ILOG_OPTIMIZATION SEMINAR
for OR Experts

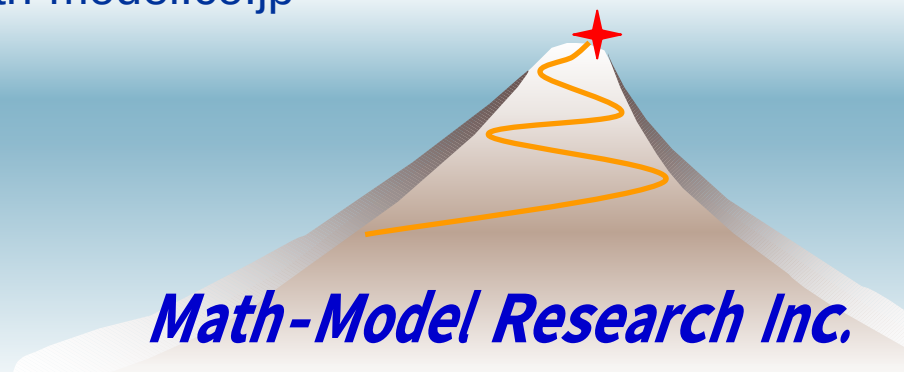
2006.12.07

混合整数計画法(ＭＩＰ)を使おう：
モデル化の実際とその背景

(株) 数理モデリング研究所

野末 尚次

URL:www.math-model.co.jp



今日の話題

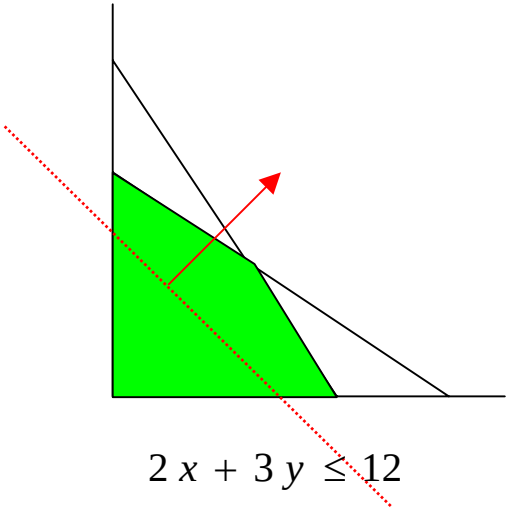
混合整数計画法による問題の定式化と解法を理解し、応用できるように、CPLEX（ILOG社）を ベースに解説します。

1) 混合整数計画法とは
2) 論理的な条件の表現の仕方
3) 整数条件を緩和した線形計画問題の利用の考え方
1) 基本的な仕組み：Branch&Bound法の考え方
4) 解の探索法－1：Branch＆Cut法の考え方
5) 解の探索法－2：集合被覆法と列生成法の考え方
6) CPLEXによる計算の体験 (変更)
7) 混合整数計画法では困難な問題
8) まとめ

Math-Model Research Inc.

最適化問題の分類（質的相違）

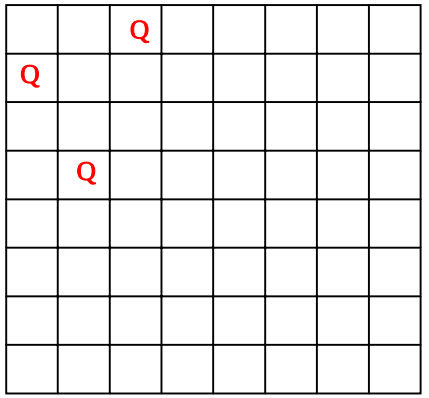
資源配分問題



$$2x + 3y \leq 12$$
$$3x + 2y \leq 12$$
$$x \geq 0,\ y \geq 0$$
$$x + y \rightarrow Max$$

線形計画法

8-Queen配置問題



$$x[i] \neq x[j],\ for\quad i \neq j$$
$$x[i]+i \neq x[j]+j,\ for\quad i \neq j$$
$$x[i]-i \neq x[j]-j,\ for\quad i \neq j$$

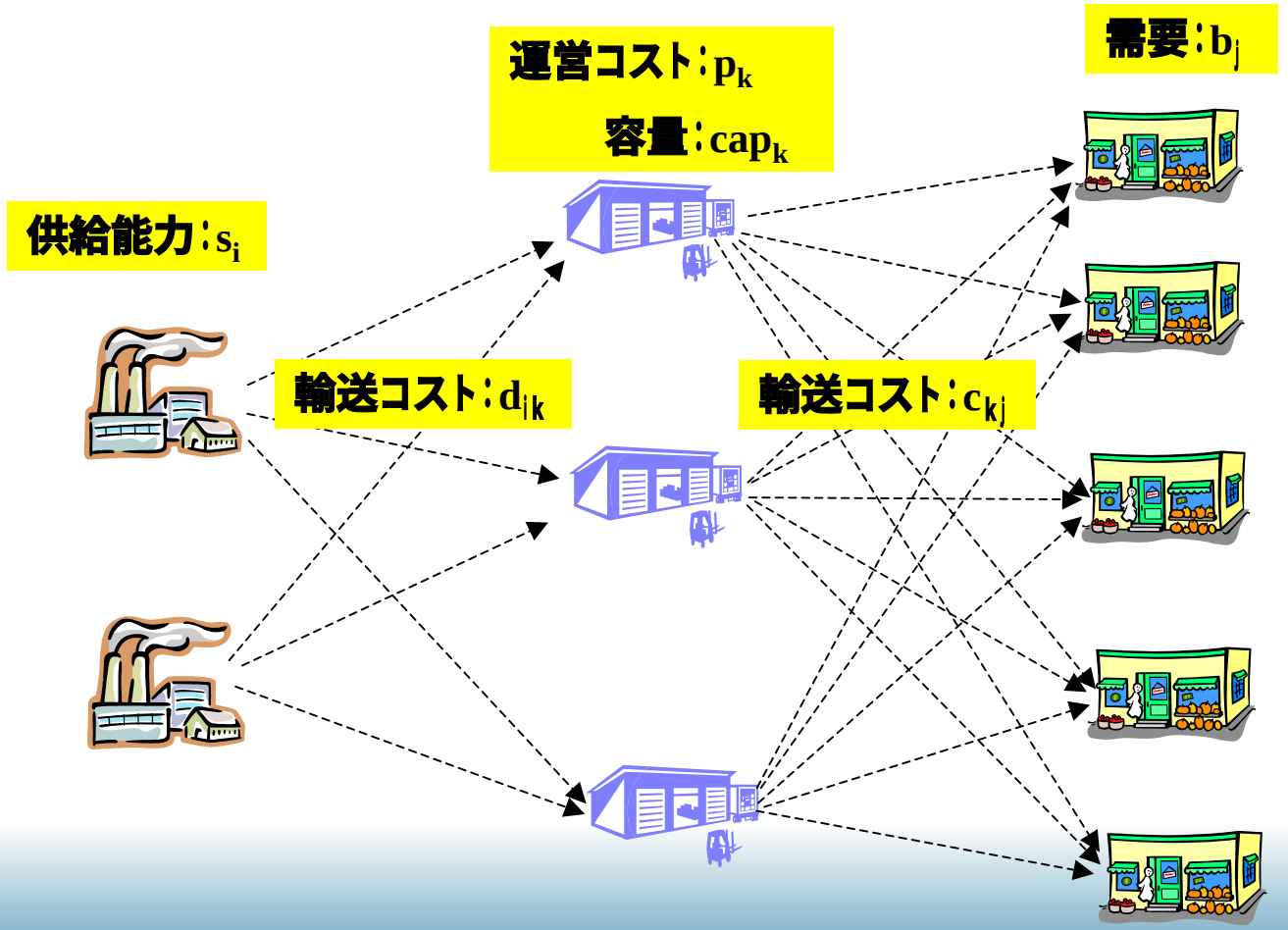
制約プログラミング

混合整数計画法

メタヒューリスティックス

倉庫の設置問題

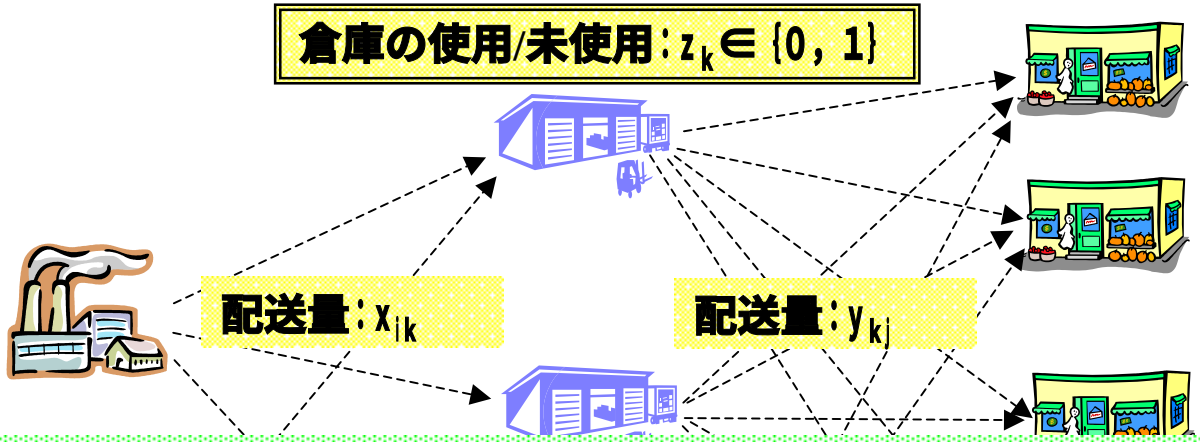
コスト最小の(複数) 倉庫の配置案？



Math-Model Research Inc.

