情報システム評価学
―整数計画法―

第1回目：整数計画法とは？

塩浦昭義
東北大学 大学院情報科学研究科 准教授

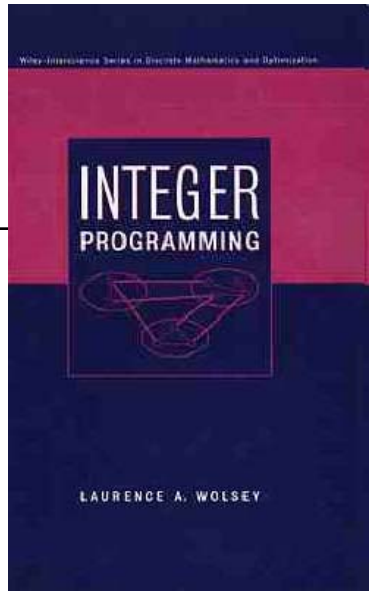
この講義について

- 授業のHP:
  - http://www.dais.is.tohoku.ac.jp/~shioura/teaching/dais08/
  - 授業に関する連絡，および講義資料等はこちらを参照
- 教員への連絡先:
  - shioura (AT) dais.is.tohoku.ac.jp
- 成績について
  - 主に数回のレポートによって得点を決めます
  - 場合によっては試験を行う可能性もあり

講義の内容

- 本年度は，整数計画法について講義をします
  - 目的：整数計画法の理論，アルゴリズムを学ぶ
    ➔整数計画法を「使える」ようになる
  - 理論：多面体理論，計算量理論，など
  - アルゴリズム：分枝限定法，動的計画法，など
  - 使い方：問題の定式化，ソルバーの利用方法，等
  - 整数計画法は情報システムの設計・評価に欠かせないツール
- 前提とする知識：線形計画法の基礎
  - 線形計画法について知らない学生は学部電気・情報系の「数理計画法」の講義（木曜3コマ）を受講してください

講義の参考書

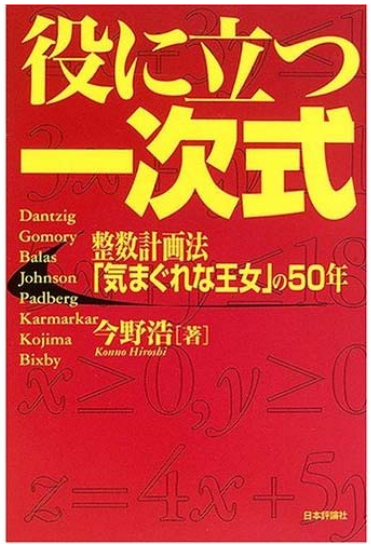


- 授業は主に下記の本に沿って進みます
  - Laurence A. Wolsey: Integer Programming, Wiley, 1998
- その他の参考書
  - G. L. Nemhauser, L. A. Wolsey: Integer and Combinatorial Optimization, Wiley, 1988
  - 今野浩, 鈴木久敏編：整数計画法と組合せ最適化，日科技連出版社，1982
  - Sven O. Krumke による lecture note

ftp://www.mathematik.uni-kl.de/pub/scripts/krumke/Notes/ip-lecture-new.pdf

講義の参考書

- おすすめ
  - 今野浩：役に立つ一次式—整数計画法「気まぐれな王女」の50年，日本評論社，2005年
  - 内容：整数計画法のこれまでの研究に関する物語



数理計画問題(mathematical programming problem)

- 数理計画問題＝最適化問題 (optimization problem)
  - ある条件式の下で目的関数を最大化（最小化）する解を求める問題

目的関数

$$\text{最大化 } f(x_1, x_2, \ldots, x_n)$$
$$\text{条件：} \ g_i(x_1, x_2, \ldots, x_n) \leq 0 \ (i = 1, 2, \ldots, m)$$

条件式

n, m: 正の整数

$f, g_i \ (i = 1, 2, \ldots, m)$: n 変数関数

線形計画問題(linear programming problem, LP)

- 目的関数と条件式が線形の数理計画問題

$$
\begin{aligned}
&\text{最大化} \quad \sum_{j=1}^{n} c_j x_j \\
&\text{条件：} \quad \sum_{j=1}^{n} a_{ij} x_j \le b_i \ (i = 1, 2, \ldots, m) \\
&\qquad\quad x_j \ge 0 \ (j = 1, 2, \ldots, n)
\end{aligned}
$$

例：

$$
\begin{aligned}
&\text{最大化} \quad 5x_1 + 9x_2 \\
&\text{条件：} \quad 2x_1 + x_2 \le 4 \\
&\qquad\quad 8x_1 + 3x_2 \le 6 \\
&\qquad\quad x_1 \ge 0, x_2 \ge 0
\end{aligned}
$$

n, m: 正の整数

$a_{ij}, b_i, c_j \ (i = 1, \ldots, m, j = 1, \ldots, n)$: 実数

行列とベクトルによる定式化

$$
\begin{aligned}
&\text{最大化} \quad c^T x \\
&\text{条件：} \quad Ax \le b, \ x \ge 0
\end{aligned}
$$

