情報システム評価学
―整数計画法―

第12回目：IPソルバーを使ってみよう

塩浦昭義（東北大学 大学院情報科学研究科 准教授）

第３回レポート

- 締切：２月１３日（金）午後５時まで
  （特別な理由により締切を延期して欲しい場合は直接問い合わせてください）
- 提出先：情報科学研究科棟８階８０３号室へ

問題１（必須）：身の回りにある問題を整数計画問題として定式化し，整数計画ソルバーを使って解きなさい

- 問題の例
  - 自分の研究テーマに関連する問題
  - 数独，ののぐらむなど，ペンシルパズル
  - その他，実社会の（整数計画問題に定式化可能な）問題

第３回レポート

問題１（続き）：

- やってもらうこと
  - 選んだ問題を整数計画問題に定式化
  - 整数計画ソルバーに入力，最適解を求め，計算時間を測定
  - 一つのデータだけでなく，様々なデータに対して問題を何回も解き，問題のサイズ（変数と制約の数）と計算時間の関係を調べる
  - 定式化を工夫して，計算時間がどの程度変化するか調べる（例：妥当不等式の追加）

第３回レポート

問題１（続き）：

- レポートに書く内容
  - 解いた問題の定義
  - ソルバーで解く際に用いた定式化とその説明
  - 得られた最適解と要した計算時間
  - 実験に関する考察（問題のサイズと計算時間の関係，定式化の工夫による影響など）

少なくとも一つ以上の問題に対して，整数計画ソルバーを使って問題を解いてください（必須）

第3回レポート

問題2：切除平面法及び妥当不等式に関する以下の問題を解きなさい

(1) 以下のように解集合 X とベクトルx が与えられたとき，x をカットする X の妥当不等式を求めよ.

(i) $X = \{(x_1, x_2, x_3) \mid x_1 + x_2 \leq 2x_3, 0 \leq x_1 \leq 1,$
$0 \leq x_2 \leq 1, x_3 \in \{0, 1\}\}, \quad x = (1, 0, 1/2)$

(ii) $X = \{(x_1, x_2) \mid 0 \leq x_1 \leq 9, x_1 \leq 4x_2 x_2 \geq 0, x_2 \in \mathbf{Z}\},$
$x = (9, 9/4)$

(iii) $X = \{(x_1, x_2, x_3) \mid x_1 + x_2 \leq 25, x_1 + x_2 \leq 8x_3, x_3 \geq 0, x_3 \in \mathbf{Z}\},$
$x = (20, 5, 125/8)$

(2) $X = \{(x_1, x_2, x_3, x_4) \in \mathbf{Z}^4 \mid 4x_1 + 5x_2 + 9x_3 + 12x_4 \leq 34,$
$x_1, x_2, x_3, x_4 \geq 0\}$
に対し、$y_2 + y_3 + 2y_4 \leq 6$ が妥当不等式であることを証明せよ。

第3回レポート

(3) 次の整数計画問題について考える。

$$\begin{aligned} \text{Minimize} \quad & x_1 + 2x_2 \\ \text{subject to} \quad & x_1 + x_2 \geq 4 \\ & 0.5x_1 + 2.5x?2 \geq 2.5 \\ & x_1, x_2 \geq 0, \text{ 整数} \end{aligned}$$

ベクトル $x^* = (15/4, 1/4)$ はこの問題のLP緩和の最適解であることを示せ。
また、この解をカットする妥当不等式を求めよ。

(4) 次の整数計画問題を、Gomoryの小数切除平面法により解きなさい。

$$\begin{aligned} \text{Minimize} \quad & 5x_1 + 9x_2 + 23x_3 \\ \text{subject to} \quad & 20x_1 + 35x_2 + 95x_3 \geq 319 \\ & x_1, x_2, x_3 \geq 0, \text{ 整数} \end{aligned}$$

整数計画ソルバーとは

- 一般の整数計画問題を厳密に解いてくれるソフトウェア
- 有償のソルバー
  - CPLEX, Xpress-MP, NUOPT, MATLAB(Optimization Toolbox)
- 無償のソルバー
  - LP_Solve
  - GLPK
- 15年前と比べて200万倍スピードアップ！
  - 20日➔1秒！
  - ハードウェア1000倍、アルゴリズム2000倍

整数計画ソルバーの使い方

- 手順

① ソルバーをPCにインストール

② 解きたい問題を定式化し，適切な入力形式で記述

③ ソルバーに問題を入力

④ 最適解が出力される

- 幾つかの方法で利用可能
  - コマンドラインから
  - 統合開発環境から
  - 他のプログラミング言語から

問題の記述形式

- 様々な形式が存在
- 初心者にはLP形式がおすすめ

$$\begin{aligned} \text{Maximize} \quad & -x_1 + 2x_2 \\ \text{subject to} \quad & 2x_1 + x_2 \leq 5 \\ & -4x_1 + 4x_2 \leq 5 \\ & x_1, x_2 \geq 0, \text{ 整数} \end{aligned}$$


```
max: -x1 + 2 x2;
2x1 + x2 < 5;
-4 x1 + 4 x2 <5;
int x2,x1;
```

LP形式による問題記述：ルール1

- ファイルの拡張子は “.lp”
- 目的関数，各制約，変数定義の各行の最後にはセミコロン “;” を付ける
- 目的関数：
  - 最小化のときは “min:”, 最大化のときは “max:” と書く
  - 大文字，小文字のどちらでも可
- 制約：
  - 等号制約の場合は“=”，不等式制約（≧）の場合は “>=” もしくは “>”，不等号制約（≦）の場合は “<=” もしくは “<”

LP形式による問題記述：ルール2

- 変数：
  - x, x1, x_1, X, X1, X_1 のようにどんな記号もOK
  - ただし，異なる記号は異なる変数を表す
  - 何も宣言しなければ，非負の実数変数を表す
  - 非負の整数変数 x1, x2 は “int x1, x2;” と宣言
  - 負の値も取り得る実数変数を使いたい場合は，2つの非負変数x, y を用いて x – y のように表す（LPの不等式標準形への変換を参照）
- 目的関数や制約の係数は “2x1 – 7x2” のように書けばよい

LP形式による問題記述：具体例

Minimize $-x_1 - 2x_2 + 0.1x_3 + 3x_4$

subject to
$x_1 + x_2 \leq 5$
$2x_1 - x_2 \geq 0$
$-x_1 + 3x_2 \geq 0$
$x_3 + x_4 \geq 0.5$
$x_1, x_2 \geq 0$, 実数
$x_3, x_4 \geq 0$, 整数