混合整数計画法による定式化

(インディケータ変数の導入)



制約条件：

$\Sigma_k x_{ik} \leqq s_i$

$\Sigma_k y_{kj} \geqq b_j$

$\Sigma_i x_{ik} = \Sigma_j y_{kj}$

$\Sigma_i x_{ik} \leqq cap_k * z_k$

$x_{ik} \geqq 0 \quad y_{kj} \geqq 0 \quad z_k \in \{0, 1\}$

$z_k==1$, $\Sigma_i x_{ik}==0$も許容解!!

$p_k > 0$により,
$\Sigma_i x_{ik}==0$なら、$z_k==0$

目的関数：$\Sigma d_{ik} x_{ik} + \Sigma c_{kj} y_{kj} + \Sigma p_k z_k$

論理的な条件の追加：

条件1：倉庫の使用時は、L_0以上の輸送量が必要

条件2：L_1以下の場合には、使用料をRだけ減額する

条件1：$L_0 z_k \leqq \Sigma_i x_{ik}$

条件2：δ_kの導入：　$\delta_k = 1$　if 使用量がL_1以下の場合

$= 0$　　上記以外

$\Sigma_i x_{ik} \leqq L_1 \delta_k + M(1 - \delta_k)$　　(Mは大きな定数)

目的関数：$\Sigma d_{ik} x_{ik} + \Sigma c_{kj} y_{kj} + \Sigma p_k z_k - \Sigma R \delta_k$

⇩

誤った解の生成：　$\delta_k = 1$　$z_k = 0$　$\Sigma_i x_{ik} = 0$

⇩

追加条件：　$\delta_k \leqq z_k$

厳密な制約条件の
表現が難しい！

文献2)

論理的な条件の表現：CPLEX

```
IloAnd
IloOr
IloNot
IloIfThen
==          the equivalence relation
!=          the exclusive-or relation
```

IloAnd(x+y>=10,x+y<=20) == (δ==1)

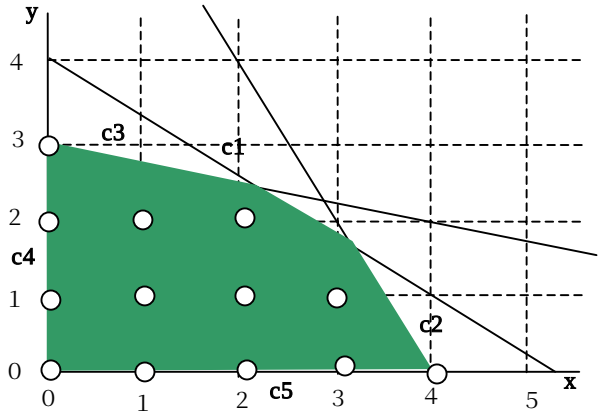
複雑な論理条件の
自動生成

(0) δの否定を指標とする：η0＝1－δ
(1) η0==0 => 次の両者がANDで成立
(x + y >= 10)： x + y >= 10 (1－η0)
(x + y <= 20)： x + y <= 20 (1－η0)＋ 1000000 η0
(2) η0==1 => 次の片方が成立： η1かη2が1の値を取る
η0=η1＋η2
!(x + y >= 10)： x + y <= 9.99999 η1＋ 1000000 (1－η1)
!(x + y <= 20)： x + y >= 20.00001 η2
追加変数：η0、η1、η2は、{0，1}変数

整数条件による解空間の構造

z ＝ 7x ＋ 8y －＞ 最大
Sub. to
c1: 3x + 4y <= 16
c2: 2x + 1y <= 8
c3: 1x + 4y <= 12
c4: x ≧ 0
c5: y ≧ 0
x: integer
y: integer

⇔



ハッチングしてある部分（凸多角形：P）に含まれる整数値の座標点が許容解を与える。

線形計画法のイメージ：凸多面体

Prob：スラック変数の導入

z ＝ 7x ＋ 8y －＞ 最大

Sub. to

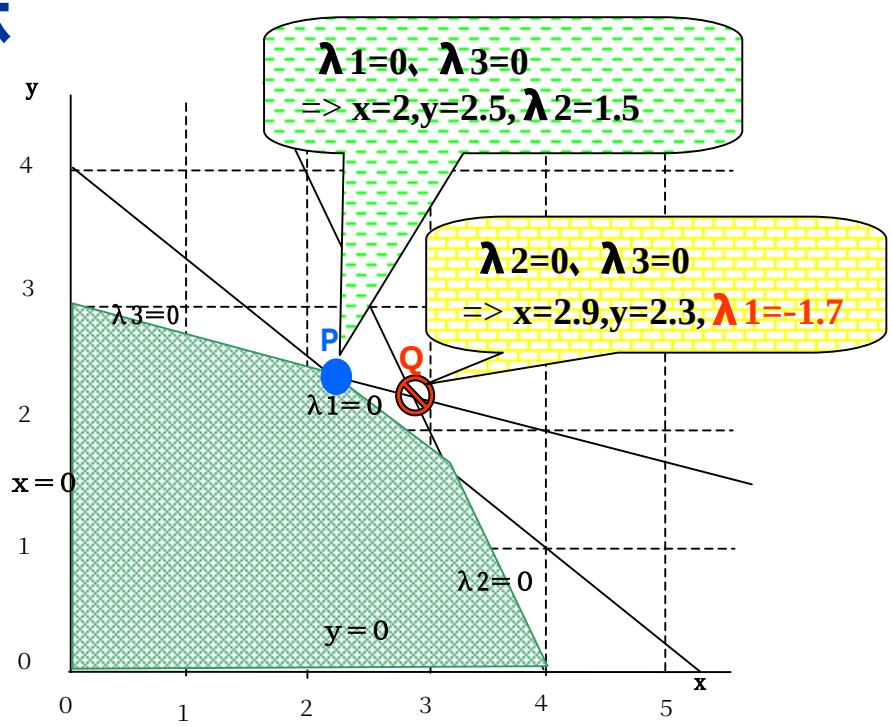
3x＋4y＋λ1 ＝16

2x＋1y ＋λ2 ＝ 8

1x＋4y ＋λ3＝12

x≧0、y≧0、

λ1≧0、λ2≧0、λ3≧0



スラックを導入した形式の意味：

1) 各変数＝0が境界を構成する。

2) この方程式には、

変数の数(5) － 式の数(3) ＝ 自由度(2)

