第10回　整数計画法II
— 分枝限定法と動的計画法 —

10.1　分枝限定法

しらみつぶしに実行可能解について調べるために場合分けを行う際に，不必要な場合分けを省略して計算量を削減する方法．ここでは，ナップザック問題を例にして分枝限定法を説明する．

まず，前節の問題：

$$\begin{aligned} \text{目的関数} &: 7x_1 + 8x_2 + x_3 + 2x_4 \rightarrow \text{最大化} \\ \text{制約条件} &: 4x_1 + 5x_2 + x_3 + 3x_4 \leq 6 \\ & x_i \in \{0, 1\} \end{aligned}$$

に対して，

$$\begin{aligned} \text{目的関数} &: 7x_1 + 8x_2 + x_3 + 2x_4 \rightarrow \text{最大化} \\ \text{制約条件} &: 4x_1 + 5x_2 + x_3 + 3x_4 \leq 6 \\ & 0 \leq x_i \leq 1 \ (1 \leq i \leq n) \end{aligned}$$

という問題を考える．この問題は，元の問題の 0-1 条件を，＿＿＿＿＿＿＿＿＿＿ でよいという条件に緩めた問題である．このように作った問題を，＿＿＿＿＿＿＿＿＿＿ という．この問題は，欲張り法を用いて最適解を求めることができる．つまり，最適解は $\mathbf{x} = (1, 2/5, 0, 0)$ となる．これを，もとのナップザック問題の ＿＿＿＿＿＿＿＿ という．実数最適解は次の性質を持つ．

- [0-1 問題の実行可能領域] ＿＿ [連続緩和問題の実行可能領域]
  であるから，
  [0-1 問題の最大値] ＿＿ [連続緩和問題の最大値]
  となっている．つまり，実数最適解における目的関数の値は，0-1 問題における最大値の**上界**となる．

- 実数最適解の変数が全て 0-1 条件を満たしている場合，これは 0-1 問題の最適解である．

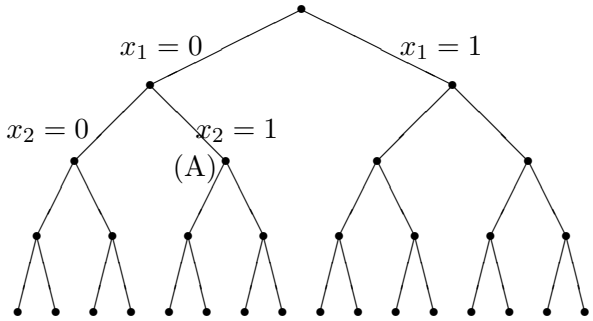

図 10.1: 分枝図

- 連続緩和問題が実行可能解を持たないとき，0-1 問題は＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

さて，0-1 問題の最適解は 2^n 個の実行可能解の中に存在する．それらを全て調べ尽くすには，図 10.1 に示すような**分枝図**を使うとよい.

- 図中の一番上の点は元の 0-1 問題を示す.
- ２段目の２つの点は x_1 をそれぞれ０か１に固定した場合の問題を示す．同様に３，４段目の点は x_1, x_2 および x_1, x_2, x_3 を固定して残りの変数についての問題を示す.
- 最下段の点は全ての変数を固定した状態を示す．つまり，これらの点の中で実行可能解を示すものについて，目的関数の値を計算して比べることによって，最適解を求めることができる.

2,3,4 段目の点が示す問題を，元の問題の ＿＿＿＿＿＿ と呼び，0 に固定されている変数の添え字の集合 J_0 と，1 に固定されている変数の添え字の集合 J_1 によって $P(J_0, J_1)$ と書く．たとえば，図中の (A) の点は $J_0 = \{1\}, J_1 = \{2\}$ に対応し，上に示したナップザック問題の例に対する部分問題 $P(\{1\}, \{2\})$ は

$$\begin{aligned} \text{目的関数} &: 8 + x_3 + 2x_4 \rightarrow \text{最大化} \\ \text{制約条件} &: 5 + x_3 + 3x_4 \leq 6 \\ &\quad x_i \in \{0, 1\} \end{aligned}$$

と書ける．この問題も，ナップザック問題となっていることに注意.