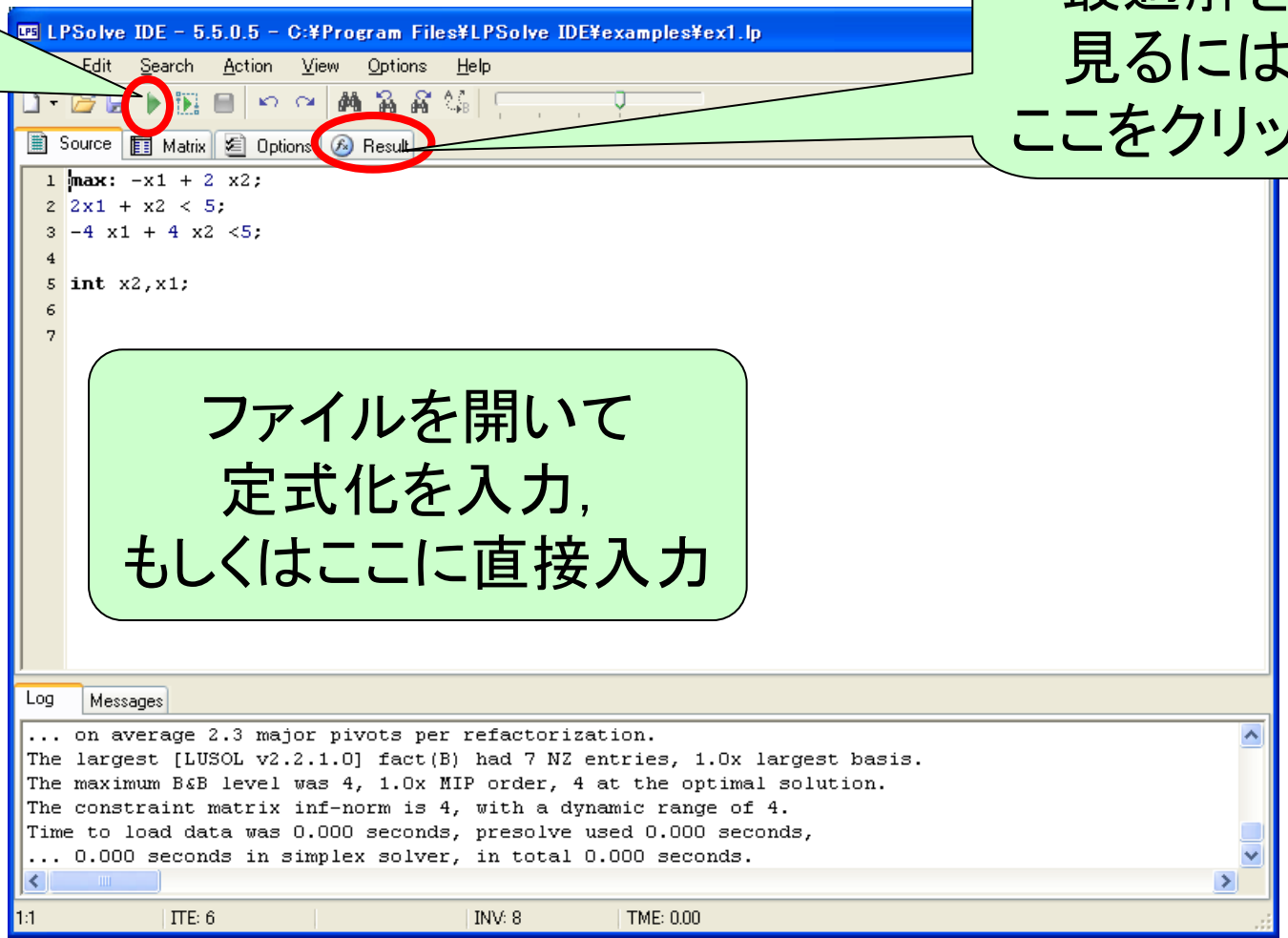

```
min: -x1 -2 x2 +0.1 x3 +3 x4;
+x1 +x2 <= 5;
 +2 x1 -x2 >= 0;
-x1 +3 x2 >= 0;
+x3 +x4 >= 0.5;
x3 >= 1.1;
int x3, x4;
```

LP_Solve の等号開発環境のインストール

- http://sourceforge.net/projects/lpsolve よりダウンロード可能
  - 上記のページの “Download” をクリック➔ “lpsolve” をクリック ➔ “lp_solve_5.5.*.*_IDE_Setup.exe” をダウンロードしてインストール

