2015.6.11

数理計画と最適化
ー組み合わせ最適化2ー

精密工学科

淺間 一

簑原 凜，徐 彬斌，楊 濘嘉（TA）

asama@robot.t.u-tokyo.ac.jp

	ナップザック問題 原問題	ナップザック問題の緩和問題
目的関数	$\sum_{i=1}^{n} c_i x_i \to \max$	$\sum_{i=1}^{n} c_i x_i \to \max$
制約条件	$\sum_{i=1}^{n} a_i x_i \le b$	$\sum_{i=1}^{n} a_i x_i \le b$
	$x_i = 0,1 (i = 1, \cdots, n)$	$\boxed{0 \le x_i \le 1 (i = 1, \cdots, n)}$
ただし	$\frac{c_1}{a_1} \ge \frac{c_2}{a_2} \ge \cdots \ge \frac{c_n}{a_n}$	$\frac{c_1}{a_1} \ge \frac{c_2}{a_2} \ge \cdots \ge \frac{c_n}{a_n}$

分枝（分岐）限定法 厳密解法
(Branch and Bound Method)

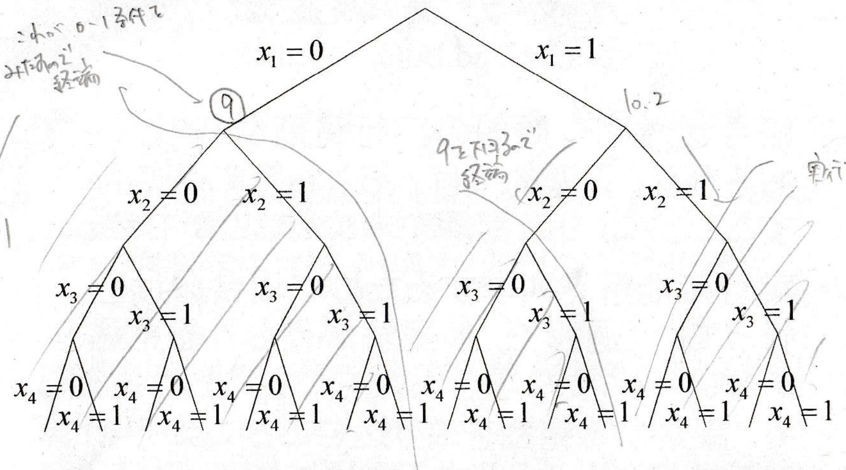
- 組合せ最適化問題の実行可能解は有限個であるから，原理的には，それらをすべて列挙することにより厳密解を求めることができる．
- 実行可能解を列挙するために場合分けを行っていく過程で，最適解が得られる見込みのない不必要な場合分けをできるだけ省略して，探索する範囲を絞り込むことにより，計算時間を短縮しようとする方法である．

実数最適解の有用な性質

- 連続緩和問題の実行可能領域は原問題の実行可能領域を含むので，前者の問題の最大値は後者の問題の最大値より大きいかまたは等しい．すなわち，前者の問題の最大値は後者の問題の最大値に対する一つの上界値を与える．

  実行可能領域：
- 実数最適解には0-1条件を満たさない変数は高々一つしかない．その変数の値を0とおき，そのかわり値が0になっている変数のなかに1に変更できるものがあればそれを1とおくことにより，原問題の近似最適解を容易に得ることができる．いまの場合は，変数x_2を0とするかわりに変数x_3を1とすれば，原問題の近似解$\boldsymbol{x}=(1,0,1,0)^{\mathrm{T}}$が得られる．

原問題の最適解は各変数の値を0または1とおいた16個の組合せのなかにある.



問題の変数を一つずつ値を0または1に固定していく様子を表しており, 最上部の節点はまだどの変数も固定されていない状態に, 最下部の16個の節点はすべての変数が0または1に固定された状態に対応している. また, 途中に現れる節点は, 原問題において一部の変数の値が0または1に固定され, それ以外の変数は固定されていないような問題を表している.

