia A. Phillips: Strengthening Integrality Gaps for Capacitated Network Design and Covering Problems，Proceedings of the 11th Annual ACM-SIAM Symposium on Discreate Algorithms (2000)，pp. 106-115

[10] J. マトウシェク，J. ネシェトリル 著，根上生也，中本敦浩 訳: 離散数学への招待 下，丸善出版 (2012)

付録 A　疑似コード

Algorithm 1 最小化ナップサック問題の 2-近似アルゴリズム

Input: $V, a_j, c_j (j \in V)$ and b.

Output: $\tilde{\boldsymbol{x}}$ and $\tilde{\boldsymbol{y}}$

$\boldsymbol{x} \leftarrow \boldsymbol{0}$
$\boldsymbol{y} \leftarrow \boldsymbol{0}$
$S \leftarrow \emptyset$
$\bar{b} \leftarrow b$
for $j \in V$ **do**
    $\bar{c}_j \leftarrow c_j$
end for
while $\bar{b} > 0$ **do**
    for $j \in V \setminus S$ **do**
        $a_j(S) \leftarrow \min\{a_j, \bar{b}\}$
    end for
    $s \leftarrow \underset{j \in V \setminus S}{\arg\min} \left\{ \dfrac{\bar{c}_j}{a_j(S)} \right\}$
    $y(S) \leftarrow \frac{\bar{c}_s}{a_s(S)}$
    $x_s \leftarrow 1$
    $S \leftarrow S \cup \{s\}$
    for $j \in V \setminus S$ **do**
        $\bar{c}_j \leftarrow \bar{c}_j - a_j(S) y(S)$
    end for
    $\bar{b} \leftarrow \bar{b} - a_s$
end while
$\tilde{\boldsymbol{x}} \leftarrow \boldsymbol{x}$
$\tilde{\boldsymbol{y}} \leftarrow \boldsymbol{y}$

Algorithm 2 問題 (3.2) の厳密解法

Input: $V, \mathcal{P}, c_j (j \in V)$

Output: $\tilde{\boldsymbol{x}}$

$\boldsymbol{x} \leftarrow \boldsymbol{0}$
for $P \in \mathcal{P}$ **do**
    $s = \arg\min_{j \in P} c_j$
    $x_s = 1$
end for
$\tilde{\boldsymbol{x}} \leftarrow \boldsymbol{x}$

Algorithm 3 MKPPC の 3-近似アルゴリズム

Input: $V, \mathcal{P}, \boldsymbol{a}, b, \boldsymbol{c}$

Output: $\tilde{S}$

$S_1 \leftarrow \emptyset$
$S_2 \leftarrow \emptyset$
for $P \in \mathcal{P}$ **do**
    $s = \arg\min_{j \in P} c_j$
    $S_1 \leftarrow S_1 \cup \{s\}$
end for
$\bar{b} \leftarrow b - \sum_{j \in S_1} a_j$
for $j \in V \setminus S_1$ **do**
    $\bar{c}_j \leftarrow c_j$
end for
while $\bar{b} > 0$ **do**
    for $j \in V \setminus \{S_1 \cup S_2\}$ **do**
        $a_j(S_2) \leftarrow \min\{a_j, \bar{b}\}$
    end for
    $s \leftarrow \arg\min_{j \in V \setminus S_2} \left\{ \dfrac{\bar{c}_j}{a_j(S_2)} \right\}$
    $S_2 \leftarrow S_2 \cup \{s\}$
    $\bar{b} \leftarrow \bar{b} - a_s$
end while
$\tilde{S} \leftarrow S_1 \cup S_2$

Algorithm 4 MKPFH の k-近似アルゴリズム

Input: $H = (V, \mathcal{E}), a_j, c_j (j \in V)$ and b.

Output: $\tilde{\boldsymbol{x}}$ and $(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}})$

$\boldsymbol{x} \leftarrow \boldsymbol{0}$
$(\boldsymbol{y}, \boldsymbol{z}) \leftarrow (\boldsymbol{0}, \boldsymbol{0})$
$S \leftarrow \emptyset$
$\bar{\mathcal{E}} \leftarrow \mathcal{E}$
$\bar{b} \leftarrow b$
for $j \in V$ **do**
    $\bar{c}_j \leftarrow c_j$
end for
while $\bar{\mathcal{E}} \neq \emptyset$ **do**
    $\min_{E \in \bar{\mathcal{E}}} |E|$ となる E を選択
    if $\sum_{j \in E} x_j \geq 1$ **then**
        $\bar{\mathcal{E}} \leftarrow \bar{\mathcal{E}} \setminus \{E\}$
    else if $\sum_{j \in E} x_j = 0$ **then**
        $s \leftarrow \arg\min_{j \in E} \{\bar{c}_j\}$
        $z_E \leftarrow \bar{c}_s$
        $x_s \leftarrow 1$
        $S \leftarrow S \cup \{s\}$
        $\bar{\mathcal{E}} \leftarrow \bar{\mathcal{E}} \setminus \{E\}$
        for $j \in E$ **do**
            $\bar{c}_j \leftarrow \bar{c}_j - z_E$
        end for
        $\bar{b} \leftarrow \bar{b} - a_s$
    end if
end while
while $\bar{b} > 0$ **do**
    for $j \in V \setminus S$ **do**
        $a_j(S) \leftarrow \min\{a_j, \bar{b}\}$
    end for
    $s \leftarrow \arg\min_{j \in V \setminus S} \left\{ \dfrac{\bar{c}_j}{a_j(S)} \right\}$
    $y(S) \leftarrow \frac{\bar{c}_s}{a_s(S)}$
    $x_s \leftarrow 1$
    $S \leftarrow S \cup \{s\}$
    for $j \in V \setminus S$ **do**
        $\bar{c}_j \leftarrow \bar{c}_j - a_j(S) y(S)$
    end for
    $\bar{b} \leftarrow \bar{b} - a_s$
end while
$\tilde{\boldsymbol{x}} \leftarrow \boldsymbol{x}$
$(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}}) \leftarrow (\boldsymbol{y}, \boldsymbol{z})$