LP_Solve等号開発環境の使い方

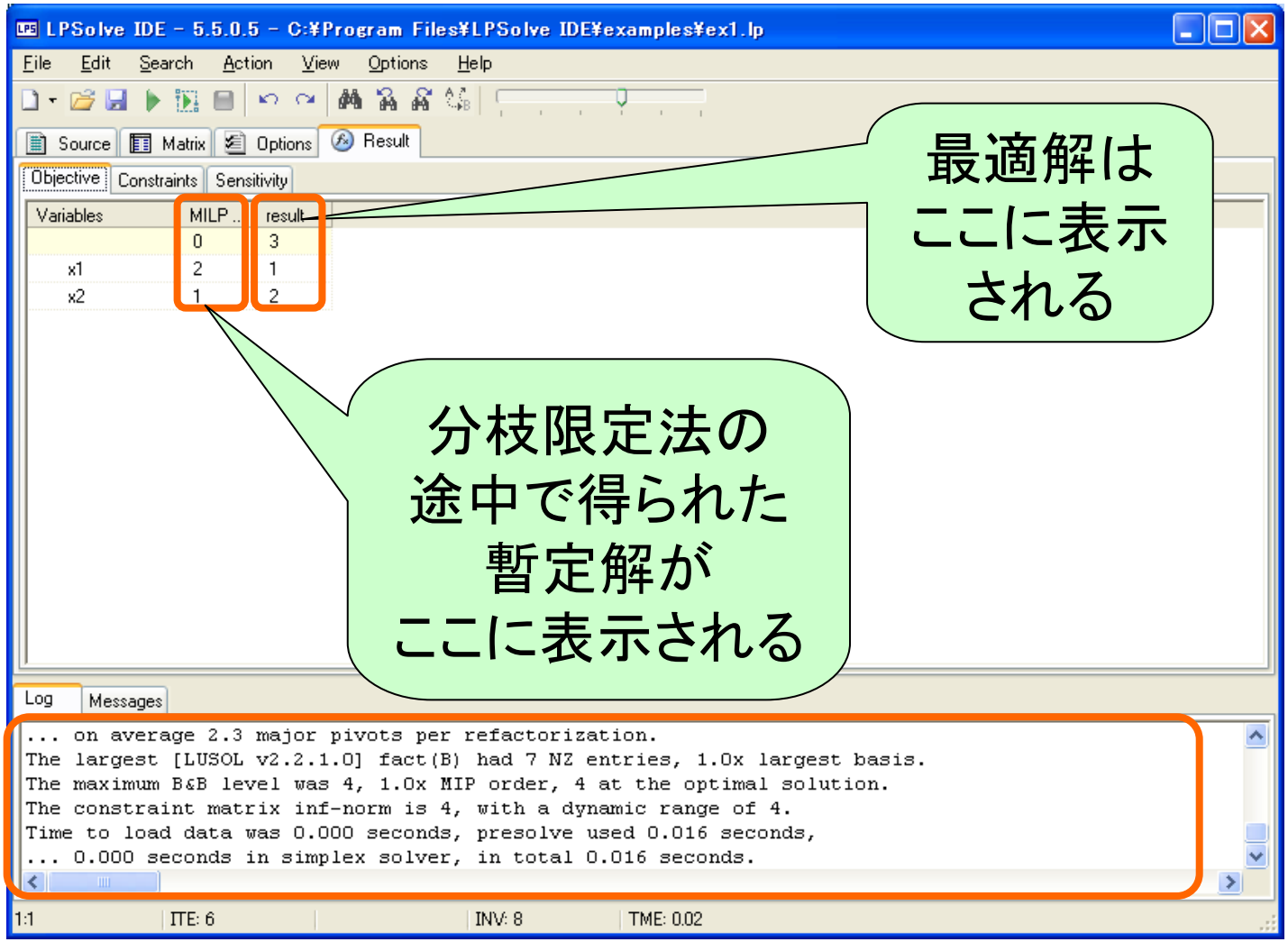
このボタンを押すと問題を解いてくれる



最適解を見るにはここをクリック

ファイルを開いて定式化を入力，もしくはここに直接入力

LP_Solve等号開発環境の使い方



問題を
解いたときの
ログはここに
表示される

ログの詳細

Model name: " - run #1
Objective: Maximize(R0)

SUBMITTED
Model size: 2 constraints, 2 variables, 4 non-zeros.
Sets: 0 GUB, 0 SOS.

最適解を
得るまでの
過程が
書かれている

Using DUAL simplex for phase 1 and PRIMAL simplex for phase 2.
The primal and dual simplex pricing strategy set to 'Devex'.

Relaxed solution 3.75 after 2 iter is B&B base.

Feasible solution 0 after 3 iter, 1 nodes (gap 78.9%)
Improved solution 3 after 6 iter, 5 nodes (gap 15.8%)

+Optimal solution 3 after 7 iter, 6 nodes (gap 15.8%).
Excellent numeric accuracy ||*|| = 0

MEMO: lp_solve version 5.5.0.5 for 32 bit OS, with 64 bit REAL variables.
In the total iteration count 7, 0 (0.0%) were bound flips.
There were 3 refactorizations, 0 triggered by time and 0 by density.
... on average 2.3 major pivots per refactorization.
The largest [LUSOL v2.2.1.0] fact(B) had 7 NZ entries, 1.0x largest basis.
The maximum B&B level was 4, 1.0x MIP order, 4 at the optimal solution.
The constraint matrix inf-norm is 4, with a dynamic range of 4.
... load data was 0.000 seconds, presolve used 0.016 seconds,
... 0.000 seconds in simplex solver, in total 0.016 seconds.

最適解を求めるの
に費やした時間

難しい整数計画問題の特徴

- 変数、制約の数が大きい≠難しい
- 難しい問題の特徴
  - 問題のサイズが大きい
  - 数値的に不安定➔誤差が出やすい
  - 対称性が高い➔最適解が非常に多い
  - 定式化が悪い➔LP緩和が悪い
  - IPが得意でない問題
  - 許容解が見つかりにくい、または存在しない

難しい整数計画問題への対処法1

- 高性能のハードウェアを買う
- 最新版のソルバーを買う
- 計算に多くの時間を費やす
- 定式化を工夫（これにより1000倍のスピードアップもありうる）
- IPの利用をあきらめる
  - 異なるアプローチで最適解を求める
  - 最適解をあきらめ、近似解で我慢する

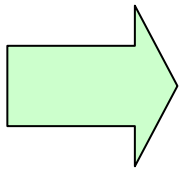
難しい整数計画問題への対処法2

- 問題のサイズが大きい
  ➔子問題への分割は可能か？
- 数値的に不安定
  ➔データの数値の桁数を減らす
- 対称性が高い
  ➔対称性を減らす工夫（制約の追加など）
- 許容解が見つかりにくい
  ➔近似解法などを利用し、許容解を求める

パズルの整数計画による定式化：数独の場合

- 数独（ナンバープレース）のルール
  - 9×9の各マスに1から9の数字を入れる
  - 各行、各列、各3×3ブロックには、1から9の数字がちょうど一回ずつ現れる

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



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

数独の定式化：変数の定義

$$x_{ijk} = \begin{cases} 1 & (\text{i 行 j 列に数字 k が入る}) \\ 0 & (\text{それ以外}) \end{cases}$$

5行4列のマスを見ると

$x_{547}=1$

$x_{541}=x_{542}=\ \ldots=0$

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

数独の定式化：制約その１

$$\sum_{k=1}^{9} x_{ijk} = 1 \ (\forall i = 1, \ldots, 9, \forall j = 1, \ldots, 9)$$

i 行 j 列のマスには、1から9のいずれかの数字が一つだけ入る

$$\sum_{j=1}^{9} x_{ijk} = 1 \ (\forall i = 1, \ldots, 9, \forall k = 1, \ldots, 9)$$

i 行のいずれかのマスに数字 k がちょうど一回現れる

$$\sum_{i=1}^{9} x_{ijk} = 1 \ (\forall j = 1, \ldots, 9, \forall k = 1, \ldots, 9)$$

j 列のいずれかのマスに数字 k がちょうど一回現れる

数独の定式化：制約その2

$$\sum_{i=s}^{s+2}\sum_{j=t}^{t+2} x_{ijk} = 1$$

$$(\forall (s,t) = (1,1),(1,4),(1,7),(4,1),(4,4),(4,7),(7,1),(7,4),(7,7),$$
$$\forall k = 1,\ldots,9)$$