暫定解

$(x_1\ x_2\ x_3\ x_4) = (1\ 0\ 1\ 0)$

$f(x) = 8$

$P(J_0, J_1)$

$P(\phi, \phi)$

$P(\{1\}, \phi)$

$P(\phi, \{1\})$

$P(\{2\}, \{1\})$

$P(\phi, \{1,2\})$

$x_1 = 0$

$8x_2 + x_3 + 2x_4 \to \max$

$5x_2 + x_3 + 3x_4 \leq 6$

$\boxed{(x_1\ x_2\ x_3\ x_4) = (0, 1, 1, 0) \quad f(x) = 9}$

$x_1 = 1 \quad (1, \frac{2}{5}, 0, 0) \to 10.2$

$7 + 8x_2 + x_3 + 2x_4 \to \max$

$4 + 5x_2 + x_3 + 3x_4 \leq 6$

$5x_2 + x_3 + 3x_4 \leq 2$

$x_1 = 1 \quad x_2 = 0 \quad (1, 0, 1, \frac{1}{3}) \to 8.67$

$7 + x_3 + 2x_4 \to \max$

$4 + x_3 + 3x_4 \leq 6$

$x_3 + 3x_4 \leq 2$

$x_1 = 1, \quad x_2 = 1$

$7 + 8 + x_3 + 2x_4 \to \max$

$4 + 5 + x_3 + 3x_4 \leq 6$

$x_3 + 3x_4 \leq -3$

終端条件

- それまでに得られている最良の実行可能解を**暫定解**とし, その目的関数値をその時点での**暫定値**とする. 部分問題$P(J_0,J_1)$の連続緩和問題を解いて問題$P(J_0,J_1)$の上界値を求めたとき, それが暫定値より小さいならば, その部分問題がもとの問題の最適解を与える可能性はない. よって部分問題$P(J_0,J_1)$は終端できる.
- 連続緩和問題の最適解（実数最適解）が0-1条件を満たすならば, それは部分問題$P(J_0,J_1)$の最適解になっている. この場合も部分問題は終端できる. さらに, 得られた解の目的関数値が暫定値より大きいなら, それを新しい暫定値とする.
- 部分問題$P(J_0,J_1)$が実行可能解をもたないことが判明すれば部分問題は終端できる.

部分問題

- 一部の変数の値が0または1に固定され, それ以外の変数は固定されていないような問題を部分問題と呼び, $P(J_0,J_1)$と書く.
- ここでJ_0とJ_1はそれぞれ0に固定されている変数と1に固定されている変数（の添字）の集合を表している. また, 固定されていない変数を自由変数といい, その添字の集合をFで表す. たとえば, $F=\{1,3\}$, $J_0=\{2\}$, $J_1=\{4\}$に対する部分問題$P(\{2\},\{4\})$は次のようになる.

原問題

$7x_1 + 8x_2 + x_3 + 2x_4 \to \max$

$4x_1 + 5x_2 + x_3 + 3x_4 \leq 6$

$P(\{2\},\{4\})$ 目的関数 $7x_1 + x_3 + 2 \to \max$

制約条件 $4x_1 + x_3 + 3 \leq 6$

$$x_i = 0,1 (i \in F = \{1,3\})$$

この問題もまたナップサック問題の形になっている

[反復1] 原問題の実数最適解を修正して得られる近似解$x=(1,0,1,0)^T$を暫定解とし, その目的関数値8を暫定値z^*とする. 原問題, すなわち部分問題$P(\varphi,\varphi)$に分枝操作を施し, 二つの部分問題$P(\{1\},\varphi)$と$P(\varphi,\{1\})$を生成する. 活性部分問題の集合は$A=\{P(\{1\},\varphi)$と$P(\varphi,\{1\})\}$となる.