A: $m \times n$ 実数行列

b: m次元ベクトル, c: n次元ベクトル,

x：n次元ベクトル(変数)

整数計画問題(integer programming problem)

- 「変数が整数である」と言う条件をもつ数理計画問題

例１：

最大化 $5x_1 + 9x_2$

条件： $2x_1 + x_2 \leq 4$

$8x_1 + 3x_2 \leq 6$

$x_1 \geq 0, x_2 \geq 0$

x_1, x_2 は整数

線形計画問題

＋「全部の変数が整数」という条件

➔（線形）整数計画問題

(linear integer programming problem, IP)

整数計画問題(integer programming problem)

- 「変数が整数である」と言う条件をもつ数理計画問題

例２：

最大化 $5x_1 + 9x_2$

条件： $2x_1 + x_2 \leq 4$

$8x_1 + 3x_2 \leq 6$

$x_1, x_2 \in \{0, 1\}$

「全部の変数が0または1」という条件

➔0-1整数計画問題

(0-1 integer programming problem, 0-1IP)

※一般には，「整数計画問題(IP)」といったら

（線形）整数計画問題，もしくは 0-1整数計画問題，

のことを指す

整数計画問題(integer programming problem)

- 「変数が整数である」と言う条件をもつ数理計画問題

例３：

最大化 $5x_1 + 9x_2 + 4y$

条件： $2x_1 + x_2 - 3y \leq 4$

$8x_1 + 3x_2 + 7y \leq 6$

$x_1 \geq 0, x_2 \geq 0,\ y \geq 0$

x_1, x_2 は整数

線形計画問題

＋「一部の変数が整数」という条件

➔（線形）混合整数計画問題

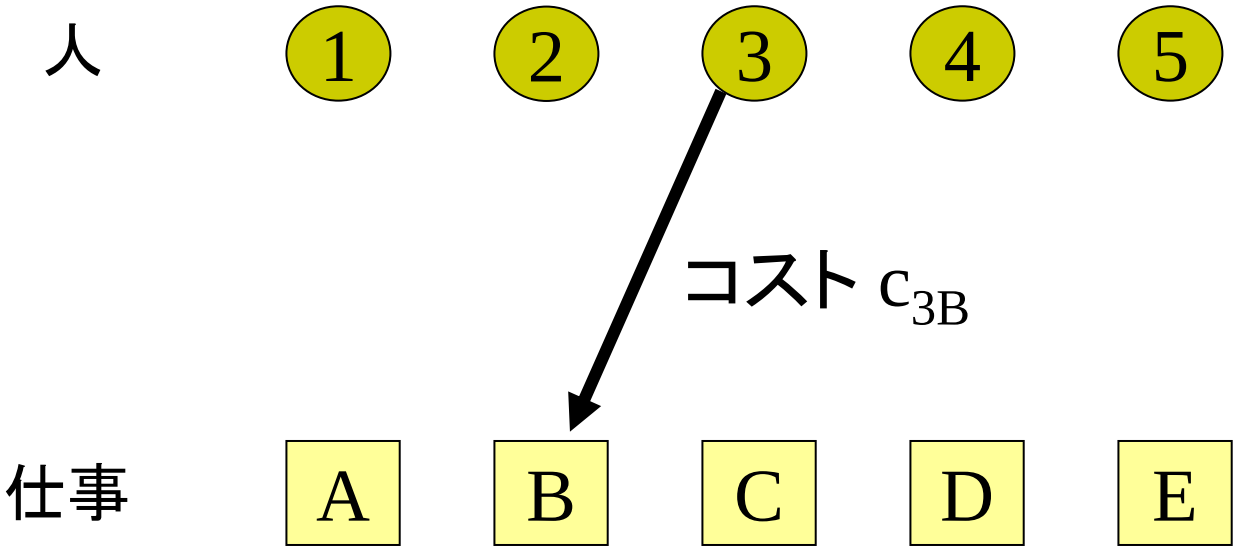
(linear mixed integer programming problem, MIP)

様々な問題のIPによる定式化

- 最適化に関わる多くの問題はIPとして定式化可能
- 定式化の一般的な手順
  1. 問題の入力データと変数を明確に区別する
  2. 必要な変数を定める
  3. 変数を使って必要な条件式を定める
  4. 変数を使って目的関数を定める

割当問題の定式化

- 割当問題(assignment problem)
  - n 人がn 種類の仕事を処理，一人当り一つの仕事
  - i さんが仕事 j を処理➔コスト c_{ij}
  - 目的：総コストの最小化



割当問題の定式化

変数の定義

$x_{ij} = 1$ (iさんを仕事jに割り当てたとき), $x_{ij} = 0$ (それ以外のとき)

