

平成 29 年度電気通信大学情報・通信工学科 卒業論文

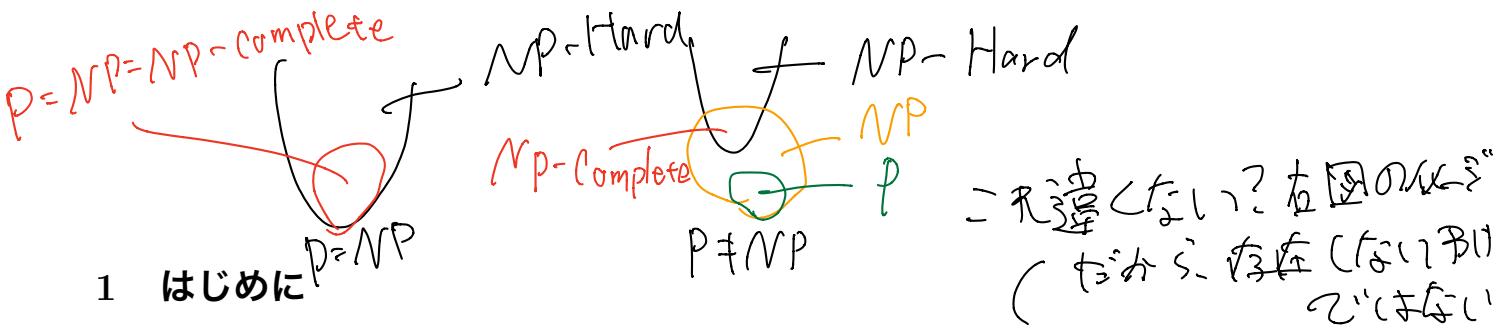
Partition 制約付き最小化ナップサック問題に対する
3-近似アルゴリズムについて

指導教員 村松 正和 教授

平成 29 年 12 月 20 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。つまり、全ての入力に対して多項式時間で最適解を求めるようなアルゴリズムが存在しないような問題が存在する。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などを諦め、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 gurobi や CPLEX のようなソルバー開発により研究されている。しかし、この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、または最適解を導出することができない場合があるということが挙げられる。

もう 1 つは最適解を求めるなどを諦め、十分良い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。^[1]

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多項式時間アルゴリズムを、その最適化問題に対する α 近似アルゴリズムと呼ぶ。^[1] 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。

最小化ナップサック問題は商品選択などの意思決定に用いられる離散最適化問題である。この問題は 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である。^[2]

2 章では既存研究などを含めた事前知識について説明し、3、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ついての近似アルゴリズムを記す。^[3]

2 事前知識

2.1 主双対法

近似アルゴリズムの代表的設計手法として主双対法がある。^[2]

この手法は厳密アルゴリズムに起源を持つ。線形計画問題の最適解は相補条件をすべて満たす性質があり、これを用いる。離散最適化問題に対する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、そして相補条件を緩和して用いることで近似アルゴリズムの設計を行う。

ある整数計画問題の LP 緩和を

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

max では?

とする。さらにこの問題を主問題とする双対問題,

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

を考える。

この 2 の問題(2.1)

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{x} 、双対問題の許容解を \bar{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。

目的関数

□

よって補題 2.1 を満たすような \bar{x} から得られる値は最適値の $\alpha \beta$ 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 緩和すると整数ギャップが生じてしまう場合があるので注意しないではならない。整数ギャップとは、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ことである。

この問題の関数
は2^nを用いた
1.2. LPの問題
のC=3に属する

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 $V = \{1, \dots, n\}$ を用いて、

$$\min \sum_{j \in V} c_j x_j \quad (2.7)$$

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

$$x_j \in \{0, 1\} \quad \forall j \in V \quad \text{目的関数} \quad (2.9)$$

と表現される。この時 a, b, c はそれぞれ $a, c \in \mathbb{R}_{++}^n, b \in \mathbb{R}_{++}$ である。式(2.7)は目的関数と呼ばれ、選択した品物の重さの総和を最小化することを表している。式(2.8)と式(2.9)はそれぞれ

式(2.8): ナップサック制約(選択した品物の価値の総和を b 以上に保つ)

式(2.9): バイナリ制約(変数 x が 0 と 1 どちらかの値しかとらない)

