8　整数計画問題

これまで、線形計画問題の解法としてシンプレックス法 (単体法) を紹介し、次に、線形計画問題の非常に特殊な場合として輸送計画問題に注目し、その解法を説明しました。また、非線形の関数、解析的な形になっていない関数を含んだ問題の解法として、動的計画法を解説しました。今度は、『問題の変数が整数でなければならない』といった離散的な条件のついた数理計画問題 (整数計画問題) について考えましょう。

8.1　整数計画問題とは

一般に、『変数の全部、あるいは一部が整数でなければならない』という条件を含んだ数理計画問題を整数計画問題とよびます。線形の整数計画問題、すなわち、線形計画問題に整数条件をつけた問題に関しては多くの研究がなされており、現実の問題への応用例も多々あります。一方、非線形の整数計画問題は線形の場合に比べて、厳密に最適解を求めるのが格段に難しくなります。ここでは、線形の整数計画問題に説明を限定して、以下では簡単に整数計画問題とよぶことにします。整数計画問題は以下のように記述できます。

$$
\begin{aligned}
\textbf{maximize} \quad & \sum_{j=1}^{n} c_j x_j & \\
\textbf{subject to} \quad & \sum_{j=1}^{n} a_{ij} x_j \leq b_i & (i = 1, 2, \ldots, m) \\
& x_j \geq 0 & (j = 1, 2, \ldots, n) \\
& x_j \text{は整数} & (j = 1, 2, \ldots, p, p \leq n)
\end{aligned}
$$

整数計画問題は全ての変数が整数でなければならない、という条件を線形計画問題に付け加えた全整数計画問題と、変数の一部が整数であるという条件がつけられた混合整数計画問題に分けられます。さらに、整数と指定された変数が **0** と **1** しかとりえない場合には、それぞれ、**0-1** 全整数計画問題、**0-1** 混合整数計画問題とよばれます。

整数計画問題の例を **3** つ挙げます。

- 分割不可能な財を含んだ生産計画問題
  生産計画問題において、資源や製品のなかに分割できない財 (例えば、自動車、船、建造物など) を含んでいるときには、整数計画問題となります。しかしながら、分割できなくてもそれらの量が非常に多量に用いられるときには、整数条件を外した線形計画問題を解き、その解の小数部分を四捨五入あるいは切捨てすることにより、近似的な最適解が得られます。

- 施設配置問題
  関東地方を中心に営業を行なってきた輸入品販売業の **K** 社では、関西地方に活動を拡大するため、京阪神地方に倉庫の賃貸を行なう計画を立てています。賃借の候補となる倉庫は m ケ

所あって、倉庫 i $(i = 1, \ldots, m)$ の保有可能な製品量は a_i で、その賃借料は d_i です。販売店 j $(j = 1, 2, \ldots, n)$ での製品の必要量を b_j とし、倉庫 i から販売店 j への製品 1 単位あたりの輸送費用を c_{ij} とします。倉庫の賃借料と販売店への輸送費用を考慮して、m ケ所ある倉庫の賃借候補地から数カ所を選び出したい。

この問題は、施設配置問題、または固定費つき輸送問題とよばれています。整数計画問題として定式化してください (問題 **8.1**)。

- ナップザック問題

  ある会社では、b 万円の資金を n 種類のプロジェクトの幾つかに投資することを考えています。j 番目のプロジェクトへの投資の単位は a_j 万円であり、何口でも投資できるものとし、一口当たり c_j 万円の利潤が期待できます。このとき、総利潤を最大にするような投資計画を立てることを考えます。

  j 番目のプロジェクトへの投資する単位を x_j とすると、この問題は全整数計画問題：

$$\begin{aligned} \textbf{maximize} \quad & \sum_{j=1}^{n} c_j x_j \\ \textbf{subject to} \quad & \sum_{j=1}^{n} a_j x_j \leq b \\ & x_j \in \{0, 1, \ldots\} \quad (j = 1, \ldots, n) \end{aligned}$$

  として定式化されます。この問題は、資金をナップザックの容量、各プロジェクトをそれに詰め込む物に例えて、ナップザック問題とよばれています。この問題の特徴は、整数条件 $x_j \in \{0, 1, \ldots\}$, $j = 1, \ldots, n$ 以外の制約式が 1 本しか存在しないことです。

