平成 29 年度電気通信大学情報・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-近似アルゴリズムについて

指導教員　村松 正和 教授

平成 30 年 1 月 31 日

電気通信大学情報・通信工学科

木村 優貴

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．そのような問題に対して用いられるアプローチは 2 つある．

多項式時間で終了する保証はないが，最適解を導出するようなアプローチである．この手法については切除平面法 [1]，分枝切除法 [2] などが研究されている．しかし，これら手法での問題点としては問題のサイズによっては最適解を求めるために膨大な時間がかかるということが挙げられる．

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．離散最適化問題に対する近似アルゴリズムとは，最適値に十分に近い解を求めるアルゴリズムである [3][4].

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ナップサック問題の似た問題で，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，詰め込む品物の重さの総和を最小化することを目的とした問題である [3]. この問題もナップサック問題と同様に 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，クラスタの役割を持つ集合を与え制約とする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]．ここで例として

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

を挙げる．この節ではこの問題 (2.1) に対する主双対法について説明する．ただし，$(a_{ij}) \in \mathbb{R}_{++}^{n \times m}, \boldsymbol{b} \in \mathbb{R}_{++}^{m}, \boldsymbol{c} \in \mathbb{R}_{++}^{n}, \sum_{j=1}^{n} a_{ij} \geq b_i (i = 1, \ldots, m)$ とする．

この手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

$$\begin{aligned} \min \quad & \boldsymbol{c}^T \boldsymbol{x} \\ \text{s.t.} \quad & A\boldsymbol{x} \geq \boldsymbol{b} \\ & x_j \geq 0 \quad (j = 1, \ldots, n) \end{aligned}$$

とその双対問題

$$\begin{aligned} \min \quad & \boldsymbol{b}^T \boldsymbol{y} \\ \text{s.t.} \quad & A^T \boldsymbol{y} \geq \boldsymbol{c} \\ & y_i \geq 0 \qquad (i = 1, \ldots, m) \end{aligned}$$

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

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

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]．

離散最適化問題に対する主双対法とは整数計画問題を線形計画問題へと緩和 (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られるようなアルゴリズムを構築することである．

問題 (2.1) の LP 緩和は

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

となる．この問題の双対問題は，

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

となる．この 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，その共通の最適値を 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

$$\bar{x}_j > 0 \implies \sum_{i=1}^{m} a_{ij} \bar{y}_i \geq c_j / \alpha \tag{2.4}$$

$$\bar{y}_i > 0 \implies \sum_{j=1}^{n} a_{ij} \bar{x}_j \leq \beta b_i \tag{2.5}$$

を満たすならば，

$$\sum_{j=1}^{n} c_j \bar{x}_j \leq \alpha\beta\theta \tag{2.6}$$

が成立する．

証明 関係 (2.4) より，

$$\bar{x}_j > 0 \implies \alpha \sum_{i=1}^{m} a_{ij} \bar{y}_i \geq c_j$$

が得られる．これと式 (2.5) を用いると式 (2.6) の左辺は

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

となる．双対問題は最大化問題であることから，

$$\sum_{i=1}^{m} b_i \bar{y}_i \leq \theta$$

なので，

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

が得られる． □

よって補題 2.1 を満たすような $\bar{\boldsymbol{x}}$ 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して主双対法を用いるためには LP 緩和を行う必要がある．

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，

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

$$\text{s.t.} \sum_{j \in V} a_j x_j \geq b \tag{2.8}$$

$$x_j \in \{0, 1\} \qquad \forall j \in V \tag{2.9}$$

と表現される．この時 $\boldsymbol{a}, b, \boldsymbol{c}$ はそれぞれ $\boldsymbol{a}, \boldsymbol{c} \in \mathbb{Z}^n_{++}, b \in \mathbb{Z}_{++}, a_j < b (\forall j \in V), \sum_{j \in V} a_j > b$ である．式 (2.7) 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 (2.8) 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 (2.9) はバイナリ制約と呼ばれる．