条件式の定義

各人は一つの仕事を処理:

$$\sum_{j=1}^{n} x_{ij} = 1 \quad (i = 1, \ldots, n)$$

各仕事は一人の人により処理される:

$$\sum_{i=1}^{n} x_{ij} = 1 \quad (j = 1, \ldots, n)$$

各変数は 0-1: $x_{ij} \in \{0, 1\}$ $(i = 1, \ldots, n, j = 1, \ldots, n)$

目的関数の定義

割当コストを最小に:

最小化 $\sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$

0-1ナップサック問題の定式化

- 0-1ナップサック問題 (0-1 knapsack problem)
  - 今年度のプロジェクトへの投資予算 b 円
  - プロジェクト j の必要経費 a_j, 期待利益 c_j
  - 目的: 予算の範囲内で投資すべきプロジェクトを選択, 期待利益の合計を最大化

変数の定義

$x_j = 1$ (プロジェクト j を選んだとき), $x_j = 0$ (それ以外のとき)

条件式の定義

予算の範囲内で投資: $\sum_{j=1}^{n} a_j x_j \leq b$

各変数は 0-1: $x_j \in \{0, 1\}$ $(j = 1, \ldots, n)$

0-1ナップサック問題の定式化

変数の定義

$x_j = 1$ (プロジェクトjを選んだとき), $x_j = 0$ (それ以外のとき)

条件式の定義

予算の範囲内で投資: $\sum_{j=1}^{n} a_j x_j \leq b$

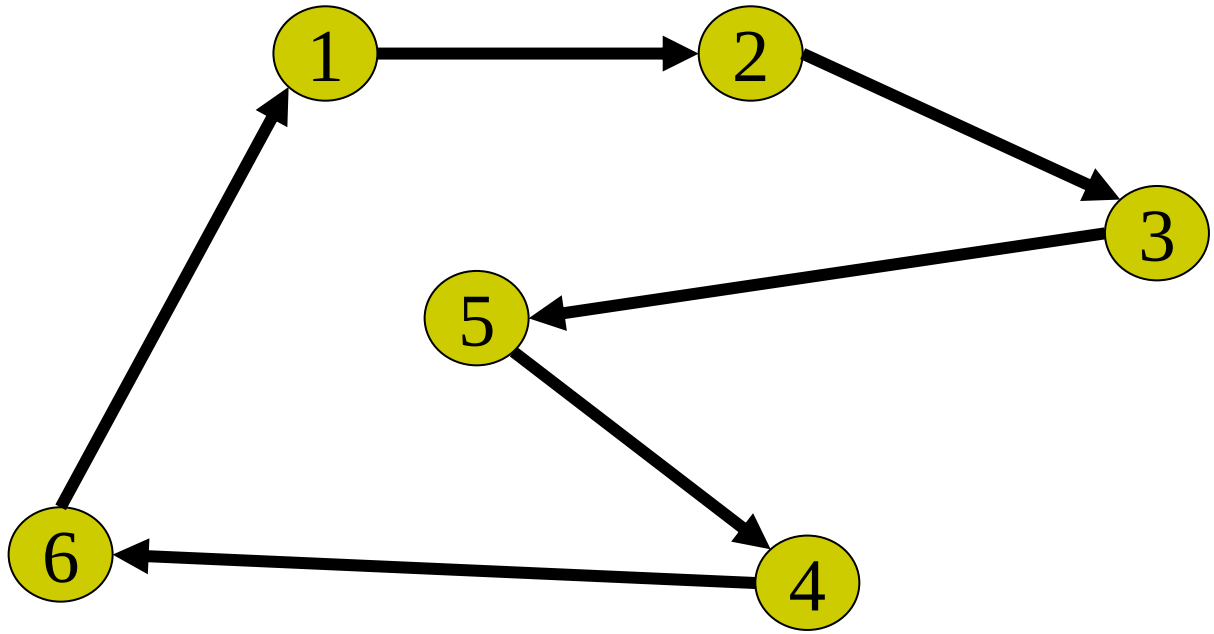
各変数は 0-1: $x_j \in \{0, 1\}$ $(j = 1, \ldots, n)$

