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ている手法としては 2 つある．

1 つは現実的な実行時間で解を求めることを諦め，その代わりに最適解を求める手法である．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手法，近似アルゴリズムの設計である．つまり離散最適化問題に対する近似アルゴリズムとは，解の値に基づいて最適値に十分に近い値をもつ近似解を求めるアルゴリズムである [3][4]．

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．

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]．したがって，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最小化版である．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の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　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]．この手法は厳密アルゴリズムに起源を持つ．主問題

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

とその双対問題

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

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]．

主双対法は整数計画問題を LP 緩和し，その双対問題を考え，そして緩和した LP の相補性条件を緩和することで近似アルゴリズムの設計を行う．

ここで例として，

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

の近似アルゴリズムを主双対法を用いて構築する．

この問題の LP 緩和は

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

となる．この問題を主問題とする双対問題は，

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

となる．この 2 つの問題には

補題 2.1 主問題 (2.1) と双対問題 (2.2) はそれぞれ実行可能であるとする．また，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，これらが

$$\forall j \in \{1,\dots,n\}, \bar{x}_j > 0 \implies \sum_{i=1}^{m} a_{ij}\bar{y}_i \geq c_j/\alpha \tag{2.3}$$

$$\forall i \in \{1,\dots,m\}, \bar{y}_i > 0 \implies \sum_{j=1}^{n} a_{ij}\bar{x}_j \leq \beta b_i \tag{2.4}$$

(ただし $\alpha, \beta > 1$) を満たすならば，

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

が成立する．

証明　主問題と双対問題に許容解が存在することから，双対定理より最適解が存在しその最適値は一致する．ここで (2.3) を変形すると，

$$\forall j \in \{1,\dots,n\}, \bar{x}_j > 0 \implies \alpha\sum_{i=1}^{m} a_{ij}\bar{y}_i \geq c_j$$

となる．これと式 (2.4) を用いると式 (2.5) の左辺は

$$\begin{aligned}\sum_{j=1}^{n} c_j\bar{x}_j &\leq \alpha\sum_{j=1}^{n}\sum_{i=1}^{m} a_{ij}\bar{x}_j\bar{y}_i \\ &\leq \alpha\beta\sum_{i=1}^{m} b_i\bar{y}_i\end{aligned} \tag{2.6}$$

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，これを用いて式 (2.6) を変形させると，

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

となる．

□

という関係が存在する．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．

先に記したように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．しかし，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．

2.2 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 $\boldsymbol{a}, \boldsymbol{c} \in \mathbb{R}^n_{++}, b \in \mathbb{R}_{++}$ 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 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) は変数 x が 0 と 1 どちらかの値しかとらないということを意味し，バイナリ制約と呼ばれる．

2.2.1 LP 緩和と双対問題

最小化ナップサック問題に対して主双対法を用いて近似アルゴリズムを構築するために，問題を LP 緩和し，双対問題を考える必要がある．

そのまま LP 緩和を行うと上述したように整数ギャップが生じてしまう．したがって，より整数ギャップを小さくするために品物の全ての部分集合 $A \subseteq V$ のそれぞれに対して新しい制約を導入し，

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

とする [3][9]．ただし，

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

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

である．この問題の双対問題は

$$\begin{aligned} \max \quad & \sum_{A \subseteq V} b(A) y(A) & & \\ \text{s.t.} \quad & \sum_{A \subseteq V : j \notin A} a_j(A) y(A) \leq c_j & \forall j \in V & \\ & y(A) \geq 0 & \forall A \subseteq V & \end{aligned} \tag{2.13}$$

となる [3]．ここで変数 $y(A)$ の個数は任意の V の部分集合に対して考えているので $2^{|V|}$ 個になる．

2.2.2 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 (2.13) を用いて主双対法によるアルゴリズムの構築を行う.

補題 2.2 主問題 (2.10) の許容解を $\bar{\boldsymbol{x}}$ とし，双対問題 (2.13) の許容解を $\bar{\boldsymbol{y}}$ とする．最適値を θ_{mkp} とする．それぞれの許容解が

$$\forall j \in V, \bar{x}_j > 0 \implies \sum_{A \subseteq V : j \notin A} a_j(A)\bar{y}(A) = c_j \tag{2.14}$$

$$\forall A \subseteq V, \bar{y}(A) > 0 \implies \sum_{j \in V} a(A)_j \bar{x}_j \leq 2b(A) \tag{2.15}$$

を満たすならば

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

が成り立つ.

