第９回：１２月２日（水）

9　第９回

9.1　分枝限定法 branch-and-bound method

今回の講義では，広汎な離散最適化問題を厳密に解くためのアルゴリズム手法である，**分枝限定法**について説明する．

分枝限定法は，本質的にはすべての解を総当たりに調べ，実行可能解のうち最良の目的関数値を達成するものを求めるだめの手法だが，**分枝操作**および**限定操作**と呼ばれる二つの操作を用いて解の探索を行う．

- 分枝操作 (branching operation): 元の問題あるいは部分問題を，いくつかの部分問題に分割する操作．

- 限定操作 (bounding operation) 部分問題の計算を終端させる操作．例えば，
  部分問題の最適値が分かった，あるいは
  部分問題が元の問題の最適値を与えないことが分かった
  ときなどである．

分枝操作は,「すべての解を総当たりに調べる」ことを系統的に行うための操作である．ただし「すべての解を総当たりに調べる」ことは一般に指数時間を要するため，実際にかかる計算時間を少しでも短くしたい．限定操作は，部分問題を解く必要があるかどうかを上記のような根拠に基づいて判定し，もし解く必要がなければその部分問題を終端するための操作である．

以下では 0,1-ナップサック問題（最大化問題）を対象として分枝限定法を説明する．前回同様，n 個の要素集合を $V = \{1, \ldots, n\}$, 各要素 $j \in V$ に対するサイズおよび利得をそれぞれ a_j と利得 c_j, ナップサックのサイズを b とする．

9.2　0,1-ナップサック問題の LP 緩和問題

LP 緩和問題

最適化問題 Π に対し，その制約条件を「緩める」ことで得られる最適化問題 $\bar{\Pi}$ を，Π の**緩和問題** (relaxation problem) という．（一般の LP や ILP のように）制約式が明示的に与えられた問題の場合，一部の制約式を取り除くことで緩和問題が得られる．また，変数が取り得る値の範囲が制約条件として課されている場合，その範囲を（元の範囲を含んだまま）広げることで緩和問題が得られる．

特に ILP 問題に対し，その整数制約（変数は指定された整数値のいずれかを取らなければならないという制約）のすべてを，その変数が取り得る実数値の範囲に関する制約に緩和すると LP 問題が得られるが，この問題を（元の ILP に対する）**LP 緩和問題**という．

Π を最大化問題，$\bar{\Pi}$ をその緩和問題とする．以下の二点は重要な性質である．

1. **$\bar{\Pi}$ の最適値は，Π の最適値以上となる．すなわち，前者は後者の上界となる．**このことは，$\bar{\Pi}$ の実行可能領域が Π の実行可能領域を含むことから理解されよう．

2. $\bar{\Pi}$ の最適解 x^* が Π の実行可能解のとき，x^* は Π の最適解でもある．

0,1-ナップサック問題の LP 緩和問題に対する貪欲法

0,1-ナップサック問題を例として分枝限定法を説明するが，元の問題および部分問題の最適値の上界を得るために LP 緩和問題を用いる．0,1-ナップサック問題の LP 緩和問題は，単位サイズあたり利得 c_j/a_j に関する貪欲法を用いて解けることを示す．

証明は LP 問題に関する強双対定理を用いて示す．0,1-ナップサック問題の LP 緩和問題を $\bar{P}_0$ とするが，証明の都合上，最小化問題として捉える．

問題 $\bar{P}_0$	最小化	$-\sum_{j=1}^{n} c_j x_j$
	制約条件	$\sum_{j=1}^{n} a_j x_j \le b,$
		$0 \le x_j \le 1,\ j = 1, \ldots, n.$

定理. 問題 $\bar{P}_0$ において, $c_1/a_1 \ge \cdots \ge c_n/a_n$ が成り立つとする. このとき q を,

$$\sum_{j=1}^{q-1} a_j \le b \quad \text{かつ} \quad \sum_{j=1}^{q} a_j > b \tag{5}$$

を満たす要素とすると, 問題 $\bar{P}_0$ の最適解は次式で与えられる.

$$x_j = \begin{cases} 1 & j = 1, \ldots, q-1 \text{ のとき}, \\ (b - \sum_{j=i}^{q-1} a_i)/a_q & j = q \text{ のとき}, \\ 0 & j = q+1, \ldots, n \text{ のとき}. \end{cases} \tag{6}$$

証明. 強双対定理を用いて示す. $\bar{P}_0$ の双対問題は以下の通りである.