2.3.1 LP 緩和と双対問題

この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．
整数計画問題を LP 緩和すると一般に元問題の最適値と緩和問題の最適値がかけ離れてしまう整数ギャップが生じてしまう．Carr ら [9] は品物の部分集合 $A \subseteq V$ に対して新しい制約

$$\text{s.t.} \quad \sum_{j \in V \setminus A} a_j(A) x_j \geq b(A) \tag{2.10}$$

を導入した．ただし，

$$b(A) \quad = \max\{0, b - \sum_{j \in A} a_j\} \quad \forall A \subseteq V \tag{2.11}$$

$$a_j(A) \quad = \min\{a_j, b(A)\} \quad \forall A \subseteq V, \forall j \in V \setminus A \tag{2.12}$$

である．この制約は部分集合 A に含まれているような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ような制約である．

さらに Carr らは最小化ナップサック問題の LP 緩和を，制約 (2.10) 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，

$$\begin{aligned} \min \quad & \sum_{j \in V} c_j x_j & & \\ \text{s.t.} \quad & \sum_{j \in V \setminus A} a_j(A) x_j \geq b(A) & \forall A \subseteq V & \\ & x_j \geq 0 & \forall j \in V & \end{aligned} \tag{2.13}$$

とすることにより，整数ギャップを小さくした．この問題の双対問題は

$$\begin{aligned} \max \quad & \sum_{A \subseteq V} b(A) y(A) & & \\ \text{s.t.} \quad & \sum_{A \subseteq V : j \notin A} a_j(A) y(A) \leq c_j & \forall j \in V & \\ & y(A) \geq 0 & \forall A \subseteq V & \end{aligned} \tag{2.14}$$

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．

問題 (2.7)〜(2.9) の許容解は LP 緩和した問題 (2.13) の許容解でもある．また，問題 (2.7)〜(2.9) の最適値を θ^*_{mkp}，問題 (2.13) の最適値を θ_{mkp} とすると，

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

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 (2.13) とその双対問題 (2.14) を用いて主双対法によるアルゴリズムの構築を行う．その準備として，以下の補題を示す．

補題 2.2 主問題 (2.13) の許容解を $\bar{\boldsymbol{x}}$ とし，双対問題 (2.14) の許容解を $\bar{\boldsymbol{y}}$ とする．問題 (2.13) と問題 (2.14) の共通の最適値を θ_{mkp} とする．もし

$$\bar{x}_j > 0 \implies \sum_{A \subseteq V : j \notin A} a_j(A) \bar{y}(A) = c_j \tag{2.15}$$

$$\bar{y}(A) > 0 \implies \sum_{j \in V \setminus A} a(A)_j \bar{x}_j \leq 2b(A) \tag{2.16}$$

が満たされるならば，

$$\sum_{j \in V} c_j \bar{x}_j \leq 2\theta_{mkp} \tag{2.17}$$

が成り立つ．

証明　式 (2.15) と式 (2.16) を用いると式 (2.17) の左辺は

$$\begin{aligned}\sum_{j\in V} c_j\bar{x}_j &= \sum_{j\in V}\sum_{A\subseteq V: j\notin A} a_j(A)\bar{x}_j\bar{y}(A)\\ &= \sum_{A\subseteq V}\sum_{j\in V\setminus A} a_j(A)\bar{x}_j\bar{y}(A)\\ &\leq 2\sum_{A\subseteq V} b(A)\bar{y}(A)\end{aligned}$$

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，

$$\sum_{A\subseteq V} b(A)\bar{y}(A) \leq \theta_{mkp}$$
$$2\sum_{A\subseteq V} b(A)\bar{y}(A) \leq 2\theta_{mkp}$$

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 (2.13) の 0 と 1 のみからなる許容解を $\tilde{\boldsymbol{x}}$，問題 (2.14) の許容解を $\tilde{\boldsymbol{y}}$ とする．これらが補題 2.2 の条件を満たすとする．この時，問題 (2.7)〜(2.9) の最適値を θ^*_{mkp} とすると．

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

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 を満たすような解 $\tilde{\boldsymbol{x}}$ と $\tilde{\boldsymbol{y}}$ を得るような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．

