整数最適化アプローチへの入門

2019年3月22日（金）

IEICE 総合大会

宮代 隆平（東京農工大学）

なぜ整数最適化？

- なぜいま整数最適化か？
  → もともと種々の最適化問題を
  　よく表現できることに加え…
  - とにかく速くなった
    - 1991年→2015年で**4500億倍**のスピードアップ
  - 整数最適化のニーズが増えてきた
    - ヒューリスティクスを超えた最適化がほしい

- 本講演が（他分野の方にも）
  整数最適化を知ってもらうきっかけとなれば

前置き

- 整数計画法 / 計画問題　or　整数最適化？
  - 整数計画法 (integer programming; IP)
  - 整数計画問題 (integer program; IP)
- 近年の流れ
  - 整数計画法，整数計画問題 (IP)
    → 整数最適化，整数最適化問題 (IO)
    - programming (計画法) → optimization(最適化)
- 本日は「整数計画法 / 計画問題」でいきます

もくじ

- 整数計画アプローチへの入門
  - 整数計画問題の定義とそのクラス分け
  - 整数変数の使い道，定式化の例
  - 0-1変数を用いた定式化のトリック
  - 問題の規模と計算時間
  - 分枝限定法と良い定式化，難しい問題の特徴
  - 整数計画ソルバーについて
  - おわりに，参考資料

線形計画問題

- 線形計画問題 (linear program; LP)
  - もっとも基本的な 数理計画問題
  - 目的関数は線形，制約式も線形
  - 変数は連続変数（小数値を取れる）
    - 整数計画問題ではない
  - 単体法，内点法で効率的に解ける

$$\begin{aligned} \text{minimize} \quad & \boldsymbol{c}^\top \boldsymbol{x} \\ \text{subject to} \quad & A\boldsymbol{x} \leq \boldsymbol{b}. \end{aligned}$$

連続最適化問題

- 他の連続な数理計画問題
  - 凸二次計画問題 (QP)
  - 凸二次制約計画問題 (QCP)
  - 二次錐計画問題 (SOCP)
  - 半正定値計画問題 (SDP, LMI)
  - 凸計画問題 (convex nonlinear program)
    - 以上は連続最適化の範疇，内点法
  - 非凸な非線形計画問題 (nonconvex NLP)
    - 大域的最適化の範疇，NP困難で難しい

整数計画問題の定義

- 整数計画問題とは (Integer Programming; IP)
  - 変数の一部または全部に
    「整数値を取る」という条件,
    つまり整数条件が課された数理計画問題
  - 整数変数: 整数条件つきの変数
  - NP困難

- 総称して IP, MIP などと呼ぶ
  - M は整数変数と連続変数の「混合」
    (Mixed-Integer Programming; MIP)

整数計画問題のクラス分け

- 整数計画問題 (IP) のクラス分け
  - 線形整数計画問題 (MILP)
    - 0-1 整数計画問題 (0-1 IP)
  - 凸二次整数計画問題 (MIQP)

そこそこ解ける☺

  - 整数二次制約計画問題 (MIQCP)
  - 整数二次錐計画問題 (MISOCP)
  - 整数半正定値計画問題 (MISDP)

かなり難しい☹

- 基本的に分枝限定法（後述）で解く

整数変数の使い道

- 整数変数は何に使うのか？
  - 一般の整数変数
    - 分割できないモノの個数

→ 何らかの数量を表す時には使われる

  - **0-1変数**
    - 何らかの yes/no を表す
    - 定式化の中心的な役割

0-1 変数の使い方

- よくある 0-1 変数の使い方
  - 1~4 日目に作業 A, B, C, D のどれをやるか？
  - 1 以上 4 以下の値をとる
    整数変数 x_a, x_b, x_c, x_d を準備し,
    「$x_a = 3$ なら作業 a を 3 日目にやる」とする
    - 制約が線形制約で書きにくい & 解きにくい
  - 0-1変数 x_{a1}, x_{a2}, x_{a3}, x_{a4}, x_{b1}, … を準備し,
    「x_{a3} が 1 なら作業 a を 3 日目にやる」とする

- IP で主な役割を果たすのは 0-1 変数

IP の適用例

- IP の適用例
  - 非常に多彩な最適化問題が
    定式化できる(しやすい)
  - 他分野への応用例
    - ロジスティクス，配送計画，意思決定，
      スケジューリング，都市工学，…
    - グラフ理論，画像処理，信号処理，機械学習，…
  - 「●● problem　integer programming」で検索

