ＯＲ学会春季研究発表会チュートリアル
【数理計画法（RAMP）研究部会企画】

はじめよう整数計画法

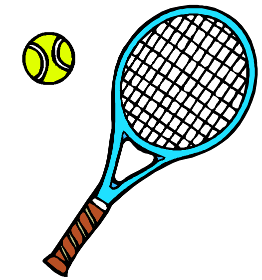
藤江哲也
兵庫県立大学大学院経営研究科

2014年3月7日（金） 大阪大学豊中キャンパス

OR学会誌2012年4月号

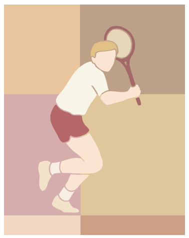
特集 はじめよう整数計画

本チュートリアルの目指すところ

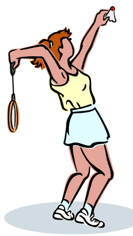


- はじめてコース　初めてラケットを持つ方のコースです。簡単なルールやマナーなども覚えます。
- 初級コース　基本からしっかりと学びたいという方のコースです。楽しくラリーが続けられることを目指します。

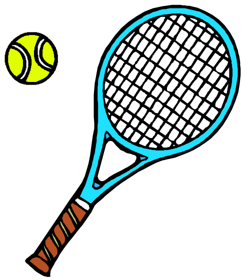
- 初中級コース　ラリーを少しつなげられる方のコースです。各ショットのレベルアップを目指します。



- 中級コース　ストローク･ボレー･サーブ･スマッシュがある程度コントロールできるという方のコースです。
- 上級コース　ダブルスにおける攻守を理解し、一通り実践できる方のコースです。



本チュートリアルの目指すところ



- **はじめてコース**
  初めてラケットを持つ方のコースです。
  簡単なルールやマナーなども覚えます。
- **初級コース**
  基本からしっかりと学びたいという方のコースです。
  楽しくラリーが続けられることを目指します。

1. 整数計画とは何か
2. 今なぜ整数計画なのか
3. 整数計画をはじめよう
   - 道具をそろえよう
   - 道具を使ってみよう

線形計画問題(LP: Linear Programming)

- テーブルとチェアを製造販売
- 1個当たり所要時間、利益、および製造工程の使用可能時間

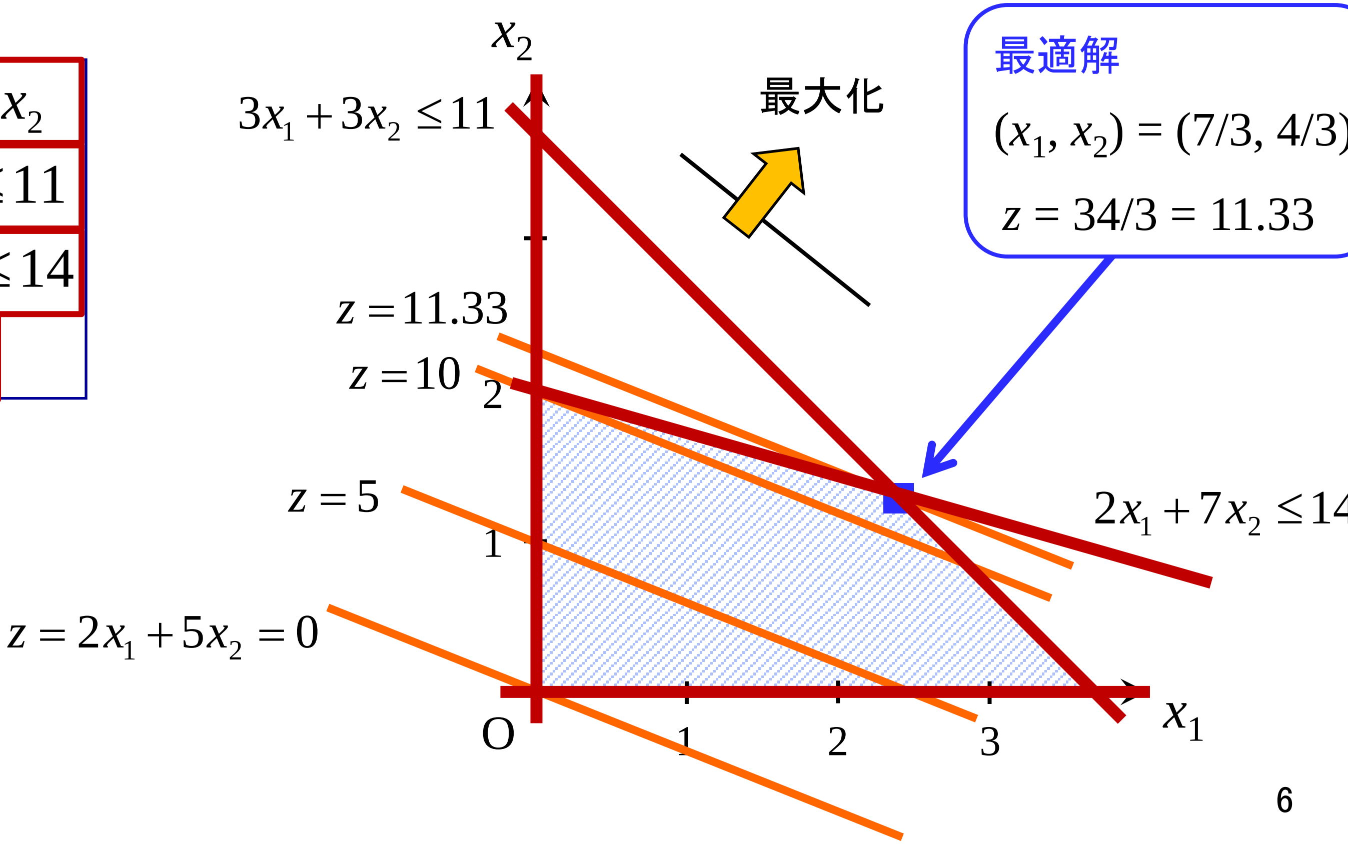
	テーブル	チェア	使用可能時間
工程1	3時間	3時間	11時間
工程2	2時間	7時間	14時間
利益	2千円	5千円	

- 利益を最大にするテーブルとチェアの製造数は?

線形計画問題（LP: Linear Programming）

	テーブル x_1	チェア x_2	使用可能時間
工程1	3時間	3時間	11時間
工程2	2時間	7時間	14時間
利益	2千円	5千円	

最大化	$z = 2x_1 + 5x_2$
条件	$3x_1 + 3x_2 \leq 11$
	$2x_1 + 7x_2 \leq 14$
	$x_1, x_2 \geq 0$



最適解

$(x_1, x_2) = (7/3, 4/3)$

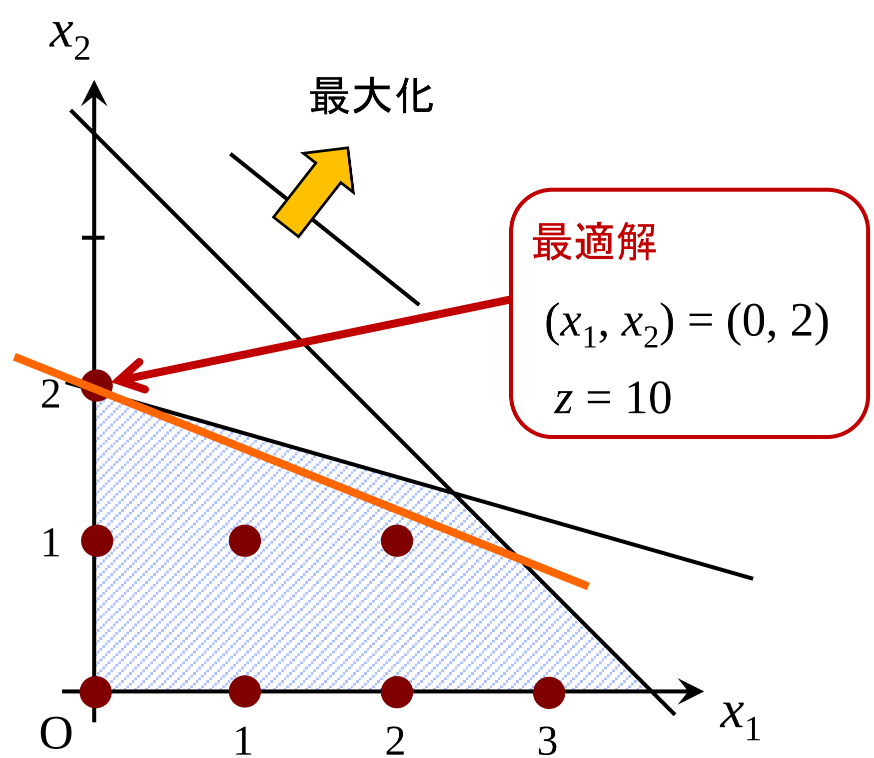
$z = 34/3 = 11.33$

整数計画問題，整数線形計画問題

IP (**I**nteger **P**rogramming)

ILP (**I**nteger **L**inear **P**rogramming)

最大化	$z = 2x_1 + 5x_2$
条件	$3x_1 + 3x_2 \leq 11$
	$2x_1 + 7x_2 \leq 14$
	$x_1, x_2 \geq 0$
	x_1, x_2 : 整数

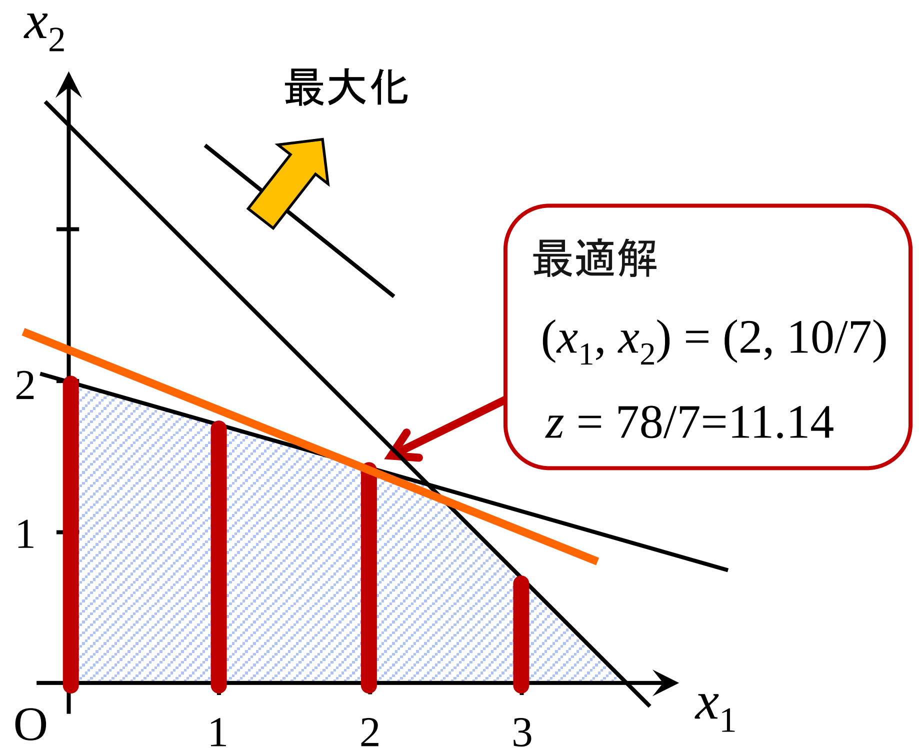


「混合」整数計画問題

MIP (Mixed Integer Programming)

整数条件:一部またはすべての変数

最大化	$z = 4x_1 + 5x_2$
条件	$2x_1 + 2x_2 \le 7$
	$3x_1 + 5x_2 \le 14$
	$x_1, x_2 \ge 0$
	x_1 : 整数



バイナリ変数（0–1変数）

『 $x_j = 0$ または 1 』

◇バイナリ変数も「線形不等式＋整数変数」で記述できる

「$x_j = 0$ または 1」 $\Leftrightarrow$ 「$0 \le x_j \le 1$, x_j：整数」

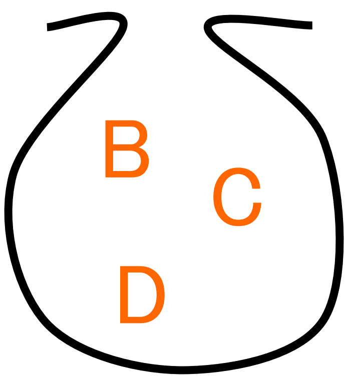
◇しかし、バイナリ変数は整数変数と区別されるのが一般的

- 高い表現能力
- 0–1の特性に基づくアルゴリズム開発

バイナリ変数の例：ナップサック問題

	A ポテトチップス	B チョコレート	C マシュマロ	D アメ	E ガム	F せんべい
満足度	5点	7点	4点	2点	3点	8点
値段	100円	130円	80円	50円	70円	110円