Input: V: 品物の集合，$\boldsymbol{a}$: 品物の価値のベクトル，b: 需要，$\boldsymbol{c}$: 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，$S=\emptyset(S=\{s\mid x_s=1\}), \bar{b}=b, \bar{c}_j=c_j(\forall j\in V)$ 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: $y(S)=\dfrac{\bar{c}_s}{a_s(S)}, x_s=1$，$S=S\cup\{s\}$，$\bar{b}=b-a_s$，$\bar{c}_j=\bar{c}_j-a_jy(S)(\forall j\in V\setminus S)$ とし，Step1 の初めに戻る．

アルゴリズムの詳細を付録 A の Algorithm2 に示す.

補題 2.3 Algorithm2 より得られた解は補題 2.2 の条件 (2.15) と条件 (2.16) を満たす.

証明 Algorithm2 は $\boldsymbol{x} = \boldsymbol{0}, \boldsymbol{y} = \boldsymbol{0}$ から始まり，$x_j = 1$ となるのは

$$a_j(S)y(S) = c_j - \sum_{A \subseteq S \setminus \{j\}} a_j(A)y(A)$$

とした時のみである．よって $x_j > 0$ となるのは $x_j = 1$ の時のみであり，この時

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

を満たすことがわかる.

$\tilde{S} = \{j \in V | \tilde{x}_j = 1\}$ とする．また，$\tilde{x}_l$ を Algorithm2 の最後に 0 から 1 へと更新された変数とする．ここで $\tilde{y}(A) > 0$ であるような A について，Algorithm2 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,

$$\sum_{j \in \tilde{S} \setminus \{l\}} a_j < b \tag{2.18}$$

である．式 (2.12) より $a_j(A) \leq a_j$ なので，$\tilde{y}(A) > 0$ であるような 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 (2.11) より $b(A) \geq b - \sum_{j \in A} a_j$ なので，これと式 (2.18) 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 (2.12) 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，Algorithm2 は条件 (2.16) を満たす.

以上より，Algorithm2 より得られた解は補題 2.2 の条件 (2.15) と条件 (2.16) を満たす. □

補題 2.4 問題 (2.7)〜(2.9) が実行可能ならば，$\boldsymbol{x} = (1, \ldots, 1)$ は主問題 (2.13) の許容解である.

証明　問題 (2.7)〜(2.9) 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，(2.11) 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 2.5　Algorithm2 から得られた $\tilde{\boldsymbol{x}}$ は主問題 (2.13) の 0 と 1 からなる許容解であり，$\tilde{\boldsymbol{y}}$ は双対問題 (2.14) の許容解である．

証明　補題 2.4 より，問題 (2.7)〜(2.9) が実行可能ならば，$\boldsymbol{x} = (1, \ldots, 1)$ は主問題 (2.13) の許容解である．Algorithm2 は $\boldsymbol{x} = \boldsymbol{0}$ から始まり，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．したがって，$\tilde{\boldsymbol{x}}$ は主問題 (2.13) の 0 と 1 のみからなる許容解である．

さらに，$\boldsymbol{y} = \boldsymbol{0}$ は双対問題の許容解である．Algorithm2 はこれを初期値とし，補題 2.3 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 (2.14) の許容解である．　□

定理 2.1　Algorithm2 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　Algorithm2 から得られる $\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}$ は補題 2.3 を満たしている．

この時，最小化ナップサック問題 (2.7)〜(2.9) 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，Algorithm2 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 を分割した集合 $\mathcal{P}$ を以下のように定義する: $\mathcal{P} \subseteq 2^V$，任意の 2 つの要素 $P_1, P_2 \in \mathcal{P}(P_1 \neq P_2)$ は互いに素である．

