最適化 1 (第 7 回)

番原睦則 (banbara@i.nagoya-u.ac.jp)

名古屋大学情報学研究科

組合せ計画 (再掲)

組合せ最適化問題 *

一般に，**有限個**の要素からなる実行可能集合のなかで，目的関数が最小 (あるいは最大) となるようなものを見つける問題

- 組合せ最適化問題の例
  - 整数計画問題全般
    - スケジューリング問題，時間割問題，パッキング問題，etc.
  - 最小木問題・最短路問題
  - 巡回セールスマン問題
- 実行可能解の数は有限だが，それらをすべて列挙して目的関数値を比較するのは非現実的 (**組合せ爆発**)
- 現実の組合せ最適化問題の多くは **NP 困難**と呼ばれるクラスに属する.

*組合せ計画問題とも呼ばれる.

組合せ最適化問題の解法 (再掲)

- **欲張り法** *
  - 解を段階的に構築する際に，常にその段階で最善と思われるものを選択する方法 (後のことは考えない).
  - 一般に，最適解が構築できる保証はない.
- **厳密解法**
  - 厳密な最適解を求める方法
  - 例) **分枝限定法** †，動的計画法
- **近似解法・発見的解法**
  - 広い意味で，厳密解法ではない方法．最適解を厳密に求めることは諦め，良い近似最適解を比較的短時間で求める方法
  - 例) 局所探索法，メタヒューリスティックス

---

*貪欲法とも呼ばれる

†ブランチ・アンド・バウンド法とも呼ばれる

分枝限定法

- 組合せ最適化問題に対する厳密解法の一つ.
- 様々な問題に対して用いることができる一般的な計算原理.

分枝限定法の基本的な考え方

1. **分枝操作**: 変数の値を固定するなどして場合分けを行い，部分問題を生成する.
2. **限定操作**: 部分問題に対して，(元の問題の) 最適解を与える可能性があるか調べ，ないと判定された場合は，その部分問題を解くことを止める.
3. 上記の操作を繰り返し，探索範囲を絞り込むことにより計算時間を短縮する.

ナップサック問題

ナップサック問題を取り上げ，分枝限定法を説明する．

$$\text{目的関数:} \quad \sum_{i=1}^{n} c_i x_i \longrightarrow \text{最大}$$

$$\text{制約条件:} \quad \sum_{i=1}^{n} a_i x_i \leq b$$

$$x_i \in \{0, 1\} \qquad (i = 1, 2, \ldots, n)$$

ただし，c_i, a_i, b はすべて正の整数とし，

$$\frac{c_1}{a_1} \geq \frac{c_2}{a_2} \geq \cdots \geq \frac{c_n}{a_n}$$

となるように，変数の添字 i は並べ換えられていると仮定．

連続緩和問題

0-1 ナップサック問題

目的: $7x_1 + 8x_2 + x_3 + 2x_4 \longrightarrow$ 最大
制約: $4x_1 + 5x_2 + x_3 + 3x_4 \leq 6$
$x_i = 0, 1 \, (i = 1, 2, 3, 4)$

連続ナップサック問題

目的: $7x_1 + 8x_2 + x_3 + 2x_4 \longrightarrow$ 最大
制約: $4x_1 + 5x_2 + x_3 + 3x_4 \leq 6$
$0 \leq x_i \leq 1 \, (i = 1, 2, 3, 4)$

- **緩和問題**とは，与えらえた問題の**制約を緩めて**得られる問題
- 整数条件を緩和した問題を**連続緩和問題**と呼ぶ.
  - 例) 0-1 ナップサック問題と連続ナップサック問題
- 連続ナップサック問題は線形計画問題 (多項式時間で解ける)

$$\boldsymbol{x} = (1, \frac{2}{5}, 0, 0)^T$$

- この $\boldsymbol{x}$ を元の 0-1 ナップサック問題の**実数最適解**という.

連続緩和問題の性質

0-1 ナップサック問題

目的: $7x_1 + 8x_2 + x_3 + 2x_4 \longrightarrow$ 最大
制約: $4x_1 + 5x_2 + x_3 + 3x_4 \leq 6$
$x_i = 0, 1 \, (i = 1, 2, 3, 4)$

連続ナップサック問題

目的: $7x_1 + 8x_2 + x_3 + 2x_4 \longrightarrow$ 最大
制約: $4x_1 + 5x_2 + x_3 + 3x_4 \leq 6$
$0 \leq x_i \leq 1 \, (i = 1, 2, 3, 4)$