[反復2] 活性部分問題$P(\{1\},\varphi)$を選ぶ. 部分問題

$P(\{1\},\varphi)$ 目的関数: $8x_2+x_3+2x_4 \to \max$

制約条件: $5x_2+x_3+3x_4 \leqq 6$

$x_i=0, 1\ (i=2, 3, 4)$

の実数最適解は$(x_2,x_3,x_4)^T=(1,1,0)^T$となる. これは0-1条件を満たすので, 部分問題$P(\{1\},\varphi)$は終端できる. また, この解に対する目的関数値9は暫定値8より大きいので, 暫定解を$x=(0,1,1,0)^T$に, 暫定値を9に置き換える. 活性部分問題の集合は$A=\{P(\varphi,\{1\})$となる.

[反復3] 活性部分問題$P(\varphi,\{1\})$を選ぶ. 部分問題

$P(\varphi,\{1\})$ 目的関数: $7+8x_2+x_3+2x_4 \to \max$

制約条件: $4+5x_2+x_3+3x_4 \leqq 6$

$x_i=0, 1\ (i=2, 3, 4)$

の実数最適解は$(x_2,x_3,x_4)^T=(2/5,0,0)^T$であり, その目的関数値は51/5=10.2となる. これは現在の暫定値9より大きいので, この時点で部分問題$P(\varphi,\{1\})$を終端することはできない. そこで, この部分問題において変数x_2を0または1に固定した部分問題を生成する. その結果, $A=\{P(\{2\},\{1\}),P(\varphi,\{1,2\})\}$となる.

[反復4] 活性部分問題$P(\{2\},\{1\})$を選ぶ．部分問題

$P(\{2\},\{1\})$ 目的関数： $7+x_3+2x_4 \to \max$

制約条件： $4+x_3+3x_4 \leqq 6$

$$x_i=0,\ 1\ (i=3,\ 4)$$

の実数最適解は$(x_3,x_4)^T=(1,1/3)^T$であり，その目的関数値は$8+2/3=8.67$となる．これは部分問$P(\{2\},\{1\})$の上界値であるが，現在の暫定値9より小さい．よって，部分問題$P(\{2\},\{1\})$が最適解を与える可能性はないので，この部分問題を終端できる．その結果，$A=\{P(\varphi,\{1,2\})\}$となる．

[反復5] 活性部分問題$P(\varphi,\{1,2\})$を選ぶ．ところが，この部分問題

$P(\varphi,\{1,2\})$ 目的関数： $7+8+x_3+2x_4 \to \max$

制約条件： $4+5+x_3+3x_4 \leqq 6$

$$x_i=0,\ 1\ (i=3,\ 4)$$

は明らかに実行可能解をもたないので，ただちに終端できる．その結果，$A=\varphi$となる．

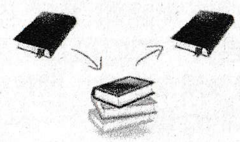
[反復6] 活性部分問題がなくなったので，計算を終了する．このとき得られている暫定解

$x=(0,1,1,0)^T$が原問題の最適解である．

スタック (Stack)

First In Last Out (FILO)

上から積んで上から取る…最初に積んだ本を最後に取る



予め用意された配列

top (「一番上」を示す)

push (スタックに積む)

top が示す位置に値を入れて…

top を一つずらす

pop (スタックから取り出す)

top を一つずらす

top が示す位置の値を取り出す

キュー (Queue)

- キュー: First In First Out (FIFO)
  待ち行列…最初に並んだ人に最初の順番が来る

予め用意された配列

head (先頭) tail (最後尾)

push (キューに入れる)

head tail の位置に値を入れて…

head tail を一つずらす

pop (キューから取り出す)

tail

head の位置の値を取り出し

tail

head を一つずらす

Octave の List

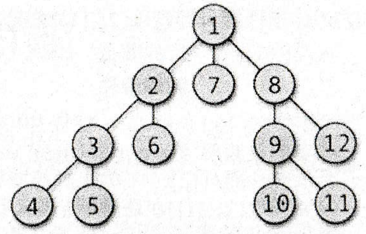
- **list**(*a1, a2, …*)
  - 新たなリストの生成
  - **list**() で空リストの生成
- **append**(*list, a1, a2, …*)
  - リスト*list*に*a1, a2, …*を追加することにより生成される新たなリストを返す。
- **splice**(*list_1, offset, length, list_2*)
  - リスト*list_1*について，*offset*番目から*length*個の要素を，*list_2*の成分で置き換える
- *Nth*(*L, j*)
  - リスト*list L*の *j* 番目の要素を参照する．（配列として返る）