最小化ナップサック問題に $\mathcal{P}$ を受け取った問題

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

$$\text{s.t.} \sum_{j \in V} a_j x_j \geq b \tag{3.2}$$

$$\sum_{j \in P} x_j \geq 1 \qquad \forall P \in \mathcal{P} \tag{3.3}$$

$$x_j \in \{0, 1\} \qquad \forall j \in V \tag{3.4}$$

を考える．制約 (3.3) 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し，Partition 制約と呼ぶ．この問題を Partition 制約付き最小化ナップサック問題 (Minumum Knapsack Problem with Partition Constraint: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

$$\begin{aligned} \min \quad & \sum_{j \in V} c_j x_j \\ \text{s.t.} \quad & \sum_{j \in P} x_j \geq 1 \quad \forall P \in \mathcal{P} \\ & x_j \in \{0, 1\} \quad \forall j \in V \end{aligned} \tag{3.5}$$

と，

$$\begin{aligned} \min \quad & \sum_{j \in V} c_j x_j \\ \text{s.t.} \quad & \sum_{j \in V} a_j x_j \geq b \\ & x_j \in \{0, 1\} \quad \forall j \in V \end{aligned} \tag{3.6}$$

を考える．

問題 (3.5) は Partition 制約 (3.3) を持った問題であり，問題 (3.6) はナップサック制約 (3.2) を持った問題である．

前章により，問題 (3.6) 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 (3.5) に対する厳密解法

問題 (3.5) は集合カバー問題の部分問題である．しかし，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，任意の $P \in \mathcal{P}$ に対して $\arg\min_{j \in P} c_j$ を計算すること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．

Algorithm 1 問題 (3.5) の厳密解法

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

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

1: $\boldsymbol{x} \leftarrow \boldsymbol{0}$
2: **for** $P \in \mathcal{P}$ **do**
3: $\quad s = \arg\min_{j \in P} c_j$
4: $\quad x_s = 1$
5: **end for**
6: $\tilde{\boldsymbol{x}} \leftarrow \boldsymbol{x}$

3.3 MKPPC の 3-近似アルゴリズム

問題 (3.5) と問題 (3.6) に対するアルゴリズムを用いて，MKPPC の 3-近似アルゴリズムの設計を行った．アルゴリズムをここに示す．

Input: V: 品物の集合，$\mathcal{P}$: 品物を分割した集合，$\boldsymbol{a}$: 品物の価値の集合，b: ナップサック内に保ちたい価値，$\boldsymbol{c}$: 品物の重さの集合

Output: $\tilde{S}$: 選択した品物の集合

Step0: $S_1 = S_2 = \emptyset$ とする．

Step1: 問題 (3.5) を満たすように Algorithm1 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 $V' = V \backslash S_1$ と $\bar{b}$ で問題 (3.6) を構成．Algorithm2 を適用．$S_2 = \{j \mid x_j = 1\}$ とする．

Step3: $\tilde{S} = S_1 \cup S_2$ とし，解を出力．終了

アルゴリズムの詳細を付録 A の Algorithm3 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，$\sum_{j \in V} c_j x_j \leq 3\theta$ を満たす．

証明 Algorithm3 より得られた解は問題 (3.5) と問題 (3.6) 両方の制約を満たす．また，アルゴリズムは $\boldsymbol{x} = \boldsymbol{0}$ から始まり，2 つの制約を満たすように x_s を 0 から 1 へと変更している，したがってこの解は MKPPC の 0-1 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．ここで S_1 と S_2 はそれぞれ Algorithm1 と Algorithm2 により得られた解なので，問題 (3.5) の最適値を θ_1，問題 (3.6) の最適値を θ_2 とおくと，問題 (3.1)〜(3.4) は問題 (3.5) にナップサック制約を加えたものなので，$\theta_1 \leq \theta$．同様の理由から $\theta_2 \leq \theta$ 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

この章では 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 [11]．

これ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 $|E| \geq 2$．

最小化ナップサック問題に対し Forcing Hypergraph を受け取った問題

$$\begin{aligned}\min \quad & \sum_{j\in V} c_j x_j \\ \text{s.t.} \quad & \sum_{j\in V} a_j x_j \geq b \\ & \sum_{j\in E} x_j \geq 1 \qquad \forall E \in \mathcal{E} \\ & x_j \in \{0,1\} \qquad \forall j \in V\end{aligned} \tag{4.1}$$