前の章 (動的計画法) では動的計画法の適用例として、変数 x_j を **0** または **1** に限定した **0-1** ナップザック問題を紹介しました。しかし、問題規模が大きくなる場合には、次に紹介する分枝限定法の方が効率的でしょう。実際のところ、制約式を複数本含むような、一般的な整数計画問題を厳密に解く場合には、分枝限定法を用いるより手がないようです。

8.2 分枝限定法

整数計画問題を解く最も素朴な方法は実行可能解を数え挙げることでしょう。**0-1** 変数問題のように、変数の取りうる値があらかじめ有限個であることが分かっている場合には、これらをすべて列挙してやれば有限回の手続きで最適解が求まるはずです (図 **2** 参照)。ところが、**0-1** 整数計画問題の場合でも変数が n 個あれば、可能な組み合わせは 2^n に達するから、$n = 30$ ともなるとこのすべてをしらみつぶしに調べるのは大変な作業です。最適解を与える見込みのない実行可能解の生成を省略して、最適解を得ることはできないでしょうか。そこで、全体のごく一部を調べることによって “実質的” にすべてを調べたのと同じ効果を持つ手法、分枝限定法について説明します。

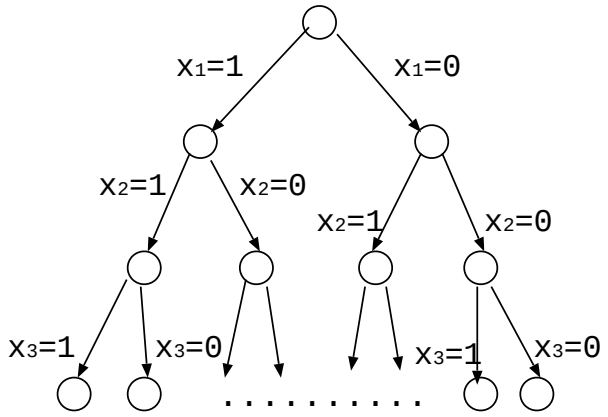

図 2: 0-1 整数計画問題の実行可能解の列挙

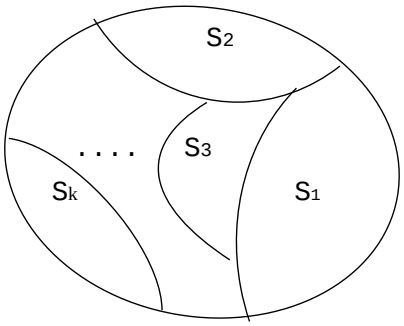

図 3: 実行可能領域 S の分割

分枝限定法の考え方は、一般的な数理計画問題に適用可能ですが、主として整数計画問題を解くために使われます。直接的にうまく解く解法がまだ見つかっていない最適化問題：

$$(P)\left|\begin{array}{ll}\textbf{maximize} & f(\boldsymbol{x})\\ \textbf{subject to} & \boldsymbol{x}\in S\end{array}\right.$$

が与えられたとします。ここで、その実行可能領域 S をいくつかの部分領域 S_i, $i=1,\ldots,k$ に分割し、その各々に対応する問題：

$$(P_i)\left|\begin{array}{ll}\textbf{maximize} & f(\boldsymbol{x})\\ \textbf{subject to} & \boldsymbol{x}\in S_i\end{array}\right.$$

を考えます。これを (P) の子問題とよび、$S=\cup_{i=1}^{k}S_i$ が成り立っているものとします（図 3 参照）。この一群の子問題 (P_i), $i=1,\ldots,k$ を解いて得られた解の中で $f(\boldsymbol{x})$ が最大のものを見つければ、それが元問題 (P) の最適解となることは明らかです。子問題 (P_i) がまだ難しく直接解けない場合には、S_i を再びいくつかの領域に分けて、(P_i) をさらにいくつかの問題に分解していきます。この手順を繰り返し適用すれば、いつかは直接解ける問題に到達し、それらの解を総合すれば (P_i) の最適解が得られ、結局 (P) が解けるであろう、というのが分枝限定法の基本的な考え方です。