問題 $\bar{D}_0$	最大化	$-b\lambda - \sum_{j=1}^{n} \mu_j$
	制約条件	$\mu_j + \lambda a_j \ge c_j,\ j = 1, \ldots, n,$
		$\lambda, \mu_j \ge 0,\ j = 1, \ldots, n.$

強双対定理によれば, $\bar{P}_0$ の実行可能解と $\bar{D}_0$ の実行可能解の目的関数値が等しいとき, 両実行可能解はそれぞれの問題における最適解である. 以下の解 $(\lambda, \mu_1, \ldots, \mu_n)$ は $\bar{D}_0$ に対する実行可能解である.

$$\lambda = \frac{c_q}{a_q}, \quad \mu_j = \begin{cases} c_j - \lambda a_j & j = 1, \ldots, q \text{ のとき}, \\ 0 & j = q+1, \ldots, n \text{ のとき}. \end{cases}$$

この解の目的関数値は

$$\begin{aligned} -\lambda b - \sum_{j=1}^{n} \mu_j &= -\frac{c_q b}{a_q} - \sum_{j=1}^{q-1} (c_j - \frac{a_j c_q}{a_q}) \\ &= -\sum_{j=1}^{q-1} c_j - (b - \sum_{j=1}^{q-1} a_j) \frac{c_q}{a_q} \end{aligned}$$

となり, 式 (6) で与えられる $\bar{P}_0$ の実行可能解の目的関数値と一致する. □

すなわち，$c_1/a_1 \geq c_2/a_2 \geq \cdots \geq c_n/a_n$ が成り立つとき，$j = 1, 2, \ldots$ の順に要素 j を「すべて」ナップサックに詰め,「すべて」が入りきらない最初の要素 $j = q$ (この q は式 (5) を満たす) については，ナップサックのサイズが一杯になるまでその一部を詰める．そのようにして得られる解 (式 (6)) は，LP 緩和問題 $\bar{P}_0$ の最適解となるのである.

以下では 0,1-ナップサック問題 P の LP 緩和問題の最適値を $f_{\mathrm{LP}}(P)$ と記す．すなわち

$$f_{\mathrm{LP}}(P) \triangleq \max \left\{ \sum_{j=1}^{n} c_j x_j \mid \sum_{j=1}^{n} a_j x_j \leq b, 0 \leq x_j \leq 1, \ j = 1, \ldots, n \right\}.$$

9.3 0,1-ナップサック問題に対する分枝限定法

0,1-ナップサック問題 P の最適値を $f(P)$ と記す．すなわち

$$f(P) \triangleq \max \left\{ \sum_{j=1}^{n} c_j z_j \mid \sum_{j=1}^{n} a_j z_j \leq b, z_j \in \{0, 1\}, \ j = 1, \ldots, n \right\}.$$

0,1-ナップサック問題に対する分枝限定法の設計例を示そう.

分枝操作

分枝操作は，変数 z_j を 1 か 0 に固定することとする．これにより問題が 2 個生成される．再帰的に分枝操作を適用すると，未決定変数が無くなったときには合計 2^n 個の問題が生成される．この分枝操作の問題生成の過程は完全二分木の形になっている．これを分枝木と呼び，分枝木の点をノード (node) や節点と呼ぶ．各ノードに対する問題をもとの問題の部分問題と呼ぶ.

♠「講義資料と課題」2 ページ: 0,1-ナップサック問題に対する分枝木 $(n = 4)$　このスライドでは $n = 4$ の場合の分枝木を示す.

- この木の根に対応する問題 1 は，本来解きたい元の問題に対応する.
- 問題 1 について，要素 4 に関する変数 z_4 を 1 に固定した場合は問題 2，0 に固定した場合は問題 3 が生成される.
- このように再帰的に分枝操作を適用すると，未決定変数が無くなったときには $2^n = 2^4 = 16$ 個の問題が生成される．これは分枝木の葉に対応する.

n がある程度大きい場合，分枝木の部分問題をすべて生成すると膨大な計算が必要になるので，分枝木の部分木で計算しなくて済むような場所を検出し，調べる部分問題の個数を減らしたい.