- おかしの合計金額は300円まで
- 各種類1つまで
- 満足度の合計が最大とするには？



ナップサックの容量
$c = 300$

合計金額 = 130 + 80 + 50 = 260 <= 300

満足度の合計 = 7 + 4 + 2 = 13

バイナリ変数の例:ナップサック問題

	A	B	C	D	E	F
満足度	5点	7点	4点	2点	3点	8点
値段	100円	130円	80円	50円	70円	110円

$$x_1 = \begin{cases} 1 & \text{A を選ぶとき} \\ 0 & \text{A を選ばないとき} \end{cases}$$

などとすると

$$\text{最大化} \quad 5x_1 + 7x_2 + 4x_3 + 2x_4 + 3x_5 + 8x_6$$

$$\text{条件} \quad 100x_1 + 130x_2 + 80x_3 + 50x_4 + 70x_5 + 110x_6 \leq 300$$

$$x_1, x_2, \cdots, x_6 = 0 \text{ または } 1$$

x_1=1 のとき 100

x_1=0 のとき 0

バイナリ変数の例：部分和問題

	A ポテトチップス	B チョコレート	C マシュマロ	D アメ	E ガム	F せんべい
値段	100円	130円	80円	50円	70円	110円

合計金額300円以内で300円に最も近い組み合わせは？

$$\text{最大化} \quad 100x_1 + 130x_2 + 80x_3 + 50x_4 + 70x_5 + 110x_6$$

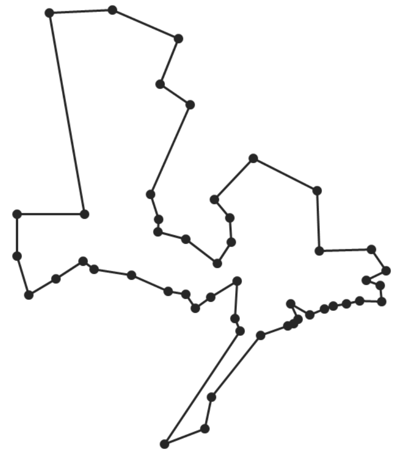
$$\text{条件} \quad 100x_1 + 130x_2 + 80x_3 + 50x_4 + 70x_5 + 110x_6 \leq 300$$

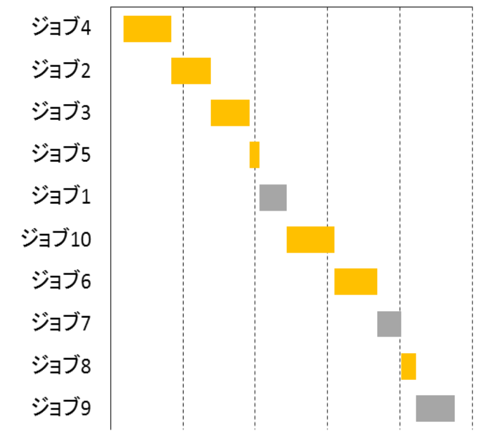
$$x_1, x_2, \cdots, x_6 = 0 \text{ または } 1$$

バイナリ変数(0-1変数)を用いた表現

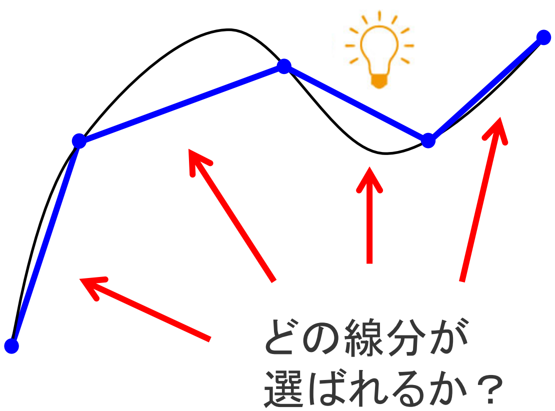
◇ 組合せ最適化問題

- 巡回セールスマン問題
- 集合分割問題
- 集合被覆問題
- スケジューリング問題
- 施設配置問題
- ・・・



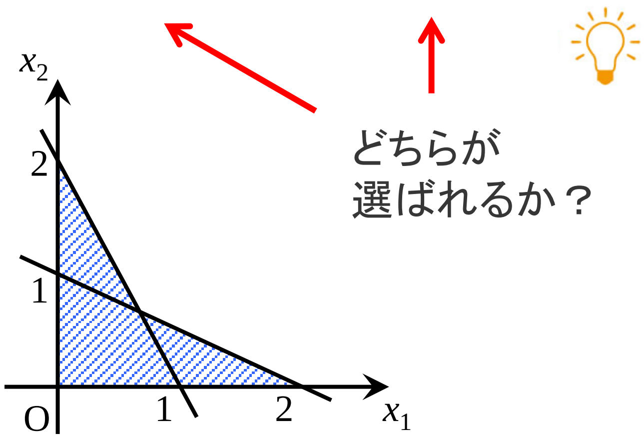


◇ 非線形関数の線形近似



◇ 離接(disjunctive)制約

$x_1 + 2x_2 \leq 2$ または $2x_1 + x_2 \leq 2$



バイナリ変数(0–1変数)を用いた表現

◇ 整数変数

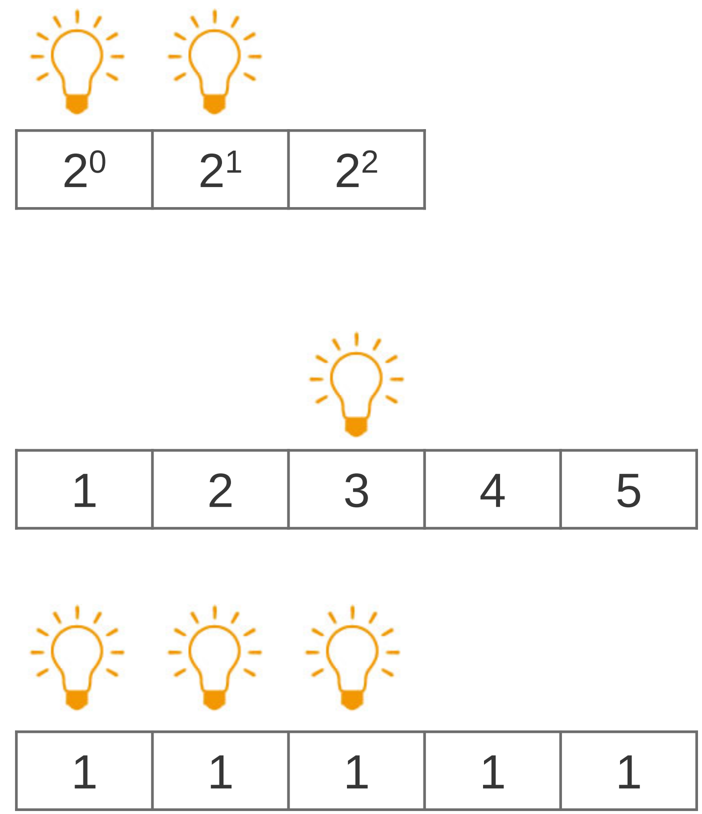
$0 \leq x \leq 5 \quad x$：整数

$$x = y_1 + 2y_2 + 2^2 y_3$$
$y_1, y_2, y_3 =$ 0 または 1

$$x = y_1 + 2y_2 + 3y_3 + 4y_4 + 5y_5$$
$$y_1 + y_2 + y_3 + y_4 + y_5 \leq 1$$
$y_1, y_2, y_3, y_4, y_5 =$ 0 または 1

$$x = y_1 + y_2 + y_3 + y_4 + y_5$$
$y_1, y_2, y_3, y_4, y_5 =$ 0 または 1

$x = 3$ の場合



線形計画LP と 整数計画MIP

	LP	MIP
応用例	生産スケジューリング 配送 施設配置 DEA 交通 ネットワーク流	切出し 詰込み 金融 シフトスケジューリング 時間割作成 選挙区割
応用範囲	広い	「整数条件」でさらに広がる！！
知名度	高い	低い？

「はじめよう」の理由

線形計画LP と 整数計画MIP

	LP	MIP
代表的解法	単体法，内点法	切除平面法，分枝限定法 分枝カット法
	1947年 単体法（Dantzig）	1957～60年 分枝限定法（Markowitz–Manne, Eastman, Land–Doig） 切除平面法（Gomory）
解きやすさ（理論的）	easy (P)	hard (NP)
（実際的）	大規模問題も解ける	解ける問題規模が拡大中

「はじめよう」
の理由

線形計画LP

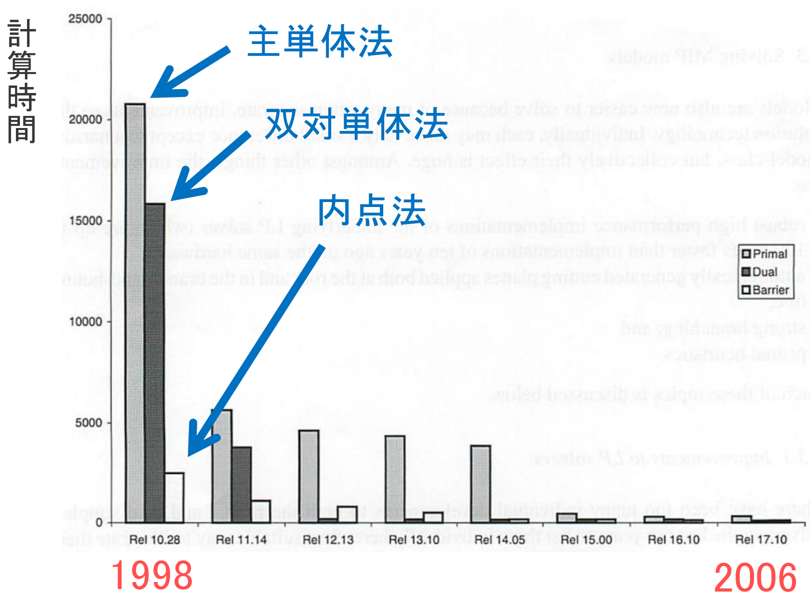
CPLEX LP 1988 → 2004（16年）

	Improvement factor
Algorithmic improvement (machine independent)	
Best of barrier, primal simplex, and dual simplex:	3300×
Machine improvement:	1600×
Total improvement (3300 · 2000):	5,280,000×

アルゴリズム　3300倍
計算機　1600倍
トータル　528万倍

R. E. Bixby, ``A Brief History of Linear and Mixed-Integer Programming Computation,'' In: Grötschel, M. (ed.) Optimization Stories, pp.107-121 (2012)

Xpress-MP 1998 → 2006（8年）



制約式　変数

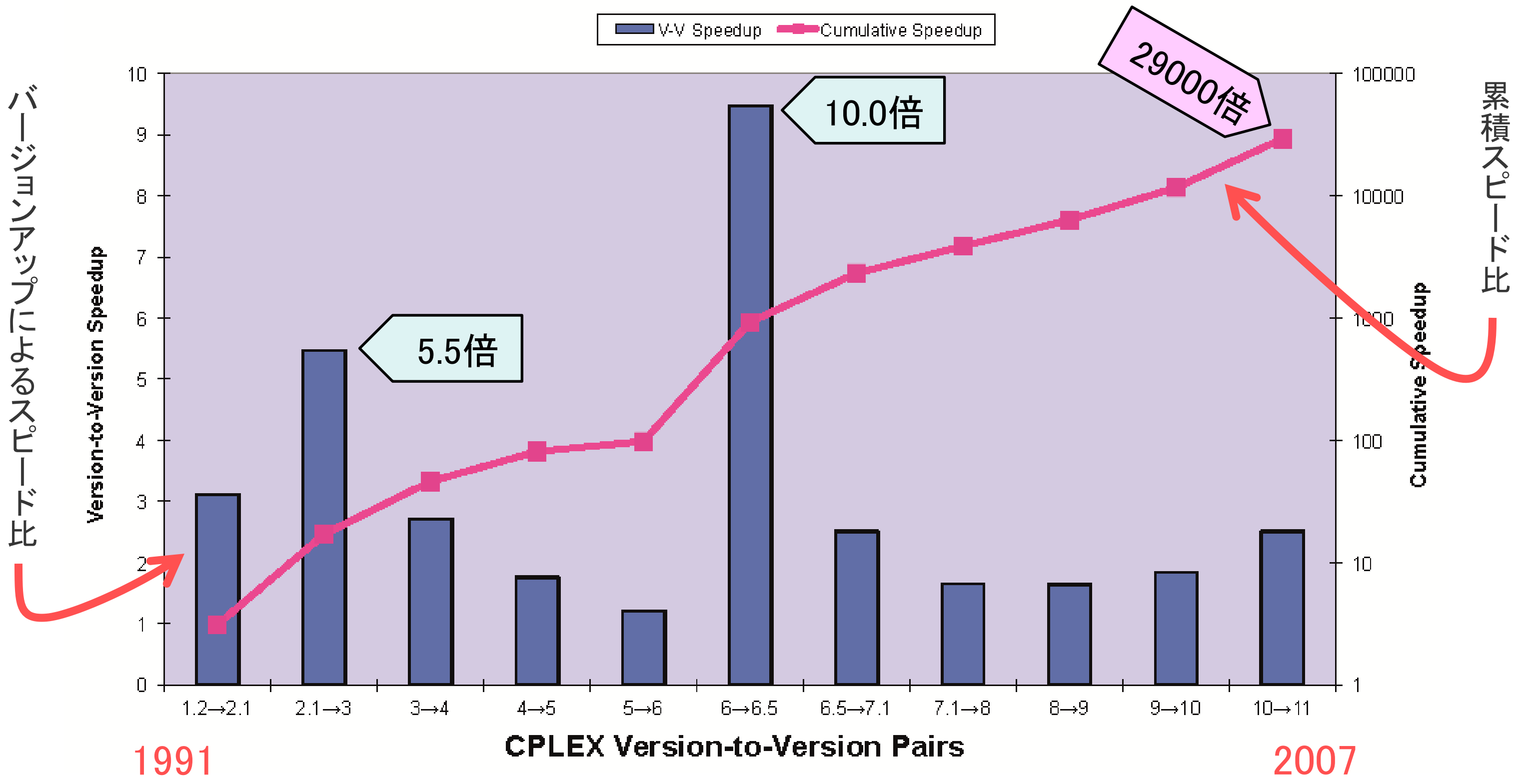
Models	Rows	Columns	Nonzeros
Artur	27908	8111	190570
Chinese	36521	81814	754366
Energy	31941	38591	106778
Ken-18	105128	154699	512719
Prele	108286	100560	638628
Watson_1	201156	383927	1053564