分枝限定法でナップザック問題を解くには,

1. まず根にあたる元の問題 $P(\emptyset, \emptyset)$ の連続緩和問題を解いてみて，実数最適解を求める．実数最適解が 0-1 条件を満たしているならば ＿＿＿＿＿＿＿＿＿＿ ので終了．そうでなければ,（欲張り法などを用いて）近似最適解を求め，暫定解とする．また，そのときの目的関数の値を暫定値とする.

2. １つの変数を 0 および 1 に固定した部分問題 $P(\{1\},\emptyset), P(\emptyset,\{1\})$ を作成する．

3. 部分問題の中の１つ $P(J_0, J_1)$ を選び，その問題の連続緩和問題を解く．その結果により，以下の４つの場合が考えられる．

   (a) **連続緩和問題に実行可能解が存在しない場合**: 部分問題 $P(J_0, J_1)$ にも実行可能解が存在しない．

   (b) **連続緩和問題によって得られた実数最適解が 0-1 条件を満たす場合**: 実数最適解は $P(J_0, J_1)$ の最適解となる．最適解の目的関数の値が暫定値より大きいならば，その解はより良い暫定解とすることができる．

   (c) **連続緩和問題によって得られた $P(J_0, J_1)$ の上界値が，現在の暫定値よりも小さい場合**: $P(J_0, J_1)$ の実行可能解の中には元の問題の最適解が存在することはない．

   (d) **連続緩和問題によって得られた $P(J_0, J_1)$ の上界値が，現在の暫定値よりも大きく，実数最適解が 0-1 条件を満たさない場合**: $P(J_0, J_1)$ の実行可能解の中に最適解が存在する可能性がある．

   (a)(c) では，注目した部分問題の実行可能解の中には元の問題の最適解が存在することはないことがわかるので，これ以上この部分問題を調べて厳密に解くことをやめる．また，(b) では注目した部分問題を厳密に解くことができたのでさらに部分問題を調べる必要がない．このように，さらなる部分問題を調べないでおわることを，部分問題の _____ という．(d) の場合は，注目した部分問題をさらに厳密に解く必要がある．そこで，$P(J_0, J_1)$ の自由変数を１つ選んで 0 または 1 に固定することによって２つの部分問題を新たに追加する．これを分枝操作という．

4. 全ての部分問題が終端してしまったら終了．そうでなければ 3. に戻る．

ある時点で，終端されていない部分問題を _____ といい，その集合を $\mathcal{A}$ で表す．具体例をあげよう．先に示した問題例を考える．

$$\begin{aligned} \text{目的関数} &: 7x_1 + 8x_2 + x_3 + 2x_4 \rightarrow \text{最大化} \\ \text{制約条件} &: 4x_1 + 5x_2 + x_3 + 3x_4 \leq 6 \\ &\quad x_i \in \{0, 1\} \end{aligned}$$

(0) 近似最適解は欲張り法を用いて _____ とする．つまり，暫定解は _____ で暫定値は ____ ． ２つの部分問題 _____, _____ を作る．$\mathcal{A}$ =_____ である．

(1) 活性部分問題 $P(\{1\},\emptyset)$ を選ぶ．これは，$x_1 = 0$ と置くことで，

$$\begin{aligned} \text{目的関数} \quad &: \quad _____ \rightarrow \text{ 最大化} \\ \text{制約条件} \quad &: \quad _____ \\ & \quad x_i \in \{0,1\} \end{aligned}$$

となる．この問題の連続緩和問題を解くと（欲張り法），実数最適解は $(x_2, x_3, x_4) =$_____ となり，0-1 条件を満たしている．したがって，上の ____ の場合である．$P(\{1\},\emptyset)$ における目的関数の最大値は 9 となり，現在の暫定値より大きいので，$(0,1,1,0)$ を暫定解とし，9 を暫定値とする．$P(\{1\},\emptyset)$ は終端できるので，$\mathcal{A} = \{P(\emptyset,\{1\})\}$ となる．

(2) 活性部分問題 $P(\emptyset,\{1\})$ を選ぶ．これは，

$$\begin{aligned} \text{目的関数} \quad &: \quad _____ \rightarrow \text{ 最大化} \\ \text{制約条件} \quad &: \quad _____ \\ & \quad x_i \in \{0,1\} \end{aligned}$$

となる．この問題の連続緩和問題を解くと，実数最適解は $(x_2, x_3, x_4) =$ _____ となり，上界は _____ である．したがって，上の ____ の場合である．$P(\emptyset,\{1\})$ の部分問題によって最適解が得られる可能性があるので，部分問題 _____ , _____ を作る．$\mathcal{A} =$ _____ となる．