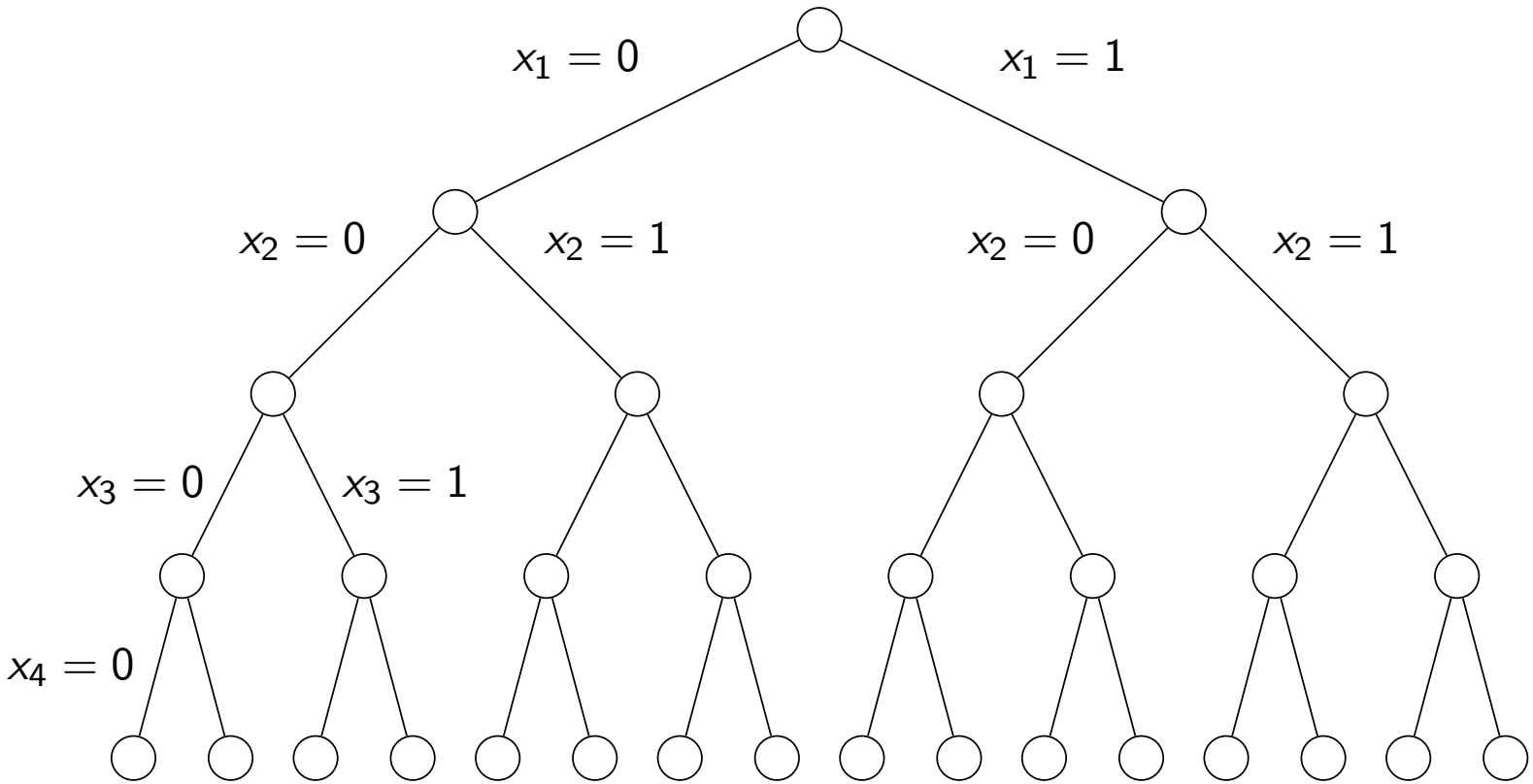
1. 連続緩和問題の実行可能領域は元の問題のそれを含む.
   - 連続緩和問題の最大値は元の問題の最大値より大きいか，または等しい.
   - 連続緩和問題の最大値は元の問題の最大値に対する一つの**上界値**を与える.
2. 実数最適解を少し変更することにより，元の問題の実行可能解が得られることが多い.
   - $\boldsymbol{x} = (1, \frac{2}{5}, 0, 0)^T$ に対して，x_2 を 0 と置き，x_3 を 1 とする.

$$\boldsymbol{x}' = (1, 0, 1, 0)^T$$

分枝図

0-1 ナップサック問題の最適解は 2^4 個の組合せのなかにある.

- 中間ノードは一部の変数の値が固定された**部分問題**に対応.



部分問題

0-1 ナップサック問題の部分問題を $\mathcal{P}(J_0, J_1)$ で表す.