証明 補題 2.1 と同様にして容易に証明することができる. □

系 2.1 問題 (2.10) の 0-1 許容解を $\tilde{\boldsymbol{x}}$，問題 (2.13) 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.

証明 問題 (2.7)〜(2.9) が実行可能ならば，その許容解は全て LP 緩和した問題 (2.10) の許容解となる.
また，LP 緩和の性質から，問題 (2.10) の最適値を θ_{mkp} とすると，

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

が成り立つ．したがって，$\tilde{\boldsymbol{x}}$ と $\tilde{\boldsymbol{y}}$ が補題 2.2 を満たすことから，

$$\begin{aligned} \sum_{j \in V} c_j \tilde{x}_j &\leq 2\theta_{mkp} \\ &\leq 2\theta^*_{mkp} \end{aligned}$$

が成り立つ. □

この条件を満たすようなアルゴリズムが過去にいくつか研究されている [3].

2.2.3 最小化ナップサック問題の 2-近似アルゴリズム [3]

補題 2.2 を満たす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: $\tilde{\boldsymbol{x}}$: 主問題の解, $\tilde{\boldsymbol{y}}$: 双対問題の解

Step1: $\boldsymbol{x}=\boldsymbol{0}, \boldsymbol{y}=\boldsymbol{0}$ 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.

Step2: $\bar{b} \leq 0$ 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, $s=\underset{j \in V \backslash S}{\arg\min}\left\{\dfrac{\bar{c}_j}{a_j(S)}\right\}$ を計算し, $y(S)=\dfrac{\bar{c}_s}{a_s(S)}, x_s=1$, $\bar{b}=b-a_s$, $\bar{c}_j=\bar{c}_j-a_j y(S)(\forall j \in V \backslash S)$ とし, Step2 の初めに戻る.

アルゴリズムの詳細を付録 A の Algorithm1 に示す.

補題 2.3 Algorithm1 から得られた $\tilde{\boldsymbol{x}}$ は主問題 (2.10) の 0-1 許容解であり, $\tilde{\boldsymbol{y}}$ は双対問題 (2.13) の許容解である.

証明 問題 (2.7)〜(2.9) 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. Algorithm1 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 (2.10) の 0-1 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. Algorithm1 はこれを初期値とし, 次の補題 2.4 で示すように, 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 (2.13) の許容解である. □

補題 2.4 Algorithm1 より得られた解を $\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}$ とした時, この解は補題 2.2 の条件 (2.14) と条件 (2.15) を満たす.

証明 Algorithm1 は $\boldsymbol{x}=\boldsymbol{0}, \boldsymbol{y}=\boldsymbol{0}$ から始まり, Step2 では $a_j(S)y(S)=c_j-\sum_{A \subseteq S \backslash\{j\}} a_j(A)y(A)$ とした時に $x_j=1$ としている. よって条件 (2.14) において $x_j>0$ であり, $\sum_{A \subseteq V: j \notin A} a_j(A)y(A)=c_j$ を満たすことがわかる.

次に条件 (2.15) だが, Step2 での動作を考える. $\tilde{S}=\{j \in V | \tilde{x}_j=1\}$ 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 $\tilde{y}(A)>0$ であるような 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

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

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,

$$\sum_{j \in \tilde{S} \backslash\{l\}} a_j<b \tag{2.16}$$

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 (2.16) 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，Algorithm1 は条件 (2.15) を満たす．

以上より，Algorithm1 より得られた解は補題 2.2 の条件 (2.14) と条件 (2.15) を満たす． □

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)〜(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，Algorithm1 より得られ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対し，V を分割した集合 $P(\mathcal{P} \subseteq 2^V)$ を受け取り Partition 制約を考えた問題

$$\begin{aligned} \min & \sum_{j \in V} c_j x_j & & \\ \text{s.t.} & \sum_{j \in V} a_j x_j \geq b & & (3.1) \\ & \sum_{j \in P} x_j \geq 1 & \forall P \in \mathcal{P} & (3.2) \\ & x_j \in \{0,1\} & \forall j \in V & \end{aligned}$$