目的関数の定義

期待利益を最大に: 最大化 $\sum_{j=1}^{n} c_j x_j$

巡回セールスマン問題

- セールスマンが n 都市をちょうど一回ずつ巡回する
- 都市 i から j への距離は c_{ij}
- 目的:都市を巡回する際の総距離を最小にする



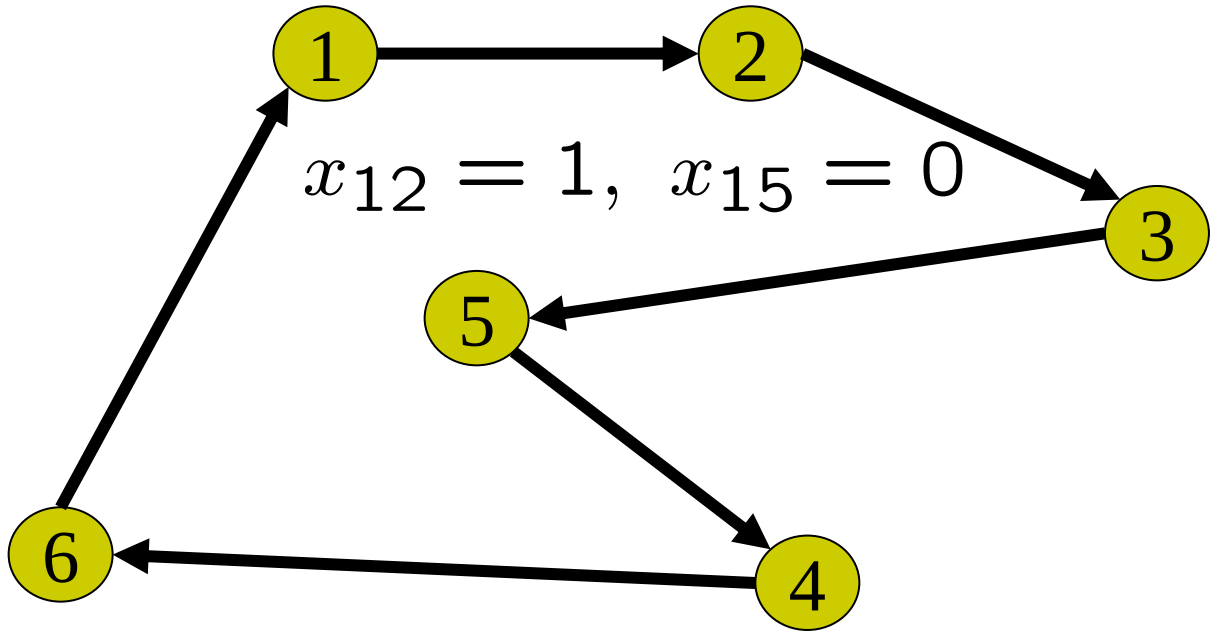
巡回セールスマン問題の定式化: 変数

変数の定義

$x_{ij} = 1$ (セールスマンが都市iの次に都市jに行くとき)

$x_{ij} = 0$ (それ以外のとき)

(変数 x_{ii} は考えない)



巡回セールスマン問題の定式化: 条件式

変数の定義

$x_{ij} = 1$ (セールスマンが都市iの次に都市jに行くとき)
$x_{ij} = 0$ (それ以外のとき)
(変数 x_{ii} は考えない)

条件式の定義

各変数は 0-1: $x_{ij} \in \{0, 1\}$ $(i = 1, \ldots, n, j = 1, \ldots, n, i \neq j)$

都市iを出発する回数はちょうど一回: $\sum_{j:j \neq i} x_{ij} = 1$ $(i = 1, \ldots, n)$

都市jに来る回数はちょうど一回: $\sum_{i:i \neq j} x_{ij} = 1$ $(j = 1, \ldots, n)$

この条件で十分か？

部分巡回路

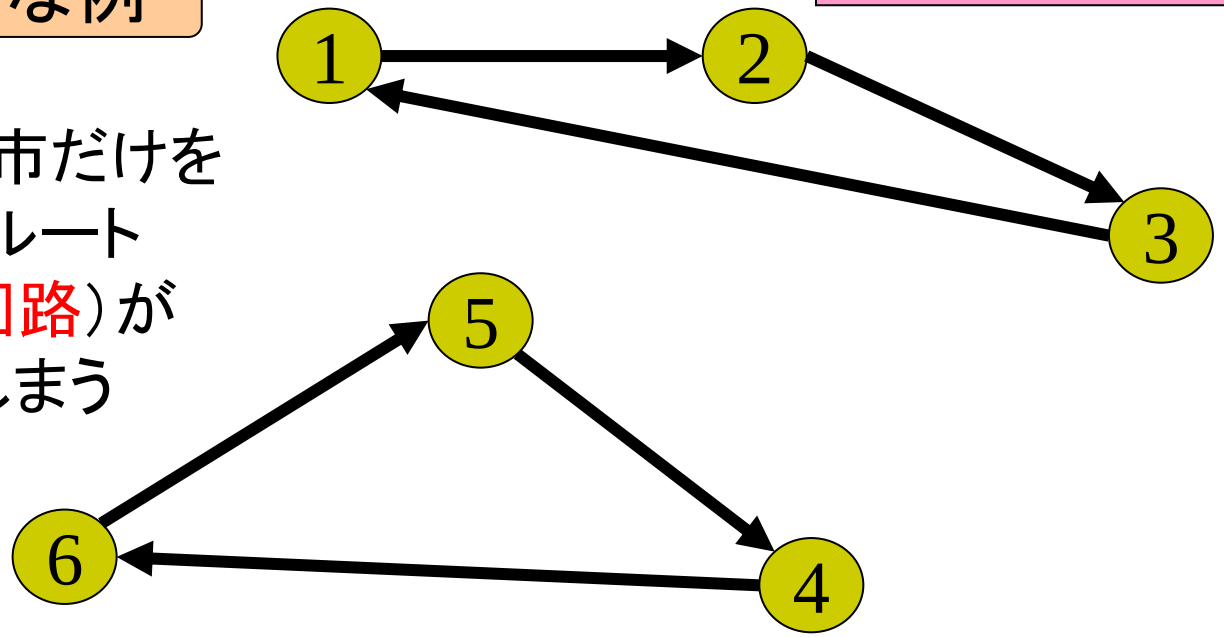
都市iを出発する回数はちょうど一回: $\sum_{j:j\neq i} x_{ij} = 1 \ (i = 1, \ldots, n)$