R. Ashford, ``Mixed Integer Programming: A Historical Perspective with Xpress-MP,'' *Annals of Operations Research* 149, 5-17 (2007)

整数計画MIP

CPLEX MIP 1991 → 2007（16年）

問題数1892
タイムリミット30000秒＝8.3時間
少なくとも一方で解けた問題を比較
速度比の幾何平均

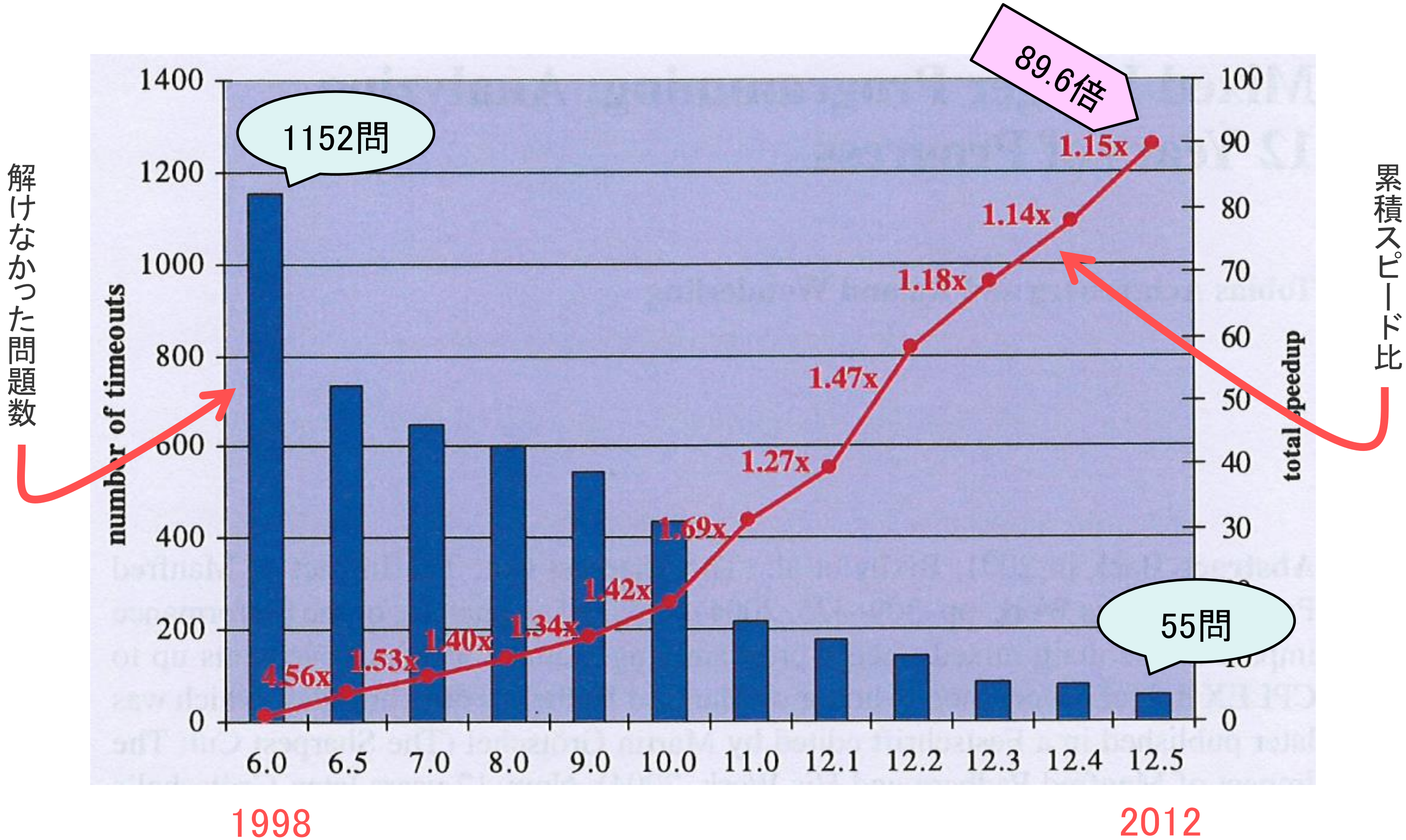


R. E. Bixby, ``A Brief History of Linear and Mixed-Integer Programming Computation,''
In: Grötschel, M. (ed.) Optimization Stories, pp.107-121 (2012)

整数計画MIP

CPLEX MIP 1998 → 2012（14年）

問題数1753
タイムリミット10000秒＝2.8時間
解けなかった問題数
解けた問題の計算時間の幾何平均



T. Achterberg and R. Wunderling, ``Mixed Integer Programming: Analyzing 12 Years of Progress,'' In: M. Jünger and G. Reinelt (eds.) Facets of Combinatorial Optimization, pp.449-481 (2013)

整数計画MIP

MIPLIB2010 (http://miplib.zib.de/) の一部

- Easy : 商用ソルバで1時間以内に解ける
- Hard : 解かれてはいるが，時間や手間がかかる
- Open : 未解決

Status	Name	Sets	C	制約式	変数	NZs	整数	バイナリ	連続	Objective
	bley_xl1	B	BP	175620	5831	869391		5831		190
	blp-ar98	T	MBP	1128	16021	200601		15806	215	6205.21
	blp-ic97	C	MBP	923	9845	118149		9753	92	4025.02
	bnatt350	BPR	BP	4923	3150	19061		3150		0
	bnatt400	CR	BP	5614	3600	21698		3600		1
	buildingenergy	C	MIP	277594	154978	788969	26287		128691	33283.9
	cdma	CU	MBP	9095	7891	168227		4235	3656	?
	circ10-3	CR	BP	42620	2700	307320		2700		?
	co-100	C	BP	2187	48417	1995817		48417		2.63994e+06
	core2536-691	B	MBP	2539	15293	177739		15284	9	689
	core4872-1529	C	MBP	4875	24656	218762		24645	11	?
	cov1075	B	BP	637	120	14280		120		20
	csched007	T	MBP	351	1758	6379		1457	301	351
	csched008	RT	MBP	351	1536	5687		1284	252	173
	csched010	B	MBP	351	1758	6376		1457	301	408
	d10200	C	IP	947	2000	57637	1267	733		12430
	d20200	C	IP	1502	4000	189389	819	3181		?
	dano3mip	CR	MBP	3202	13873	79655		552	13321	?
	danoint	B	MBP	664	521	3232		56	465	65.6667
	datt256	C	BP	11077	262144	1503732		262144		?
	dc1c	C	MBP	1649	10039	121158		8380	1659	1.7679e+06
	dc1l	C	MBP	1653	37297	448754		35638	1659	?

Easy

Open

今なぜ整数計画MIPなのか

非常に！

◇「整数条件」で応用範囲が広がる

◇組合せ最適化（離散最適化）を含む，汎用的なモデル

◇汎用的ゆえに実用性は絶望視されていた

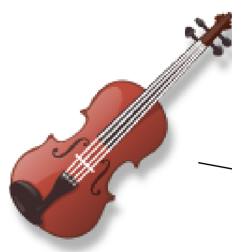
◇しかし，MIPソルバーが高速化

- LPソルバーの進化，切除平面法の併用，高速化のための技術

以外と？

◇整数計画をはじめるのは難しくない　　次の話題

〇百万円



数千円

MIPではフリーソフトも充実
十分高性能で使いやすさも向上

アカデミックフリーのソフトも！

Excelソルバー

- ◇Microsoft Office または Excel をインストールすると利用できるアドイン
- ◇Excelソルバーの解説書籍・解説
  - ✓ 高井，真鍋(編著)「問題解決のためのオペレーションズ・リサーチ入門」，日本評論社，2000年
  - ✓ 柏木「Excelで学ぶ意思決定論」，オーム社，2006年
  - ✓ 阿部「Excelで学ぶ統計解析」，ソシム，2006年
  - ✓ 藤澤，後藤，安井「Excelで学ぶOR」，オーム社，2011年
  - ✓ 後藤: Excelで学ぶ数理最適化，オペレーションズ・リサーチ，Vol.57, No.4, pp.175-182 (2012)

Excelソルバー

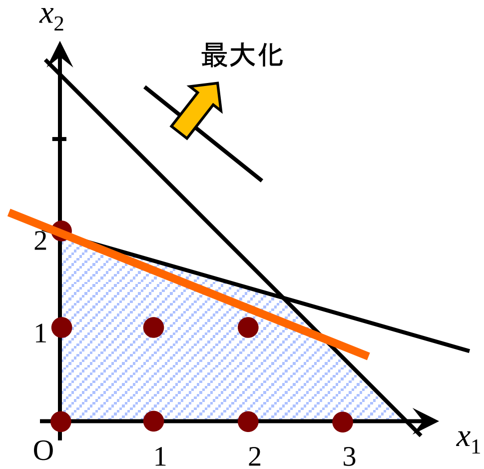
最大化　$z = 2x_1 + 5x_2$

条件　$3x_1 + 3x_2 \le 11$

$2x_1 + 7x_2 \le 14$

$x_1, x_2 \ge 0$

x_1, x_2 : 整数



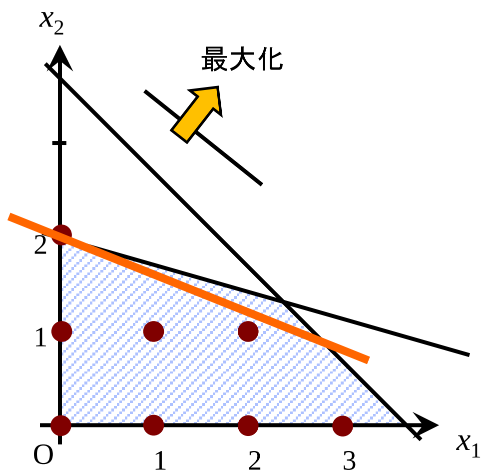
データの入力

解を書き入れるセル
（B2:C2）

	A	B	C	D	E
1		x1	x2	(左辺)	(右辺)
2					
3		2	5		
4		3	3		11
5		2	7		14

Excelソルバー

最大化	$z = 2x_1 + 5x_2$
条件	$3x_1 + 3x_2 \le 11$
	$2x_1 + 7x_2 \le 14$
	$x_1, x_2 \ge 0$
	x_1, x_2 :整数



数式の入力

または

```
=SUMPRODUCT(B$2:C$2,B3:C3)
=SUMPRODUCT(B$2:C$2,B4:C4)
=SUMPRODUCT(B$2:C$2,B5:C5)
```

	A	B	C	D	E
1		x1	x2	(左辺)	(右辺)
2					
3		2	5	=B3*B$2+C3*C$2	
4		3	3	=B4*B$2+C4*C$2	11
5		2	7	=B5*B$2+C5*C$2	14