- J_0: 値が 0 に固定されている変数集合
- J_1: 値が 1 に固定されている変数集合
- 固定されたない変数を**自由変数**と呼び，F で表す.

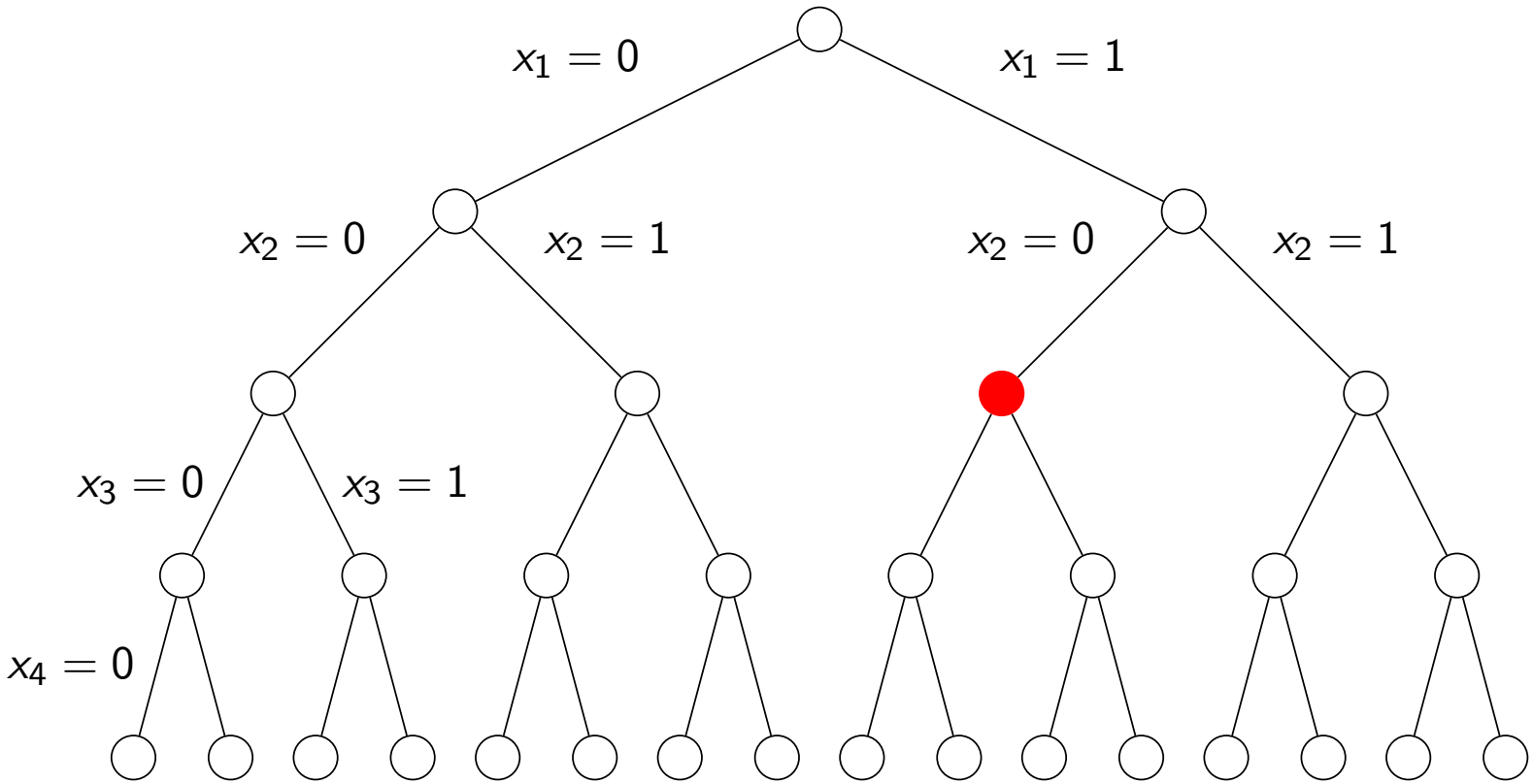
0-1 ナップサック問題

$$
\begin{aligned}
\text{目的:} \quad & 7x_1 + 8x_2 + x_3 + 2x_4 \longrightarrow \text{最大} \\
\text{制約:} \quad & 4x_1 + 5x_2 + x_3 + 3x_4 \leq 6 \\
& x_i = 0, 1 \, (i = 1, 2, 3, 4)
\end{aligned}
$$

部分問題 $\mathcal{P}(\{2\}, \{1\})$, $F = \{3, 4\}$

$$
\begin{aligned}
\text{目的:} \quad & 7 + x_3 + 2x_4 \longrightarrow \text{最大} \\
\text{制約:} \quad & 4 + x_3 + 3x_4 \leq 6 \\
& x_i = 0, 1 \, (i = 3, 4)
\end{aligned}
$$

部分問題 $\mathcal{P}(\{2\},\{1\}),\ F=\{3,4\}$



部分問題の終端

分枝図に現れるすべての部分問題を数え上げる必要はない.