があり、任意の2変数を独立変数とできる。

3) 任意の2変数 { u,v } の境界の交点で、点Pが定まる。

4) 定義式に、u = 0、v = 0を代入した時、

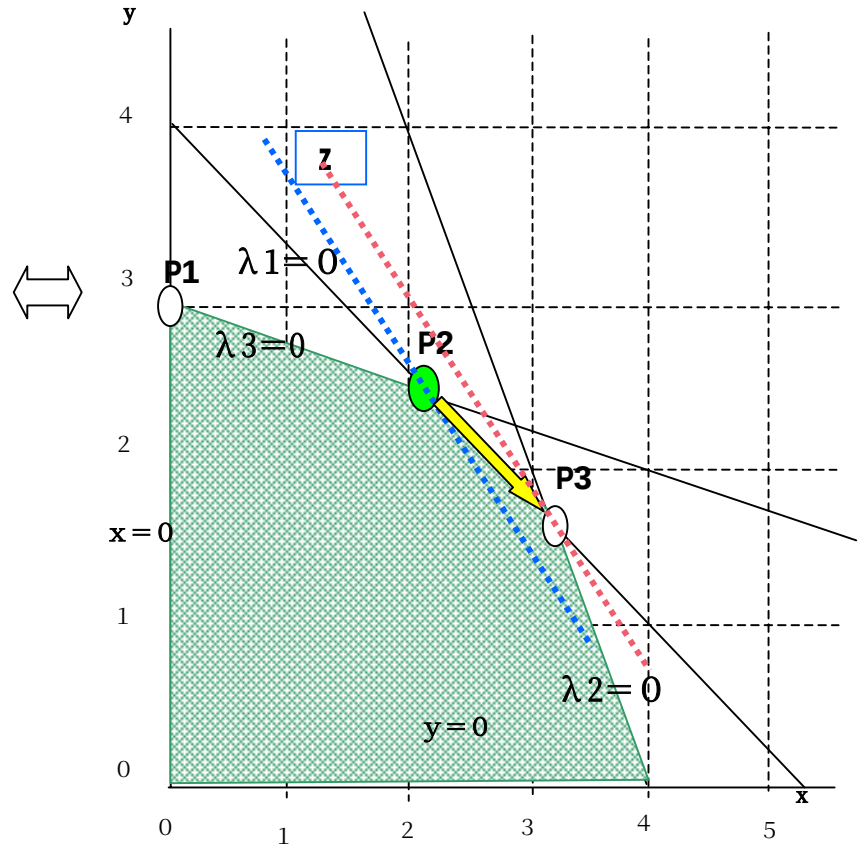
残りの変数が非負なら、凸多面体Pの端点である。

独立変数と限界コスト(Reduced Cost)

1) 独立変数として、{λ1、λ3} を選択
2) λ1=0、λ3=0の交点として、P2が定まる。
3) 方程式を独立変数で解いた形式に表現

$$x = 2 - 1/2\lambda 1 + 1/2\lambda 3$$
$$y = 5/2 + 1/8\lambda 1 - 3/8\lambda 3$$
$$\lambda 2 = 3/2 + 7/8\lambda 1 - 5/8\lambda 3$$

$$z = 34 - 5/2\lambda 1 + 3\lambda 3$$
(reduced cost)



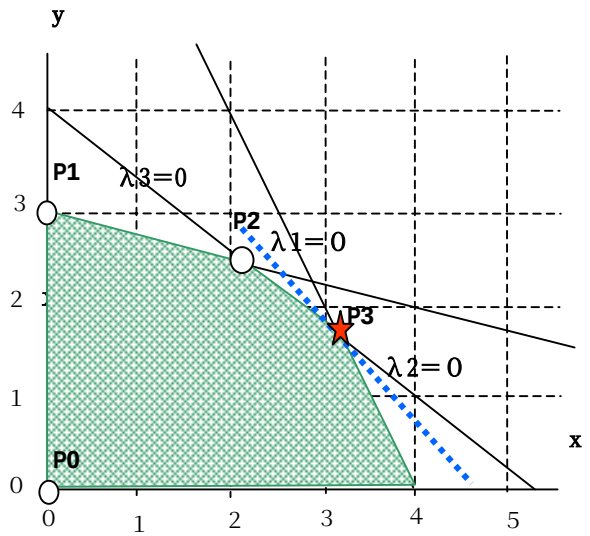
4) λ3を0から増した方が、zが増加する。

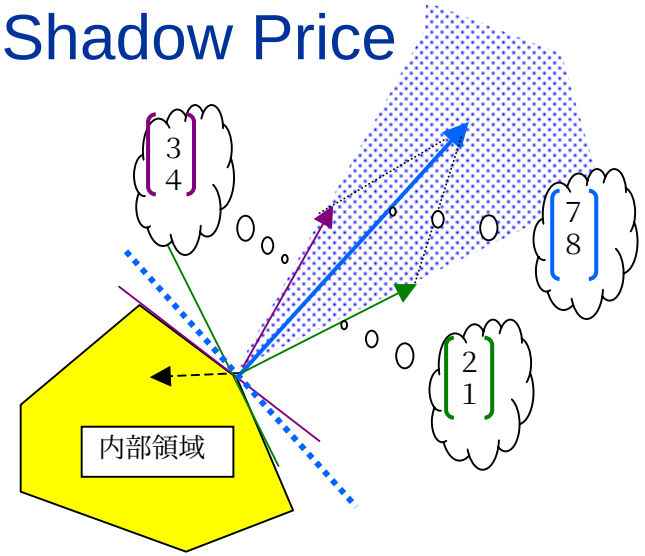
5) λ1=0のままで、λ3を増やすと、
最初にλ2＝0となる。

6) 独立変数 {λ1、λ2} のP3に移る。

最適性条件とShadow Price

P3 {λ1、λ2} が最適 ⇔





交点の最適性条件：
目的関数の係数ベクトルcが、交点Pでアクティブな制約条件の係数ベクトルa_kの凸1次結合で表される。

$$c = v a_2 + w a_3 + \cdots \quad \text{但し} \quad v \geqq 0,\ w \geqq 0, \cdots$$

$$x = 3.2 + 0.2\lambda 1 - 0.8\lambda 2$$
$$y = 1.6 - 0.4\lambda 1 + 0.6\lambda 2$$
$$\lambda 3 = 2.4 + 1.4\lambda 1 - 1.6\lambda 2$$
$$z = 35.2 - 1.8\lambda 1 - 0.8\lambda 2$$

$$\begin{pmatrix} 7 \\ 8 \end{pmatrix} = 1.8 \begin{pmatrix} 3 \\ 4 \end{pmatrix} + 0.8 \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

全てのReduced cost：$\gamma_j \leqq 0$

全てのShadow price：$\pi_i \geqq 0$
(dual変数)

線形計画法のまとめ

「最適解X*」～「shadow price π」～「reduced cost γ」

問題:

$Z = c'X \quad \rightarrow$ 最大

$AX \leqq b$

$X \geqq 0$

3) $c'X^* = \pi'b$

$c = \pi_1 a_1 + \pi_2 a_2 + \cdots$

$c'X^* = \pi_1 a_1'X^* + \pi_2 a_2'X^* + \cdots$

$= \pi_1 b_1 + \pi_2 b_2 + \cdots$

アクティブな制約 $a_k'X^* = b_k$

最適解X*の条件

1) reduced cost $\gamma \leqq 0$

2) shadow price $\pi \geqq 0$

[アクティブでない制約kのπ_kは0]

4) $\gamma_k = c_k - \Sigma_j \pi_j a_{jk}$

X_kをΔ増やすと、
・c_kΔだけ利益が増す
・現在の解は b_iがa_{ik}Δ減るので、
利益は、$(\Sigma_j \pi_j a_{jk})$Δだけ減少

k列

$$\cdots + \begin{pmatrix} c_k \\ a_{1k} \\ a_{2k} \\ a_{3k} \\ \vdots \\ a_{nk} \end{pmatrix} \Delta = \begin{pmatrix} Z \\ b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{pmatrix}$$

CPLEXによる表現

```
Maximize
 obj: x0 + 2x1 + 3x2
Subject To
 c0:  -x0 +  x1 + x2  <= 20
 c1:   x0 - 3x1 + x2  <= 30
Bounds
 0 <= x0 <= 40
 0 <= x1
 0 <= x2
End
```

定式化2：式中心のモデル化

```
IloEnv   env;
   IloModel model(env);
   // 変数のオブジェクトの配列定義
   IloNumVarArray x(env);
   // 各変数のオブジェクトの定義
   x.add(IloNumVar(env, 0.0, 40.0));
   x.add(IloNumVar(env));
   x.add(IloNumVar(env));
   // 式オブジェクトの配列定義
   IloRangeArray  cst(env);
   // 各式のオブジェクトとレンジ定義
   cst.add(  -x[0] +   x[1] + x[2] <= 20);
   cst.add(   x[0] - 3*x[1] + x[2] <= 30);
   // モデルへの登録
   model.add(cst);
   // 目的関数のモデルへの登録
   model.add(IloMaximize(env, x[0] + 2*x[1] + 3*x[2]));
   // モデルの抽出
   IloCplex cplex(model);
   // 最適解の探索
   cplex.solve();
   // 解の印刷
   cplex.out() << "Optimal=" << cplex.getObjValue() << endl;
env.end();
```

Branch＆Bound法

整数条件を除いた緩和問題ＲＰを解いた時に、
　整数変数が整数となっていない場合には、整数解の探索を行う。

1) Branch：　整数となっていない整数変数 x_k　(現在の解 d_k)