都市jに来る回数はちょうど一回: $\sum_{i:i\neq j} x_{ij} = 1 \ (j = 1, \ldots, n)$

この条件で十分か？

不十分な例

一部の都市だけを
巡回するルート
（部分巡回路）が
出てきてしまう

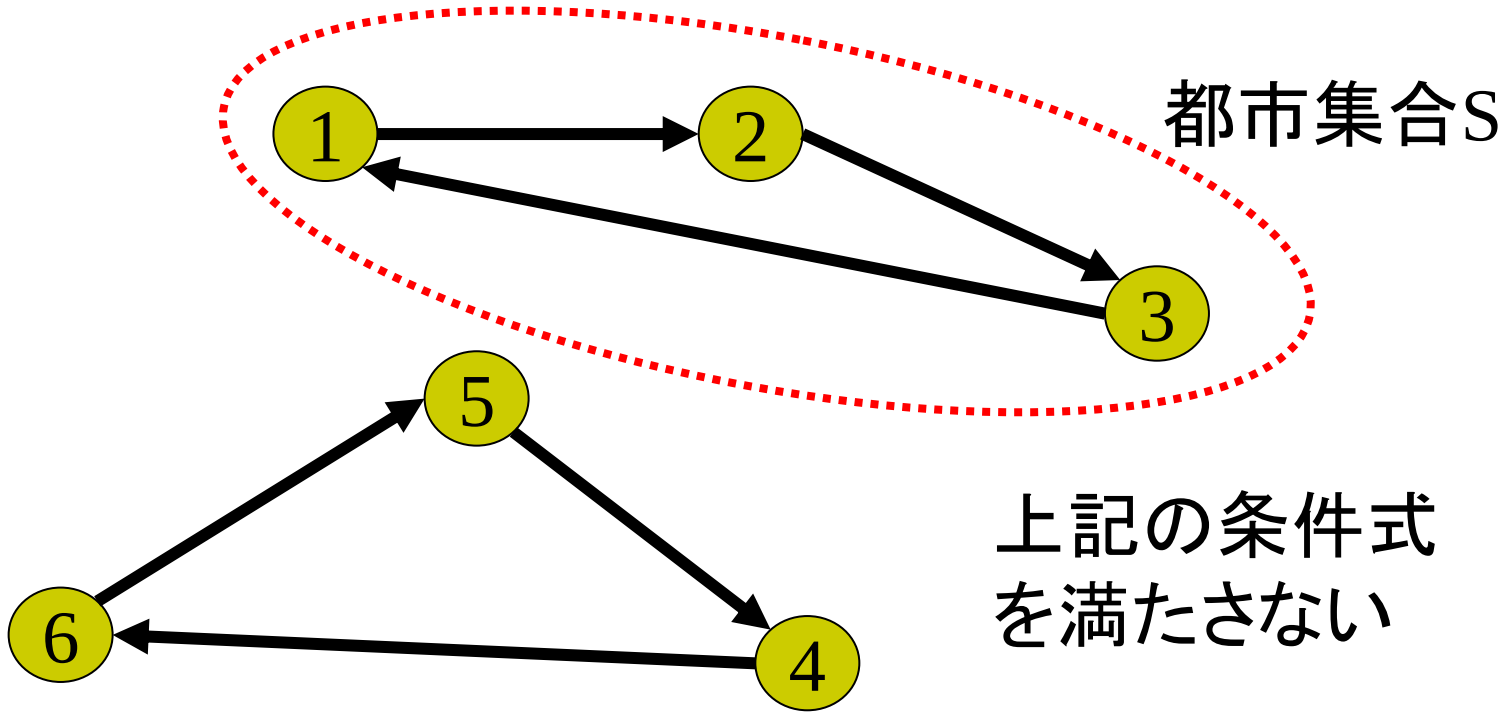


巡回セールスマン問題の定式化: 部分巡回路の除去

条件式の定義(続き)