を表している。

2.2.1 LP 緩和と双対問題

最小化ナップサック問題の双対問題を考えるため、LP 緩和を行う。最小化ナップサック問題でそのまま LP 緩和を行うと整数ギャップが生じてしまう場合がある。したがって LP 緩和ははより少ない整数ギャップにするために $\sum_{j \in A} a_j < b$ となるような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、

$$\begin{aligned} &\text{より整数} && \min \sum_{j \in V} c_j x_j \\ &\text{もやう} && \text{上記} \\ &\text{かよくする} && \text{s.t. } \sum_{j \in V} a_j(A)x_j \geq b(A) \quad \forall A \subseteq V \\ &\text{ため} && x_j \geq 0 \quad \forall j \in V \end{aligned} \quad (2.10)$$

とする。[3]ただし、参考文献[1]によると、この問題の双対問題は

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

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

である。この問題の双対問題は

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

となる。ここで変数 $y(A)$ の個数は $|2^V|$ 個になる。

又々大問題の発現か、あるいは参考文献

2.2.2 主双対法

最小化ナップサック問題に対し、主双対法を適用する。

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

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

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

を満たすならば

$$\sum_{j \in V} c_j \bar{x}_j \leq 2\theta_{mfp} \quad (\text{この式が成り立つ})$$

が成り立つ。

系 2.1 問題 (2.10) の 0-1 許容解を \tilde{x} 、問題 (2.13) の許容解を \tilde{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_{mfp}^* とすると。

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

が成り立つ。

これを満たすようなアルゴリズムが過去の研究で設計されている。
[3] ～もって参考文献ないかな？
この条件は常に満たされています。

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-近似アルゴリズムの概要を示す。

Input: V, a, b, c

Output: \tilde{x}, \tilde{y}

Step1: $x = 0, y = 0$ 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 $\tilde{x} = x, \tilde{y} = y$ とし、アルゴリズムを停止する。そうでないならば、 $s = \arg \min_{j \in V \setminus S} \left\{ \frac{\bar{c}_j}{a_j(S)} \right\}$ を計算し、 $y(S) = \frac{\bar{c}_s}{a_s(S)}, x_s = 1$ とし、 \bar{b}, \bar{c} を更新後 Step2 の初めに戻る。

アルゴリズムの詳細を Algorithm1 に示す。

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{x} and \tilde{y}

```
1:  $x \leftarrow \mathbf{0}$ 
2:  $y \leftarrow \mathbf{0}$ 
3:  $S \leftarrow \emptyset$ 
4:  $\bar{b} \leftarrow b$ 
5: for  $j \in V$  do
6:    $\bar{c}_j \leftarrow c_j$ 
7: end for
8: while  $\bar{b} > 0$  do
9:   for  $j \in V \setminus S$  do
10:     $a_j(S) \leftarrow \min \{a_j, \bar{b}\}$ 
11:   end for
12:    $s \leftarrow \arg \min_{j \in V \setminus S} \left\{ \frac{\bar{c}_j}{a_j(S)} \right\}$ 
13:    $y(S) \leftarrow \frac{\bar{c}_s}{a_s(S)}$ 
14:    $x_s \leftarrow 1$ 
15:    $S \leftarrow S \cup \{s\}$ 
16:   for  $j \in V \setminus S$  do
17:      $\bar{c}_j \leftarrow \bar{c}_j - a_j(S)y(S)$ 
18:   end for
19:    $\bar{b} \leftarrow \bar{b} - a_s$ 
20: end while
21:  $\tilde{x} \leftarrow x$ 
22:  $\tilde{y} \leftarrow y$ 
```

補題 2.3 Algorithm1 から得られた \tilde{x} は主問題 (2.10) の 0-1 許容解であり, \tilde{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, $x = \mathbf{1}$ は主問題 (2.10) の許容解である. Algorithm1 は $x = \mathbf{0}$ 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{x} は主問題 (2.10) の 0-1 訸容解である.

さらに, $y = \mathbf{0}$ は双対問題の許容解である. Algorithm1 はこれを初期値とし, 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{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{x}, \tilde{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 $x = \mathbf{0}, y = \mathbf{0}$ から始まり, Step2 では $a_j(S)y(S) = c_j - \sum_{A \subseteq S \setminus \{j\}} a_j(A)y(A)$ とした時に $x_j = 1$ としている. よって条件 (2.14)において $x_j > 0$ であるならば $\sum_{A \subseteq S: j \notin A} a_j(A)y(A) = c_j$ 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 $S = S \cup \{s\}$ 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,

$$\sum_{j \in \tilde{S} \setminus \{l\}} a_j < b \quad (2.16)$$

である. 式 (2.12) より $\tilde{y}(A) > 0$ であるような A について,

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

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. 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. \square

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.