Excelソルバー

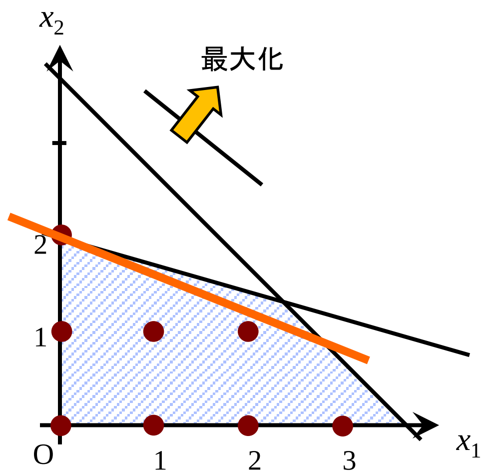
最大化　$z = 2x_1 + 5x_2$

条件　$3x_1 + 3x_2 \le 11$

$2x_1 + 7x_2 \le 14$

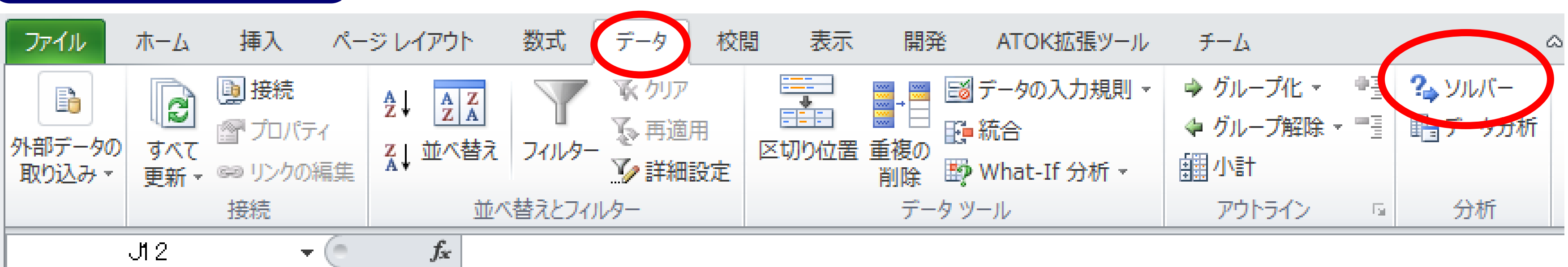
$x_1, x_2 \ge 0$

x_1, x_2 : 整数



見当たらない場合は，
「ファイル」→「オプション」→「アドイン」から

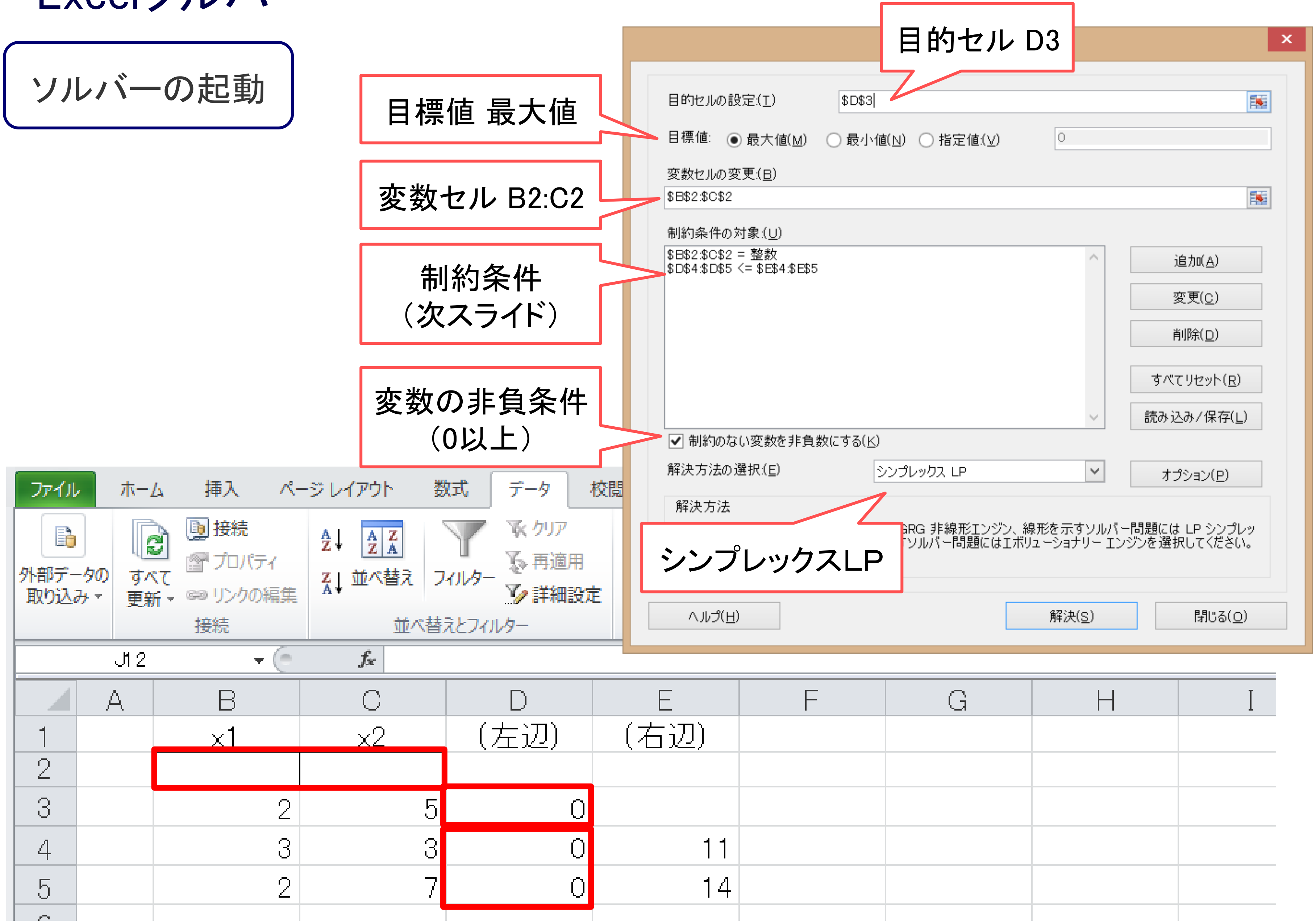
ソルバーの起動



	A	B	C	D	E	F	G	H	I
1		x1	x2	（左辺）	（右辺）				
2									
3		2	5	0					
4		3	3	0	11				
5		2	7	0	14				

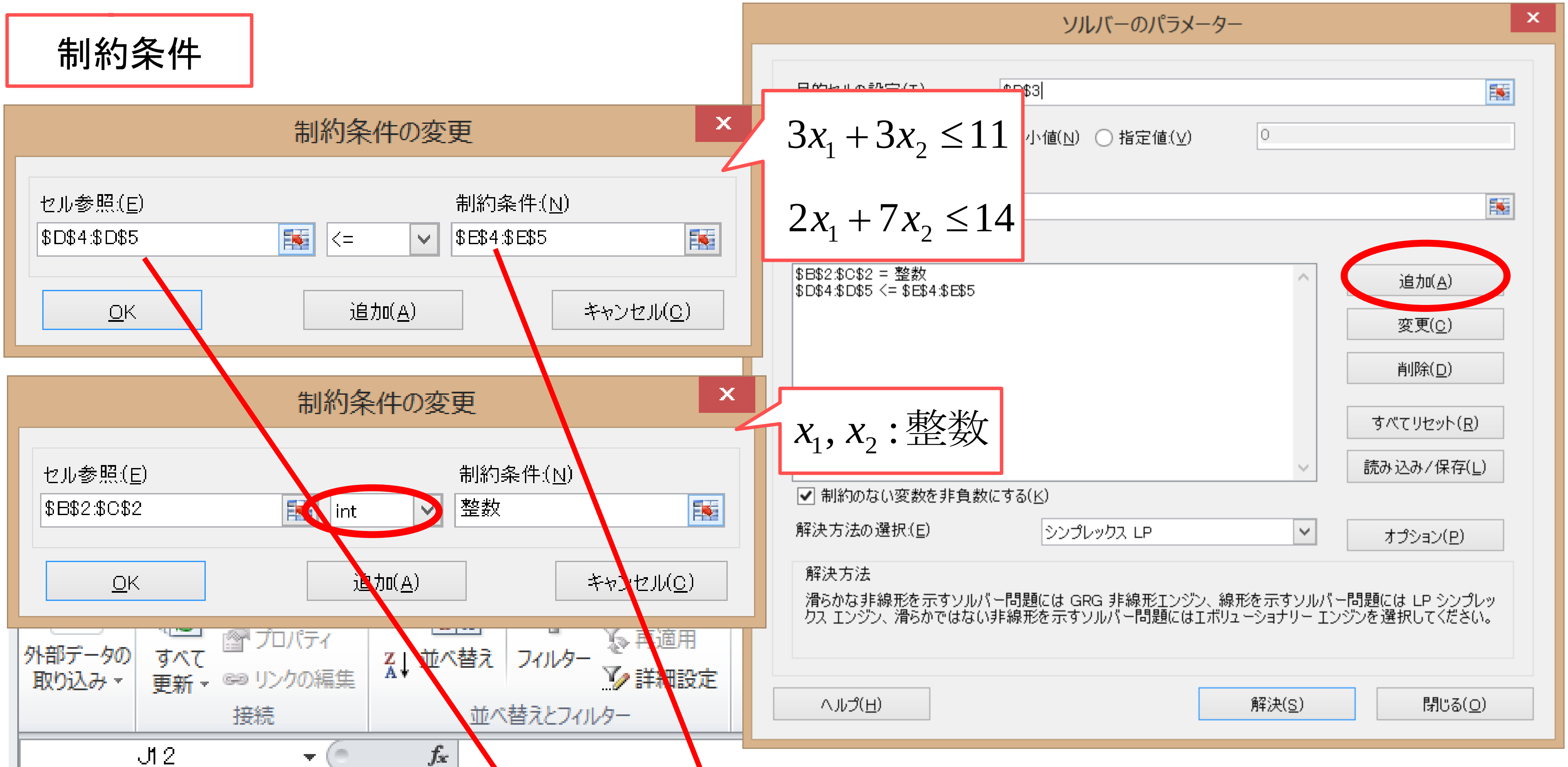
Excelソルバー

ソルバーの起動



Excelソルバー

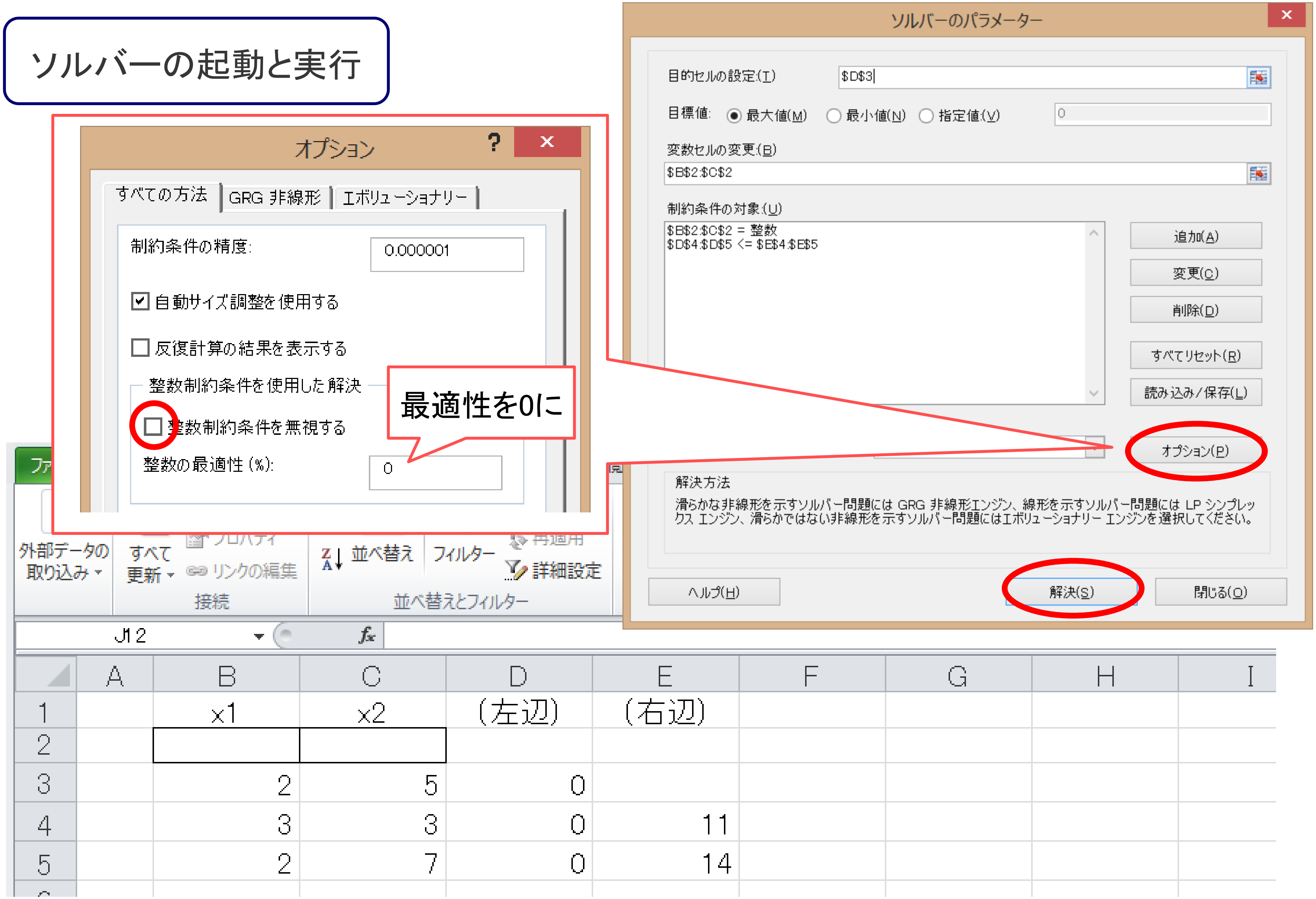
制約条件



	A	B	C	D	E
1		x1	x2	(左辺)	(右辺)
2					
3		2	5	0	
4		3	3	0	11
5		2	7	0	14

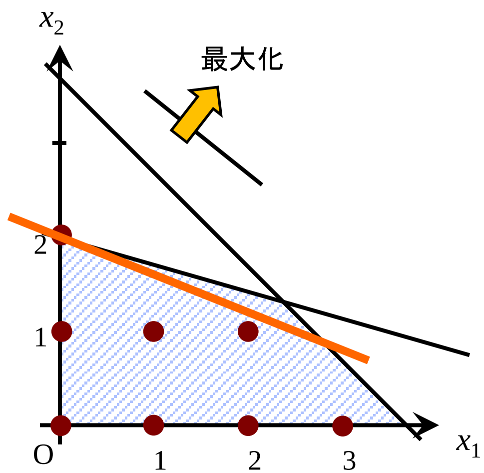
Excelソルバー

ソルバーの起動と実行



Excelソルバー

$$
\begin{array}{ll}
\text{最大化} & z = 2x_1 + 5x_2 \\
\text{条件} & 3x_1 + 3x_2 \le 11 \\
 & 2x_1 + 7x_2 \le 14 \\
 & x_1, x_2 \ge 0 \\
 & x_1, x_2 : \text{整数}
\end{array}
$$