- 分枝操作を適用して固定した変数 z_j に対するアイテムのサイズ a_j の和がナップサックサイズ b を超えたら，その部分問題は実行不可能であり，この部分問題に対する再帰的計算は終端させることができる.
- 次で述べる限定操作により，そのような部分木を検出することができる.

限定操作

概要. 限定操作では，先に述べた LP 緩和問題と，その最適値に関する性質を利用する．部分問題 P' を訪れる前に何らかの方法で元の問題 P_1 に対する実行可能解が得られていて，その利得（目的関数値）を LB とする（複数の実行可能解が得られている場合は，それらの利得のうち最大値を LB とする）．すると，$\mathrm{LB} \geq f_{\mathrm{LP}}(P')$ が成り立つのであれば，部分問題 P' を解く必要がないことを結論づけられる．それは

$$\mathrm{LB} \geq f_{\mathrm{LP}}(P') \geq f(P')$$

が成り立つため，部分問題 P' の最適値 $f(P')$ が LB を超えることはない，つまり頑張って P' を解いても，これまで求めた実行可能解より良い解を得ることはないからである．そのような P' は終端することができる.

実行可能解と暫定値. 元の問題 P_1 の実行可能解は次のように見つかる.

- 分枝木の探索開始前に，発見的方法で 1 個特別に元の問題 P_1 の実行可能解を見つけておく.
- 分枝操作で n 個の変数を 1 か 0 に固定し，未決定変数が無くなったとき，実行可能であれば，実行可能解が手に入る.

計算中に見つけた実行可能解のうち最も利得の大きい値を LB に格納しておく．これを**暫定値** (incumbent value) と呼ぶ．暫定値 LB の意味は，

「元の問題 P_1 の最適値（最大利得）$f(P_1)$ は LB 以上である」

ということである．計算中に新しい実行可能解が得られるたびに LB の値を更新する．

上界値．分枝操作によりいくつかの変数が 1 か 0 に固定されたが，まだ未決定変数を持つ部分問題 P' を考える．この部分問題は，未決定変数が関与する部分だけを見ると（都合の良いことに）やはり 0,1-ナップサック問題の構造をしているので，LP に緩和したときの最大利得 $f_{\mathrm{LP}}(P')$ が容易に計算できる（固定済みの変数の寄与する利得も含める）．この部分問題 P' の利得の上界値として $\mathrm{UB}(P') := f_{\mathrm{LP}}(P')$ とする．上界値 $\mathrm{UB}(P')$ の意味は，

「部分問題 P' の利得（固定済みの変数の寄与する利得も含め）の最大値は $\mathrm{UB}(P')$ 以下である」

ということである．

以上の議論から，計算中に生成された部分問題 P' の上界値 $\mathrm{UB}(P')$ とこの時点での暫定値 LB を比べたとき以下のことが結論できる．

「$\mathrm{LB} \geq \mathrm{UB}(P')$ であれば，部分問題 P' を正確に解いても LB よりも大きな利得の解は得られない.」

従って，この部分問題に対する再帰的計算は終端させることができる．これが限定操作である．

0,1-ナップサック問題に対する分枝限定法

1. 貪欲法[6]により元の問題 P_1 の実行可能解（初期暫定解）を構築する．この利得を LB（グローバル変数）とする．
2. 現在の部分問題 P_j が実行不可能のとき（選択したアイテムの和がナップサックサイズ b を超える場合），P_j に対する計算を終端する．
3. 部分問題 P_j が実行可能であるとき．
   (i) 部分問題 P_j に未決定変数がないとき，唯一の実行可能解の利得 $f(P_j)$ を計算する．もし，$\mathrm{LB} < f(P_j)$ であれば，$\mathrm{LB} := f(P_j)$ と更新する．
   (ii) 部分問題 P_j が未決定変数をもつとき，貪欲法により P_j の LP 緩和解を構築する．この利得 $f_{\mathrm{LP}}(P_j)$ を $\mathrm{UB}(P_j)$ とする．