AIC最小化問題

- AIC値（BIC値）が最小になるような
  説明変数の部分集合と偏回帰係数 $\boldsymbol{a}$ を求める
  - n 個のサンプル $\{(y_i; x_{i1}, x_{i2}, \ldots, x_{ip}) \mid i = 1, 2, \ldots, n\}$
  - p 個の説明変数 $\boldsymbol{y} = X\boldsymbol{a} + \boldsymbol{\varepsilon}$

  - AIC値： $n \log(||\boldsymbol{y} - X\boldsymbol{a}||_2^2/n) + 2k$
    - k は選択した説明変数の個数
      （k も数理計画の変数）

→ AICの最小化問題は MISOCP
（混合整数二次錐計画問題）として定式化できる

AIC最小化問題の定式化

$$\text{minimize} \quad f$$

$$\text{subject to}$$

$$\sum_{i=i}^{n} \left(y_i - \left(a_0 + \sum_{j=1}^{p} a_j x_{ij} \right) \right)^2 \leq f \cdot \sum_{j=0}^{p} \left(w_j \cdot \exp\left(-\frac{2j}{n} \right) \right) \quad (i = 1, 2, \ldots, n),$$

$$\sum_{j=0}^{p} (j \cdot w_j) = \sum_{j=1}^{p} z_j, \qquad \sum_{j=0}^{p} w_j = 1,$$

$$-Mz_j \leq a_j \leq Mz_j \quad (j = 1, 2, \ldots, p),$$

$$\boldsymbol{a} \in \mathbb{R}^{n+1},\ f \in \mathbb{R}_+,\ \boldsymbol{w} \in \{0,1\}^{p+1},\ \boldsymbol{z} \in \{0,1\}^{p}.$$

Miyashiro, Takano (2015)
In EJOR

LDPC符号の最尤復号化

- LDPC符号の最尤復号化

$$\begin{aligned} \text{minimize} \quad & \boldsymbol{\lambda}^\top \boldsymbol{x} \\ \text{subject to} \quad & H\boldsymbol{x} = 2\boldsymbol{z}, \\ & \boldsymbol{x} \in \{0,1\}^n,\ \boldsymbol{z} \in \mathbb{Z}_+^m. \end{aligned}$$

  - 線形整数計画問題 (MILP)
    - ただし，一見してそのままでは難しそうと感じる

参考：柿沼, 高野,　整数最適化復号法の性能評価,
情報科学研究 38 (2017), pp. 17-24

IP の表現能力

- IP の表現能力は低いか？
  - 基本的には線形の制約で表したい
    - よくて凸二次制約，二次錐制約
  - 現実の制約条件を表現できないのでは？

- 0-1 変数を用いたいくつかのトリック

0-1 変数の積

- 0-1 変数 x_1, x_2 の積
  - $y = x_1 \cdot x_2$ となるような 0-1 変数 y を作りたい
    → このままでは線形にならない
  - 0-1 変数 y と以下の不等式を導入
    - $y \geqq x_1 + x_2 - 1$
    - $y \leqq x_1$
    - $y \leqq x_2$

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

big-M 法

- big-M法： 論理関係 (if-then) を線形不等式で
  - 0-1 変数 y が 0 なら連続変数 x は 0,
    y が 1 なら x に制約なし

→ $x \cdot (1-y) = 0$ と書きたいが非線形に

  - 何らかの理由で $|x|$ の上界値 M が既知
    （y に関わらず $-M \leqq x \leqq +M$）

→ $-My \leqq x \leqq +My$

    - 0-1 変数を「制約式のスイッチ」に

定式化のトリックについて

- 他にも様々なトリックがある
  - 多彩な問題が表現できる
  - ただし，計算が遅くなる要因の一つでもある

- 基本的なものは，以下によくまとまっています

藤江哲也： 整数計画法による定式化入門.
オペレーションズ・リサーチ, 57 (2012),
pp. 190−197

うまくいかない場合

- IPアプローチがうまくいったら？
  - 真の最適解が求まる ☺
    - IP の一番の魅力

- IPアプローチがうまくいかない時は？
  - 計算が終わらない ☹
    - 得られた暫定的な許容解に最適性の保証が無い
    - かなり解の目的関数値が悪い場合も…