最適解

最適解

最適値

	A	B	C	D	E	F	G	H	I
1		x1	x2	(左辺)	(右辺)				
2		0	2						
3		2	5	10					
4		3	3	6	11				
5		2	7	14	14				

MIPソルバー

◇商用

- FICO Xpress (Fair Isaac Corporation)
- Gurobi Optimizer (Gurobi Optimization)
- IBM ILOG CPLEX (IBM)
- LINDO (LINDO Systems)
- NUOPT (NTTデータ 数理システム)
- SOPT (Saitech, Inc.) 等

◇非商用

- COIN/CBC
- GNU GLPK
- lp_solve
- SCIP 等

問題ファイルの作成

最大化 $z = 2x_1 + 5x_2$

条件 $3x_1 + 3x_2 \le 11$

$2x_1 + 7x_2 \le 14$

$x_1, x_2 \ge 0$

x_1, x_2 : 整数

CPLEX LP形式(sample1.lp)

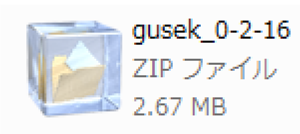
```
maximize
  2 x1 + 5 x2
subject to
  3 x1 + 3 x2 <= 11
  2 x1 + 7 x2 <= 14
general
  x1 x2
end
```

MPS形式(sample1.mps)

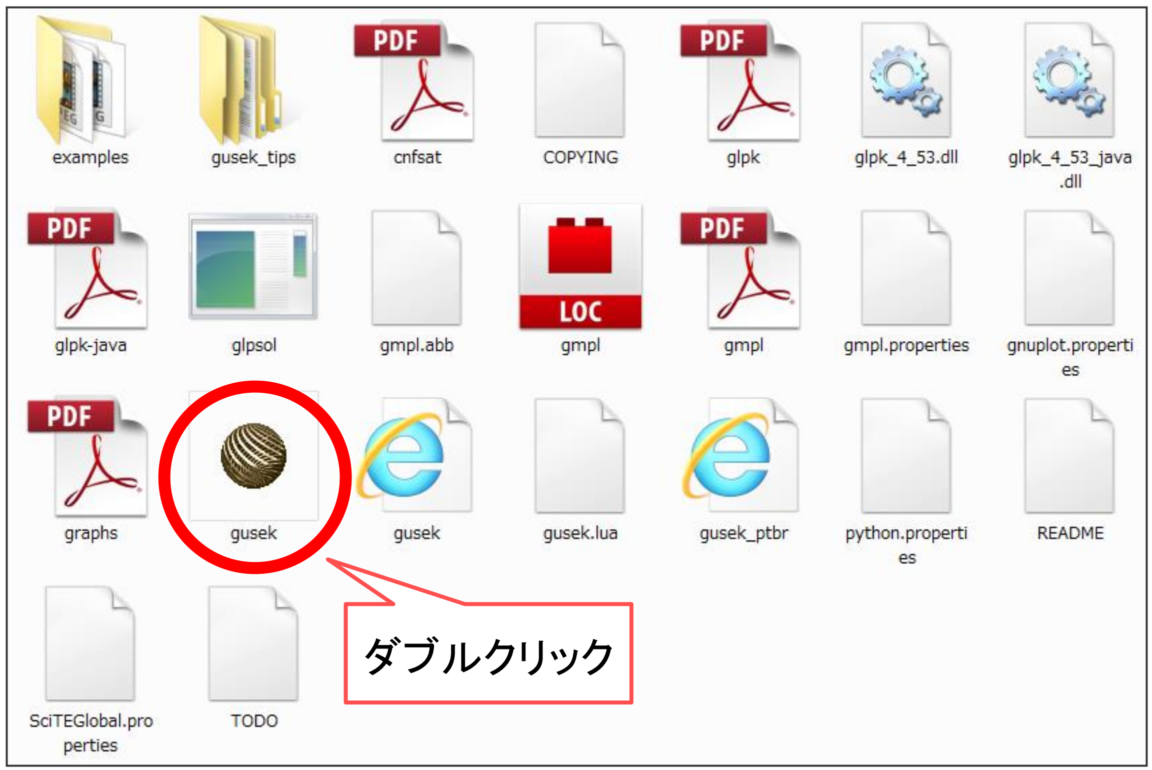
```
NAME          sample1.mps
ROWS
 N  z
 L  r1
 L  r2
COLUMNS
    M1        'MARKER'                 'INTORG'
    x1        z                   -2   r1                    3
    x1        r2                   2
    x2        z                   -5   r1                    3
    x2        r2                   7
    M2        'MARKER'                 'INTEND'
RHS
    RHS1      r1                  11   r2                   14
BOUNDS
 PL BOUND     x1
 PL BOUND     x2
ENDATA
```