グラフ探索問題

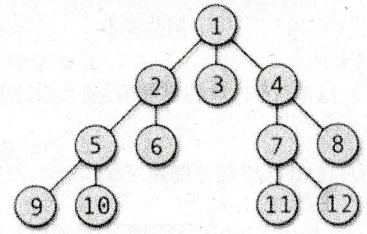
- 頂点 $v \in V$ に対してそれと辺でつながれた頂点の集合を $T(v)$ とする.
- グラフ探索問題：出発点となる頂点 v_0 と関数 T とが与えられたときすべての頂点を列挙する問題
  - 分枝限定法では，関数Tの返値は分枝操作により得られる頂点
- 基本的な戦略：
  - 探索すべき頂点の集合をAとする.
  - 集合Aから頂点を一つ取り出し、関数 T によって新しい頂点が見つかったらそれを集合Aに追加する
  - 分枝限定法では，終端条件を満たしたときは追加を行わない
- 手続きの非一義性：頂点を一つ取り出す…データ構造依存
  - スタック：深さ優先探索、キュー：幅優先探索

(Ref. 杉原厚吉: “データ構造とアルゴリズム”, 共立出版)

スタックの場合（縦型探索）
○List (1)
○リストの一番前の1を取り出す. List()
1の評価，評価の結果分枝が必要な場合
T(1)→2, 7, 8
リストの前に加える. List(2, 7, 8)
ListはΦ？→No
○リストの一番前の2を取り出す. List(7, 8)
2の評価，評価の結果分枝が必要な場合
T(2)→3,6
リストの前に加える. List(3,6,7,8)
ListはΦ？→No
○リストの一番前の3を取り出す. List(6, 7, 8)



キューの場合（横型探索）
○List (1)
○リストの一番前の1を取り出す. List()
1の評価，評価の結果分枝が必要な場合
T(1)→2, 3, 4
リストの後に加える. List(2, 3, 4)
ListはΦ？→No
○リストの一番前の2を取り出す. List(3, 4)
2の評価，評価の結果分枝が必要な場合
T(2)→5,6
リストの後に加える. List(3,4,5,6)
ListはΦ？→No
○リストの一番前の3を取り出す. List(4, 5, 6)

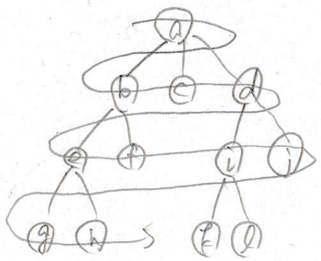


【演習問題】

$$\max \quad 4x_1 + 5x_2 + 12x_3 + 14x_4$$
$$\text{s.t.} \quad 2x_1 + 3x_2 + 5x_3 + 6x_4 \le 9$$
$$x_i = 0 \text{ または } 1 \left(i = 1,\ldots,4\right)$$

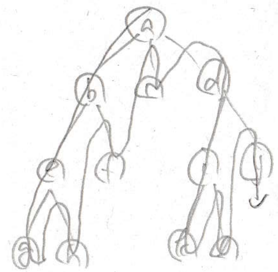
■ 欲張り法の解：(1,0,1,0)，目的関数値16
■ けちけち法の解：(0,0,1,0)，目的関数値12

分枝限定法で解いたときの解？



キュー
List(a)
a List()
T(a) b,c,d List(b,c,d)
b List(c,d)
T(b) e,f List(c,d,e,f)

幅優先探索
(横型 〃)



深さ優先探索
(縦型 〃)

スタック
a List(a) List()
T(a) → b,c,d
b List(b,c,d) (List(c,d)
T(b) = e,f List(e,f,c,d)
e T(e) = g,h List(g,h,f,c,d)
g