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 \mathcal{P} ($\mathcal{P} \subseteq 2^V$, $\forall P_1, P_2 \in \mathcal{P}$ ($P_1 \neq P_2$), $P_1 \cap P_2 = \emptyset$, $\bigcup_{j=1}^{|\mathcal{P}|} P_j = V$) を受け取り Partition 制約とする Partition 制約付き最小化ナップサック問題 (Minimum Knapsack Problem with Partition Constraint MKPPC) を

$$\begin{aligned} & \min \sum_{j \in V} c_j x_j \\ \text{s.t. } & \sum_{j \in V} a_j x_j \geq b \end{aligned} \tag{3.1}$$

$$\begin{aligned} & \sum_{j \in P} x_j \geq 1 \quad \forall P \in \mathcal{P} \\ & x_j \in \{0, 1\} \quad \forall j \in V \end{aligned} \tag{3.2}$$

と定式化する。この時 \mathcal{P} は品物のクラスタの集合を表している。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式(3.2)で表される Partition 制約を追加した問題である。

3.1 MKPPC の分割

MKPPC の 2 つの制約 (3.1) と (3.2) をそれぞれ別の問題に分割する。まず、制約 (3.2) についてを考えてみる。

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

を考える。これを問題 1 と呼ぶことにする。

制約 (3.1) については式 (2.7)~(2.9) で定式化される最小化ナップサック問題と同様の問題を考える。これを問題 2 と呼ぶことにする。

3.2 問題 1 に対する厳密解法

問題 1 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 $\forall P_1, P_2 \in \mathcal{P}$ ($P_1 \neq P_2$), $P_1 \cap P_2 = \emptyset$ であることから、 $\forall P \in \mathcal{P}$ に対して $\arg \min_{j \in P} c_j$ を計算することで簡単に最適解を導出することが可能である。厳密解法の詳細を Algorithm 2 に示す。

何故 最適解を導出できるのかをもとめよう。

Algorithm 2 問題 1 の厳密解法

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

Output: \tilde{x}

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

本研究で(↑)

3.3 MKPPC の 3-近似アルゴリズム

問題 1 と問題 2 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。アルゴリズムの概要を示す。

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

Output: \tilde{S}

↑ これの↓
↓ ここに(←)

Step0: $S_1 = S_2 = \emptyset$ とする。

Step1: 問題 1 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 2 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 Algorithm3 に示す。

筆の下
ルート
の
枝葉
を
同様

Algorithm 3 MKPPC の 3-近似アルゴリズム

Input: V, \mathcal{P}, a, b, c **Output:** \tilde{S}

```
1:  $S_1 \leftarrow \emptyset$ 
2:  $S_2 \leftarrow \emptyset$ 
3: for  $P \in \mathcal{P}$  do
4:    $s = \arg \min_{j \in P} c_j$ 
5:    $S_1 \leftarrow S_1 \cup \{s\}$ 
6: end for
7:  $\bar{b} \leftarrow b - \sum_{j \in S_1} a_j$ 
8: for  $j \in V$  do
9:    $\bar{c}_j \leftarrow c_j$ 
10: end for
11: while  $\bar{b} > 0$  do
12:   for  $j \in V \setminus S$  do
13:      $a_j(S) \leftarrow \min \{a_j, \bar{b}\}$ 
14:   end for
15:    $s \leftarrow \arg \min_{j \in V \setminus S} \left\{ \frac{\bar{c}_j}{a_j(S)} \right\}$ 
16:    $S_2 \leftarrow S_2 \cup \{s\}$ 
17:    $\bar{b} \leftarrow \bar{b} - a_s$ 
18: end while
19:  $\tilde{S} \leftarrow S_1 \cup S_2$ 
```

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)

証明 Algorithm3 より得られた解は問題 1 と問題 2 両方の制約を満たす。また、アルゴリズムは $x = \mathbf{0}$ から始まり、2つの制約を満たすように x_j を 0 から 1 へと変更している、したがってこの解は MKPPC の 0-1 許容解である。

さらに問題 1 の最適値を θ_1 、問題 2 の最適値を θ_2 とおくと、

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

→ この不等式の意味を
説明

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 $\mathcal{O}(|E| + |V|^2)$ である。