GUSEK（GLPKのIDE）による実行

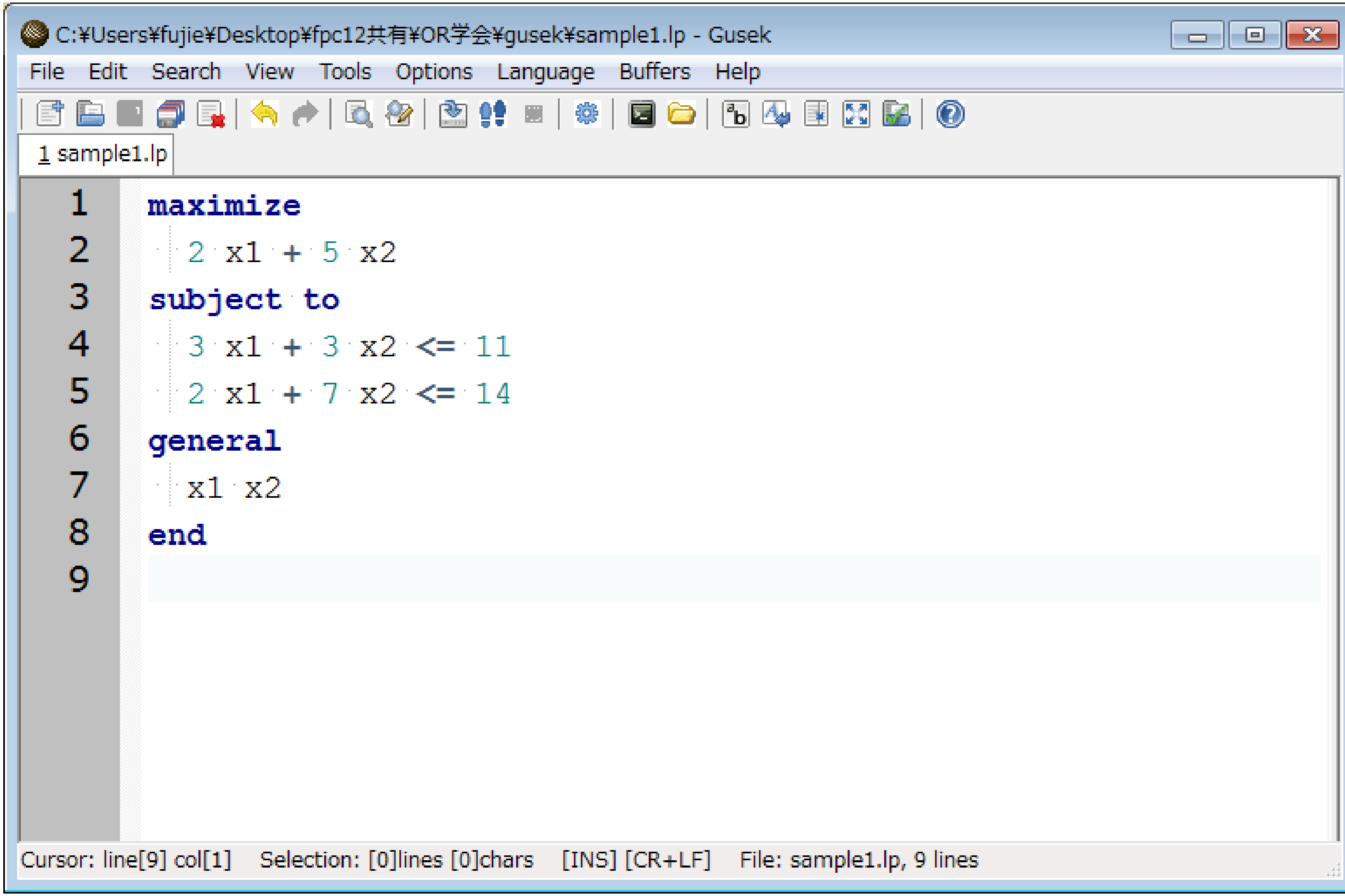
sourceforge.jp等からダウンロード → zipファイルを解凍



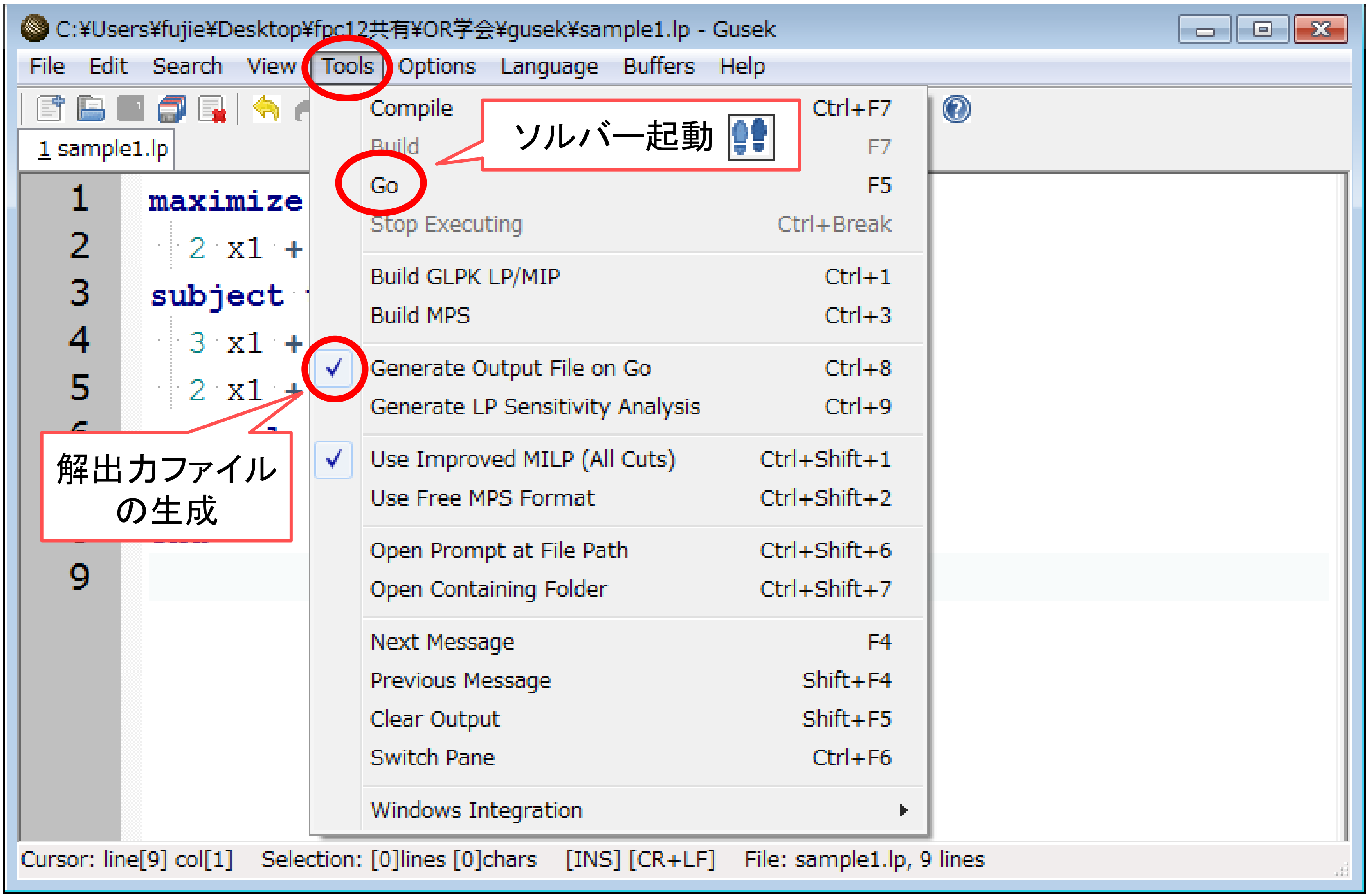
解凍



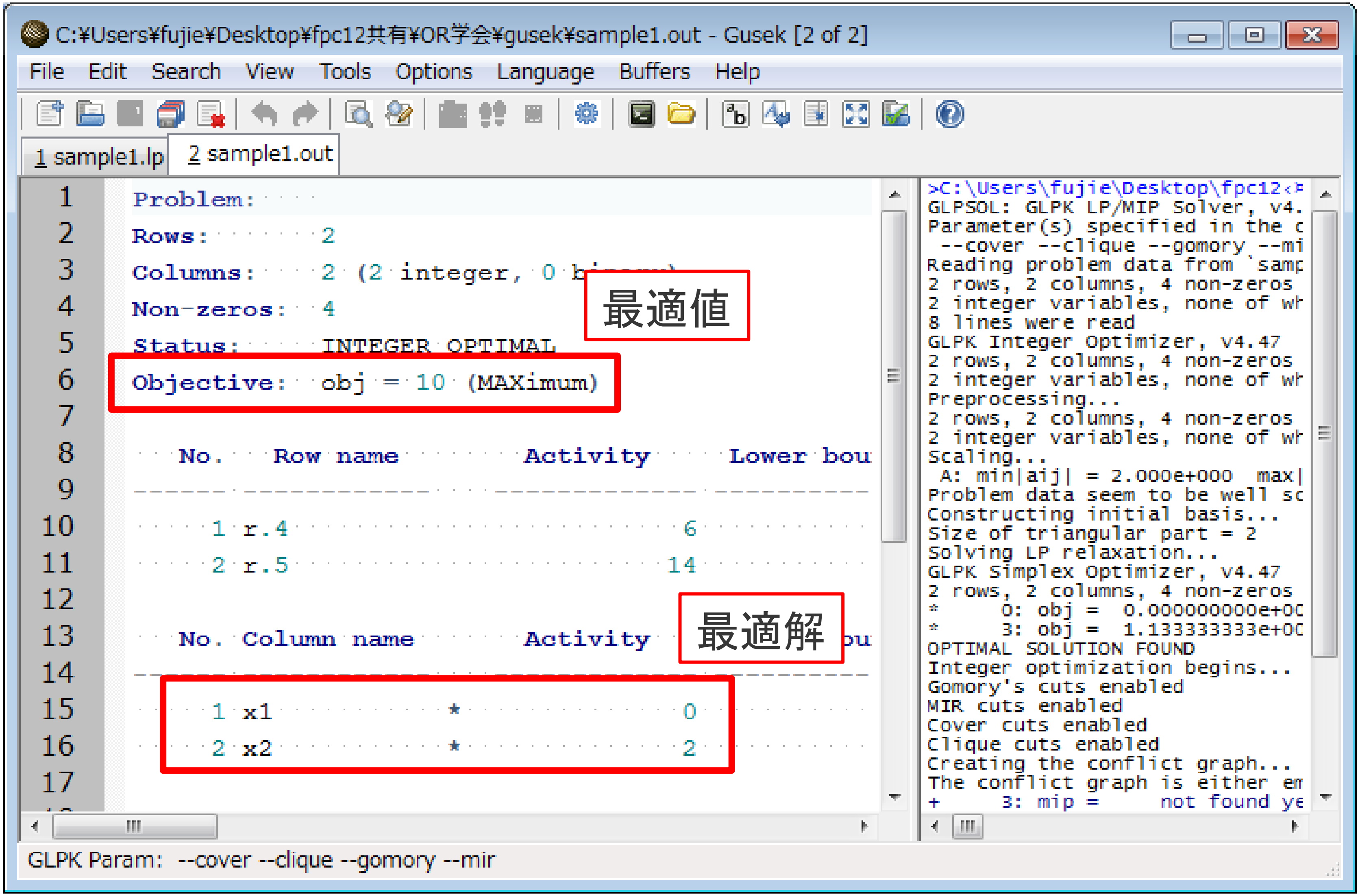
GUSEK（GLPKのIDE）による実行



GUSEK（GLPKのIDE）による実行

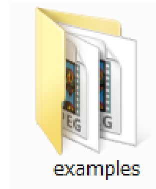


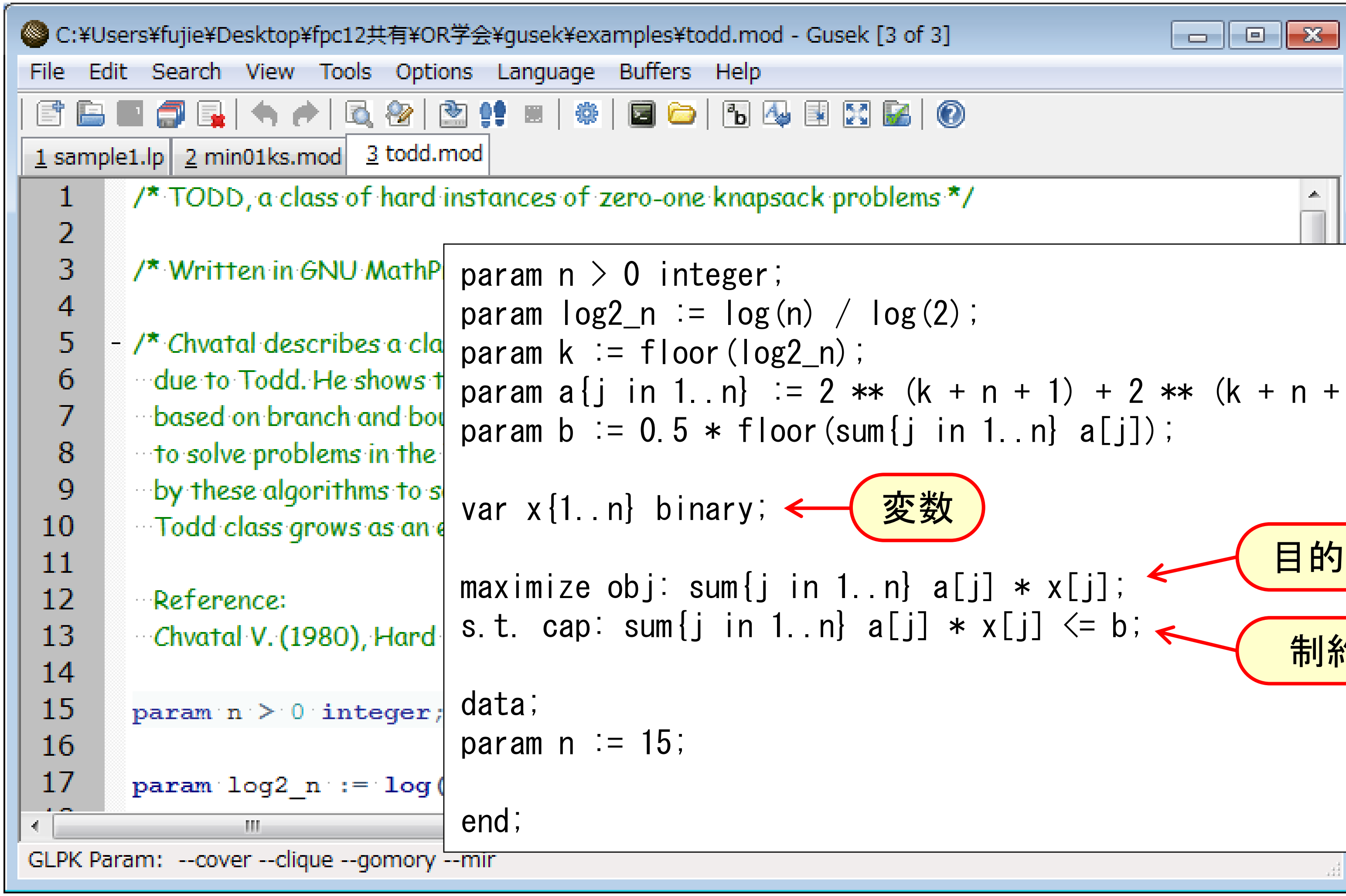
GUSEK（GLPKのIDE）による実行



モデリング言語(MathProg)

examples¥todd.mod (ナップサック問題)






```
param n > 0 integer;
param log2_n := log(n) / log(2);
param k := floor(log2_n);
param a{j in 1..n} := 2 ** (k + n + 1) + 2 ** (k + n + 1 - j) + 1;
param b := 0.5 * floor(sum{j in 1..n} a[j]);

var x{1..n} binary;

maximize obj: sum{j in 1..n} a[j] * x[j];
s.t. cap: sum{j in 1..n} a[j] * x[j] <= b;

data;
param n := 15;

end;
```

変数

目的関数

制約式

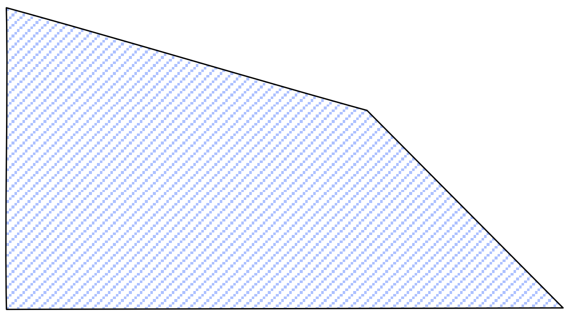
ここまでのまとめ

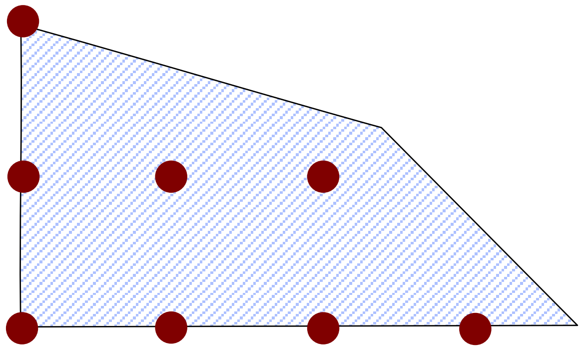
1. 整数計画MIPとは何か
2. 今なぜ整数計画なのか
   - 応用範囲が広い
   - ソルバーが劇的に高速化
3. 整数計画をはじめよう
   - 道具をそろえよう → *意外と身近*
   - 道具を使ってみよう → *意外と容易*

4. 定式化について

線形計画LP と 整数計画MIP

	LP	MIP
実行可能解集合の形	多面体	格子点(全整数の場合)



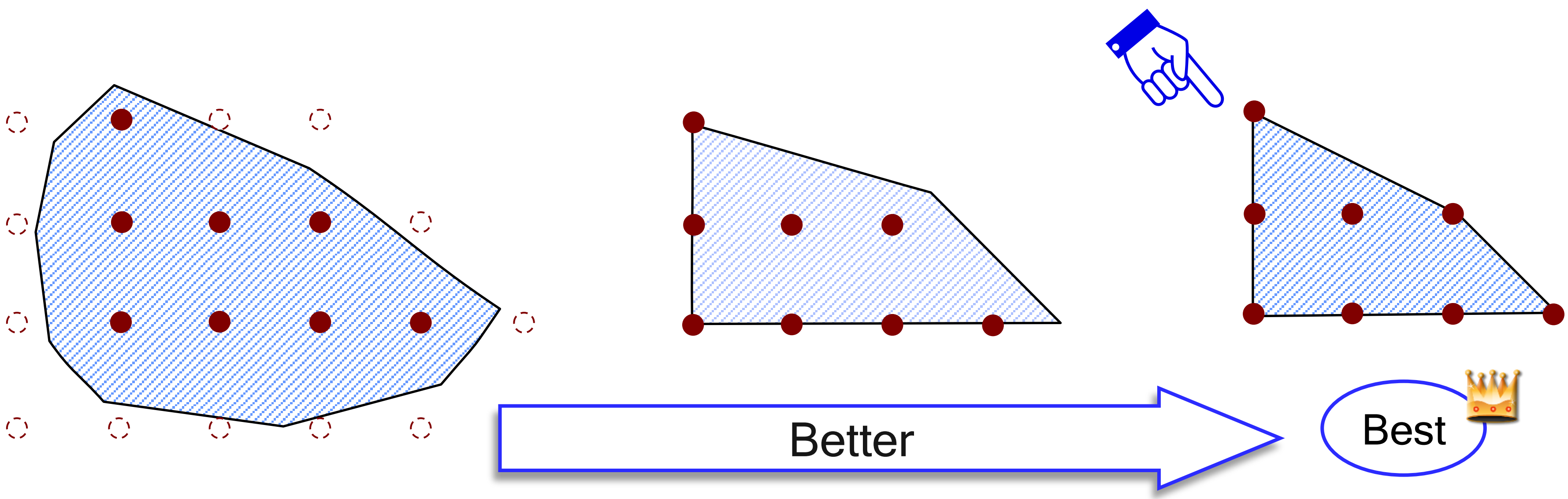


整数計画MIP

◇ 様々な定式化が可能

理想的な定式化（凸包）

- ✓ これがわかれば『MIP = LP』
- ✓ しかし，一般に知ることは困難
- ✓ わかったとしても，膨大な数の制約式になる可能性が大きい



定式化について

◇ 定式化は複数ありうる

- 変数の定義も複数ありうる

定式化の例：数独

7				5			4	
		9			3	1		
6	3	8	4				7	
		3	6			9		2
5	9		1			3		
8		2			5			7
	4			3	1	7		5
					9		6	
9	1		2	7				4