を考える．この問題は最小化ナップサック問題に，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ことを意味する Forcing 制約を加えた問題である．この問題を 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 緩和と双対問題

問題 (4.1) の LP 緩和は，整数ギャップを小さくする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

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

を導入し，

$$\begin{aligned} \min \quad & \sum_{j \in V} c_j x_j & & \\ \text{s.t.} \quad & \sum_{j \in V \setminus A} a_j(A) x_j \geq b(A) & & \forall A \subseteq V \\ & \sum_{j \in E} x_j \geq 1 & & \forall E \in \mathcal{E} \\ & x_j \geq 0 & & \forall j \in V \end{aligned} \tag{4.2}$$

と表される．ただし，

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

$$a_j(A) = \min\{a_j, b(A)\} \quad \forall A \subseteq V, \forall j \in V \setminus A \tag{4.4}$$

とする．さらに問題 (4.2) の双対問題は

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

と表される．

4.4 主双対法

式 (4.2) 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 (4.2) 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 (4.2) と問題 (4.5) の共通の最適値を θ_{mkpfh} とする．もし，

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

$$\bar{z}_E > 0 \implies \sum_{j \in E} \bar{x}_j \leq k \tag{4.7}$$

$$\bar{y}(A) > 0 \implies \sum_{j \in V \setminus A} a_j(A)\bar{x}_j \leq kb(A) \tag{4.8}$$

が満たされるならば,

$$\sum_{j \in V} c_j \bar{x}_j \leq k\theta_{mkpfh} \tag{4.9}$$

が成り立つ．

証明 式 (4.6)〜(4.8) を用いると，式 (4.9) の左辺は

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

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，

$$\sum_{A \subseteq V} b(A)\bar{y}(A) + \sum_{E \in \mathcal{E}} \bar{z}_e \leq \theta_{mkpfh}$$

$$k \left\{ \sum_{A \subseteq V} b(A)\bar{y}(A) + \sum_{E \in \mathcal{E}} \bar{z}_e \right\} k \leq \theta_{mkpfh}$$

が成り立つので,

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

が得られる． □

系 4.1 問題 (4.2) の 0 と 1 のみからなる許容解を $\tilde{\boldsymbol{x}}$，問題 (4.5) の許容解を $(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}})$ とする．これらが補題 4.1 の条件を満たすとする．この時，問題 (4.1) の最適値を θ^*_{mkpfh} とすると．

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

が成り立つ．

4.5 MKPFH に対する k-近似アルゴリズム

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

Input: $H = (V, \mathcal{E})$(V: 品物の集合，$\mathcal{E}$: 品物のクラスタの集合)，$\boldsymbol{a}$: 品物の価値の集合，b: 需要，$\boldsymbol{c}$: 品物の重さの集合

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 $\min_{E \in \bar{\mathcal{E}}} |E|$ となるような E を 1 つ選択し，$\bar{E}$ とする．$\bar{\mathcal{E}} = \bar{\mathcal{E}} \backslash \{\bar{E}\}$ 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，$S = S \backslash \{s\}$，$\bar{b} = b - a_s$，$\bar{c}_j = \bar{c}_j - z_E (\forall j \in E)$ 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: $s = \arg\min_{j \in V \backslash S} \left\{ \dfrac{\bar{c}_j}{a_j(S)} \right\}$ を計算し，$y(S) = \frac{\bar{c}_s}{a_s(S)}$，$x_s = 1$ とする．さらに，$S = S \backslash \{s\}$，$\bar{b} = b - a_s$，$\bar{c}_j = \bar{c}_j - a_j y(S) (\forall j \in V \backslash S)$ とし，Step3 の初めに戻る．

アルゴリズムの詳細を付録 A の Algorithm4 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)〜(4.8) を満たす．

証明 Algorithm4 は $\boldsymbol{x} = \boldsymbol{0}, (\boldsymbol{y}, \boldsymbol{z}) = (\boldsymbol{0}, \boldsymbol{0})$ から始まり，16 行目と 31 行目から Step1 では $z_E = c_j$，Step2 では $a_j(A)y(A) = c_j - \sum_{E \in \mathcal{E}: j \in E} z_E$ とした時に $x_j = 1$ としている．したがって条件 (4.6) において $x_j > 0$ であるならば $\sum_{A \subseteq V: j \notin A} a_j(A)y(A) + \sum_{E \in \mathcal{E}: j \in E} z_E = c_j$ を満たすことがわかる．