二つの子問題を生成：

P_up：追加制約条件：　$x_k \geqq \mathrm{ceiling}(d_k)$

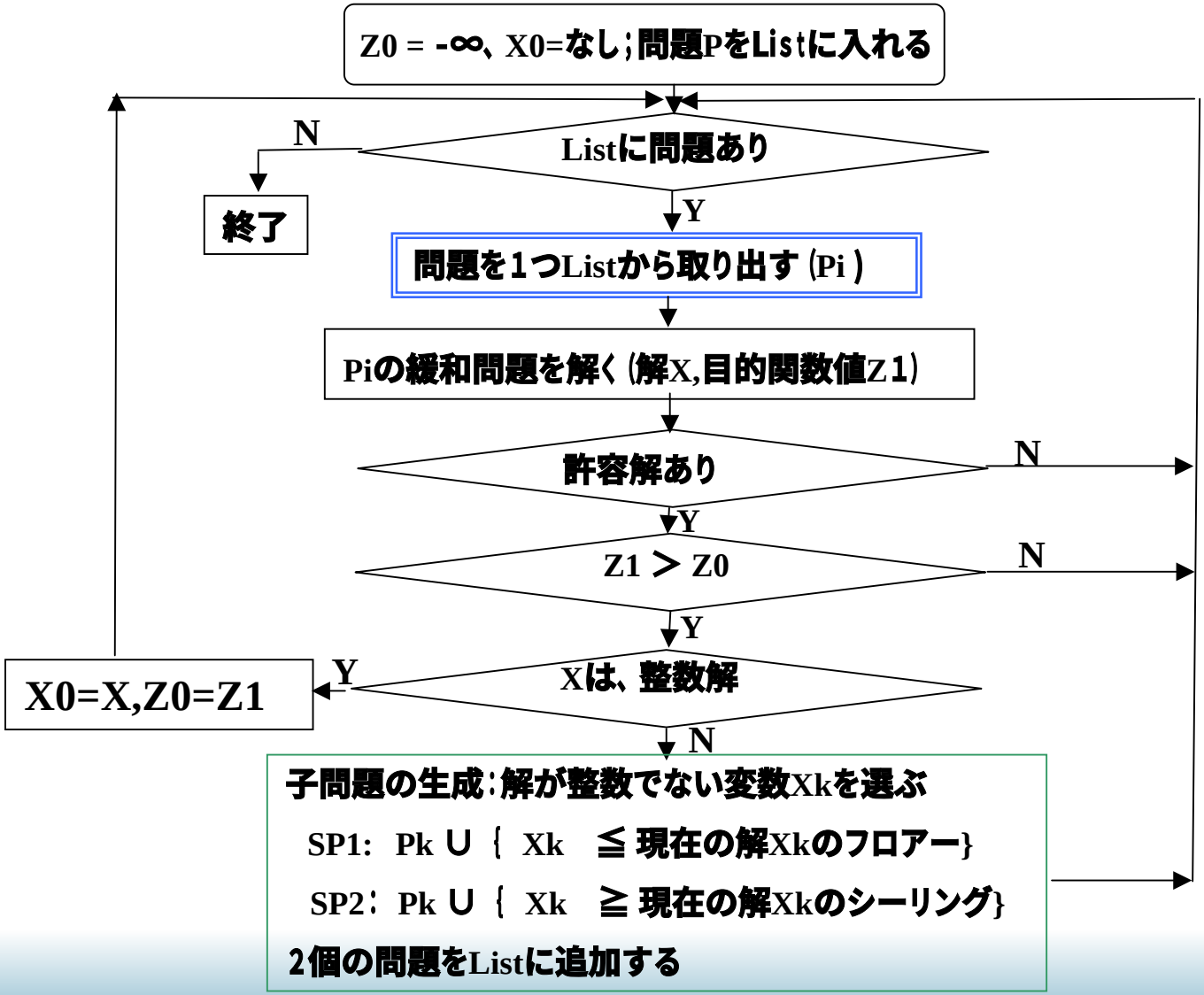
P_dw：追加制約条件：　$x_k \leqq \mathrm{floor}(d_k)$

2) Bound：子問題の緩和問題を解き、

「緩和問題の最適値　≦ 既に得られている整数解の最良値」

＝＞ この子問題は、最適解を生成しないのでスキップする。

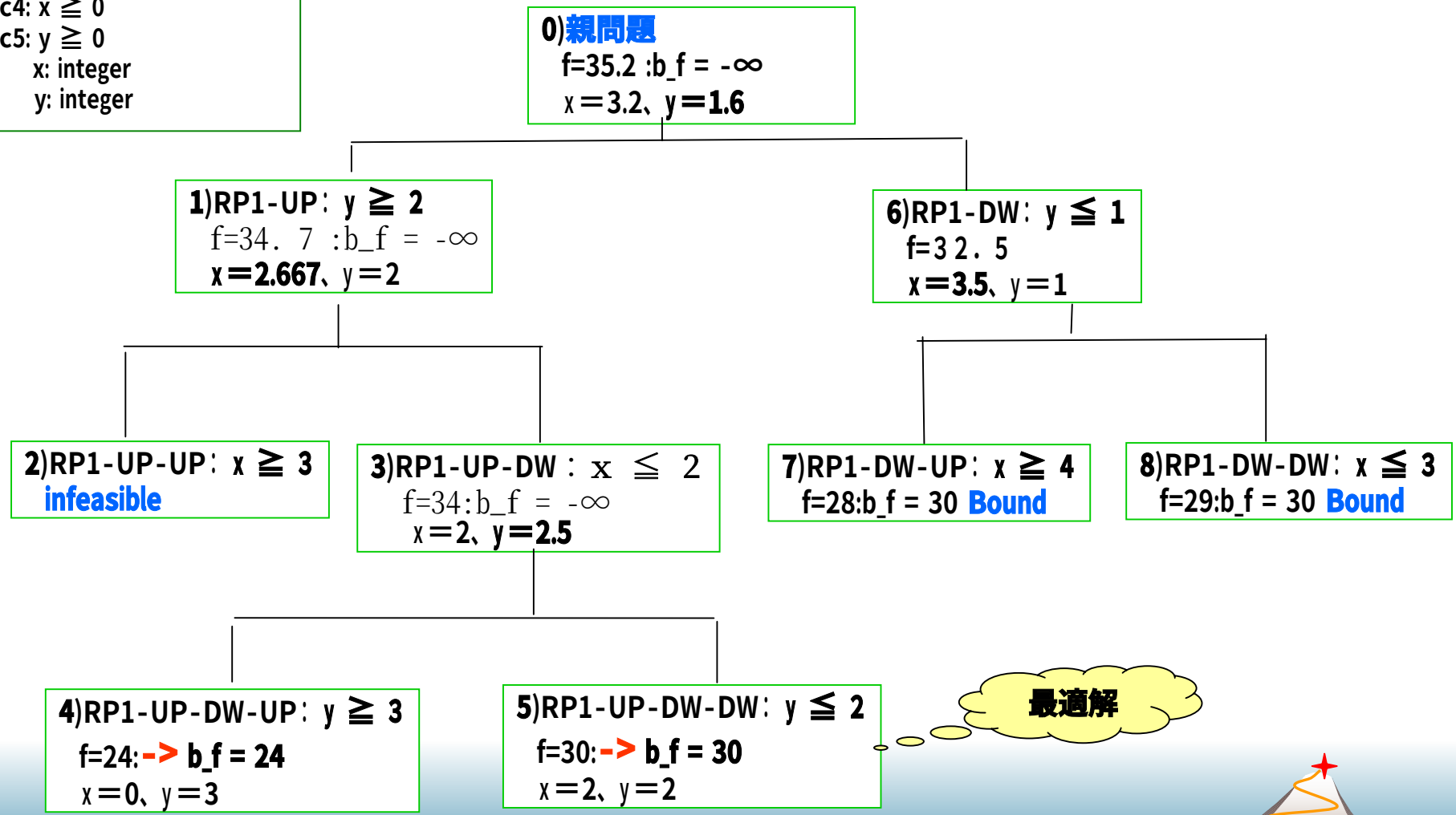
Branch and Bound法 (標準)