各3×3のブロックに、
数字 k がちょうど一回現れる

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

赤い丸は(s,t)の
位置の候補を表す

数独の定式化：制約その3

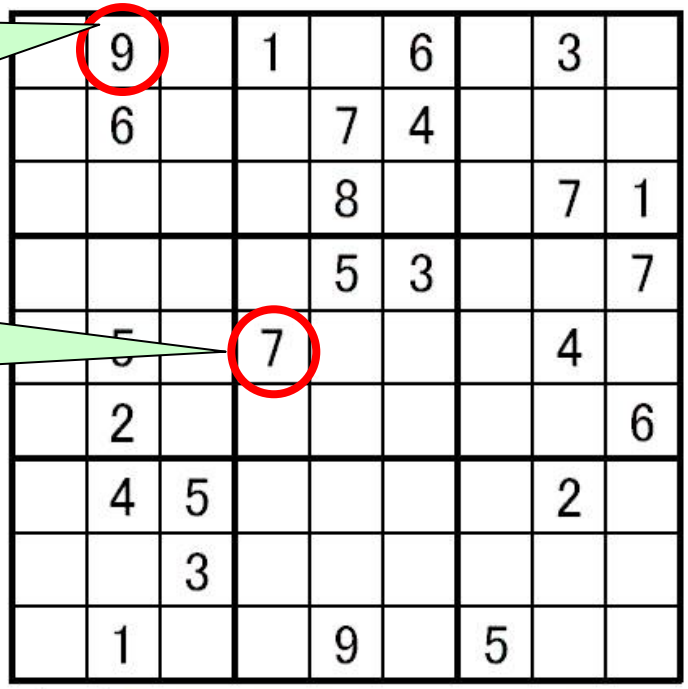
$x_{ijk} = 1$ (i行j列にすでに数字kが入っているとき)

1行2列には9が入っている

$x_{129}=1$

5行4列には7が入っている

$x_{547}=1$



数独の定式化：目的関数

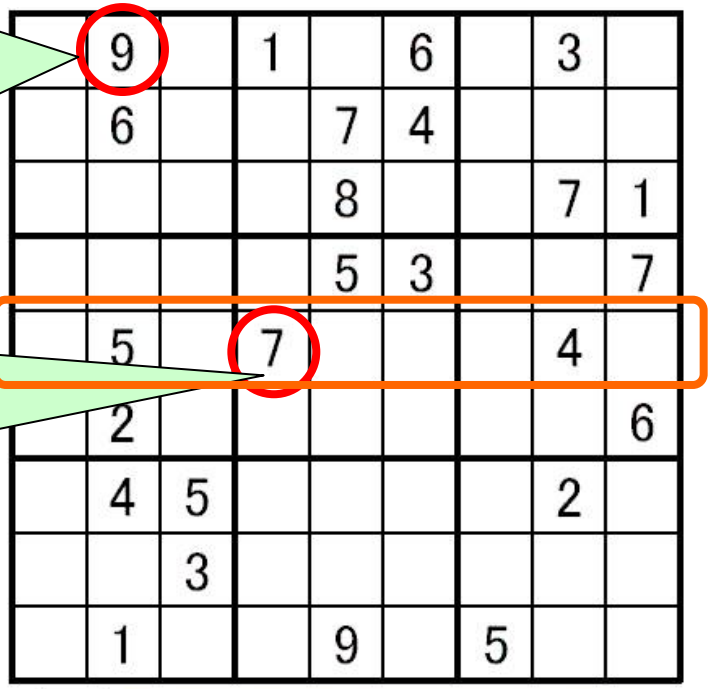
- 実は何でも良い
  - 許容解を求める問題なので
  - 普通の数独では、許容解は1つしかない
- 定数　0　としても可
- 〔検討課題〕目的関数を変化させると計算時間はどう変わるか？

数独の定式化：高速化のための工夫

- 固定可能な変数は事前に固定する

1行2列には9が入っている
➔１行２列に9以外の数字は入らない
➔$x_{12k}=0$ $(k \neq 9)$

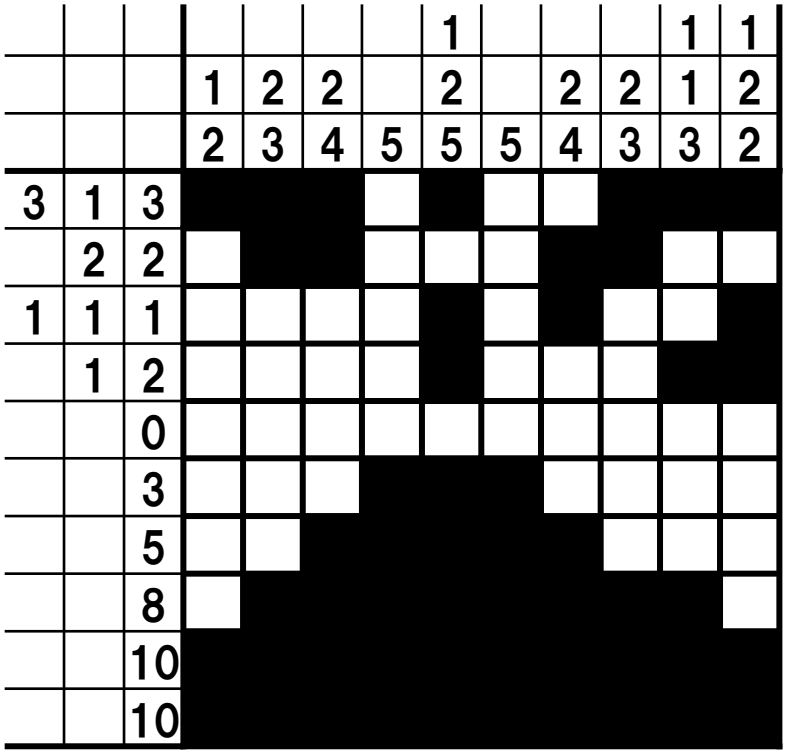
5行4列には7が入っている
➔5行の他のマスには7は入らない
➔$x_{5j7}=0$ $(j \neq 4)$



パズルの整数計画による定式化：ののぐらむの場合

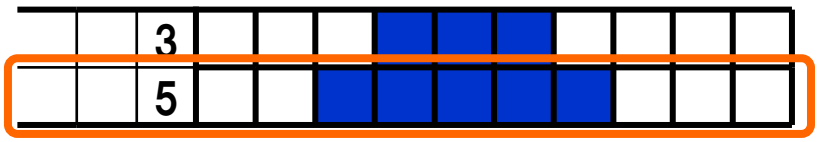
- ののぐらむ（イラストロジック）のルール
  - マス目の一部を黒く塗りつぶす

							1				1	1
			1	2	2		2		2	2	1	2
			2	3	4	5	5	5	4	3	3	2
3	1	3										
	2	2										
1	1	1										
	1	2										
		0										
		3										
		5										
		8										
		10										
		10										