7	2	1	9	**5**	8	6	**4**	3
4	5	**9**	7	6	**3**	**1**	2	8
6	**3**	**8**	**4**	1	2	5	**7**	9
1	7	**3**	**6**	8	4	**9**	5	**2**
5	**9**	4	**1**	2	7	**3**	8	6
8	6	**2**	3	9	**5**	4	1	**7**
2	**4**	6	8	**3**	**1**	**7**	9	**5**
3	8	7	5	4	**9**	2	**6**	1
9	**1**	5	**2**	**7**	6	8	3	**4**

- あいているマスに 1–9 までのどれかの数字を入れる。
- 縦・横の各列及び、太線で囲まれた3×3のブロックに同じ数字が入ってはいけない。

Wikipediaの「数独」より

数独の定式化

方法1

x_{ij} : (i, j)マスに入る数字（つまり$1 \le x_{ij} \le 9$）

方法2

$$x_{ijk} = \begin{cases} 1 & (i, j)\text{マスに入る数字が } k \text{ のとき} \\ 0 & \text{そうでないとき} \end{cases}$$

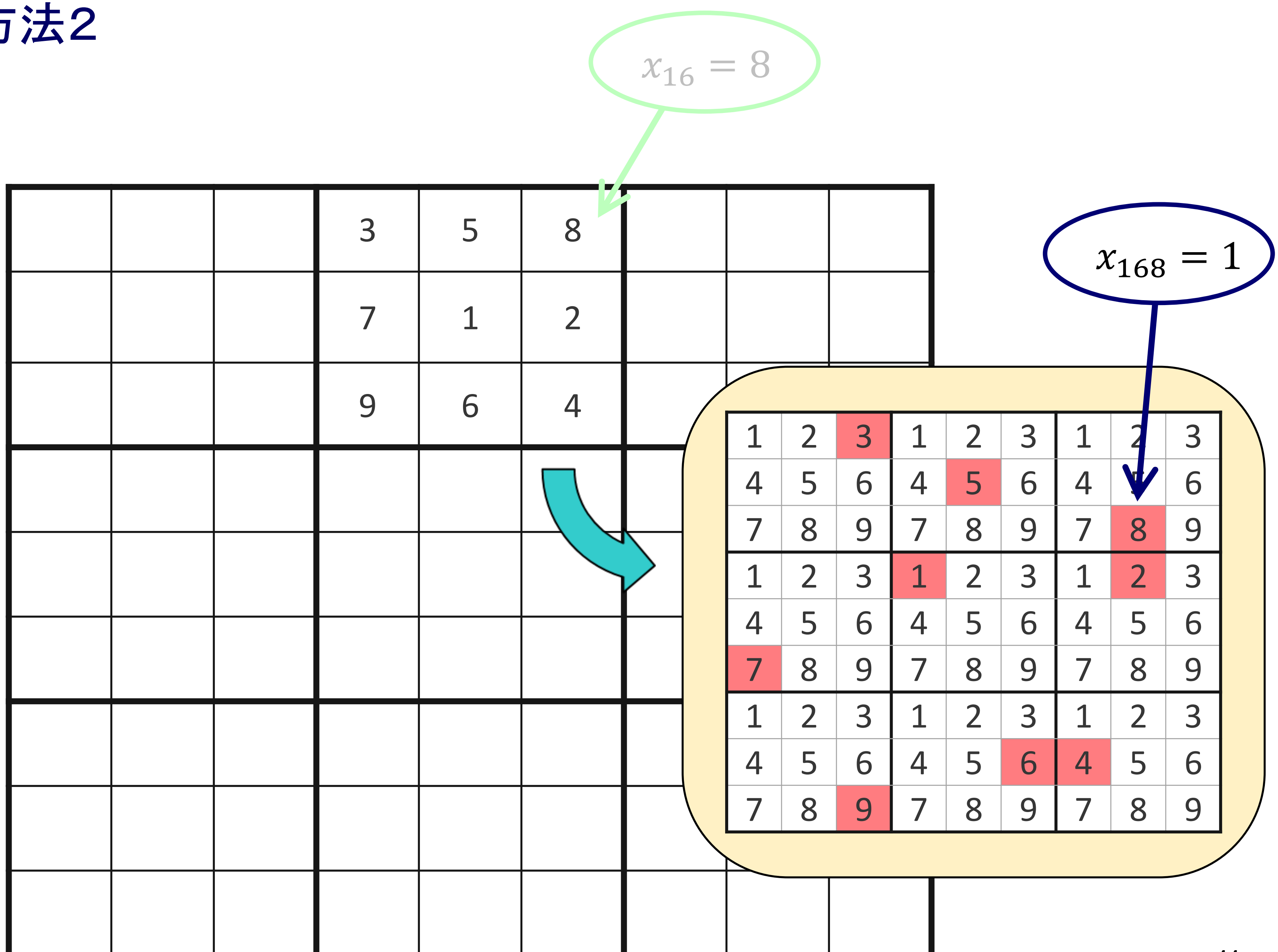
T. Koch, "Rapid Mathematical Programming or How to Solve Sudoku Puzzles in a Few Seconds", Operations Research Proceedings 2005

方法1

$$x_{16} = 8$$

			3	5	8			
			7	1	2			
			9	6	4			

方法2



定式化:方法1

x_{ij} : (i, j)マスに入る数字（つまり$1 \le x_{ij} \le 9$）

最小化　*arbitrarily*

条件

$$\left| x_{ij} - x_{i\ell} \right| \ge 1 \qquad \left(i, j, \ell = 1, \cdots, n;\ j < \ell\right)$$

$$\left| x_{ij} - x_{kj} \right| \ge 1 \qquad \left(i, j, k = 1, \cdots, n;\ i < k\right)$$

$$\left| x_{3(r-1)+i,3(c-1)+j} - x_{3(r-1)+k,3(c-1)+\ell} \right| \ge 1 \qquad \left(i, j, k, \ell = 1,2,3;\ 3i + j < 3k + \ell\right)$$

$$1 \le x_{ij} \le 9 \qquad (i, j = 1, \cdots, n)$$

$$x_{ij} = k \qquad \text{(初期配置で}(i, j)\text{マスが } k \text{ のとき)}$$

$$x_{ij} : \text{整数} \qquad \left(i, j = 1, \cdots n\right)$$

$n = 9$

定式化：方法1

注1

$$\left| x_{ij} - x_{i\ell} \right| \geq 1 \Leftrightarrow \begin{cases} x_{ij} - x_{i\ell} = x^{+} - x^{-} \\ b^{+} \leq x^{+} \leq 8b^{+} \\ b^{-} \leq x^{-} \leq 8b^{-} \\ b^{+} + b^{-} = 1 \\ b^{+}, b^{-} = 0 \text{ または } 1 \end{cases}$$

注2　$\text{all\_different}(x_{i1}, \cdots, x_{in})$　という表現をすることも多い

定式化:方法2

$$x_{ijk} = \begin{cases} 1 & (i, j)\text{マスに入る数字が } k \text{ のとき} \\ 0 & \text{そうでないとき} \end{cases}$$

最小化 $arbitrarily$

条件

$$\sum_{k=1}^{n} x_{ijk} = 1 \qquad (i, j = 1, \cdots, n)$$

$$\sum_{j=1}^{n} x_{ijk} = 1 \qquad (i, k = 1, \cdots, n)$$

$$\sum_{i=1}^{n} x_{ijk} = 1 \qquad (j, k = 1, \cdots, n)$$

$$\sum_{i=3(r-1)+1}^{3r} \sum_{j=3(c-1)+1}^{3c} x_{ijk} = 1 \qquad (r, c = 1,2,3;\ k = 1, \cdots n)$$

$$x_{ijk} = 1 \qquad (\text{初期配置で}(i, j)\text{マスが } k \text{ のとき})$$

$$x_{ijk} = 0 \text{ または } 1 \qquad (i, j, k = 1, \cdots n)$$

定式化の比較

◇定式化1

- 変数 3970(一般整数変数 2025、0-1変数 1944)
- 制約 4860＋固定制約数
- 計算時間 41.5秒(Intel Core2 Duo 6600@2.40GHz, メモリ2.0GB, CPLEX 12.5)
  ※Kochの論文によると、CPLEX9.03では6時間経っても解けなかった

◇定式化2

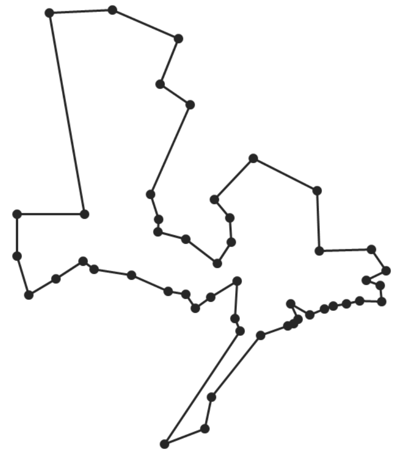
- 変数 729(0-1変数 729)
- 制約 346＋固定制約数
- 計算時間 0.02秒

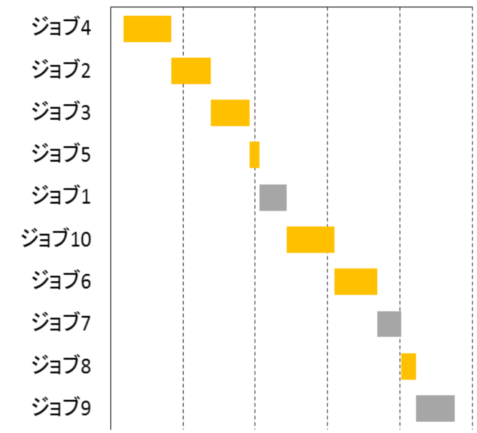
定式化について

- ◇ 定式化は複数ありうる
  - 変数の定義も複数ありうる
- ◇ MIPとLPがかけ離れている（LP緩和が弱い）定式化は望ましくない
  - big-Mを含む表現

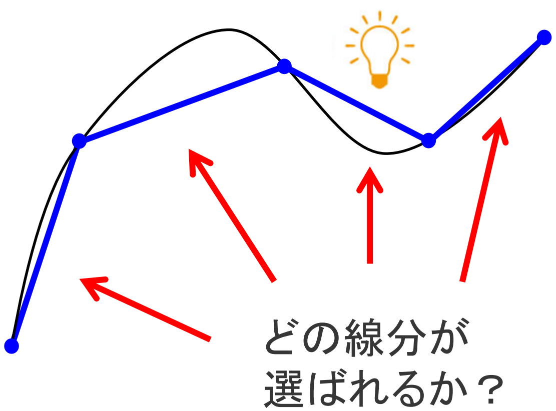
バイナリ変数(0-1変数)を用いた表現

◇ 組合せ最適化問題
- 巡回セールスマン問題
- 集合分割問題
- 集合被覆問題
- スケジューリング問題
- 施設配置問題
- ・・・



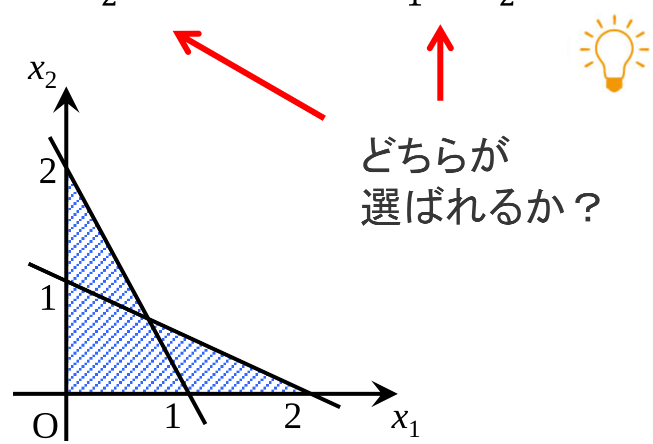


◇ 非線形関数の線形近似



◇ 離接(disjunctive)制約

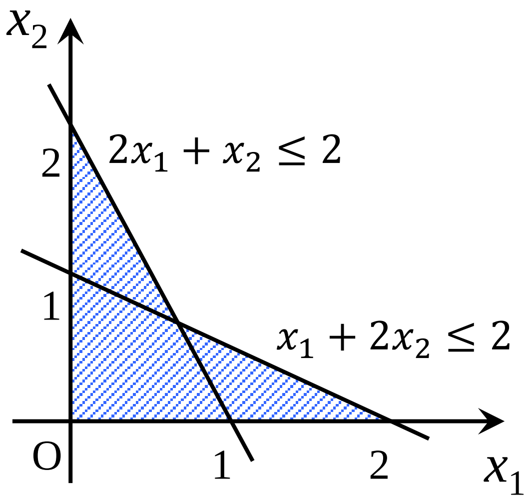
$x_1 + 2x_2 \leq 2$ または $2x_1 + x_2 \leq 2$