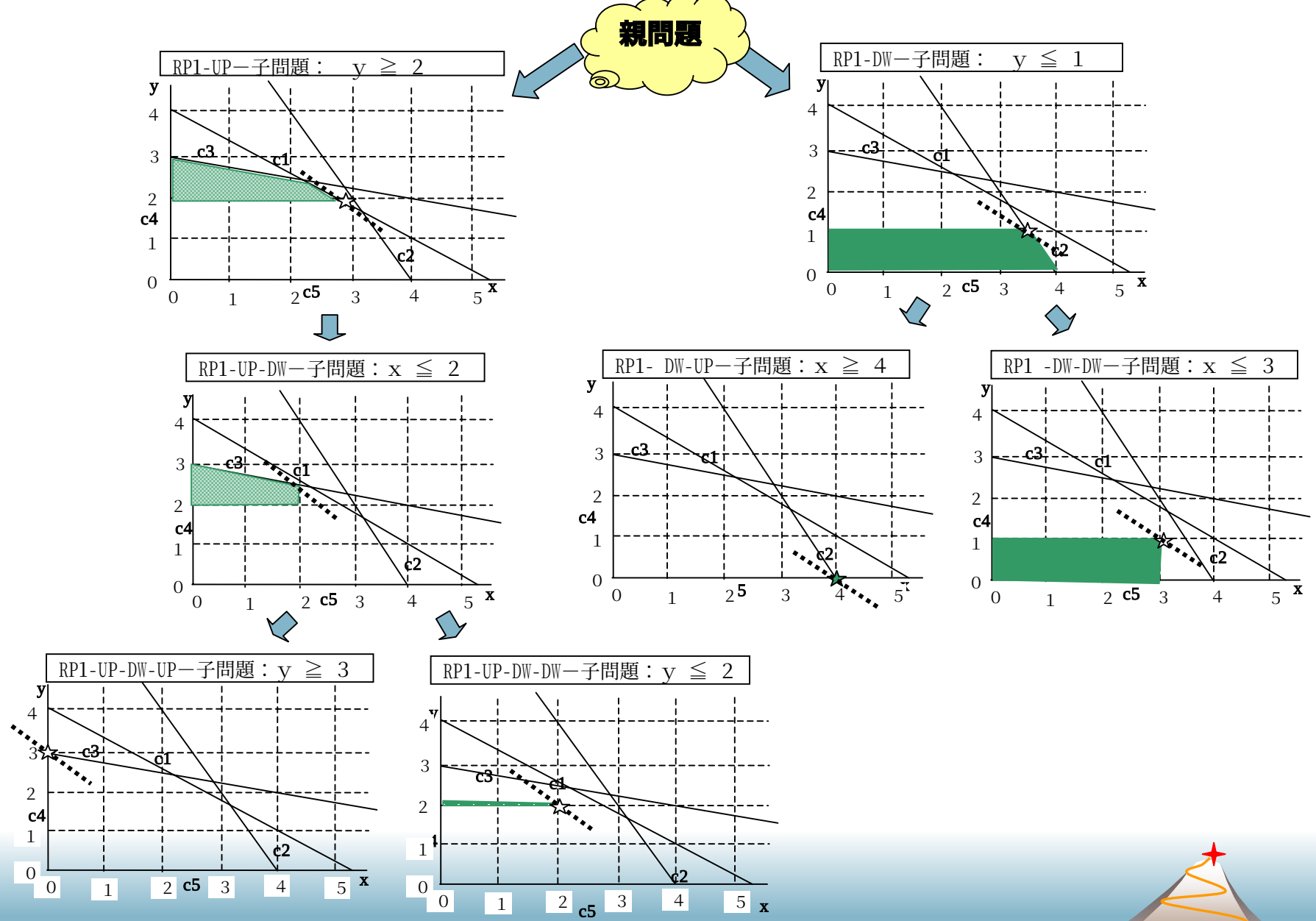


基本的なアルゴリズム

z ＝ 7x ＋ 8y －> 最大
Sub. to
c1: 3x + 4y <= 16
c2: 2x + 1y <= 8
c3: 1x + 4y <= 12
c4: x ≧ 0
c5: y ≧ 0
x: integer
y: integer

実行例





Branch&Bound法の課題

- 緩和問題の最適解と混合整数問題の最適解の関係
  - 両者の最適値の差が大きいと、boundingが起こりにくい
  - 両者の最適解が離れていると、Branchのステップが多くなる

  ＝＞ valid cutと実行時の付加機能の導入

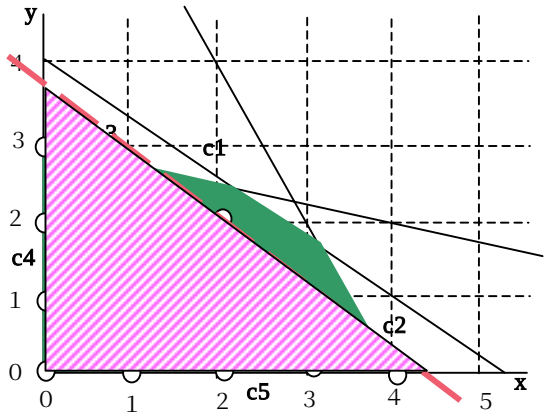
  ＝＞ Branch&Cut法の導入

- 良い実行可能解を早く得ると、boundingが起こりやすい。
  ＝＞primal heuristicsの導入

- Branchする変数の選択が重要
  ＝＞node selectorの導入

Valid inequality： Cut

緩和問題の領域を削るが、許容整数解は削除しない制約



Cover Cut：x1,x2,x3は、0-1変数

制約条件： 4x1 + 3x2 + 2x3 ≦ 5

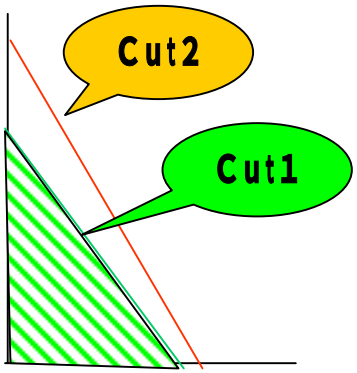
Minimal cover：変数の値を1にした時に、

制約条件が無効となる変数群の極小集合

S1={x1,x2} S2={x1,x3}

Cover cut： x1 + x2 ≦ 1

x1 + x3 ≦ 1



Cutの優劣：

2個の cut1 ($\Sigma a_i x_i \leqq b$) とcut2 ($\Sigma c_i x_i \leqq d$) に対して、cut1がcut2より優勢である

或る z >0 が存在して， $a_i \geqq z c_i$ 、 $b \leqq z d$ とできる。

優勢なcutの みを使用する

文献3)

CPLEXの生成可能なCut

- Clique Cuts
- Cover Cuts
- Disjunctive Cuts
- Flow Cover Cuts
- Flow Path Cuts
- Gomory Fractional Cuts
- GUB Cover Cuts
- Implied Bound Cuts
- Mixed Integer Rounding Cuts

Cutの例：x1,x2,x3は、非負の整数

MIR Cut:

元の制約： $x1 + 2.2x2 + 1.8x3 \leqq 2.9$

⇒ **MIR Cut：** $x1 + 2x2 + x3 \leqq 2$

Clique Cut:

元の制約： $x1 + x2 \leqq 1,$

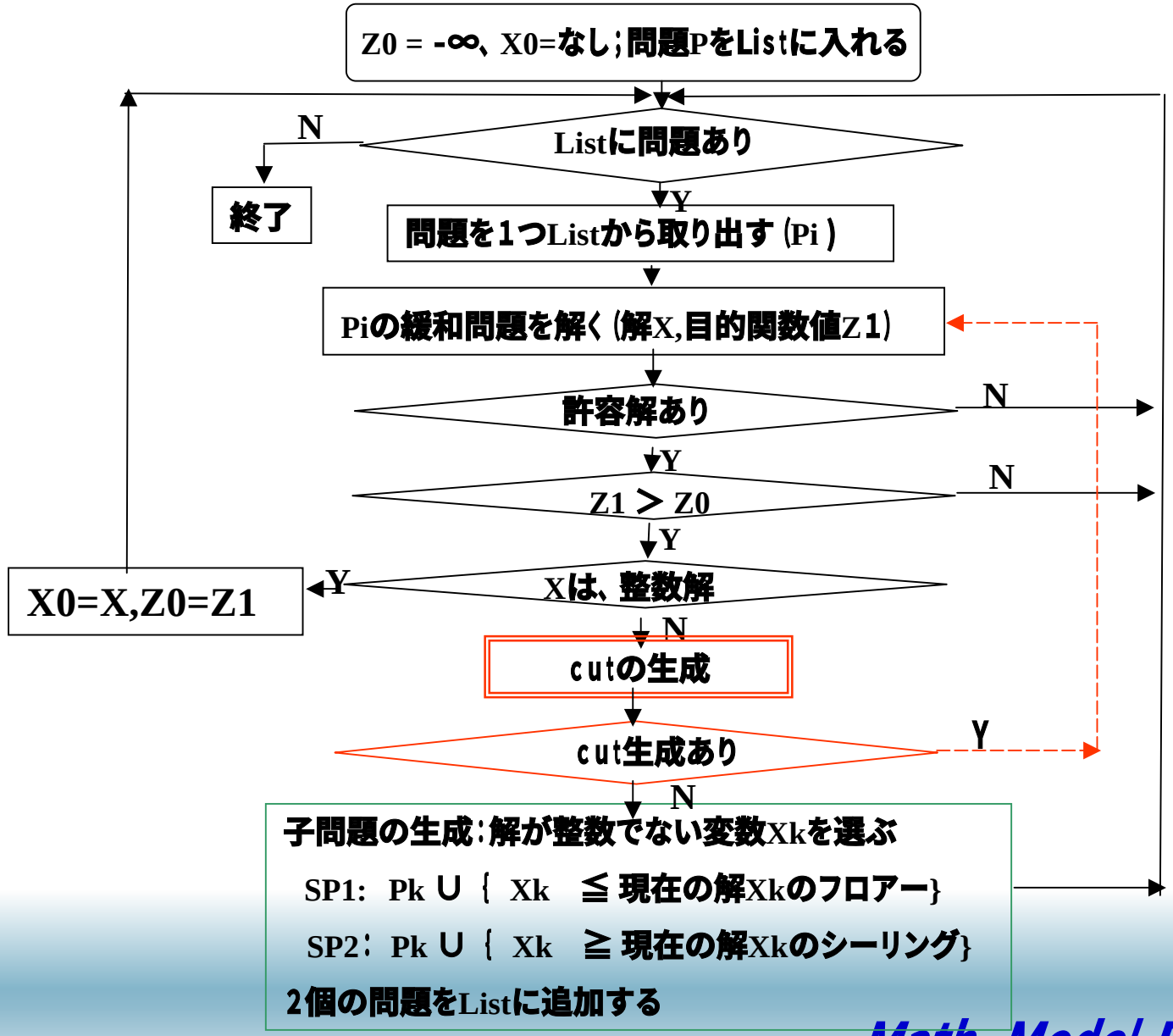
$x2 + x3 \leqq 1,$

$x1 + x3 \leqq 1$

⇒ **Clique Cut：** $x1 + x2 + x3 \leqq 1$

有効なCutを選択して適用

Branch and Cut法



Primal Heuristics

緩和問題の解が整数でない時に、近傍の整数解を探索する

Relaxation Induced Neighborhood Search (RINS)

基本的なアイデア：

緩和問題の解で整数となっている変数x_kの値$v(x_k)$と、

現在の最良解でのx_kの値が等しい変数の集合をQとする。

MIP_RINS：元の問題に、次の制約を付け加えて解く。

$x = v(x) \quad x \in Q$ ： **Qの変数の値は、現在の解に固定する**

[一定の探索内で解が得れなければ、近傍探索は失敗とする。]

処理が重いので、これを実行する頻度の設定

文献4)

Branc&Cutの実行例

0)親問題
f=35.2 :b_f = -∞
x＝3.2、y＝1.6

1)ヒューリスティックによる解の探索
b_f = 30
x＝2、y＝2

2）3個のCutの生成
f=33.5 b_f = 30
（非整数解）

3）2個のCutの生成
f=30: b_f = 30

スタート・ノード
で終了

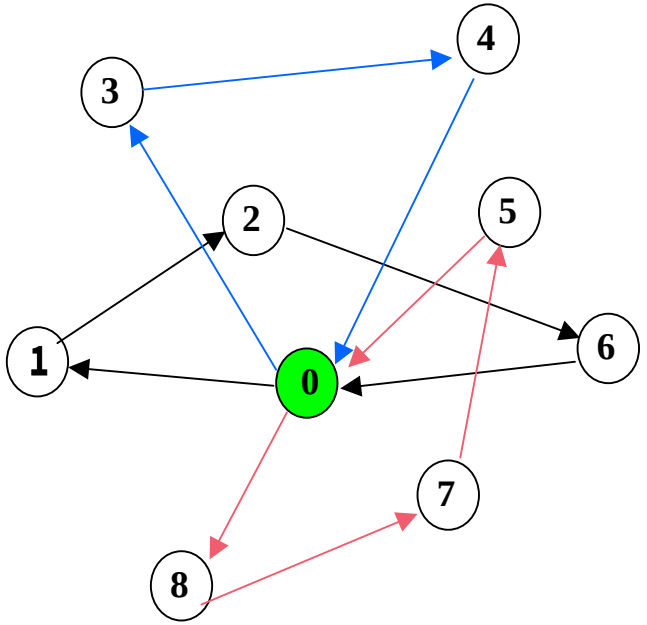
論理が複雑で全体の表現が困難な問題

複数のトラックによる配送問題

- トラック： 容量C
- 倉庫：場所P_0
- 配送先：場所P_k、購入量 Q_k

  配送時間帯 T_k、・・・

- 目的：最小のトラック数で配送



集合分割問題によるアプローチ：ベーシック

子問題：

制約を満たす配送ルートの生成

（ヒューリスティックス等で多数生成）

- p1：0->1->2->6->0
- p2：0->1->2->4->0
- p3：0->1->8->0
- p4：0->2->5->6->0
- p5：0->2->3->5->0

：

- pn-1：0->8->2->5->0
- pn：0->2->7->5->4->0

個別のルート生成

親問題：集合分割問題

$$\Sigma z_k \rightarrow \text{最小}$$

sub to

店	P1	P2	P3	P4	・・・	P_{n-1}	Pn			
1	1	1	0	0		0	0		Z1	1
2	1	1	1	1		1	1		Z2	1
3	0	0	0	1		0	0		Z3	1
4	0	1	0	0		0	1		Z4	1
5	0	0	1	1	・・・	1	1	=	：	1
6	1	0	1	0		0	0			1
7	0	0	0	0		0	1		：	1
8	0	0	0	0		1	0		Zn	1

$$z_k \in \{0, 1\}$$

Column Generationアプローチ：一括パターン生成

Cutting Stock問題：

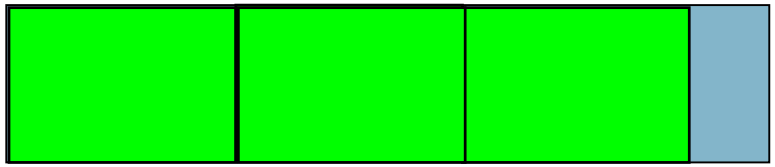
次の表の板を,幅Wの原板から製造したい。

目的：原板の枚数を最小にする切り出しパターン？

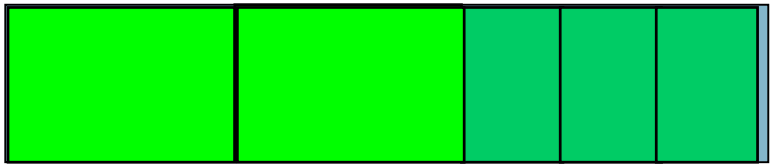
幅	必要量
W1	d1
W2	d2
W3	d3
W4	d4

切り出しパターン： P = (P1,P2,P3,P4)

P0 = (3,0,0,0)



P1= (2,3,0,0)



有効なパターンをどのように生成するか？

文献1)

Column Generationアプローチ：列ベクトルが不完全

初期パターン群の生成：PS (0)

[同一サイズのみを切り出すパターン]

t＝0

パターン y を列に追加する

$PS(t+1) = PS(t) \cup \{y\}$

t=t+1

Y

f ＜ 0

N

緩和親問題：cplex1

ΣZ_k －＞最小

sub to

cst[i]： $\Sigma_k P_{jk} Z_k \geqq d_j \quad k \in PS(t)$

$Z_k \geqq 0$ ：整数条件を緩和

Pricing 問題：cplex2

(列：パターンの生成)

$f = 1 - \Sigma \pi_j y_j$ －＞最小

sub to

$\Sigma W_j y_j \leqq W$

$y_j \geqq 0$ ：整数

Shadow priceの取得

π_i = cplex1.getDual(cst[i])

未知のパターン（列ベクトル）yで
Reduced Costが最小を求める

親問題:緩和なし
(不完全)

文献5)

2日間集荷ルートの決定問題

一台のトラックにより、2日間の集荷を行うが、毎日の農家と1日のみの農家がある。

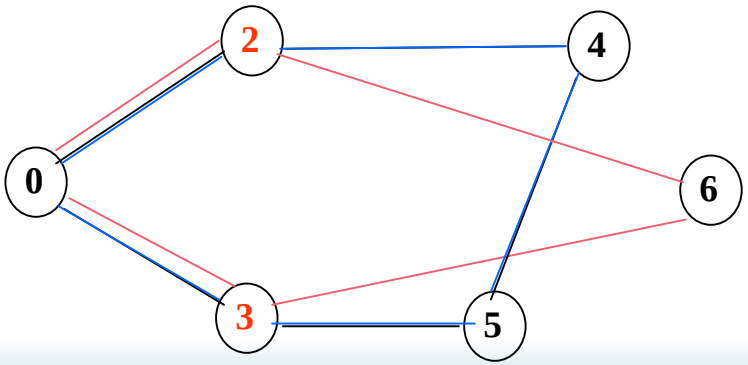
トラック：容量 C、移動コスト C_{ij}

農家：全体の集合N、毎日：N1、1日：N2

集荷量K_i

X_{ij}^k：　k日目に (i, j) 間を運行

Y_i^k：　k日目に農家 i を集荷



$$Min \sum_k \sum_{i<j} C_{ij} * X_{ij}^k$$

$$\sum_i K_i Y_i^k \le C \qquad k=1,2$$

$$Y_i^1 + Y_i^2 = 1 \qquad i \in N2$$

$$\sum_{j:j>i} X_{ij}^k + \sum_{j:j<i} X_{ji}^k = 2 \qquad k=1,2, i \in N1$$

$$\sum_{j:j>i} X_{ij}^k + \sum_{j:j<i} X_{ji}^k = 2Y_i^k \qquad k=1,2, i \in N2$$