(3) $P(\{2\},\{1\})$ を選ぶ．これは

$$\begin{aligned} \text{目的関数} \quad &: \quad _____ \rightarrow \text{ 最大化} \\ \text{制約条件} \quad &: \quad _____ \\ & \quad x_i \in \{0,1\} \end{aligned}$$

である．実数最適解は $(x_3, x_4) =$_____ となり，上界は _____ である．したがって，上の ____ の場合であり，最適解が得られる可能性はないので終端．$\mathcal{A} = \{P(\emptyset,\{1,2\})\}$.

(4) $P(\emptyset,\{1,2\})$ を選ぶ.

$$\begin{aligned} \text{目的関数} \quad &: \quad 7 + 8 + x_3 + 2x_4 \rightarrow \text{ 最大化} \\ \text{制約条件} \quad &: \quad 4 + 5 + x_3 + 3x_4 \leq 6 \\ & \quad x_i \in \{0,1\} \end{aligned}$$

となる．この問題は実行可能解を持たない．したがって，上の ____ の場合であり終端.

(5) 活性部分問題がなくなったので，現在の暫定解 $(0,1,1,0)$ を最適解として終了.

分枝限定法では，基本的には全ての解を列挙して目的関数の値を調べていることと等しいので，かならず最適解を求めることができる.

ここで取り上げたナップザック問題の場合，分枝操作によって必ず２つの部分問題が生成されたが，一般には３つ以上の部分問題が生成されることもある．また，ここでは各部分問題の目的関数の上界は実数最適解によって求めたが，計算量を削減するためには与えられた問題ごとにいろいろな工夫が必要.

- 効率的な上界値の計算方法
- 活性部分問題の探索法（複数ある活性部分問題のうち，どれを先に選択するか？）.
  深さ優先探索法 $\cdots$ 元の問題から見て，最も深いものを選ぶ.
  最良優先探索法 $\cdots$ 各活性部分問題における目的関数の上界値の推定値を求め，それが最大のものを選ぶ.

10.2 動的計画法

最適性の原理：ある問題と，それを分解した部分問題を考えると，元の問題の最適解は，部分問題の最適解を含んでいる.

最適性の原理を利用して組合せ最適化問題をとく手法を，**動的計画法**と呼ぶ.（動的計画法も列挙法の１つであり，必ず最適解が求まる.）たとえば，最短路問題の解法のひとつであるダイクストラ法は,「全体のネットワークにおける最短路は，その部分グラフにおいても最短路を与える」という性質を使っているので，動的計画法である.

別の動的計画法の例として，ここでは巡回セールスマン問題を例に挙げて説明する.

[問題 10.1] [巡回セールスマン問題] あるネットワーク $G=(V,E)$,$V=\{1,2,\ldots,n\}$ において，全ての節点をちょうど一回づつ通る巡回路のうち，もっとも長さの短いものを求めよ．ただし，各枝 (i,j) には枝の長さ a_{ij} が与えられているとする.（ここでは，無向グラフを扱う.）

一般性を失うことなく，出発点を節点１とする．巡回路は，出発店から最後に訪問する節点までの（すべての節点を通る）路と，最後に訪問する節点から出発点へ直接到達する枝とに分けることができる．ここで，出発点（節点１）から，$S\subset\{2,3,\ldots,n\}$ に含まれる全ての節点を通って節点 i に到達する路のうち，最短の長さを $F(S,i)$ で表すことにする．すると，求めたい巡回路の長さ f^* は,

$$f^*=\min_{i\in\{2,3,\ldots,n\}}(F(\{2,3,\ldots,n\},i)+a_{i1})$$

と書ける．

では，$F(S,i)$ はどのようにして求めたらよいか？この路は，出発点から i の直前に訪問する節点 j までの（S から i を除いたすべての節点を通る）路と，節点 j から節点 i へ直接到達する枝とに分けることができる．すなわち，

$$F(S,i) = \min_{j \in S \setminus \{i\}} (F(S \setminus \{i\}, j) + a_{ji})$$

と書ける．

このように，最短路の長さはより小さい問題の最短路から求めることができ，最終的には，$|S| = 1$ である問題まで分解することができる．$S = \{i\}$ の場合，明らかに $F(\{i\}, i) = a_{1i}$ である．したがって，$|S| = 1$ である集合 S から順に $F(S,i)$ を求めていけばよい．