問題の計算時間について

- 問題の計算時間は何によって決まるか？
  - 線形計画 (LP) の場合
    - 制約式の本数，変数の個数が
      わりと良い計算時間の指標に
    - いずれにせよ高速
  - 整数計画 (IP) の場合
    - 制約式の本数，変数の個数は
      あてにならない！
      （とは言っても，超巨大な問題はやはり無理）

極端なケース

- 大きくても簡単な問題
  - MIPLIB 2017 の supportcase11
  - 0-1変数：806万個，制約式：1527万本

  → 商用ソルバーで1時間以内に解ける

- 小さいのに難しい問題
  - MIPLIB 2017 の pb-market-split8-70-4
  - 0-1変数：71個，制約式：17本

  → 2019年3月現在，許容解が見つかっていない（計算が終わらない）

分枝限定法

- なぜこんなことが起きるのか？
  → IP を解くアルゴリズムの理解が必須

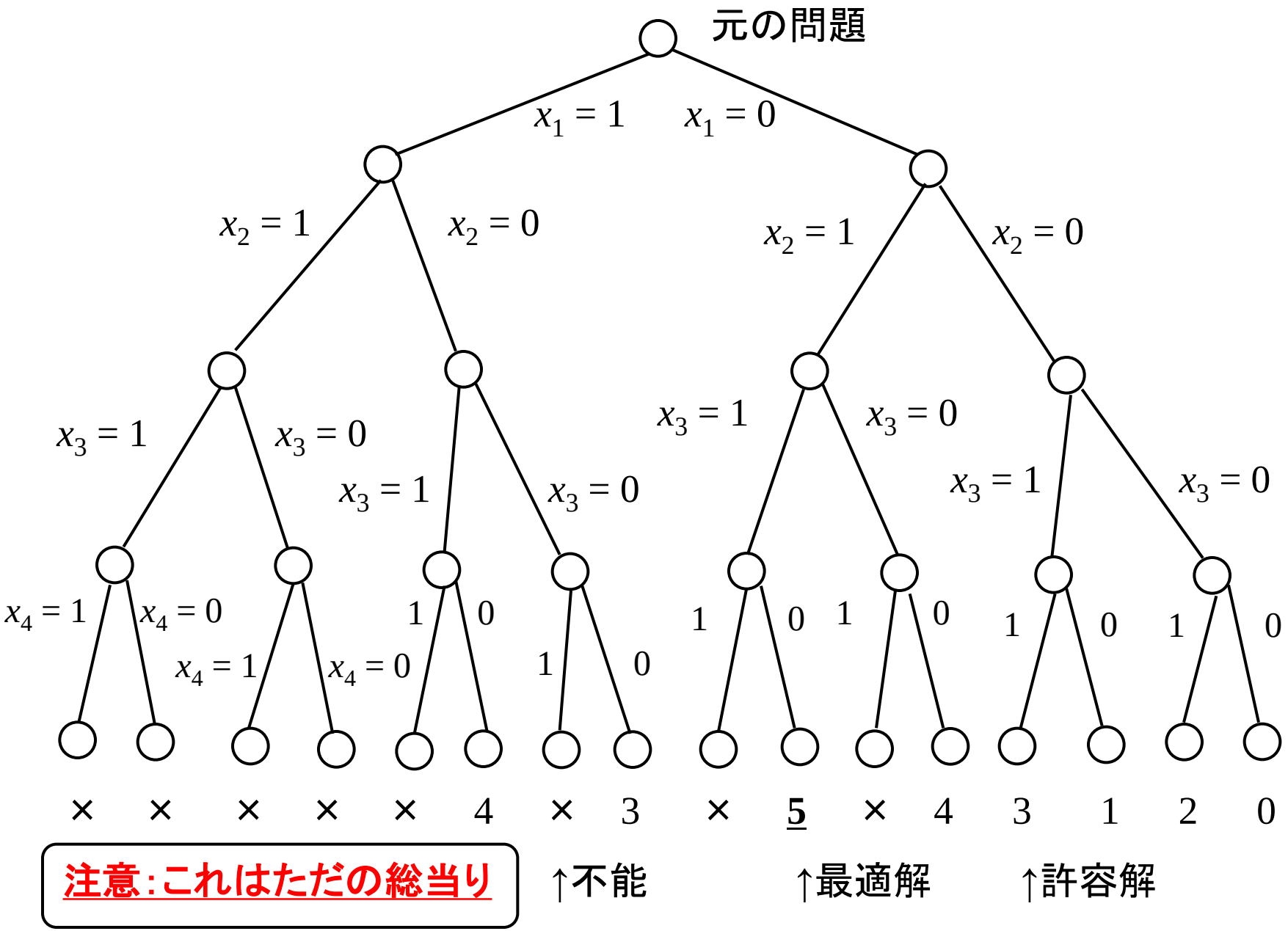
- 分枝限定法 (branch & bound method)
  - IP を場合分け（分枝）しながら，
    各ノードで対応した IP の連続緩和問題
    （IP から整数条件を除いた問題）を解き，
    その緩和問題の最適値が
    既に得られている IP の許容解より悪かったら
    枝刈り操作（限定）を行う方法