セールスマンは少なくとも一度，都市の集合Sの外に出て行く:

$$\sum_{i \in S} \sum_{j \notin S} x_{ij} \geq 1 \quad (S \subset \{1, \ldots, n\},\ S \neq \emptyset)$$



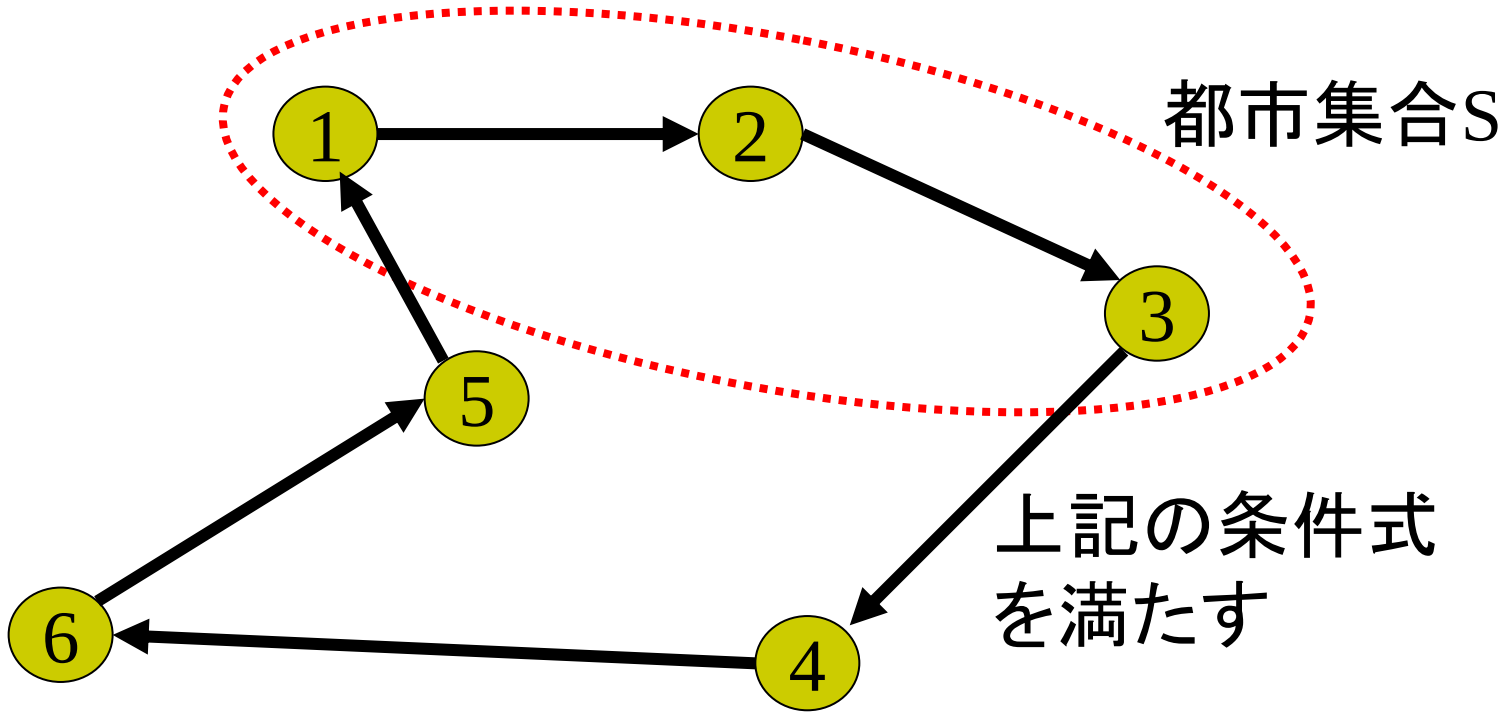
上記の条件式
を満たさない

巡回セールスマン問題の定式化: 部分巡回路の除去

条件式の定義(続き)

セールスマンは少なくとも一度，都市の集合Sの外に出て行く:

$$\sum_{i \in S} \sum_{j \notin S} x_{ij} \geq 1 \quad (S \subset \{1, \ldots, n\},\ S \neq \emptyset)$$

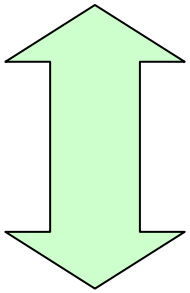


巡回セールスマン問題の定式化: 部分巡回路の除去

条件式の定義(続き)

セールスマンは少なくとも一度，都市の集合Sの外に出て行く:

$$\sum_{i \in S} \sum_{j \notin S} x_{ij} \geq 1 \ (S \subset \{1, \ldots, n\},\ S \neq \emptyset)$$



他の条件式の下で，
2つの条件は等価

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1 \ (S \subset \{1, \ldots, n\},\ 2 \leq |S| \leq n - 1)$$