パズルの整数計画による定式化：ののぐらむの場合

- ののぐらむ（イラストロジック）のルール
  - マス目の左と上の数字は，対応する行と列において連続して塗りつぶすマスの数を表す



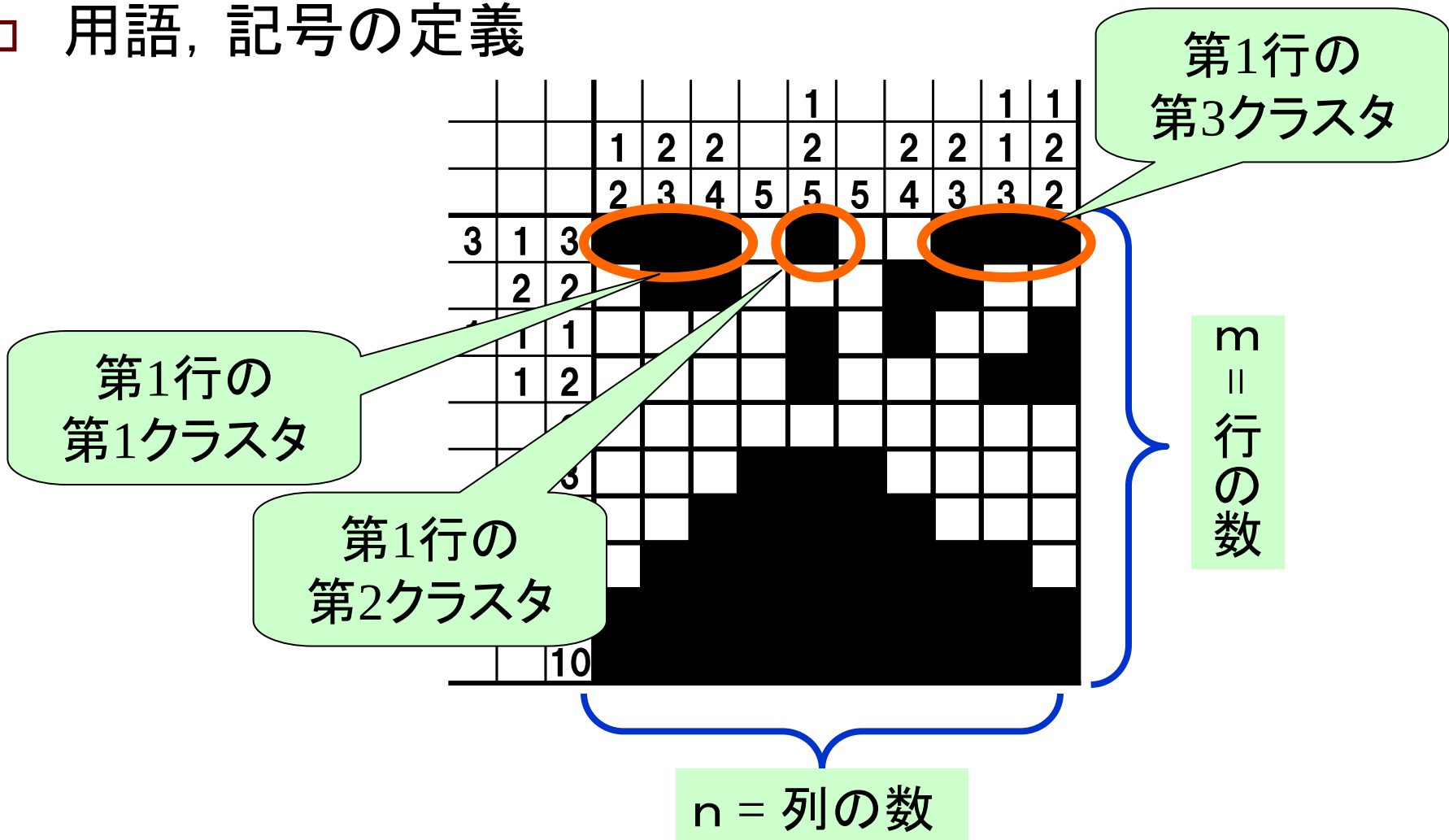
5個のマスを連続して黒く塗る

3個の黒マス，白マス1個以上，
1個の黒マス，白マス1個以上，
3個の黒マス，の順に塗る

  - 複数の数字が並んでいる場合は，並んでいる順番通りに，書いてある数字の分だけ連続してマスを塗りつぶす
  - 数字と数字の間には，1マス以上の白マスを入れる

ののぐらむの定式化：準備

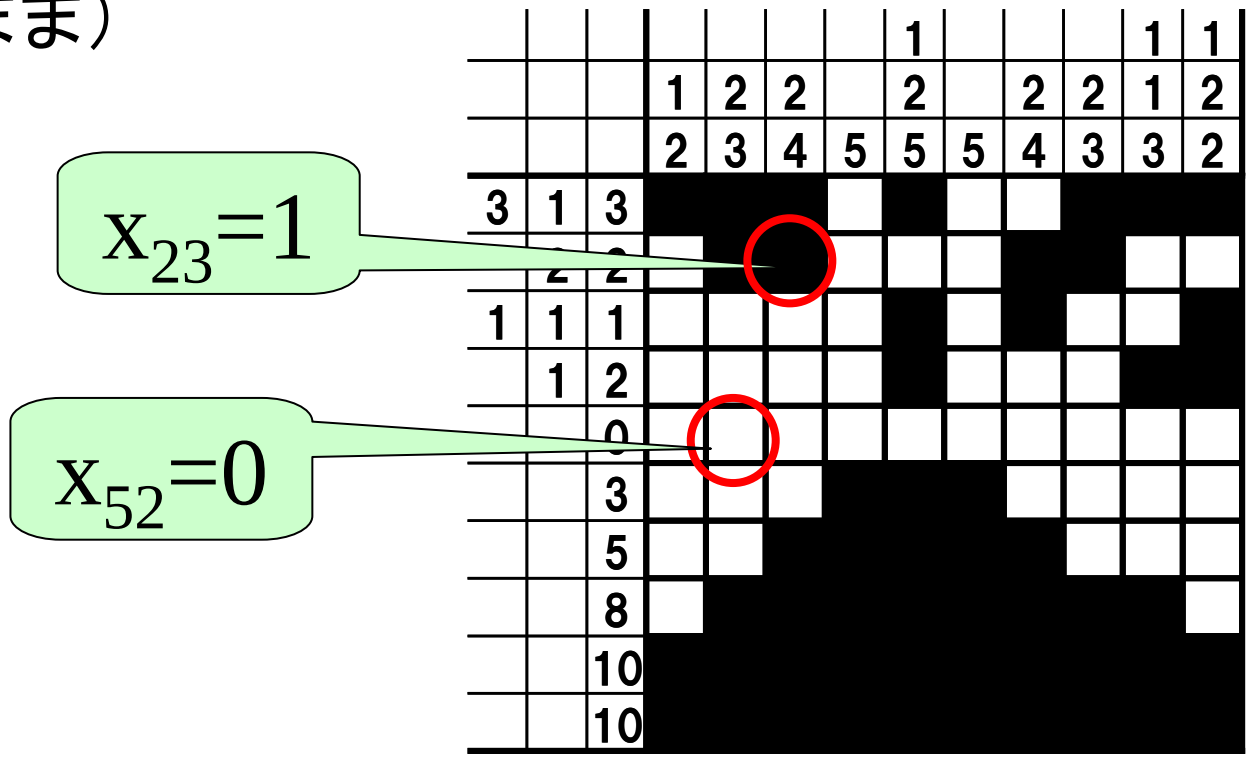
- 用語，記号の定義



ののぐらむの定式化：変数の定義その1

- Bosch (2000)による定式化