次に条件 (4.7) についてだが，Algorithm4 中で $x_j (\forall j \in V)$ 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 (2.5) を満たしている．

最後に条件 (4.8) だが，これについては Algorithm4 が Step1 から Step2 へ移行した時に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 Step2 を 1 度も反復を行わなかった場合を考える．この時は Step2 で $y(A)$ が操作されないので $y(A) = 0 (\forall A \subseteq V)$ となる．したがってこの時 Algorithm4 は条件 (4.8) を満たす．

次に Step2 で 1 回以上反復をした場合について考える．$\tilde{S} = \{j \in V | \tilde{x}_j = 1\}$ 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 $\tilde{y}(A) > 0$ であるような 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

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

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，

$$\sum_{j \in \tilde{S} \backslash \{l\}} a_j < b \tag{4.10}$$

である．式 (4.4) より $\tilde{y}(A) > 0$ であるような A について，

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

であり，式 (4.3) と式 (4.10) から，

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

であるので，

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

となることがわかる．また，式 (4.4) より，$a_j(A) \leq b(A)$ なので，

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

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，

$$\sum_{j \in V \backslash A} a_j(A) \tilde{x}_j \leq k b(A)$$

となる．したがって，Step2 で直ちに停止しない場合でも Algorithm4 は条件 (4.8) 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 (4.2) 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 (4.2) 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 (4.2) 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 $s = \arg\min_{j \in V \setminus S} \left\{ \dfrac{\bar{c}_j}{a_j(S)} \right\}$ 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ような整数mを選択した．これとVを入力として

Input: V, m
Output: $\mathcal{P}$
1: $\mathcal{P} \leftarrow \emptyset$
2: $i \leftarrow 1$
3: **while** $i < |V| + 1$ **do**
4: $\quad j \leftarrow 0$
5: $\quad P \leftarrow \emptyset$
6: $\quad$ **while** $j < m$ **do**
7: $\quad\quad P \leftarrow P \cup \{i\}$
8: $\quad\quad i \leftarrow i + 1$
9: $\quad\quad j \leftarrow j + 1$
10: $\quad$ **end while**
11: $\quad \mathcal{P} \leftarrow \mathcal{P} \cup \{P\}$
12: **end while**

とすることにより，$\mathcal{P}$を決定した．

$|V| = 1000$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．

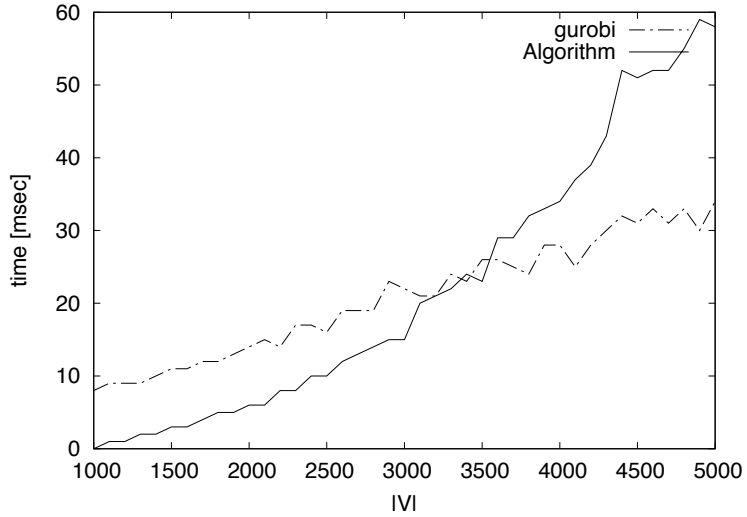

図 1: MKPPC に対する gurobi と Algorithm3 の実行時間の比較 ($\mathcal{P}$: 等分割)