巡回セールスマン問題の定式化: 目的関数

目的関数の定義

総距離を最小に: 最小化 $\sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$

閑話休題：錠を開ける難しさ

- 3種類の錠のうち，開けるのが最も難しいのはどれか？

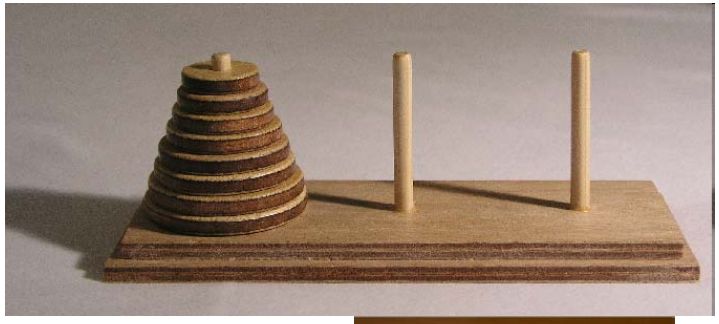
（1）4桁の1～8の数字を合わせる

（2）3桁の0，1～9の数字を合わせる

（3）0，1～9の中から4つの数字を選ぶ

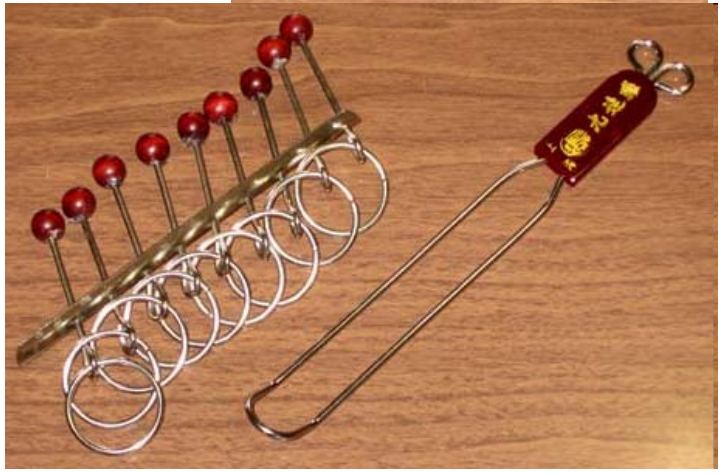
閑話休題：パズルの難しさ

- 下記のパズルは何故難しいか？



ハノイの塔

チャイニーズリング



組合せ爆発

- IPにより定式化した3つの問題の解は，ある集合の部分集合

➔すべての解を列挙すれば最適解が求められる

  - 割当問題：解は{1, …, n} の置換に対応 ➔ n! 個の解
  - ナップサック問題：高々 2^n個の解
  - 巡回セールスマン問題：最初の都市＝1，次の都市の選択肢は n-1, その次の都市は n-2，．．． ➔ (n-1)! 個の解

- すべての解の列挙は現実的に不可能

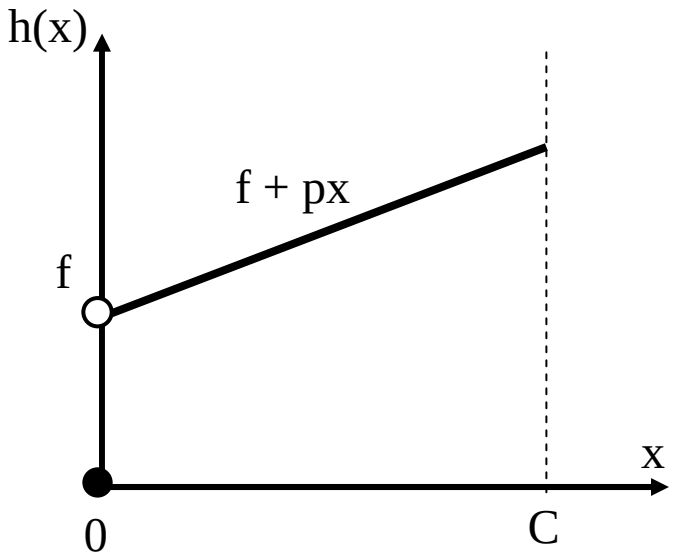
➔ 効率の良いアルゴリズムの必要性

MIPによる定式化：固定コスト関数の表現

- 生産計画などの問題では，次のような固定コスト関数が目的関数や条件式にしばしば現れる

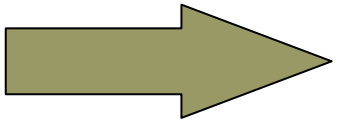
$$h(x) = \begin{cases} f + px & (0 < x \leq C) \\ 0 & (x = 0) \end{cases}$$