$$x_{ij} = \begin{cases} 1 & (\text{i 行 j 列を黒く塗る}) \\ 0 & (\text{白のまま}) \end{cases}$$



ののぐらむの定式化：変数の定義その2

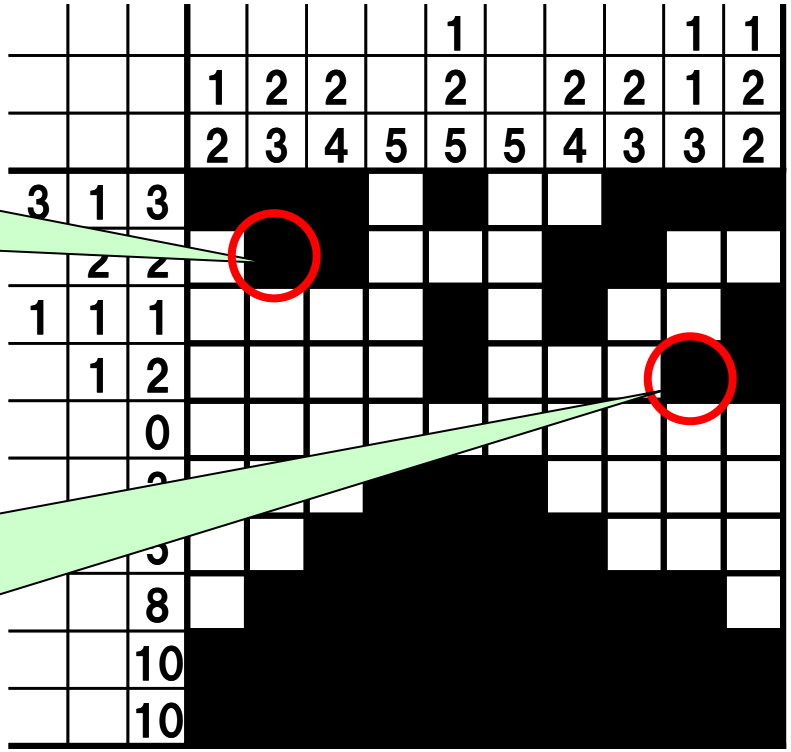
$$y_{ijk}=\begin{cases}1\ (\text{第 i 行の第 k クラスタは第 j 列より始まる})\\ 0\ (\text{それ以外})\end{cases}$$

第2行の第1クラスタは第2列より始まる

$y_{221}=1$

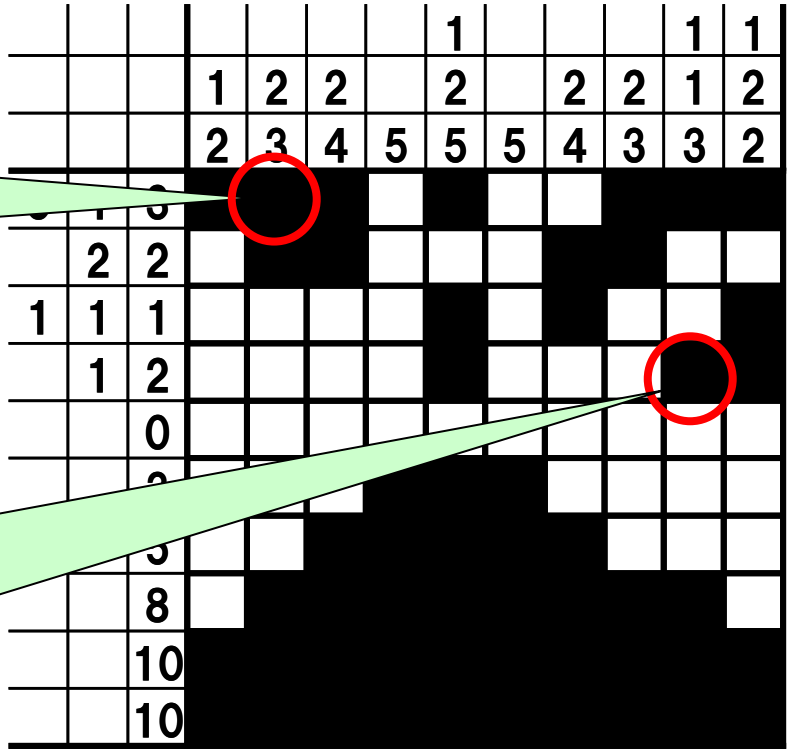
第4行の第2クラスタは第9列より始まる

$y_{492}=1$



ののぐらむの定式化：変数の定義その3

$$z_{ijk} = \begin{cases} 1 & \text{（第 j 列の第 k クラスタは第 i 行より始まる）} \\ 0 & \text{（それ以外）} \end{cases}$$