部分問題の終端

順次生成される部分問題に対して，元の問題の最適解を与える可能性があるか調べ，ないと判定すれば，その部分問題を解くことをやめる.

- 部分問題を終端すると，分枝図におけるその**部分問題の下のすべての部分問題**を調べる必要がなくなる.
- 生成される部分問題の数はその分だけ減少する.

限定操作

- 問題を解く過程で，これまでに得られている最良の実行可能解を**暫定解**，その目的関数値を**暫定値**とする．
- 部分問題の終端には以下のようにものがある．これらの操作を**限定操作**という．

限定操作

1. **部分問題が元の問題の最適解を与える可能性がない場合**
   - 部分問題 $\mathcal{P}$ の連続緩和問題を解いて得られる $\mathcal{P}$ の上界値が，暫定値より小さいなら，$\mathcal{P}$ は終端できる．
2. **部分問題の最適解が求まる場合**
   - 部分問題 $\mathcal{P}$ の連続緩和問題の最適解が 0-1 条件を満たすとき，$\mathcal{P}$ は終端できる．
   - 得られた解の目的関数値が暫定値より大きい場合は，その解を暫定解にする．
3. **部分問題が実行可能解をもたない場合**

分枝操作

分枝操作

終端できなかった部分問題に対して，その自由変数を一つ選んで固定することにより，新しい部分問題を生成する．

- 実際の計算では，終端できないとわかった部分問題に対して，直ちに分枝操作を行うとは限らない．
- 一般に，ある時点で分枝操作が適用可能な部分問題 (**活性部分問題**) は複数する．
- 分枝操作をする活性部分問題をどうやって選ぶ？

部分問題の探索法

最良優先探索法

各活性部分問題に対して，その最大値の推定値を求めておき，それが最大の問題を選ぶ.

- 計算終了までに生成される活性部分問題の数が少ない.
- 計算時間が少なくなることが期待できる.

深さ優先探索

分枝図においてルートからの深さが最大であるような活性部分問題を選ぶ.

- 計算途中で保持する活性部分問題の数が数ない.
- 限定操作で終端できなかった部分問題に対して，直ちに分枝操作をすることに対応.

分枝限定法の計算手順

0. 適当な方法で元の問題 $\mathcal{P}_0$ の近似最適解を求め，それを暫定解とする．その目的関数値を暫定値 z^* とする．$\mathcal{P}_0$ からいくつかの部分問題 $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_m$ を生成し，$\mathcal{A} := \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_m\}$ とおく．
1. 集合 $\mathcal{A}$ から部分問題 $\mathcal{P}_i$ を一つ選ぶ．
   a. $\mathcal{P}_i$ が実行可能解をもたなければ，直ちに $\mathcal{P}_i$ を終端．$\mathcal{A} := \mathcal{A} \backslash \{\mathcal{P}_i\}$ としてステップ (3) へ．
   b. $\mathcal{P}_i$ の最適解が得られ，その目的関数値 z_i が $z_i \leq z^*$ ならば，直ちに $\mathcal{P}_i$ を終端．$z_i > z^*$ ならば，$z^* := z_i$ とおき，暫定解を更新して $\mathcal{P}_i$ を終端．$\mathcal{A} := \mathcal{A} \backslash \{\mathcal{P}_i\}$ としてステップ (3) へ．
   c. $\mathcal{P}_i$ の上界値 $\overline{z_i}$ が得られ，$\overline{z_i} \leq z^*$ ならば，直ちに $\mathcal{P}_i$ を終端．$\mathcal{A} := \mathcal{A} \backslash \{\mathcal{P}_i\}$ としてステップ (3) へ．$\overline{z_i} > z^*$ ならば，ステップ (2) へ．
2. $\mathcal{P}_i$ からいくつかの部分問題 $\mathcal{P}_j, \ldots, \mathcal{P}_k$ を生成し，$\mathcal{A} := \mathcal{A} \cup \{\mathcal{P}_j, \ldots, \mathcal{P}_k\} \backslash \{\mathcal{P}_i\}$ とおく．ステップ (1) へ戻る．
3. $\mathcal{A} = \emptyset$ ならば計算終了．このとき，暫定解は元の問題 $\mathcal{P}_0$ の最適解．$\mathcal{A} \neq \emptyset$ ならば，ステップ (1) へ戻る．