よくわかるなりのぞ
でねね

証明 Step1 の 1 回の反復での計算量は $\mathcal{O}(1)$ である。また、Step1 の反復回数は $|E|$ 回である。また、Step2 の 1 回の反復での計算量は $\mathcal{O}(|V|)$ である。また、Step2 の反復回数は多くとも $|V|$ 回である。以上より、Algorithm3 の計算量は $\mathcal{O}(|E| + |V|^2)$ 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。

「制約」 「ミスアロ」

4 Forcing Hypergraph 付き最小化ナップサック問題の k-近似アルゴリズム

MKPPC を一般化した問題、Forcing Hypergraph 付き最小化ナップサック問題の近似アルゴリズムを考える。

4.1 ハイパーグラフ

V を頂点集合、 $\mathcal{E} \subseteq 2^V$ とする。この順序対 $H = (V, \mathcal{E})$ をハイパーグラフと呼ぶ [4]。これはグラフの概念の一般化となっている。

「ハイパーグラフ」 「頂点集合」

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、辺に含まれる頂点の個数が 2 個以上であるような Forcing Hypergraph $H = (V, \mathcal{E})$ ($V = \{1, \dots, n\}$, $\mathcal{E} \subseteq 2^V$, $|E| \geq 2 (\forall E \in \mathcal{E})$) を受け取り Forcing 制約と

「ハイパーグラフ」

「問題」

する Forcing Hypergraph 付き最小化ナップサック問題 (Minimum Knapsack Problem with Forcing Hypergraph: MKPFH) を

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

「MKPFH に対する」とある

と定式化する。Forcing 制約とは、 $\forall E \in \mathcal{E}$ 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である。

「付帯の」
「いらない」

4.3 LP 緩和と双対問題

問題 (4.1) の LP 緩和は、最小化ナップサック問題と同様に整数ギャップを地位 s 区するために新たな制約式を与えて、

$$\begin{aligned}
 \text{「これがいいのどう} \\
 \text{ここがいい限りなく} \\
 \text{なのがを盡く} \\
 \text{る} \quad \text{」}
 \end{aligned}
 \begin{aligned}
 \min \quad & \sum_{j \in V} c_j x_j \\
 \text{s.t.} \quad & \sum_{j \in V} a_j(A)x_j \geq b(A) \quad \forall A \subseteq V \\
 & \sum_{j \in E} x_j \geq 1 \quad \forall E \in \mathcal{E} \\
 & x_j \geq 0 \quad \forall j \in V
 \end{aligned} \tag{4.2}$$

と表される。式 (4.2) の双対問題は

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

となる。

「它表され」

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 \mathbf{x} 、双対問題 (4.3) に許容解 (\mathbf{y}, \mathbf{z}) が存在し、

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

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

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

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_{kp} とすると $\sum_{j \in V} c_j x_j \leq k\theta_{kp}$ となる。

「証明？」

4.5 MKPFH の 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以下に示す.

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

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

Step0: $x = \mathbf{0}$, $(y, z) = (\mathbf{0}, \mathbf{0})$ 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 $e \in \bar{\mathcal{E}}$ の中から $\min_{E \in \bar{\mathcal{E}}} |E|$ 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 $\{E\}$ 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{b}, \bar{c} を更新後 Step1 の初めに戻る.

Step2: $\bar{b} \leq 0$ ならば $\tilde{x} = x, (\tilde{y}, \tilde{z}) = (y, z)$ とし, アルゴリズムを停止する. そうでないならば, $s = \arg \min_{j \in V \setminus S} \left\{ \frac{\bar{c}_j}{a_j(S)} \right\}$ を計算し, $y(S) = \frac{\bar{c}_s}{a_s(S)}$, $x_s = 1$ とし, \bar{b}, \bar{c} を更新後 Step2 の初めに戻る.

アルゴリズムの詳細を Algorithm4 に記す.