Big-Mを含む問題例

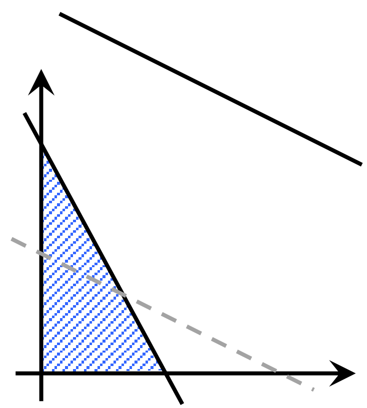
$$
\begin{aligned}
&\text{最大化} \quad z = 2x_1 + 3x_2 \\
&\text{条件} \quad x_1 + 2x_2 \le 2 \\
&\qquad\quad \text{または} \ \ 2x_1 + x_2 \le 2 \\
&\qquad\quad x_1,\, x_2 \ge 0
\end{aligned}
$$

$$
\begin{aligned}
&\text{最大化} \quad z = 2x_1 + 3x_2 \\
&\text{条件} \quad x_1 + 2x_2 - 2 \le M(1-y) \\
&\qquad\quad 2x_1 + x_2 - 2 \le My \\
&\qquad\quad x_1,\, x_2 \ge 0, \ \ y = 0 \ \text{または} \ 1
\end{aligned}
$$

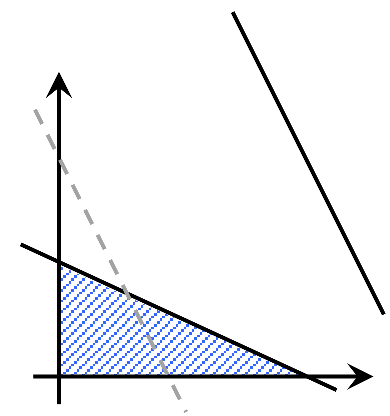


$y = 0$ または $y = 1$

$$
\begin{aligned}
x_1 + 2x_2 - 2 &\le M \\
2x_1 + x_2 - 2 &\le 0
\end{aligned}
$$

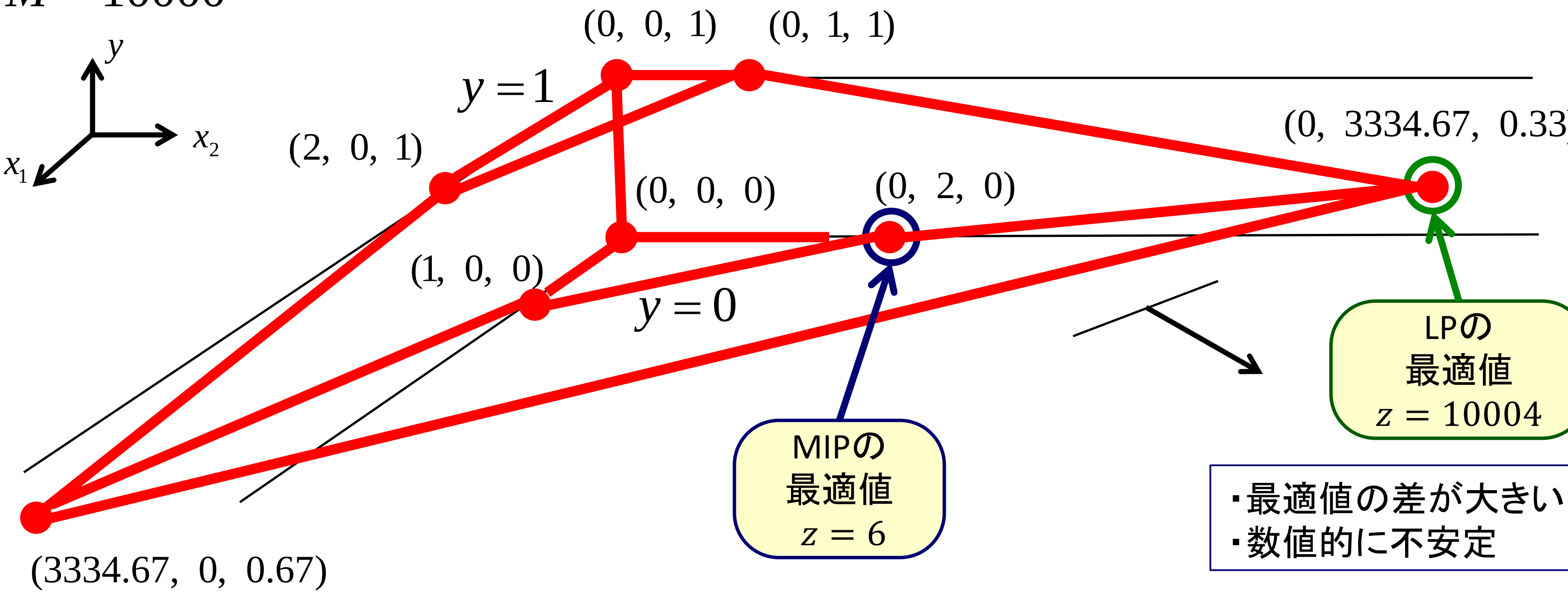


$$
\begin{aligned}
x_1 + 2x_2 - 2 &\le 0 \\
2x_1 + x_2 - 2 &\le M
\end{aligned}
$$



最大化	$z = 2x_1 + 3x_2$
条件	$x_1 + 2x_2 - 2 \le M(1 - y)$
	$2x_1 + x_2 - 2 \le My$
	$x_1, x_2 \ge 0,$ $y = 0$ または 1 $\rightarrow$ $0 \le y \le 1$

$M = 10000$



- 最適値の差が大きい
- 数値的に不安定

定式化について

◇ 定式化は複数ありうる

- 変数の定義も複数ありうる

◇ MIPとLPがかけ離れている(LP緩和が弱い)定式化は望ましくない

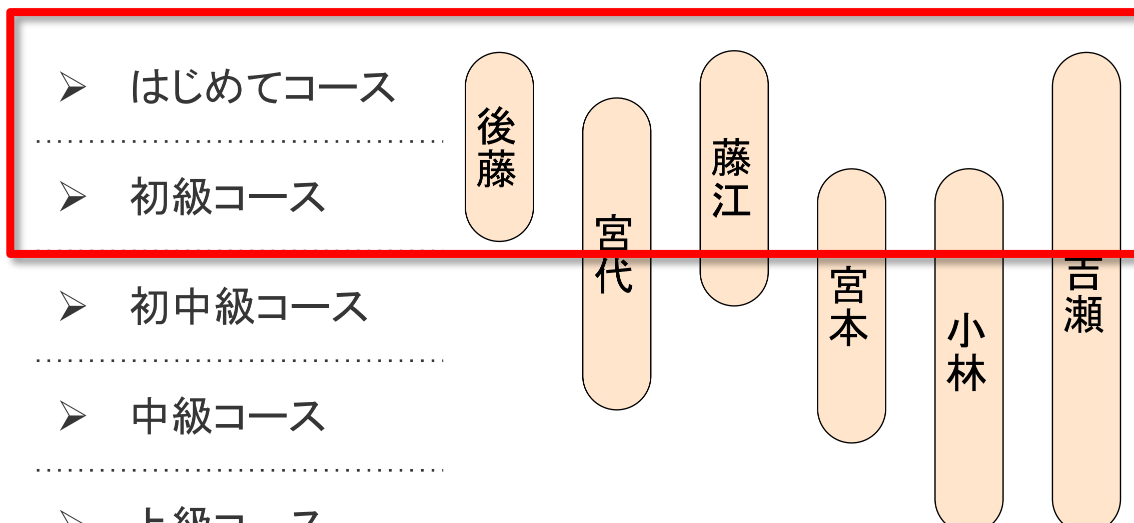
- big-Mを含む表現

◇ 緩和の強さ vs変数・制約式の数

…といろいろ話はありますが，まずは「はじめよう定式化」！

OR学会誌2012年4月号

特集　はじめよう整数計画



補足：MIPソルバーの利用方法

◇ 問題ファイルを作成し，コマンドラインから実行

◇ IDE（統合開発環境）の使用

◇ モデリング言語

- 商用

  AIMMS, AMPL, GAMS, LINGO, MOSEL, MPL, OPL, Simple, . . .

- フリー

  MathProg, ZIMPL, . . .

◇ API（Application Interface）

- プログラムから呼び出す
- C, C++, Java, Excel, Matlab, python, . . .

補足：整数計画いろいろ

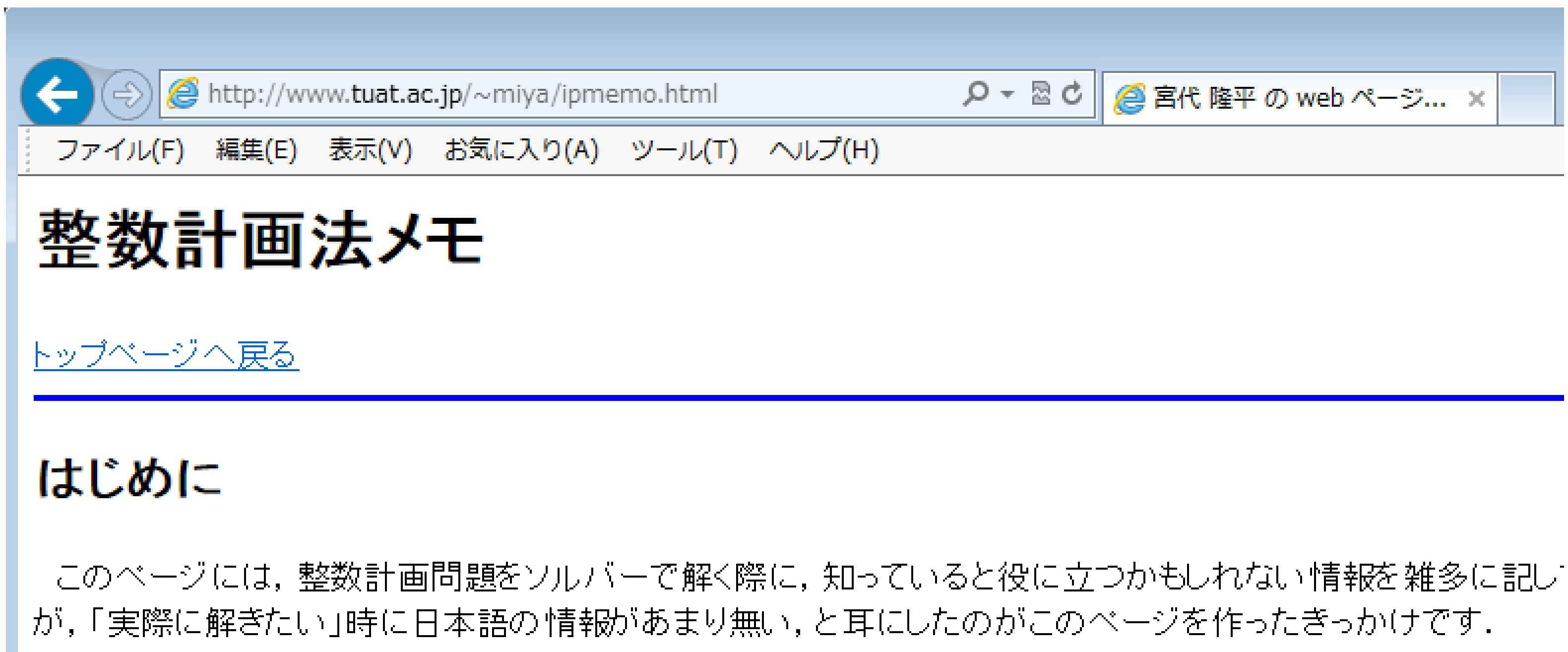
- ◇ 線形計画
- ◇ 二次計画（線形制約）
- ◇ 二次計画（二次制約）
- ◇ 二次錐計画
- ◇ 半正定値計画
- ◇ 非線形計画

＋ 整数条件

参考情報

整数計画法メモ（東京農工大 宮代先生）

http://www.tuat.ac.jp/~miya/ipmemo.html



参考情報

OR学会誌特集号（最適化）

- ◇ 2011年５月号「最適化技術の深化と広がり」
- ◇ 2013年12月号「はじめようメタヒューリスティクス」
- ◇ 2014年１月号「研究の楽しさ」
- ◇ 2014年３月号「新世代が切り拓く連続最適化」

さらに

同誌には，整数計画の適用事例を扱った論文・解説多数

参考情報

RAMPシンポジウム

http://www.orsj.or.jp/ramp/


http://www.orsj.or.jp/ramp/ RAMP Symposium ラン...

ファイル(F) 編集(E) 表示(V) お気に入り(A) ツール(T) ヘルプ(H)

日本オペレーションズ・リサーチ学会常設研究部会

「数理計画(RAMP)研究部会」

1988年に第13回のISMP(International Symposium on Mathematical Programming)が東京で開催されました。日本でこのシンポジウムが開催されたのをきっかけとし、日本オペレーションズ・リサーチ学会の常設研究部会として、数理計画(RAMP)研究部会が組織されました。

数理計画(Mathematical Programming)の理論と応用に関する研究と実施の活性化のために、それらの最先端の成果を情報交換する場として、毎年1回「RAMPシンポジウム」を開催しています。

第25回 RAMPシンポジウム ホームページ
日時：2013年10月29日(火)，30日(水)
会場：鹿児島大学稲盛会館
実行委員長：新森 修一（鹿児島大学）
プログラム委員長：岩田　覚(東京大学)

過去の「RAMPシンポジウム」の開催状況は下記の通りです。

第24回	東北大学片平キャンパスさくらホール	2011年9月27日，28日	ホームページ
第23回	関西大学　千里山キャンパス・百周年記念ホール	2011年10月24日，25日	ホームページ