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．この時 $\mathcal{P}$ は品物のクラスタの集合を表している．ただし，任意の $\mathcal{P}$ の要素は互いに素であり，$\mathcal{P}$ の要素全ての和集合は V と一致するものとする．この問題は最小化ナップサック問題に，V を分割した集合に含まれる品物のうち少なくとも一つは選択するといった Partition 制約 (3.2) を追加した問題である．

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

$$\begin{aligned} \min & \sum_{j \in V} c_j x_j & & \\ \text{s.t.} & \sum_{j \in P} x_j \geq 1 & \forall P \in \mathcal{P} & (3.3) \\ & x_j \in \{0,1\} & \forall j \in V & \end{aligned}$$

と，

$$\begin{aligned} \min & \sum_{j \in V} c_j x_j & & \\ \text{s.t.} & \sum_{j \in V} a_j(A) x_j \geq b(A) & \forall A \subseteq V & (3.4) \\ & x_j \geq 0 & \forall j \in V & \end{aligned}$$

を考える．

問題 (3.3) は Partition 制約 (3.2) を持った問題であり，問題 (3.4) はナップサック制約 (3.1) を持った問題である．

前章により，問題 (3.4) 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 (3.3) に対する厳密解法

問題 (3.3) 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付録 A の Algorithm2 に示す．

3.3 MKPPC の 3-近似アルゴリズム

問題 (3.3) と問題 (3.4) 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 (3.3) を満たすように Algorithm2 を適用．$S_1 = \{j \mid x_j = 1\}$ 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: 残った品物で問題 (3.4) 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，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 0-1 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 (3.3) と問題 (3.4) 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

$$\sum_{j \in V} c_j x_j = \sum_{j \in S} c_j = \sum_{j \in S_1} c_j + \sum_{j \in S_2} c_j \tag{3.5}$$

と変形できる．ここで S_1 と S_2 はそれぞれ Algorithm2 と Algorithm3.4 により得られた解なので，問題 (3.3) の最適値を θ_1，問題 (3.4) の最適値を θ_2 とおくと，

$$(3.5) \leq \theta_1 + 2\theta_2 \tag{3.6}$$

となる．さらに，$\theta_1 \leq \theta$，$\theta_2 \leq \theta$ なので，

$$\begin{aligned} (3.6) &\leq \theta + 2\theta \\ &= 3\theta \end{aligned}$$

より，$\sum_{j \in V} c_j x_j \leq 3\theta$ 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　疑似コードの 3〜6 行目より，Step1 の 1 回の反復での計算量は高々$\mathcal{O}(p)$ 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 $\mathcal{O}(p|\mathcal{P}|)$ であることがわかる．

次に，Step2 の 1 回の反復での計算量について考える．12〜14 行目より，$a_j(S)$ の計算量は高々$\mathcal{O}(|V|)$ である．また，15 行目の計算量も高々$\mathcal{O}(|V|)$ 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 [10]．

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

最小化ナップサック問題に対し，任意の $E \in \mathcal{E}$ について $|E| \geq 2$ であるようなハイパーグラフ $H = (V, \mathcal{E})(V = \{1, \ldots, n\})$ を受け取り Forcing 制約とする問題を

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

を Forcing Hypergraph 付き最小化ナップサック問題 (Minimum Knapsack Problem with Forcing Hypergraph:MKPFH) と呼ぶこととする Forcing 制約とは，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制約のことをいう．

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の操作をして，

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

と表される．さらに問題 (4.2) の双対問題は

$$\begin{aligned} \max \quad & \sum_{A\subseteq V} b(A) y(A) + \sum_{E\in\mathcal{E}} z_e & & \\ \text{s.t.} \quad & \sum_{A\subseteq V: j\notin A} a_j(A) y(A) + \sum_{E\in\mathcal{E}: j\in E} z_E \leq c_j & & \forall j \in V \\ & y(A) \geq 0 & & \forall A \subseteq V \\ & z_E \geq 0 & & \forall E \in \mathcal{E} \end{aligned} \tag{4.3}$$