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．

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．これと V を入力として，

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**
    $P \leftarrow \emptyset$
    $k \leftarrow \min_{j \in p} j$
    $p \leftarrow p \backslash \{k\}$
    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

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 2 にまとめた．図 2 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．

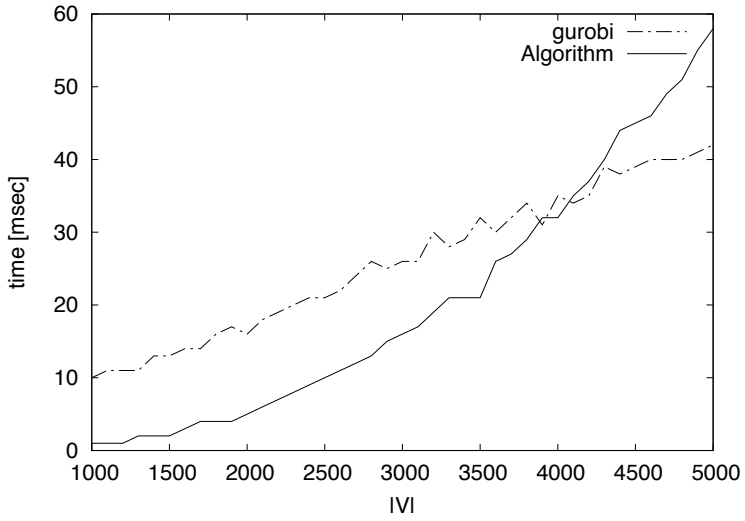

図 2: MKPPC に対する gurobi と Algorithm3 の実行時間の比較 ($\mathcal{P}$: ランダム分割)

また，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．

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．これと V を入力として，

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

$\mathcal{E} \leftarrow \emptyset$
$i \leftarrow 0$
while $i < m$ **do**
    $E \leftarrow \emptyset$
    for $j \in V$ **do**
        $k \leftarrow [0, 1]$ から一様乱数で値を 1 つ選択
        if $k < 0.3$ **then**
            $E \leftarrow E \cup \{j\}$
        end if
    end for
    $\mathcal{E} \leftarrow \mathcal{E} \cup \{E\}$
    $i \leftarrow i + 1$
end while

とし，$\mathcal{E}$ を決定した．

$|V| = 1000$ 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 3 にまとめた．

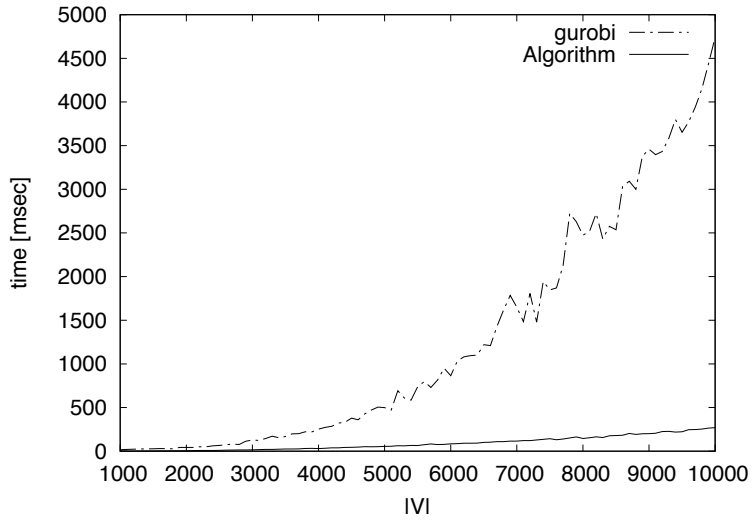

図 3: MKPFH に対する gurobi と Algorithm4 の実行時間の比較

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，よって 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，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．