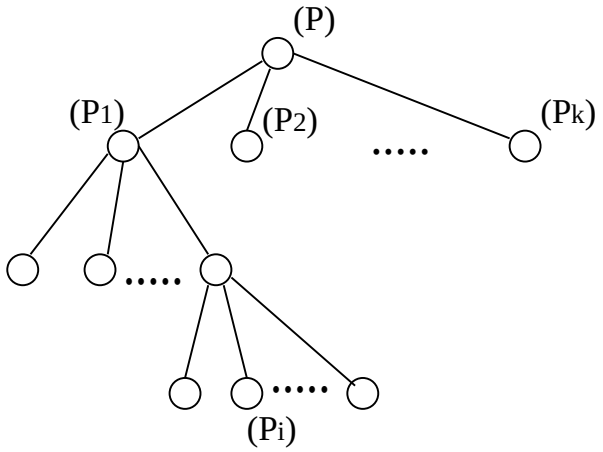

図 4: 部分問題の木構造

次に分枝限定法を構成している分枝操作と限定操作について説明します。問題 (P_i) が直接解けない場合、その実行可能領域 S_i を $S_i = \cup_{\ell=p}^{q} S_\ell$ となるいくつかの部分集合 $S_p, \ldots, S_q$ に分割して、(P_i) をその一群の子問題：

$$(P_\ell) \left| \begin{array}{ll} \textbf{maximize} & f(\boldsymbol{x}) \\ \textbf{subject to} & \boldsymbol{x} \in S_\ell \end{array} \right\} \ell = p, \ldots, q$$

でおきかえるのが 分枝操作 です。この様子は、図 4 のように、木 (tree) 構造に示すことができ、これは列挙木と呼ばれています。ある部分問題が解ければ、その問題の分枝はその時点で終了です。問題 (P_i) を分枝していく過程で、その時点までに解いた子問題の中で最大の目的関数値を達成している実行可能解を 暫定解 とよび $\hat{\boldsymbol{x}}$ で表し、また、その目的関数値 $f(\hat{\boldsymbol{x}})$ を 暫定値 といいます。より目的関数値の大きい実行可能解が分かるたびに、暫定解を更新します。

通常、ある部分問題 (P_i) を実際に分枝する前に、ある計算を行なって分枝する必要があるか否かを調べます。これは 限定操作 と呼ばれています。限定操作のためには、その部分問題の緩和問題

$$(\bar{P}_i) \left| \begin{array}{ll} \textbf{maximize} & f(\boldsymbol{x}) \\ \textbf{subject to} & \boldsymbol{x} \in \bar{S}_i \end{array} \right.$$

を構築します。ここで、$\bar{S}_i$ は S_i を含む集合であり、$(\bar{P}_i)$ が簡単に解けるように構築します。(P_i) が整数計画問題の場合には、その整数条件を取り除いた線形計画問題が緩和問題 $(\bar{P}_i)$ として利用されることが多々あります。この緩和問題を解いて、最適解 $\bar{\boldsymbol{x}}_i$ を計算します。$f(\bar{\boldsymbol{x}}_i)$ は問題 (P_i) の最適値の上界値を与えます。従って、

1. $\bar{\boldsymbol{x}}_i \in S_i$ であれば、$\bar{\boldsymbol{x}}_i$ は (P_i) の最適解となります。

2. 暫定解 $\hat{\boldsymbol{x}}$ に対して $f(\bar{\boldsymbol{x}}_i) \leq f(\hat{\boldsymbol{x}})$ であれば、(P_i) の最大値は暫定値 $f(\hat{\boldsymbol{x}})$ 以下であることが分かります。

3. $\bar{S}_i$ が空集合ならば、S_i も空と分かります。

このように、１，２，３の場合には、部分問題 (P_i) はこれ以上、分枝する必要はなく、その子問題はそこで終了します。２の場合には、問題 (P_i) のいかなる実行可能解も $\hat{x}$ より良い目的関数値を与えられないことが分かり、(P_i) をこれ以上分枝することは無意味となります。この場合には、“限定操作によって見切られて分枝停止になった (見切りがついた)” といいます。

以上のような分枝操作と限定操作を全ての部分問題が終了するまで繰り返します。

アルゴリズム (分枝限定法のプロトタイプ)