と表される．

4.4 主双対法

式 (4.2) と式 (4.3) に主双対法を適用する．

補題 4.1 主問題 (4.2) に許容解 $\boldsymbol{x}$, 双対問題 (4.3) に許容解 $(\boldsymbol{y}, \boldsymbol{z})$ が存在し，

$$\forall j \in V, x_j > 0 \implies \sum_{A\subseteq V: j\notin A} a_j(A) y(A) + \sum_{E\in\mathcal{E}: j\in E} z_E = c_j \tag{4.4}$$

$$\forall E \in \mathcal{E}, z_E > 0 \implies \sum_{j\in E} x_j \leq k \tag{4.5}$$

$$\forall A \subseteq V, y(A) > 0 \implies \sum_{j\in V\setminus A} a_j(A) x_j \leq k b(A) \tag{4.6}$$

を満たすとする．ただし $k = \max_{E\in\mathcal{E}} |E|$ とする．このとき問題 (4.2) の最適値を θ とすると $\sum_{j\in V} c_j x_j \leq k\theta$ となる．

4.5 MKPFH に対する k-近似アルゴリズム

補題 4.1 を満たすような MKPFH の k-近似アルゴリズムの概要を示す．

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: 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: $\tilde{\boldsymbol{x}}$: 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，$S = \emptyset (S = \{s \mid x_s = 1\}), \bar{\mathcal{E}} = \mathcal{E}, \bar{b} = b, \bar{c}_j = c_j (\forall j \in V)$ を初期値として与える．

Step1: $\bar{\mathcal{E}} = \emptyset$ ならば Step2 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 つ選択する．このとき $\sum_{j \in E} x_j \geq 1$ ならば $\bar{\mathcal{E}}$から要素 E を除き Step1 の初めに戻る。そうでないならば $s = \arg\min_{j \in E} \{\bar{c}_j\}$ を計算し，$z_E = \bar{c}_s, x_s = 1$ とし，$\bar{\mathcal{E}} = \bar{\mathcal{E}} \backslash \{E\}$，$\bar{b} = b - a_s$，$\bar{c}_j = \bar{c}_j - z_E (\forall j \in E)$ とし，Step1 の初めに戻る．

Step2: $\bar{b} \leq 0$ 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，$s = \arg\min_{j \in V \backslash S} \left\{ \frac{\bar{c}_j}{a_j(S)} \right\}$ を計算し，$y(S) = \frac{\bar{c}_s}{a_s(S)}, x_s = 1$，$\bar{b} = b - a_s$，$\bar{c}_j = \bar{c}_j - a_j y(S) (\forall j \in V \backslash S)$ とし，Step2 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 より得られる $\tilde{\boldsymbol{x}}$ は主問題 (4.2) の 0-1 許容解であり，$(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}})$ は双対問題 (4.3) 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，Forcing 制約と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 (4.2) の 0-1 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，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 (4.3) の許容解である． □

補題 4.3 Algorithm4 より得られた解は補題 3.1 の条件 (4.4)〜(4.6) 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 (4.4) 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 (4.5) 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 (2.4) を満たしている．

最後に条件 (4.6) 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 (4.6) 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，

$$\sum_{j \in \tilde{S} \setminus \{l\}} a_j < b \tag{4.7}$$

である．式 (2.12) より $\tilde{y}(A) > 0$ 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) と式 (4.7) 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，Step2 で直ちに停止しない場合でも Algorithm4 は条件 (4.6) を満たす．

以上より，Algorithm4 より得られた解は補題 3.1 の条件 (4.4)〜(4.6) を満たす． □

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

証明　疑似コードの 9〜24 行目より，Step1 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 全体の計算量は $\mathcal{O}(k|\mathcal{E}|)$ であることがわかる．

次に，Step2 の 1 回の反復での計算量について考える．26〜28 行目より，$a_j(S)$ の計算量は高々$\mathcal{O}(|V|)$ である．また，29 行目の計算量も高々$\mathcal{O}(|V|)$ 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，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．