ナップサック問題

- 以下，最大化の 0-1 IP の場合として解説
  具体的に，以下のナップサック問題で説明する

$$
\begin{aligned}
\text{maximize} \quad & 3x_1 + 4x_2 + x_3 + 2x_4 \\
\text{subject to} \quad & 2x_1 + 3x_2 + x_3 + 3x_4 \leqq 4, \\
& x_1, x_2, x_3, x_4 \in \{0, 1\}.
\end{aligned}
$$

- 総当りで考えると $2^4 = 16$ 通り
  - 一般には 2^n 通りで「組合せ爆発」を引き起こす


元の問題
$x_1 = 1$
$x_1 = 0$
$x_2 = 1$
$x_2 = 0$
$x_2 = 1$
$x_2 = 0$
$x_3 = 1$
$x_3 = 0$
$x_3 = 1$
$x_3 = 0$
$x_3 = 1$
$x_3 = 0$
$x_3 = 1$
$x_3 = 0$
$x_4 = 1$
$x_4 = 0$
$x_4 = 1$
$x_4 = 0$
1
0
1
0
1
0
1
0
1
0
1
0
1
0
× × × × × 4 × 3 × 5 × 4 3 1 2 0
注意：これはただの総当り
↑不能
↑最適解
↑許容解

連続緩和問題（線形緩和問題）

- 連続緩和（線形緩和）と
  連続緩和問題（線形緩和問題）
  - MILP ＝ LP＋整数条件
    → MILP から整数条件を抜くと LP に
  - 連続緩和問題（線形緩和問題）:
    元の IP から整数条件を抜いた LP

  - 例: 0-1 IP の連続緩和
    - 0-1制約 $x \in \{0, 1\}$ は「$0 \leqq x \leqq 1$ かつ $x \in \mathbb{Z}$」
      → 連続緩和すると $0 \leqq x \leqq 1$