ステップ 1： (初期設定)

$\hat{z} = -\infty$, $\mathcal{N} := \{(P)\}$, $p := 1$ とする。

ステップ 2： (最適性判定)

$\mathcal{N} := \phi$ なら終了。

ステップ 3： (部分問題の選択)

$\mathcal{N}$ から問題を選び、それを (P_k) とする。$\mathcal{N} := \mathcal{N} \setminus \{(P_k)\}$

ステップ 4： (限定操作)

緩和問題 $(\bar{P}_k)$ を定め、それを解く。

4.1: 緩和問題 $(\bar{P}_k)$ が実行可能解を持たないときはステップ 2 へ ((P_k) の分枝終了)。

4.2: 緩和問題 $(\bar{P}_k)$ が有界の場合には，最適解 $\bar{x}_k$ と最適値 $\bar{z}_k = f(\bar{x}_k)$ を求める。

- $\bar{z}_k \leq \hat{z}$ ならばステップ 2 へ ((P_k) の分枝終了)。
- $\bar{x}_k$ が (P_k) の実行可能解でないならば、$\bar{x}_k$ を加工して実行可能解 $\hat{x}_k$ と $\hat{z}_k = f(\hat{x}_k)$ を求める。

ステップ 5： (更新・分枝)

5.1： (更新操作) $\hat{z}_k > \hat{z}$ ならば $\hat{z} = \hat{z}_k$, $\hat{x} = \hat{x}_k$ とする。$\bar{x}_k$ が (P_k) の実行可能解のとき、ステップ 2 へ ((P_k) の分枝終了)。

5.2： (分枝操作)

(P_k) の子問題 $(P_p), \ldots, (P_q)$ を作り、

$$\mathcal{N} := \mathcal{N} \cup \{(P_p), \ldots, (P_q)\}, \quad p := q + 1$$

としてステップ 2 へ。

ステップ 2 において $\mathcal{N} := \phi$ で終了した場合、$\hat{z} < \infty$ なら対応する解は P の最適解となり、$\hat{z} = \infty$ なら P は実行可能解を持たない (つまり、非有界) とわかります。$\mathcal{N}$ は未分枝問題の集合とよばれ、まだ子問題に分解されておらず、分枝停止となっていないような問題の集まりを指しています。

8.3 分枝限定法の計算効率

分枝限定法の計算効率に大きく影響を及ぼす要因として **(1)** よい上界値を与えるような緩和問題をどのように作るか、**(2)** 下界を与える実行可能解をどのように求めるか、**(3)** どのように分枝するか、**(4)** 生成された部分問題をどのような順番で処理するかを挙げられます。

(1) 緩和問題の作り方： 良い上界値 (緩和問題の最適値) が得られると、多くの子問題を限定操作で排除できます。つまり、(P) の暫定値 (過去に求められた (P) の最良の実行可能解に対応する目的関数の値) を $f(\hat{x})$ として、ある子問題 (P_k) に対する緩和問題 $(\bar{P}_k)$ の最適値を $f(\bar{x}_k)$ とすると、分枝限定法では $f(\bar{x}_k) \leq f(\hat{x})$ なら (P_k) を見切って手間を省くことができるので、なるべく小さい上界値を求める方法が必要となります。より小さい上界値を得ようとするとかなりの計算量が必要となるため、“計算が簡単で良い上界値” を求めるための適切な緩和問題の構築法が望まれています。

緩和問題の作り方については、様々な組合せ問題に対して、問題の特徴をいかした緩和手法が提案されています。ここでは，連続緩和法とを紹介します。

連続緩和問題

全整数計画問題：

$$(P) \left| \begin{array}{ll} \textbf{maximize} & \boldsymbol{c}^T\boldsymbol{x} \\ \textbf{subject to} & A_1\boldsymbol{x} \leq \boldsymbol{b}^1 \quad (A_1 \in R^{m_1 \times n}) \\ & A_2\boldsymbol{x} = \boldsymbol{b}^2 \quad (A_2 \in R^{m_2 \times n}) \\ & \boldsymbol{x} \geq \boldsymbol{0}, \qquad \boldsymbol{x} \in Z^n \end{array} \right.$$