$$X_{ij}^k \le Y_i^k \qquad k=1,2,\ i,j \in N2, i<j$$

$$X_{ji}^k \le Y_i^k \qquad k=1,2,\ i \in N2, j \in N, j<i$$

$$X_{ij}^k = \{0,1\} \qquad k=1,2,\ i,j \in N\ i<j$$

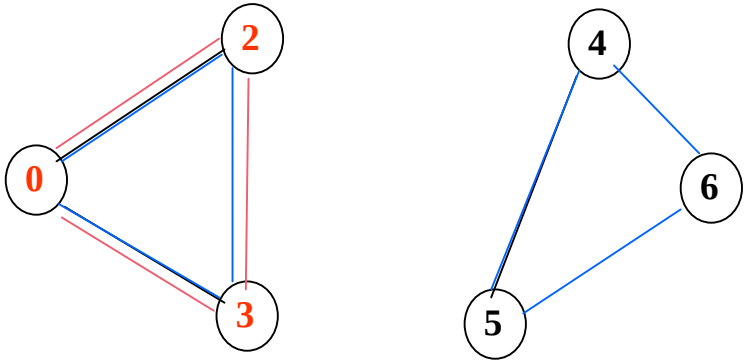
$$Y_i^k = \{0,1\} \qquad k=1,2,\ i \in N$$

文献2)

制約追加アプローチ：制約式が不完全

問題点：

流れの均衡式だけでは、サブツアーが発生する。



サブツアーの禁止制約：ノード0を含まない任意のNの部分集合 S に対して、

$$\sum_{ij:i\in S,\, j\in S,\, j>i} X_{ij}^{k} + \sum_{ji:i\in S,\, j\in S,\, j<i} X_{ji}^{k} \le |S| - 1$$

制約が多すぎる！

初期設定：

サブツアー禁止制約リスト：PS = φ

制約緩和 問題：

前頁のLPに、PSの制約を付加して解く。

サブツアーのチェック：

サブツアーがあれば、禁止制約リストPSに追加

サブツアーあり

Y

N

終了

ハイブリッド・アプローチ (CP＋LP)

制約プログラミング (CP):

長所: 複雑な論理的制約条件を表現可能

短所: 大域的な評価 (最適解、要素の感度) が困難

線形計画法 (LP):

長所: 大域的な評価 (最適解、要素の感度) が可能

短所: 利用可能な制約条件が限定

CPとLPの長所を利用

緩和問題:

CPの制約条件を緩和した**LP**問題の利用

ハイブリッド・アプローチ:

1) **CP**により厳密に問題を定式化する

2) **LP**緩和問題を解いて、上限(下限) 値、感度情報を得る

3) 上限 (下限) 値、感度情報により、**CP**を高速に解く

ハイブリッド・アプローチのイメージ

制約プログラミング (CP):

目的: $F = aX_1 + bX_2 + \cdots$ -> 最大

変数: X_1、X_2、・・・・

条件:

C1: $cX_1 + dX_2 + \cdots \leqq b_1$

C2: $\{X_1, X_2, X_3, X_4\}$ の非ゼロは2個以内

線形計画 (LP):

目的: $G = aX_1 + bX_2 + \cdots$ -> 最大

変数: X_1、X_2、・・・・

条件:

C1: $cX_1 + dX_2 + \cdots \leqq b_1$

C2: 削除

CPの解探索:

現在までの最も良い許容解 (F_0)

緩和情報を利用して、
一部の変数を設定 (部分解)

現在の部分解の評価:
$G_0 \leqq F_0$ => バックトラック

探索情報:
1) 厳しい制約を先に扱う
2) 初期値としてY_0近傍を利用

LPの解探索:

(CPで設定された値は、固定)

1) 目的関数: G_0
2) 最適化を達成するXの値 (Y_0)
3) 制約条件のシャドープライス

CPLEXによる計算の体験

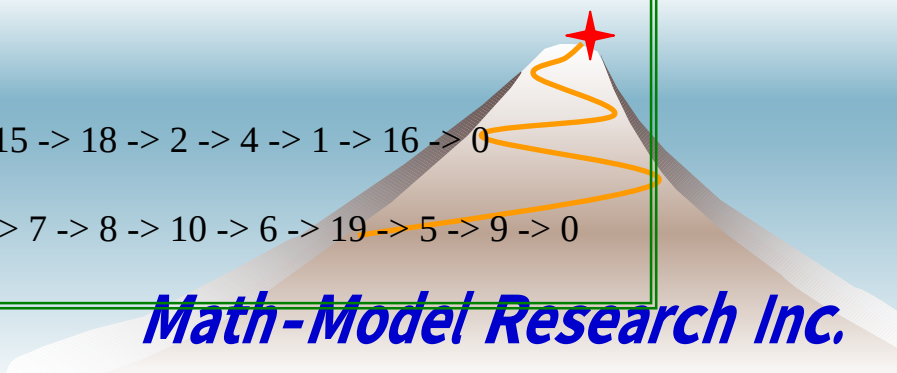
1) 2日間集荷ルートの決定問題
　　(制約条件の動的追加)

2) 切り出し問題
　　(Column generationの利用)

3) 容量制約のある配送問題
　　(ヒューリスティックスによるルート生成と集合被覆問題)

2日間集荷ルートの決定問題：N=21、N1=10、 capacity:80

```
>Release¥Milk_Collection.exe  1
******** iter: 1
Optimal value: 12160
Day:0 size: 15
 Base_Node:0 -> 7 -> 8 -> 9 -> 0
 Base_Node:1 -> 4 -> 17 -> 1
 Base_Node:2 -> 15 -> 12 -> 3 -> 18 -> 2
 Base_Node:5 -> 6 -> 19 -> 5
Day:1 size: 16
 Base_Node:0 -> 9 -> 5 -> 6 -> 10 -> 8 -> 20 -> 7 -> 11 -> 3 -> 14 -> 2 -> 13 -> 4 -> 1 -> 16 -> 0
******** iter: 2
Optimal value: 12189
Day:0 size: 17
 Base_Node:0 -> 1 -> 16 -> 5 -> 19 -> 6 -> 9 -> 0
 Base_Node:2 -> 4 -> 13 -> 15 -> 12 -> 3 -> 18 -> 2
 Base_Node:7 -> 8 -> 20 -> 7
Day:1 size: 14
 Base_Node:0 -> 1 -> 17 -> 4 -> 2 -> 14 -> 3 -> 11 -> 7 -> 8 -> 10 -> 6 -> 5 -> 9 -> 0
******** iter: 3
Optimal value: 12293
Day:0 size: 15
 Base_Node:0 -> 9 -> 5 -> 6 -> 8 -> 20 -> 7 -> 12 -> 3 -> 15 -> 18 -> 2 -> 4 -> 1 -> 16 -> 0
Day:1 size: 16
 Base_Node:0 -> 1 -> 17 -> 4 -> 13 -> 2 -> 14 -> 3 -> 11 -> 7 -> 8 -> 10 -> 6 -> 19 -> 5 -> 9 -> 0
```