ナップサック問題の連続緩和問題

- ナップサック問題の連続緩和問題

  maximize $3x_1 + 4x_2 + x_3 + 2x_4$

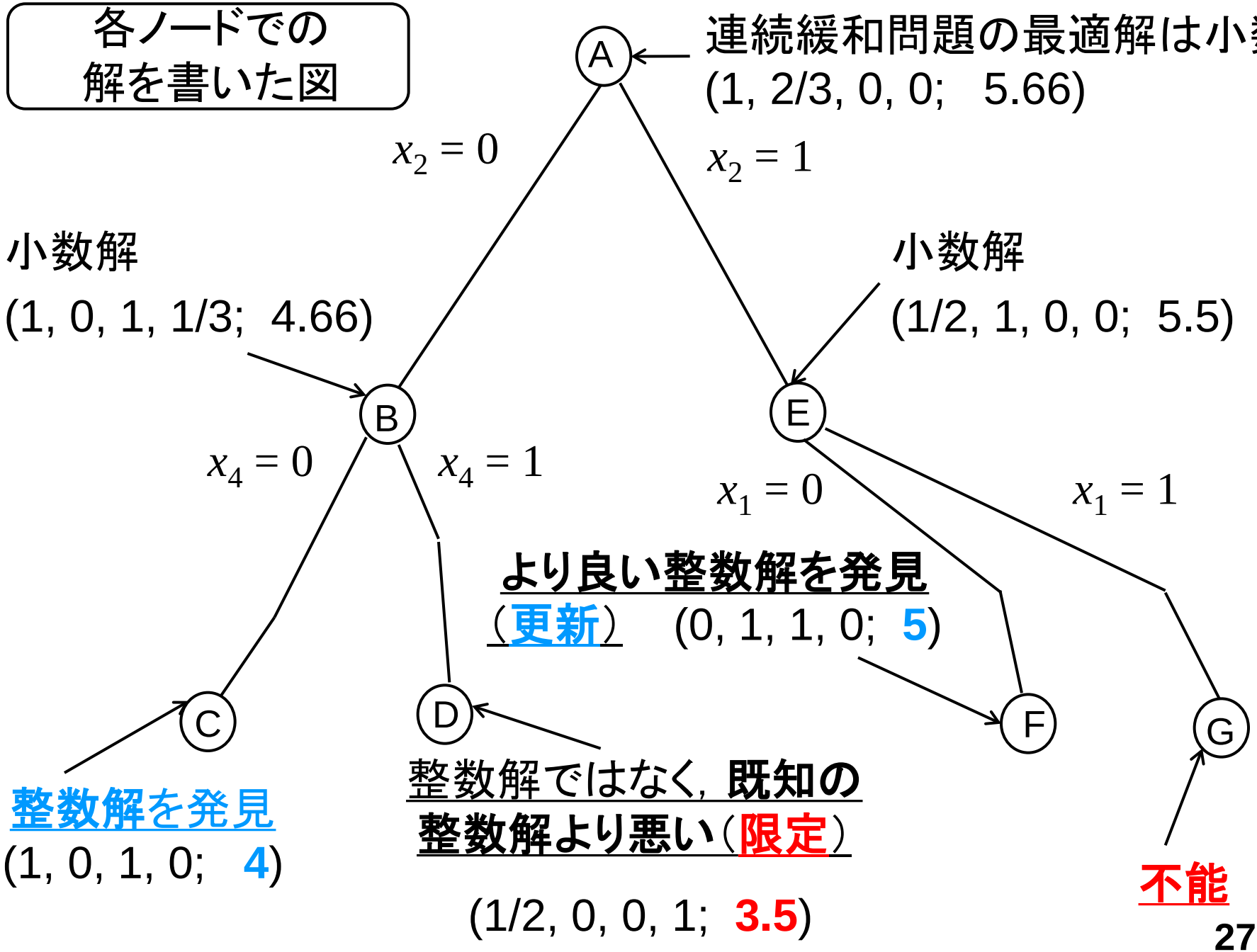
  subject to $2x_1 + 3x_2 + x_3 + 3x_4 \leqq 4,$