第2列の第1クラスタは第1行より始まる

$y_{121}=1$

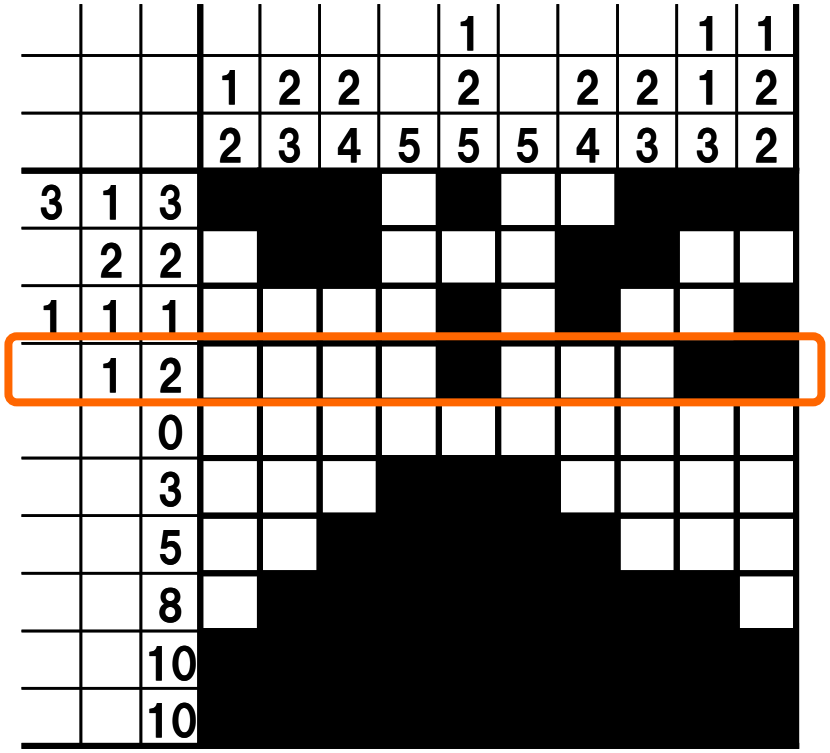
第9列の第2クラスタは第4行より始まる

$y_{492}=1$

ののぐらむの定式化：制約その1

第 i 行において，第 k クラスタが現れるのはちょうど1回

第 j 列において，第 k クラスタが現れるのはちょうど1回



例：第4行において，第2クラスタが現れるのはちょうど1回

$$y_{4,1,2} + y_{4,2,2} + y_{4,3,2} + \cdots + y_{4,8,2} + y_{4,9,2} + y_{4,10,2} = 1$$