の場合には、(P) から変数の整数条件 $\boldsymbol{x} \in Z^n$ を取り除いた問題：

$$(\bar{P}) \left| \begin{array}{ll} \textbf{maximize} & \boldsymbol{c}^T\boldsymbol{x} \\ \textbf{subject to} & A_1\boldsymbol{x} \leq \boldsymbol{b}^1, A_2\boldsymbol{x} = \boldsymbol{b}^2, \boldsymbol{x} \geq \boldsymbol{0} \end{array} \right.$$

は線形計画問題となって、(P) に比べてはるかに簡単に解くことができます。(P) とその緩和問題 $(\bar{P})$ の間には、明らかに次の関係が成立しています。

定理 8.1. *(緩和法の原理)*

(i) $(\bar{P})$ が最適解 $\bar{\boldsymbol{x}}$ を持ち、$\bar{\boldsymbol{x}} \in Z^n$ ならば $\bar{\boldsymbol{x}}$ は (P) の最適解である。

(ii) $(\bar{P})$ が実行可能解を持たないならば、(P) も実行可能解を持たない。

(iii) $(\bar{P})$ の最適解を $\bar{\boldsymbol{x}}$ とし (P) の最適解を $\boldsymbol{x}^*$ とすると $\boldsymbol{c}^T\bar{\boldsymbol{x}} \geq \boldsymbol{c}^T\boldsymbol{x}^*$ である。

$(\bar{P})$ は (P) の連続緩和問題、と呼ばれています。

(2) 実行可能解の求め方： 良い下界値 (実行可能解での目的関数値) を得られると、多くの子問題を限定操作で排除できます。上界値と同様に、良い下界値ほど多くの計算量が必要となるため、“計算が簡単で良い下界値” を用います。緩和問題の最適解を整数解に丸めることにより、簡単に整数計画問題の実行可能解が得られる場合もあります。

(3) 分枝の仕方： 分枝操作について、少し説明を加えたいと思います。いろいろな分枝の仕方が考えられますが、列挙木があまり大きくならないようにするために、ある問題から生成される子問題の個数は **2** 個ないし **3** 個であるように設計するのが普通です。整数計画問題に対して、二つの分枝操作の例を挙げておきます。

例：整数計画問題

$$\begin{aligned} &\textbf{maximize} \quad \boldsymbol{c}^T\boldsymbol{x} \\ &\textbf{subject to} \quad \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \\ &\qquad\qquad x_j \in \{\ell_j,\ \ell_j+1, \ldots,\ u_j\} \quad (j=1,\ldots,n) \end{aligned}$$

ただし、ℓ_j, u_j は $\ell_j \leq u_j$ なる整数とします。

【分枝構造 **1**】

$k \in \{1,2,\ldots,n\}$ を一つ決めて、$u_k - \ell_k + 1$ 個の場合：

$$x_k = \ell_k,\ \ x_k = \ell_k + 1, \ldots,\ \ x_k = u_k,$$

に分けます。

【分枝構造 **2**】

整数条件を外した線形計画問題の最大解を $\bar{\boldsymbol{x}} = (\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n)$、最大値を $\bar{z}$ とします。このとき、整数計画問題の最大値の上界値となります。すべての要素 $\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n$ が整数ならば $\bar{\boldsymbol{x}}$ が整数計画問題の最大解となり、分枝する必要はありません。$\bar{x}_k$ が整数でないと仮定したとき、**2** つの場合**:**

$$x_k \leq [\bar{x}_k], \quad [\bar{x}_k] + 1 \leq x_k$$

に分けます。$[a]$ は a の整数部を示しています。

(4) 部分問題の選択： 生成される部分問題 (未分枝問題) をどんな順番で解いたらよいのか、代表的な探索法を **3** つ挙げておきます。また、これらを適当に組み合わせた様々な探索法が提案されています。

最良上界 (下界) 探索：列挙木の成長を抑えるためには早い段階でよい暫定解を見つけることが **1** つの要点です。最良上界 (下界) 探索は、すべての未分枝問題の中から最も良い実行可能解を生成する公算の高いものを優先して選んでくる方法です。そのために、未分枝問題の中で最大 (最小) の上界値 (下界値) を持つ問題を選び出します。思惑通り良い実行可能解が求まれば、暫定値が改善されて未分枝問題の数が一挙に減る可能性があります。しかしその一方で、もし良い実行可能解が生成されないと一般に列挙木は横に広がり、計算効率が大幅に低下してしまうという欠点があります。