  $0 \leqq x_1, x_2, x_3, x_4 \leqq 1.$

  - これは LP → 単体法で高速に解ける

- IP を解くかわりに，対応する連続緩和問題を解くことによって情報が得られないか？

各ノードでの
解を書いた図



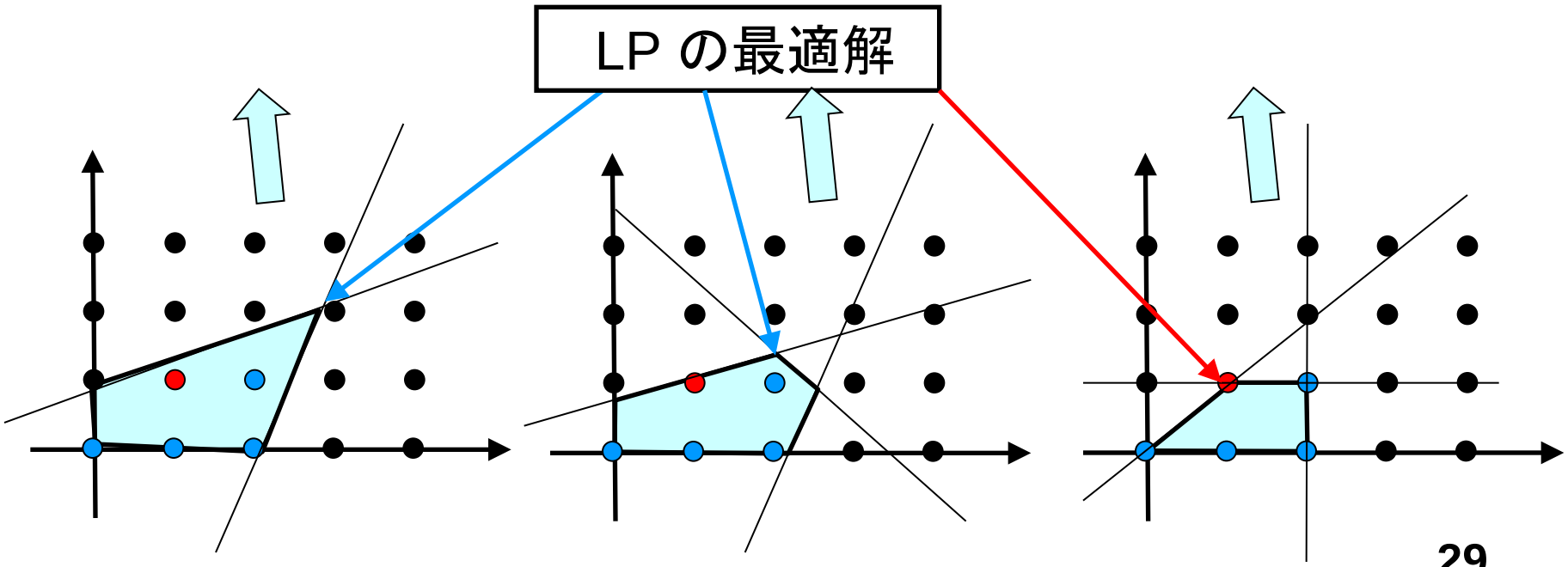
分枝限定法の基本

- 元問題の連続緩和問題を解く
  - よりよい整数解が得られたら，
    「これまでの最も良い整数解」を更新する
  - 不能なら別のノードに
  - 小数解が得られたら，目的関数値をチェック
    - 既知の整数解より悪かったら限定
    - 既知の整数解より良かったら
      制約式を追加して分枝（子問題を作成）
- これらをノードが無くなるまで行う

比較：**IPの暫定許容解の値 vs LPの最適値**

良い定式化

- 高速に解ける「良い定式化」とは
  - 限定操作（枝刈り）を有効に機能させる
    → 元のIPの最適値と連続緩和問題の最適値が
    なるべく近くなる定式化が良い定式化



定式化の違いによる計算時間の差異

- 定式化の良し悪し
  - 同一の問題でも，定式化は複数考えられ，
    定式化の良し悪しで求解速度が全く異なる
- あるスケジューリング問題の例
  - 0-1変数　612個の定式化：5時間でも終わらず
  - 0-1変数2592個の定式化：20分で計算終了
- 良い定式化を考える研究

pb-market-split8-70-4

- 難しい問題　pb-market-split8-70-4
  - 0-1変数71個，制約式17本の小さな問題
    → 前処理すると 8 本の制約式に
  - 8 本の制約式は全て「等号ナップサック制約」
    - 連続緩和問題の情報が役にたたない
    - 分枝していっても整数解がなかなか得られない

$$\sum_{j=1}^{n} a_{ij} x_j = b_i \quad (i = 1, 2, \ldots, m)$$

    - ＝ が ≦ なら非常に楽（ナップサック制約）

難しい問題の特徴

- 分枝限定法が遅くなる要因
  - 連続緩和問題の最適値が IP の最適値と非常に離れている
  - 最適解の個数が多い，または問題が不能
    - 限定操作（枝刈り）が働きにくい
  - 内部に難しい制約が含まれている，トリッキーな定式化
  - 問題自体が巨大
- 問題を読むと，なんとなくわかる
    - ただし係数データにも依存する

ソルバー

- 実際に IP を解くときは…？
  - 鉄則：自分で実装はしない
  - 出来合いの「ソルバー」を用いる
    - もちろん研究開発目的は別

- ソルバー
  - 整数計画問題を解くためのソフトウェア
  - 商用と，非商用（≒フリー）のものがある

ソルバーの進歩

- ？年間で？倍速くなった
  - 1991 → 2015 で **4500億倍**
    [Bertsimas, King, Mazumder, 2016]
  - ハードウェアの改善 ＜ アルゴリズムの改善
    （1991年のCPU：Intel i486, 16MHz）