[6] ここでいう貪欲法は元の 0,1-ナップサック問題に対するものなので，LP 緩和問題に対する貪欲法とは異なることに注意せよ．たとえば単位サイズあたり利得の降順に要素を走査し，各要素がナップサックに入るならば入れ，そうでなければ入れないことを繰り返す，といったものが考えられる．

(a) $\mathrm{LB} \geq \mathrm{UB}(P_j)$ であれば，P_j に対する計算を終端する.（P_j が元の問題 P_1 である場合は， 初期暫定解が最適解となる.）

(b) $\mathrm{LB} < \mathrm{UB}(P_j)$ であれば，P_j の未決定変数 z_i を一つ選び，部分問題 P_j において，$z_i = 1$，$z_i = 0$ をそれぞれ課した二つの部分問題 $P_{j'}$, $P_{j''}$ を生成する．$P_{j'}$, $P_{j''}$ に順に 2．を適用する.

♠「講義資料と課題」3 ページ: 0,1-ナップサック問題の例 (3) に対する分枝限定法の探索の進行　第 8 回で紹介した 0,1-ナップサック問題の例題 (3) に対して分枝限定法アルゴリズムを適用したときの計算の過程を追ってみる．問題例 (3) は以下の通り.

$$\begin{array}{llll} c_1 = 4, & c_2 = 6, & c_3 = 13, & c_4 = 15, \\ a_1 = 2, & a_2 = 4, & a_3 = 6, & a_4 = 7, \\ b = 11. & & & \end{array}$$

単位サイズあたりの利得は

$$c_1/a_1 = 4/2 = 2,\ c_2/a_2 = 6/4 = 1.5,\ c_3/a_3 = 13/6 = 2.1666\ldots,\ c_4/a_4 = 2.1426\ldots$$

となるので,

$$c_3/a_3 > c_4/a_4 > c_1/a_1 > c_2/a_2$$

であることに注意せよ.

ここで，分枝操作で固定する変数は z_4, z_3, z_2, z_1 の順に行うことにする.

節点 1: LP 緩和問題の最適値 $f_{\mathrm{LP}}(P_1) = c_3 + c_4 \times (5/7) = \frac{166}{7}$ の小数点以下の値を切り捨てることで得られる 23 は問題 P_1 の最適値の上界値 $\mathrm{UB}(P_1)$ となる.[7]

一方 P_1 に対する貪欲法を適用すると $z = (1, 0, 1, 0)$ が得られるが, その目的関数値は $c_3 + c_1 = 17$ であることから，$\mathrm{LB} = 17$ となる.

$\mathrm{LB} = 17 < \mathrm{UB}(P_1) = 23$ により，変数 z_4 で分枝して節点 2 と 3 を生成.

節点 2: $\mathrm{UB}(P_2) = [f_{\mathrm{LP}}(P_2)] = [c_4 + c_3 \times (4/6)] = [15 + 13 \times (4/6)] = 23$. ただし $[\cdot]$ は小数点以下の値の切捨てを表すものとする.

$\mathrm{LB} = 17 < \mathrm{UB}(P_2) = 23$ により，変数 z_3 で分枝し, 節点 4 と 5 を生成.

節点 4: 要素 3,4 がナップサックに入った部分問題に対応するが，$a_3 + a_4 = 13 > b = 11$ より，P_4 は実行不可能である．よって終端.

節点 5: $\mathrm{UB}(P_5) = [f_{\mathrm{LP}}(P_5)] = [c_4 + c_1 + c_2 \times (2/4)] = [15 + 4 + 6 \times (2/4)] = 22.$

$\mathrm{LB} = 17 < \mathrm{UB}(P_5) = 22$ により，変数 z_2 で分枝し, 節点 6 と 7 を生成.

[7] この例題ではすべての利得が整数なので，小数点以下の値を切り捨てた値はなお $f(P_1)$ の上界値である.

節点 6: $a_4 + a_2 = 7 + 4 = 11 = b$ より自動的に $z_1 = 0$ となり，実行可能解 $(0, 1, 0, 1)$ が得られ，その目的関数値は $c_4 + c_2 = 15 + 6 = 21$．よって $f(P_6) = 21$ である．

$f(P_6) = 21 > \mathrm{LB} = 17$ より，暫定値 LB を 21 に更新.