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

```
1:  $\mathbf{x} \leftarrow \mathbf{0}$ 
2:  $(\mathbf{y}, \mathbf{z}) \leftarrow (\mathbf{0}, \mathbf{0})$ 
3:  $S \leftarrow \emptyset$ 
4:  $\bar{\mathcal{E}} \leftarrow \mathcal{E}$ 
5:  $\bar{b} \leftarrow b$ 
6: for  $j \in V$  do
7:    $\bar{c}_j \leftarrow c_j$ 
8: end for
9: while  $\bar{\mathcal{E}} \neq \emptyset$  do
10:    $\min_{E \in \bar{\mathcal{E}}} |E|$  となるような  $E$  を選択
11:   if  $\sum_{j \in E} x_j \geq 1$  then
12:      $\bar{\mathcal{E}} \leftarrow \bar{\mathcal{E}} \setminus \{E\}$ 
13:   else if  $\sum_{j \in E} x_j = 0$  then
14:      $s \leftarrow \arg \min_{j \in E} \{\bar{c}_j\}$ 
15:      $z_E \leftarrow \bar{c}_s$ 
16:      $x_s \leftarrow 1$ 
17:      $S \leftarrow S \cup \{s\}$ 
18:      $\bar{\mathcal{E}} \leftarrow \bar{\mathcal{E}} \setminus \{E\}$ 
19:     for  $j \in E$  do
20:        $\bar{c}_j \leftarrow \bar{c}_j - z_E$ 
21:     end for
22:      $\bar{b} \leftarrow \bar{b} - a_s$ 
23:   end if
24: end while
25: while  $\bar{b} > 0$  do
26:   for  $j \in V \setminus S$  do
27:      $a_j(S) \leftarrow \min \{a_j, \bar{b}\}$ 
28:   end for
29:    $s \leftarrow \arg \min_{j \in V \setminus S} \left\{ \frac{\bar{c}_j}{a_j(S)} \right\}$ 
30:    $y(S) \leftarrow \frac{\bar{c}_s}{a_s(S)}$ 
31:    $x_s \leftarrow 1$ 
32:    $S \leftarrow S \cup \{s\}$ 
33:   for  $j \in V \setminus S$  do
34:      $\bar{c}_j \leftarrow \bar{c}_j - a_j(S)y(S)$ 
35:   end for
36:    $\bar{b} \leftarrow \bar{b} - a_s$ 
37: end while
38:  $\tilde{\mathbf{x}} \leftarrow \mathbf{x}$ 
39:  $(\tilde{\mathbf{y}}, \tilde{\mathbf{z}}) \leftarrow (\mathbf{y}, \mathbf{z})$ 
```

補題 4.2 Algorithm4 より得られる \tilde{x} は主問題 (4.2) の 0-1 許容解であり, (\tilde{y}, \tilde{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, $x = \mathbf{1}$ は主問題 (4.2) の許容解である. Algorithm4 は $x = \mathbf{0}$ 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{x} は主問題 (4.2) の 0-1 許容解である.

さらに, $(y, z) = (\mathbf{0}, \mathbf{0})$ 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{y}, \tilde{z}) は双対問題 (4.3) の許容解である. \square

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 $x = \mathbf{0}, (y, z) = (\mathbf{0}, \mathbf{0})$ 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 $S = S \cup \{s\}$ するので

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

とすることができます. 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,

$$\sum_{j \in \tilde{S} \setminus \{l\}} a_j < b \quad (4.7)$$

である. 式 (2.12) より $\tilde{y}(A) > 0$ であるような A について,

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

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,

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

≤ 2倍 (なぜか)

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、

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

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 $\mathcal{O}(|E| + |V|^2)$ である。

学ぶ

証明 Step1 の1回の反復での計算量は $\mathcal{O}(1)$ である。また、Step1 の反復回数は多くとも $|E|$ 回である。また、Step2 の1回の反復での計算量は $\mathcal{O}(|V|)$ である。また、Step2 の反復回数は多くとも $|V|$ 回である。

以上より、Algorithm4 の計算量は $\mathcal{O}(|E| + |V'|^2)$ 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

5 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] David P.Williamson・David B.Shmoys 著、浅野孝夫 訳: 近似アルゴリズムへの序論。近似アルゴリズムデザイン、共立出版(2015), pp.3-4
- [2] V.V. ヴァジラーニ 著、浅野孝夫 訳: プライマルデュアル法による集合カバー。近似アルゴリズム、シュプリングジャパン(2002), pp.127-132
- [3] David P.Williamson・David B.Shmoys 著、浅野孝夫 訳: 主双対法。近似アルゴリズムデザイン、共立出版(2015), pp.193-197
- [4] J.マトウシェク・J.ネシエトリル 著、根上生也・中本敦浩 訳: ブロック・デザイン。離散数学への招待 下、丸善出版(2012), pp. 134

まと
うの
かみ
ひい