- 同一マシン上の同一インスタンス
  - 2009年：45時間（CPLEX 12.1）
  - 2013年：　1時間（CPLEX 12.5）

ソルバー高速化の要因

- ソルバーの高速化の要因
  - ハードウェアの高速化（CPU，メモリ）
  - マルチコア化
  - LPの高速化（緩和問題を解くスピード）
  - IPの前処理の進化
  - IPのヒューリスティクスの進化（良い解を高速に）
  - カットを生成するアルゴリズム
    （自動的に妥当不等式を追加する）

ソルバーの現状

- ソルバーの現状
  - 連続最適化問題は高速に解ける
  - 各種 IP を解く速度は，ソルバーによって段違い
  - 速度：高速な商用ソルバー ≫
    ≫ 高速な非商用ソルバー
    ≒ そこそこの商用ソルバー ＞ それ以下
  - 高速な商用ソルバーと高速な非商用ソルバーで計算速度が10倍以上違います
    - 商用で数分，フリーで1日以上とかはざら

高速なソルバー

- 高速な商用ソルバー
  - 高速な商用ソルバー
    - アカデミックは無料～20万円程度，
      一般ライセンスは100万円以上のことが多い
  - クラウドの利用も

- 高速な非商用ソルバー（2019年3月現在）
  - SCIP
  - MIPCL

整数計画アプローチをとる前に

- 整数計画アプローチが向いているか？
  - 真の最適解は必要か？
    現状の解では不十分か？
  - 許される計算時間は？　問題のサイズは？
  - 入力データはどの程度正確か？
    最適解を実際に実現できるか？
  - 商用ソルバーは購入可能か？
  - 良い許容解は知っているか？
  - 難しい問題か？ 定式化は良いものか？

おわりに

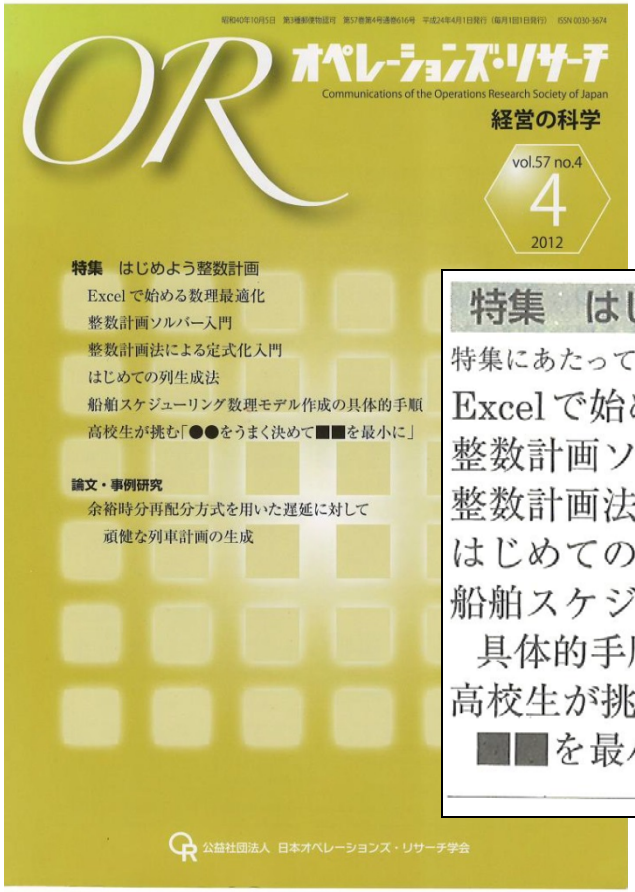
- 最適化に IP アプローチを使うと嬉しいこと
  - 真の最適解が得られる(こともある)
    - 公平性などの担保に必須
    - 計算時間はかかることが多い
      → モデル化，定式化が重要
- 解けない問題もある
    - 計算途中の解で満足する
    - 定式化や分枝限定法の改良をする
    - 緩和に基づく rounding や列生成法などを行う
    - IP の使用をあきらめる

参考ホームページ

- ソルバーの使い方については，
  より詳しい情報を
  https://www.tuat.ac.jp/~miya/ipmemo.html
  に載せています
  - 「整数計画法メモ」で検索
  - いくつかの参考文献もダウンロード可能

関連の OR 学会誌

- OR学会誌 2012年4月号
  （全記事が web から
  無料ダウンロード可）



特集　はじめよう整数計画