節点 7: $\mathrm{UB}(P_7) = [f_{\mathrm{LP}}(P_7)] = [c_4 + c_1] = [15 + 4] = 19$.

$\mathrm{LB} = 21 \geq \mathrm{UB}(P_7) = 19$ より，限定操作で終端.

節点 3: $\mathrm{UB}(P_3) = [f_{\mathrm{LP}}(P_3)] = [c_3 + c_1 + c_2 \times (3/4)] = [13 + 4 + 6 \times (3/4)] = 21$.

$\mathrm{LB} = 21 \geq \mathrm{UB}(P_3) = 21$ より，限定操作で終端.

以上より, 最適値は 21, 最適解は $z = (0, 1, 0, 1)$.

注: 探索木は, 分枝操作に用いる変数や, 部分問題を調べる順序によって大きく変わり得る.

【演習 9 − 1 】上記の分枝限定法の適用の仕方に倣い，次の 0,1-ナップサック問題の例題

$$
\begin{array}{lll}
c_1 = 8, & c_2 = 10, & c_3 = 7, \\
a_1 = 4, & a_2 = 6, & a_3 = 3, \\
b = 11. & &
\end{array}
$$

に分枝限定法の適用したときの計算の過程を示せ．分枝操作で固定する変数の順序は z_1, z_2, z_3 とすること．分枝木の絵を描き，LB, UB の計算の過程も示すこと．

9.4 分枝限定法：コンピュータサイエンスとオペレーションズ・リサーチでの違い

前回の講義でも述べたが，異なるアルゴリズムの計算速度をオーダー計算量で比較することは容易ではない．ここで一例を挙げる．グラフから互いに隣接しないように集めてきた点の集まりを独立点集合 (independent set) と呼ぶ．例えば，グラフの彩色において，同じ色で彩色された点の集合は独立点集合である．グラフを k 色で彩色するとは，頂点集合を k 個の独立点集合に分割することと等価である．大きさが最大の独立点集合を求める独立点集合問題はグラフに関する最適化問題の中でもっとも基本的なもののひとつであるが，NP-完全問題でもある．詳細は省くが，n 個のグラフの独立点集合問題を解く分枝アルゴリズムで計算量の上界が $1.4^n n^{O(1)}$ であると理論的に解析できるものが比較的容易に設計できる（現在，最良の上界は $1.1996^n n^{O(1)}$ [Xiao, Nagamochi 2017]）．一方で，独立点集合問題を整数計画問題として定式化し，専用ソルバーで解くこともできる．整数計画問題も NP-完全問題であり，これを厳密に解くには分枝限定法に頼らざるを得ないが，この場合の計算量の上界を解析することは難しく，自明な上界は $2^n n^{O(1)}$ である．オーダー計算量で比較すると，$1.4^n n^{O(1)}$ 時間のアルゴリズムのほうが $2^n n^{O(1)}$ 時間のアルゴリズムより高速であると言いたくなるかもしれないが，果たしてそうであろうか．サイズの異

なる 4 個のグラフの例題に対して両者を計算機実験により比較した結果を以下の表に示す．専用ソルバーには CPLEX という商用ソルバーを用いた．

例題の大きさ	$1.4^n n^{O(1)}$ 時間 厳密アルゴリズム (sec.)	整数計画問題 CPLEX (sec.)
点の個数 50，枝の本数 175	6.6	0.024
点の個数 60，枝の本数 210	89.4	0.067
点の個数 70，枝の本数 245	367.0	0.11
点の個数 80，枝の本数 280	5540.7	0.19

いかがであろうか．単に，個々の例題に対し後者のほうが求解時間が短いというだけでない．点の個数が 30 個増えたときの計算時間の増加は，前者は 839 倍であるが，後者は 8 倍程度であることが観察できる．単純に，$\alpha^{30} = 839$，$\beta^{30} = 8$ を解くと，実験した範囲では前者は $\alpha^n n^{O(1)} = 1.251^n n^{O(1)}$ 時間，後者は $\beta^n n^{O(1)} = 1.072^n n^{O(1)}$ 時間の振る舞いを呈している．

第 9 回の講義は以上である．