5　数値実験

2つの提案アルゴリズム，Algorithm3 と Algorithm4 について C++で実装し，数理計画ソルバー gurobi との比較をする数値実験を行なった．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．(例: $V = \{1,2,3,4,5,6\}, \mathcal{P} = \{\{1,2,3\},\{4,5,6\}\}$)

具体的な実験内容としては，$|V| = 1000$ 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，urobi 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．

5.1.2　実験 2

実験 2 では MKPPC における $\mathcal{P}$ を V のランダムな分割により与える．(例: $V = \{1,2,3,4,5,6\}, \mathcal{P} = \{\{1,2,3,4\},\{5,6\}\}$)

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．

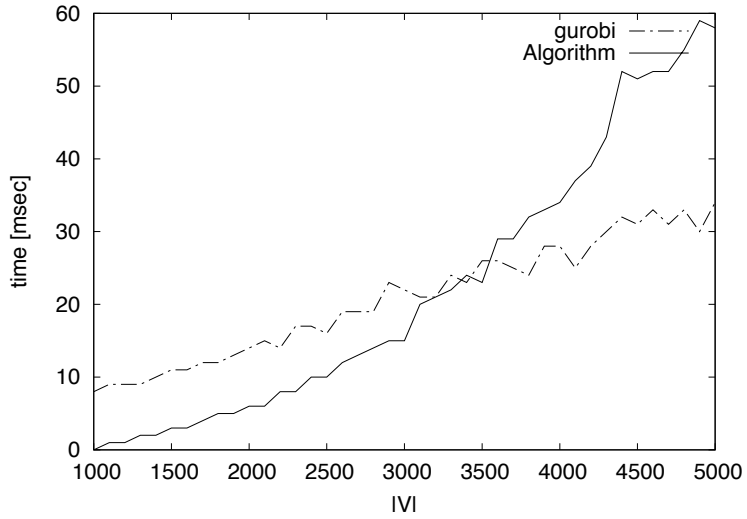

図 1: MKPPC に対する gurobi と Algorithm3 の実行時間の比較 ($\mathcal{P}$: 等分割)

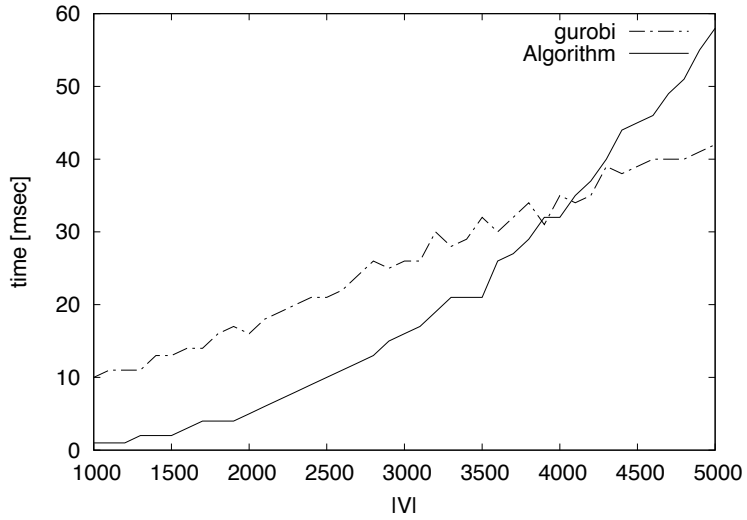

図 2: MKPPC に対する gurobi と Algorithm3 の実行時間の比較 ($\mathcal{P}$: ランダム分割)

5.1.3 考察

図 1 と図 2 から，Algorithm3 の計算量は頂点数が増加するほど $\mathcal{O}(|V|^2)$ になることが分かる．実際，頂点数が増加することにより，アルゴリズムにおける Step1 終了時には $\bar{b} < 0$ となる場合はほどんどなく，また，$\bar{b}$ の値が大きいと Step2 を多く反復することにより解を得ることになると考えられる，したがって，頂点数が増加するほど，Algorithm3 の計算量は $\mathcal{O}(|V|^2)$ になる．

また，実験 1 と実験 2 では gurobi の計算時間に変化はあるが，Algorithm3 の計算時間にはほとんど違いがないことが分かる．よって Algorithm3 は $\mathcal{P}$ の形によらず同様の実行時間になると言える．