具体的な手順は以下のとおり．

1. 全ての $i \in \{2, 3, \ldots, n\}$ に対して，

   $$F(\{i\}, i) = a_{i1}, \quad P(\{i\}, i) = 1$$

   とする．$k = 2$ とする．

2. 全ての $|S| = k$ である節点集合 $S \subseteq \{2, 3, \ldots, n\}$ と，全ての節点 $i \in S$ に対して，

   $$F(S,i) = \min_{j \in S \setminus \{i\}} (F(S \setminus \{i\}, j) + a_{ji})$$

   を求める．また，

   $$P(S,i) = \text{最小値を与える } j$$

   とする．

3. $k < n - 1$ ならば $k \leftarrow k + 1$ として，step 2 にもどる．$k = n - 1$ ならば，

   $$\begin{aligned} f^* &= \min_{i \in \{2,3,\ldots,n\}} (F(\{2, 3, \ldots, n\}, i) + a_{1i}), \\ p^* &= \text{最小値を与える } i \end{aligned}$$

   とし，終わる．

$P(S,i)$ は，節点 1 から出発して S の全ての節点を通って i へたどり着く最短路の，i の直前の節点番号を格納している．

例として，図 10.2 を解いてみよう（$n = 4$ である）．

(1) まず，

$$\begin{aligned} F(\{2\}, 2) &= a_{12} = 7, \quad P(\{2\}, 2) = 1 \\ F(\{3\}, 3) &= a_{13} = 4, \quad P(\{3\}, 3) = 1 \\ F(\{4\}, 4) &= a_{14} = 8, \quad P(\{4\}, 4) = 1 \end{aligned}$$

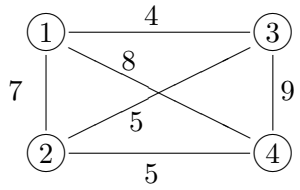

図 10.2: 巡回セールスマン問題の例

(2) $k = 2$.

$$
\begin{aligned}
&F(\{2,3\},2) = F(\{3\},3) + a_{32} = 9, \quad P(\{2,3\},2) = 3\\
&F(\{2,3\},3) = F(\{2\},2) + a_{23} = 12, \quad P(\{2,3\},3) = 2\\
&F(\{2,4\},2) = F(\{4\},4) + a_{42} = 13, \quad P(\{2,4\},2) = 4\\
&F(\{2,4\},4) = F(\{2\},2) + a_{24} = 12, \quad P(\{2,4\},4) = 2\\
&F(\{3,4\},3) = F(\{4\},4) + a_{43} = 17, \quad P(\{3,4\},3) = 4\\
&F(\{3,4\},4) = F(\{3\},3) + a_{34} = 13, \quad P(\{3,4\},4) = 3
\end{aligned}
$$

(3) $k = 3$.

$$
\begin{aligned}
F(\{2,3,4\},2) &= \min(F(\{3,4\},3) + a_{32}, F(\{3,4\},4) + a_{42}) = 18\\
P(\{2,3,4\},2) &= 4\\
F(\{2,3,4\},3) &= \min(F(\{2,4\},2) + a_{23}, F(\{2,4\},4) + a_{43}) = \underline{\quad\quad}\\
P(\{2,3,4\},3) &= 2\\
F(\{2,3,4\},4) &= \min(\underline{\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad})\\
&= \underline{\quad\quad}\\
P(\{2,3,4\},4) &= \underline{\quad\quad}
\end{aligned}
$$

(4) $k = n - 1$ となっているので,

$$
\begin{aligned}
f^* &= \min(F(\{2,3,4\},2) + a_{21}, F(\{2,3,4\},3) + a_{31}, F(\{2,3,4\},4) + a_{41})\\
&= \underline{\quad\quad}\\
p^* &= \underline{\quad\quad}
\end{aligned}
$$

として終わる.

最短巡回路は，p^* から逆にたどって (_____) となり，そのときの長さは _____ である.

上の例でわかるように，各繰り返しにおいて ${}_{n-1}C_k \times k$ 個の $F(S, i)$ を求める必要があり, このアルゴリズムは n の多項式時間では解けない．巡回セールスマン問題をもっと効率的に解くアルゴリズムが存在するが，それでも多項式時間で解くことはできないことが分かっている．