切り出し問題： Column Generation: ILOG Sample Program(cutstock.cpp)を実行

原板:110

幅	必要
20	48
45	35
50	24
55	10
75	8

幅	P0
20	5
45	0
50	0
55	0
75	0

幅	P1
20	0
45	2
50	0
55	0
75	0

幅	P2
20	0
45	0
50	2
55	0
75	0

幅	P3
20	0
45	0
50	0
55	2
75	0

幅	P4
20	0
45	0
50	0
55	0
75	1

初期パターン

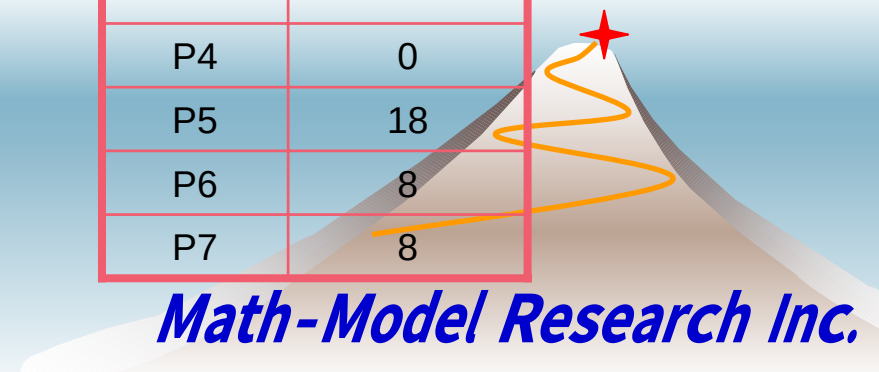
幅	P5
20	1
45	2
50	0
55	0
75	0

幅	P6
20	1
45	0
50	0
55	0
75	1

幅	P7
20	3
45	0
50	1
55	0
75	0

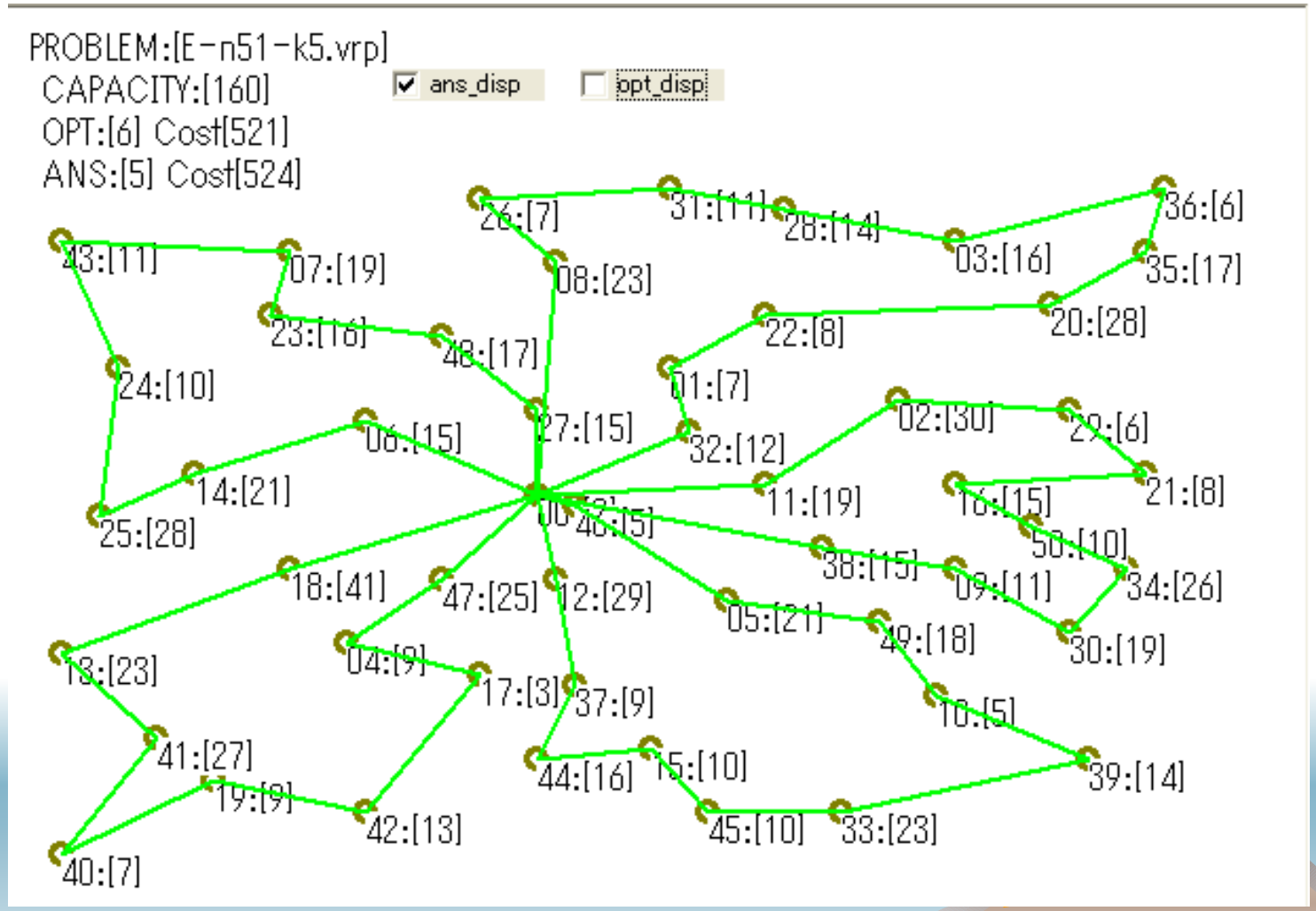
パターン	最適製造量
P0	0
P1	0
P2	8
P3	5
P4	0
P5	18
P6	8
P7	8

親問題



Math-Model Research Inc.

容量制約のある配送問題：ヒューリスティックスによる生成ルート 642



Math-Model Research Inc.

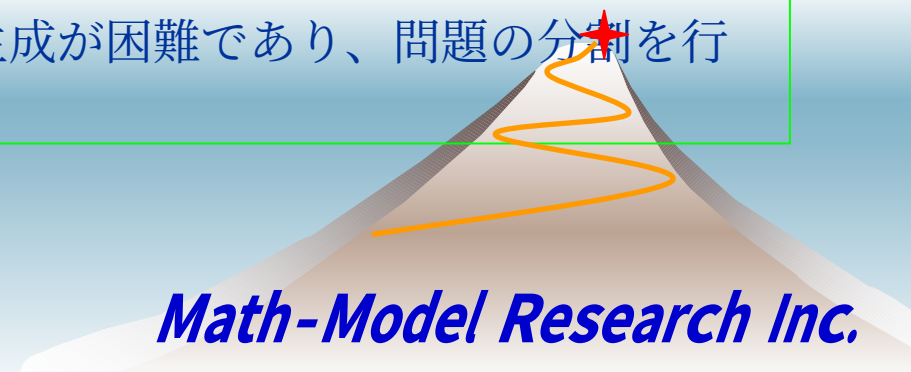
混合計画法では困難な問題

・混合整数計画法は、基本的には、
　１）緩和問題の上／下限値による探索ノードのPruning
　２）cutによる整数解の露出
をベースにしている。
　例えば、制約追加型のアプローチでは、緩和問題の最適値が最良解より良くなるため、あまり可能性がない。
　しかし、アサイクリックなネットワークの場合には、サブツアーが存在しないため、ネットワークの均衡モデルで解を得ることができる。

・良いCutの存在が分かっている問題をベースにすることも有効である。

・ネットワーク理論のように、パフォーマンスが良いアルゴリズムによる近似も試みる価値がある。

・整数変数の数が多いときは、解の生成が困難であり、問題の分割を行う必要がある。



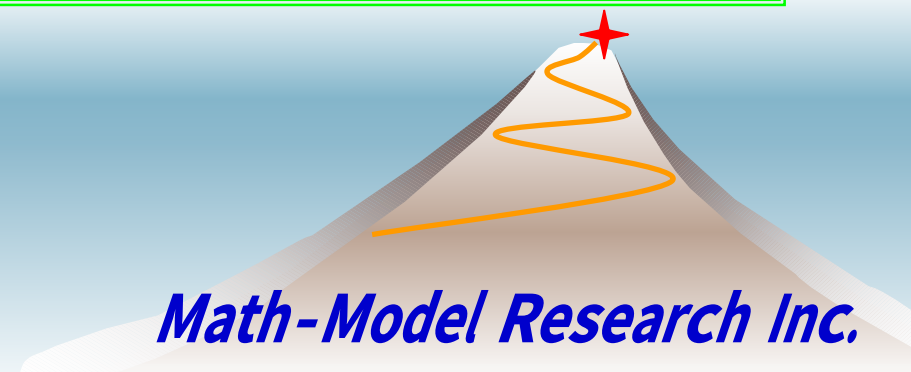
まとめ

混合整数計画法による問題の定式化と解法を理解し、応用できるように、CPLEX（ILOG社）をベースに解説を試みましたが、理論と実践ともに中途半端になってしまったと危惧しております。

私自身は、昔、MIPを使った研究も行っておりましたが、ここ１０年ほどは、制約プログラミングを中心にした開発をおこなっており、CPLEXも上記の観点から使用しております。

ILOG社の最適化パッケージは、開発チームの学術的な研究成果をベース開発されており、SCMなどで盛んに使用されています。

日本でも、これらの方法論を実務に適用するスペシャリストの出現が望まれます。



参考文献

1) ILOG CPLEX 9.1 User's Manual, ILOG
2) H.P. Williams: Model Building in Mathematical Programming,2005,Wiely
3) L.Wolsey: Integer Programming, 1998, Wiely
4) Y.Pochet: Production Planning by Mixed Integer Programming,2006,Springer
5) G.Desaulniers,etal: Column Generation,2005,Springer