ののぐらむの定式化：制約その2

第 i 行において，第 k クラスタは第 (k+1) クラスタの左に位置し，その間には1つ以上の白マスが存在

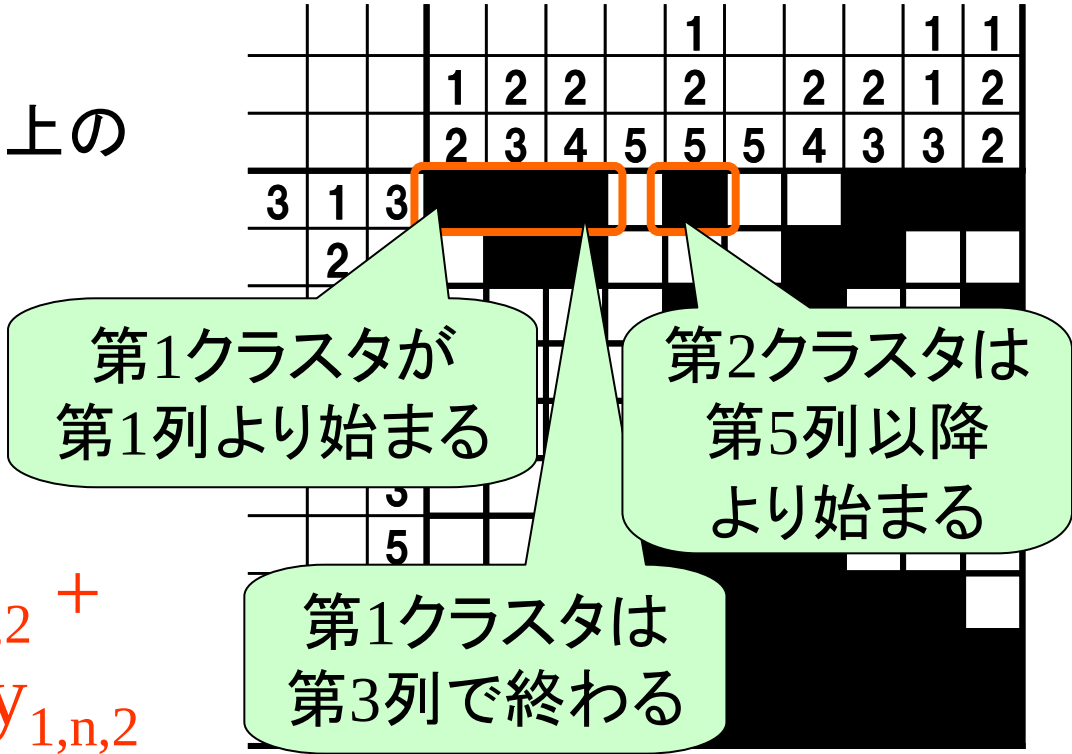
（各列に対しても同様の制約が必要）

例：第1行において，
第2クラスタは第3クラスタの
左に位置し，その間には1つ以上の
白マスが存在

➔ $y_{1,j,1} = 1$ ならば，

$y_{1,j+4,2}$, $y_{1,j+5,2}$, …$y_{1,n,2}$
のいずれかは必ず 1

➔ $y_{1,j,1} \leqq y_{1,j+4,2} + y_{1,j+5,2} +$
… $+y_{1,n,2}$



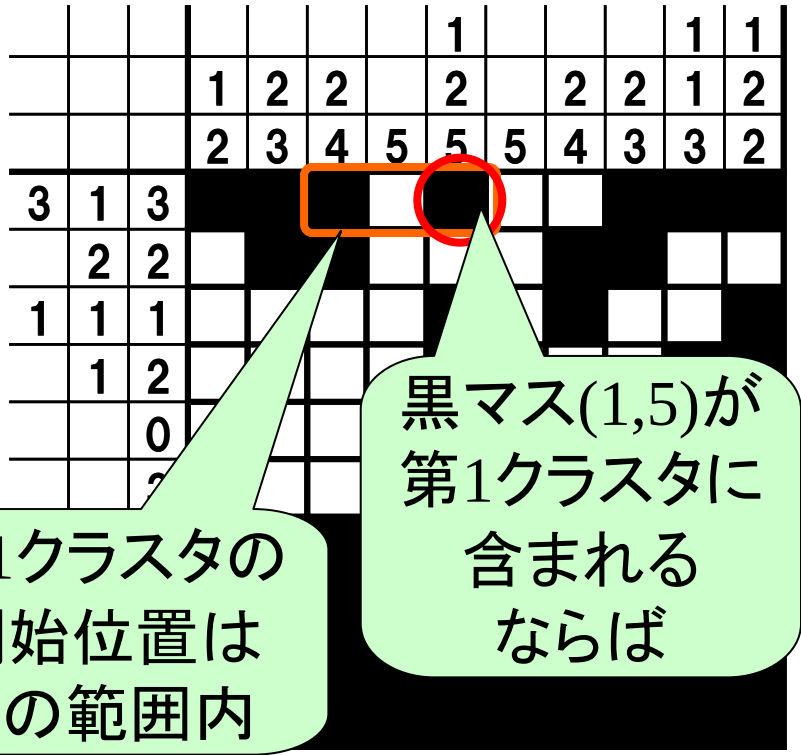
ののぐらむの定式化：制約その3

マス (i, j) が黒ならば，それは第 i 行（第 j 列）のいずれかのクラスタに含まれる

例：
黒マス(1,j)が第1クラスタに含まれる
➔ (j + 1)-3 ≦第1クラスタの開始列≦j
➔

$$x_{1,j} \leqq y_{1,j-2,1} + y_{1,j-1,1} + y_{1,j,1}$$

（第2クラスタ，第3クラスタに含まれる場合も同様の不等式が成立）



ののぐらむの定式化：制約その3（続き）

マス (i, j) が黒ならば，それは第 i 行（第 j 列）のいずれかのクラスタに含まれる

例：
黒マス(1,j)は3つのクラスタのいずれかに含まれる

第1クラスタに含まれる場合，この項＝1

$$x_{1,j} \leqq (y_{1,j-2,1} + y_{1,j-1,1} + y_{1,j,1}) + y_{1,j,2} + (y_{1,j-2,3} + y_{1,j-1,3} + y_{1,j,3})$$

第2クラスタに含まれる場合，この項＝1

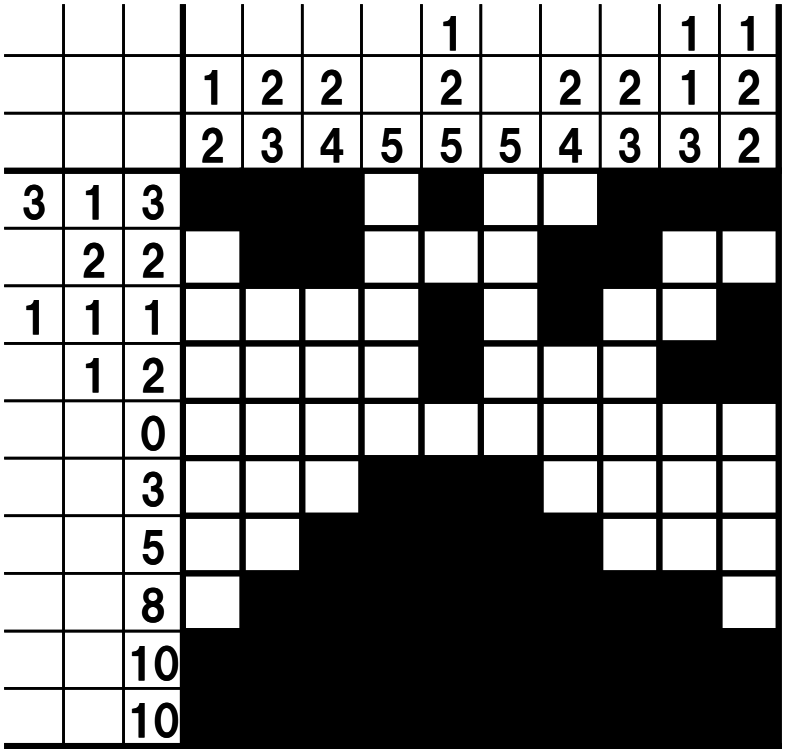
第3クラスタに含まれる場合，この項＝1

ののぐらむの定式化：制約その4

マス (i, j) が白ならば，それは第 i 行（第 j 列）のいずれのクラスタの開始位置にもなり得ない

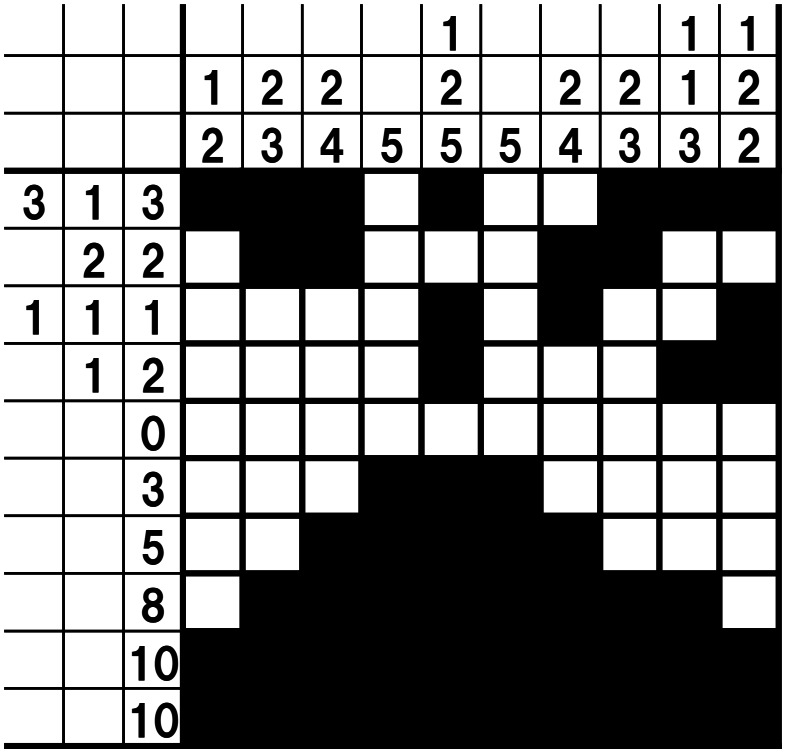
例：
(1,j)が白マス
➔ 第kクラスタの開始列≠ j
➔

$$x_{1,j} \geqq y_{1,j,k}$$



ののぐらむの定式化:補足

- この他の定式化も存在
  - 妥当不等式として他の制約式を追加してもOK
  - [検討事項]妥当不等式を追加したときの計算時間の変化は？
- 目的関数は任意の線形関数でOK



パズルの整数計画による定式化：ののぐらむの場合

- ののぐらむ（イラストロジック）のルール
  - マス目の一部を黒く塗りつぶす

							1				1	1
			1	2	2		2		2	2	1	2
			2	3	4	5	5	5	4	3	3	2
3	1	3										
	2	2										
1	1	1										
	1	2										
		0										
		3										
		5										
		8										
		10										
		10