f, p は共に正の値



MIPによる定式化：固定コスト関数の表現

$$h(x) = \begin{cases} f + px & (0 < x \leq C) \\ 0 & (x = 0) \end{cases}$$



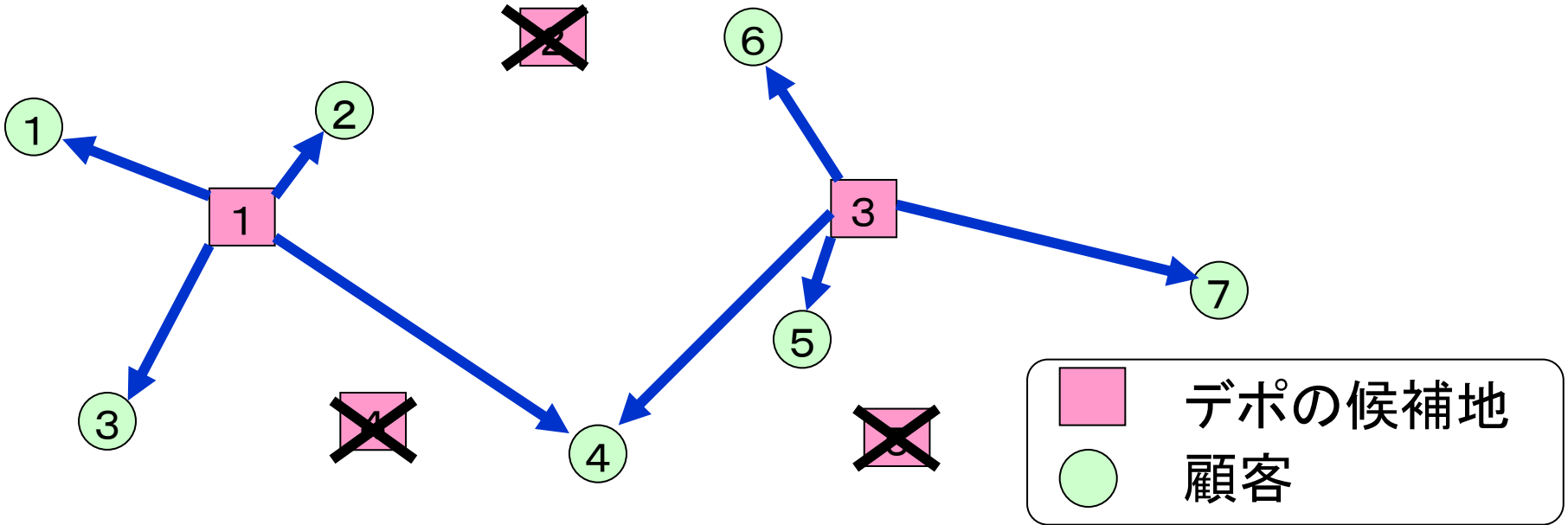
実数変数 x, 0-1 整数変数 y
を使って表現

目的関数もしくは条件式に出てくる $h(x)$ を $fy + px$ に置き換え
追加の条件式: $x \leq Cy,\ y \in \{0, 1\}$

- x > 0 ならば y = 1　　よって fy + px = f + px = h(x)
- x = 0 のときは y = 0 もしくは y = 1
  - ➔ 最適解では y = 0 を満たすことが多い
    （例：最小化問題で h(x) が目的関数に出てくる）
  - よって fy + px = 0 = h(0)

MIPによる容量なし施設配置問題の定式化

- 容量なし施設配置問題(Uncapacitated Facility Location Problem, UFL)
  - 配送デポの配置計画：デポから顧客に商品を配送
  - デポの候補地 N={1, 2, …, n}，顧客 M={1, 2, …, m}



MIPによる容量なし施設配置問題の定式化

- 条件：各顧客はいずれかのデポから配送される
- 目的：「設置コスト＋輸送コスト」をなるべく小さく
  - デポ j の設置コスト：f_j
  - デポ j から顧客 i への輸送コスト：c_{ij}

変数の定義

デポ j に関する変数 $y_j \in \{0, 1\}$ $(j = 1, \ldots, n)$：

$y_j = 1$ (デポjを設置)，$y_j = 0$ (それ以外のとき)

顧客iのデポjへの割当に関する変数$x_{ij} \in \{0, 1\}$ $(i \in M, j \in N)$:

$x_{ij} = 1$ (顧客iをデポjに割り当てたとき)，$x_{ij} = 0$ (それ以外のとき)

MIPによる容量なし施設配置問題の定式化

条件式の定義

顧客iはちょうど一つのデポに割り当てられる: $\sum_{j=1}^{n} x_{ij} = 1 \ (i = 1, \ldots, m)$

デポj が設置されているときのみ，顧客の割当が可能:

$$\sum_{i=1}^{m} x_{ij} \leq m y_j \ (j = 1, \ldots, n)$$

目的関数の定義

$$\underbrace{\sum_{j=1}^{n} f_j y_j}_{\text{デポ設置コスト}} + \underbrace{\sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}}_{\text{輸送コスト}} \text{ を最小化}$$