しかし，実験 1 では $|V| = 3600$ で，実験 2 では $|V| = 4100$ で提案アルゴリズムは gurobi より平均実行時間が長くなっていることが分かる．これについては，ナップサック問題における擬多項式時間での厳密解法 [4] が存在するため，gurobi はそれを利用することにより，高速にナップサック問題を解いているのではないかと考えられる．グラフの形と Algorithm3 の計算量からわかるように，$|V|$ が増加すると Algorithm3 の計算量は $\mathcal{O}(|V|^2)$ になるため，提案手法では常に gurobi より短い実行時間にすることは困難だと考えられる．

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．

具体的な実験内容としては，$|V| = 1000$ 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．$\mathcal{E}$ については，まず

$$|\mathcal{E}| \in \left[2, \frac{n}{2}\right]$$

とし，各 $E \in \mathcal{E}$ について各頂点が確率 0.3 で入るようにした．

実験結果を図 3 にまとめた．

5.3 考察

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 の計算量は $\mathcal{O}(|V|^2)$ になることがわかる．

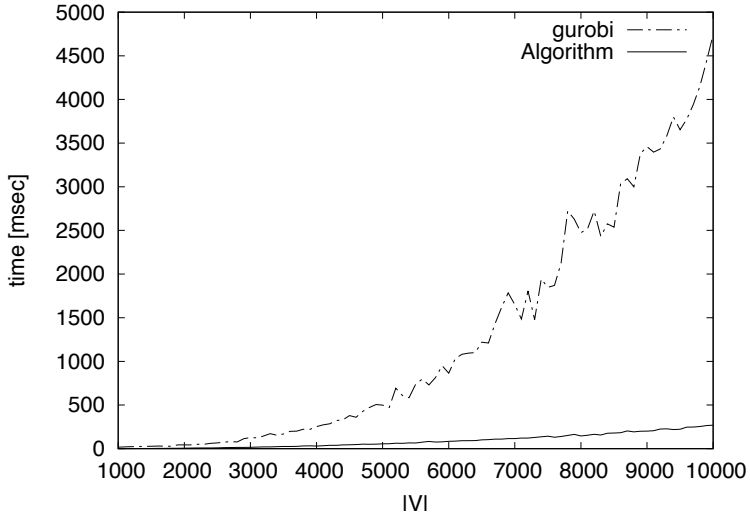

図 3: MKPFH に対する gurobi と Algorithm4 の実行時間の比較

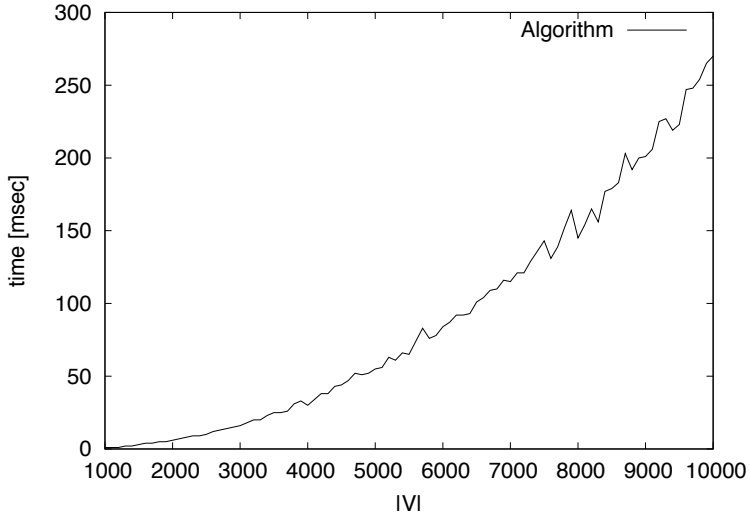

図 4: Algorithm4 の実行時間

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．

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] 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 $j \in V \backslash S$ **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 2 問題 (3.3) 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 $j \in V \backslash S_1$ **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
$\tilde{\boldsymbol{x}} \leftarrow \boldsymbol{x}$
$(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{z}}) \leftarrow (\boldsymbol{y}, \boldsymbol{z})$