幅優先探索：すべての未分枝問題の中から列挙木の最も浅い位置にある子問題を選びます。よい実行可能解がどの子問題から生じても見逃すことはありませんが、最初の実行可能解が

なかなか見つからないで、限定操作が有効に働かない危険性があります。上の２つの探索は、未分枝問題として保持される問題数が多くなる傾向があるので、大きな記憶場所を必要とします。

深さ優先探索：最も新しく生成された子問題の中から次に探索すべき問題を選ぶ方法です。深さ優先探索では目的関数の値を全く考慮せずに分枝頂点を選択するので、良い実行可能解の見つかる見込のない探索を深く続ける危険性を持っています。しかし、計算途中で記憶すべき問題数を現在探索中の子問題の深さに比例する程度に抑えることができるという長所を持つため、最良上界 (下界) 探索や幅優先探索と比べてプログラムが容易であり、通常は深さ優先探索を用いることが多いです。

最良上界 (下界) 探索の場合には、分枝限定法のアルゴリズムを修正して、未分枝問題の集合 $\mathcal{N}$ に子問題を入れる前にこの上界値 (下界値) を計算します。そして次に解く問題を選ぶとき、つまりステップ 3 では、$\mathcal{N}$ の中で最大の上界値 (最小の下界値) をもつ問題を選び出します。

8.4 演習問題

演習問題 8.1. 施設配置問題について、定式化しなさい。 何を変数にしたか、明記すること。

演習問題 8.2. 「分枝操作」を行なう理由を述べ、さらに分枝操作において具体的にどのようなことをするのか説明せよ。

演習問題 8.3. 「限定操作」を行なう理由を述べ、さらに限定操作において具体的にどのようなことをするのか説明せよ。

演習問題 8.4. 「緩和問題」を解く目的を述べ、さらに緩和問題が満たすべき条件を挙げなさい。

演習問題 8.5. 集合 $M = \{1, 2, \ldots, m\}$ と n 個の M の部分集合である $S_j \subseteq M$ $(j = 1, 2, \ldots, n)$ が与えられているとする。S_j を幾つか選んでその和集合が M の全ての要素を含むとき、M の被覆であるという。つまり添字の部分集合 $X \subseteq \{1, 2, \ldots, n\}$ に対して $\cup_{j \in X} S_j = M$ がこのとき成り立つ。各集合 S_j に対して費用 c_j が与えられた時、費用の総和が最小となる M の被覆 X を求める問題を**集合被覆問題**と呼ぶ。この問題を $0-1$ 整数計画問題として定式化せよ。

ヒント：$i \in S_j$ $(i = 1, 2, \ldots, m; j = 1, 2, \ldots, n)$ ならば $a_{ij} = 1$、そうでなければ $a_{ij} = 0$ を定義すれば良い。

演習問題 8.6. 演習問題 *8.5* の定義にて、さらに $S_j \cap S_k = \emptyset,\ \forall j, k \in X,\ j \neq k$ の条件を課すると**集合分割問題**と呼ばれる。この問題も $0-1$ 整数計画問題として定式化せよ。

演習問題 8.7. コンビニへの配送計画、郵便や新聞の配達など、複数の車両を用いて全ての顧客に荷物を集配送する問題を考える。各車両はデポと呼ばれる特定の地点を出発していくつかの顧客を訪問した後、再びデポに戻る。この時、1 台の車両が訪問する客の順序をルートと呼ぶ。顧客の集合を $M = \{1, 2, \ldots, m\}$ とする。1 台の車両で配送できるルートを全て列挙したものを $S_j\ (j = 1, 2, \ldots, n)$ とする。車両の積載能力、顧客間の移動時間および移動費用を総括したコストを各ルートに対して $c_j\ (j = 1, 2, \ldots, n)$ とおくと、この問題は集合被覆問題、集合分割問題のいずれの問題として定式化できるのか答えよ。