また，図 4 は Algorithm4 のみについて実験結果をプロットした図である．

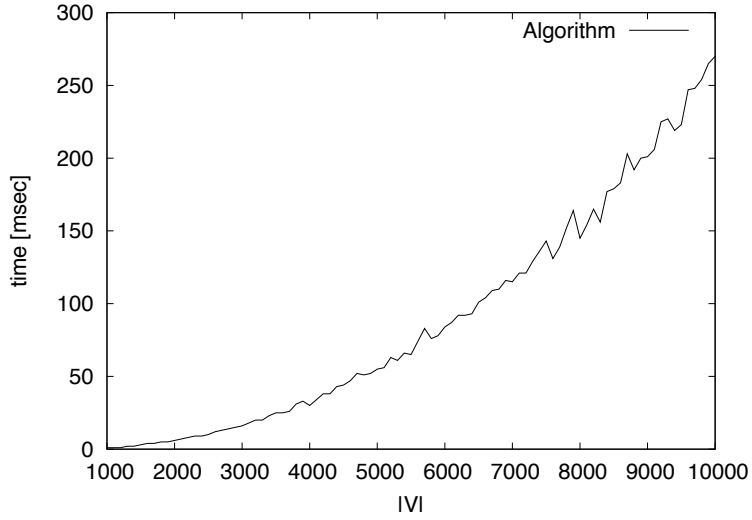

図 4: Algorithm4 の実行時間

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．

6　終わりに

本研究では最小化ナップサック問題にクラスタから少なくとも 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．

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] Tim Carnes, David Shmoys: Primal-Dual Schema for Capacitated Covering Problems, Mathematical Programing vol. 153 (2015), pp. 289-308

[11] 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 2 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 $j \in V \backslash S$ **do**
        $a_j(S) \leftarrow \min\{a_j, \bar{b}\}$
    end for
    $s \leftarrow \underset{j \in V \backslash S}{\arg\min} \left\{ \dfrac{\bar{c}_j}{a_j(S)} \right\}$
    $y(S) \leftarrow \frac{\bar{c}_s}{a_s(S)}$
    $x_s \leftarrow 1$
    $S \leftarrow S \cup \{s\}$
    for $j \in V \backslash S$ **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 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 $j \in V \backslash S_1$ **do**
    $\bar{c}_j \leftarrow c_j$
end for
while $\bar{b} > 0$ **do**
    for $j \in V \backslash \{S_1 \cup S_2\}$ **do**
        $a_j(S_2) \leftarrow \min\{a_j, \bar{b}\}$
    end for
    $s \leftarrow \arg\min_{j \in V \backslash S_2} \left\{ \dfrac{\bar{c}_j}{a_j(S_2)} \right\}$
    $S_2 \leftarrow S_2 \cup \{s\}$
    $\bar{b} \leftarrow \bar{b} - a_s$
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**
    $\min_{E \in \bar{\mathcal{E}}} |E|$ となるような E を選択
    if $\sum_{j \in E} x_j \geq 1$ **then**
        $\bar{\mathcal{E}} \leftarrow \bar{\mathcal{E}} \backslash \{E\}$
    else if $\sum_{j \in E} x_j = 0$ **then**
        $s \leftarrow \arg\min_{j \in E} \{\bar{c}_j\}$
        $z_E \leftarrow \bar{c}_s$
        $x_s \leftarrow 1$
        $S \leftarrow S \cup \{s\}$
        $\bar{\mathcal{E}} \leftarrow \bar{\mathcal{E}} \backslash \{E\}$
        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 $j \in V \backslash S$ **do**
        $a_j(S) \leftarrow \min\{a_j, \bar{b}\}$
    end for
    $s \leftarrow \arg\min_{j \in V \backslash S} \left\{ \dfrac{\bar{c}_j}{a_j(S)} \right\}$
    $y(S) \leftarrow \frac{\bar{c}_s}{a_s(S)}$
    $x_s \leftarrow 1$
    $S \leftarrow S \cup \{s\}$
    for $j \in V \backslash S$ **do**
        $\bar{c}_j \leftarrow \bar{c}_j - a_j(S) y(S)$
    end for
    $\bar{b} \leftarrow \bar{b} - a_s$
end while
$\tilde{\boldsymbol{x}} \leftarrow \boldsymbol{x}$
$(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}}) \leftarrow (\boldsymbol{y}, \boldsymbol{z})$