整数計画法を用いた Pearl Puzzleの効率的な解法

2016年3月7日

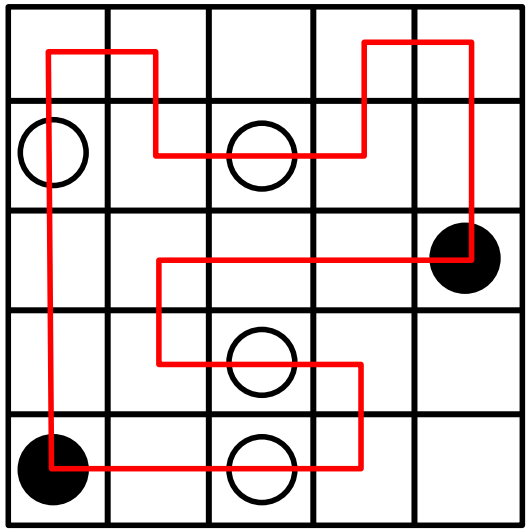
第11回組合せゲーム・パズル研究集会

大阪電気通信大学

○貴宮 京一　鈴木 裕章　上嶋 章宏

目次

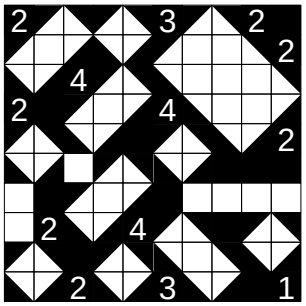
- 研究背景
  - 整数計画法や先行研究
- Pearl Puzzle
  - ルールと定義
- 定式化
  - 2つの既存手法と提案手法
- 評価
  - 計測結果と考察
- まとめと今後の課題

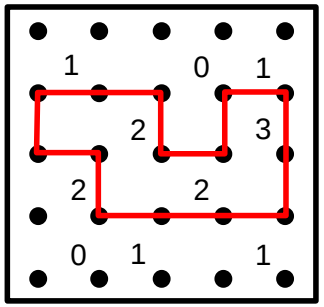


研究背景: 汎用ソルバの適用

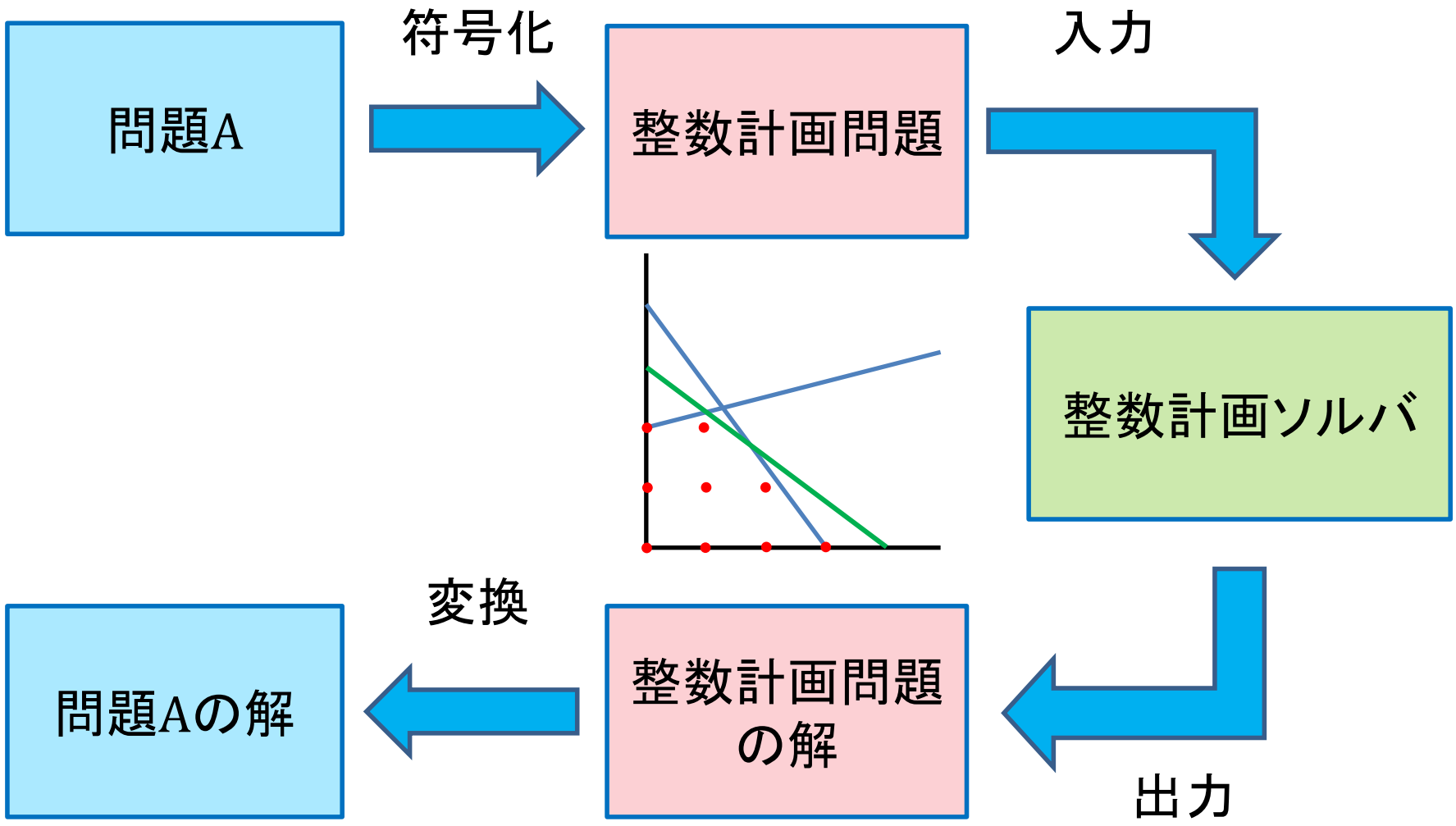
- 幅広い分野で汎用ソルバを使用した解法
- 汎用ソルバ利用の利点･欠点
  - 問題記述を与えることで，各問題に適用可能
  - 問題依存のアルゴリズムに比べ，細かなプログラム制御に不向き
  - 多くの研究成果が利用可能で，費用対効果が高い
  - 定式化の工夫が計算時間に影響を与える可能性
- 汎用ソルバを用いたパズルの高速解法

  シャカシャカ [岡本，2013]　Slitherlink [石濱，久野，2013]

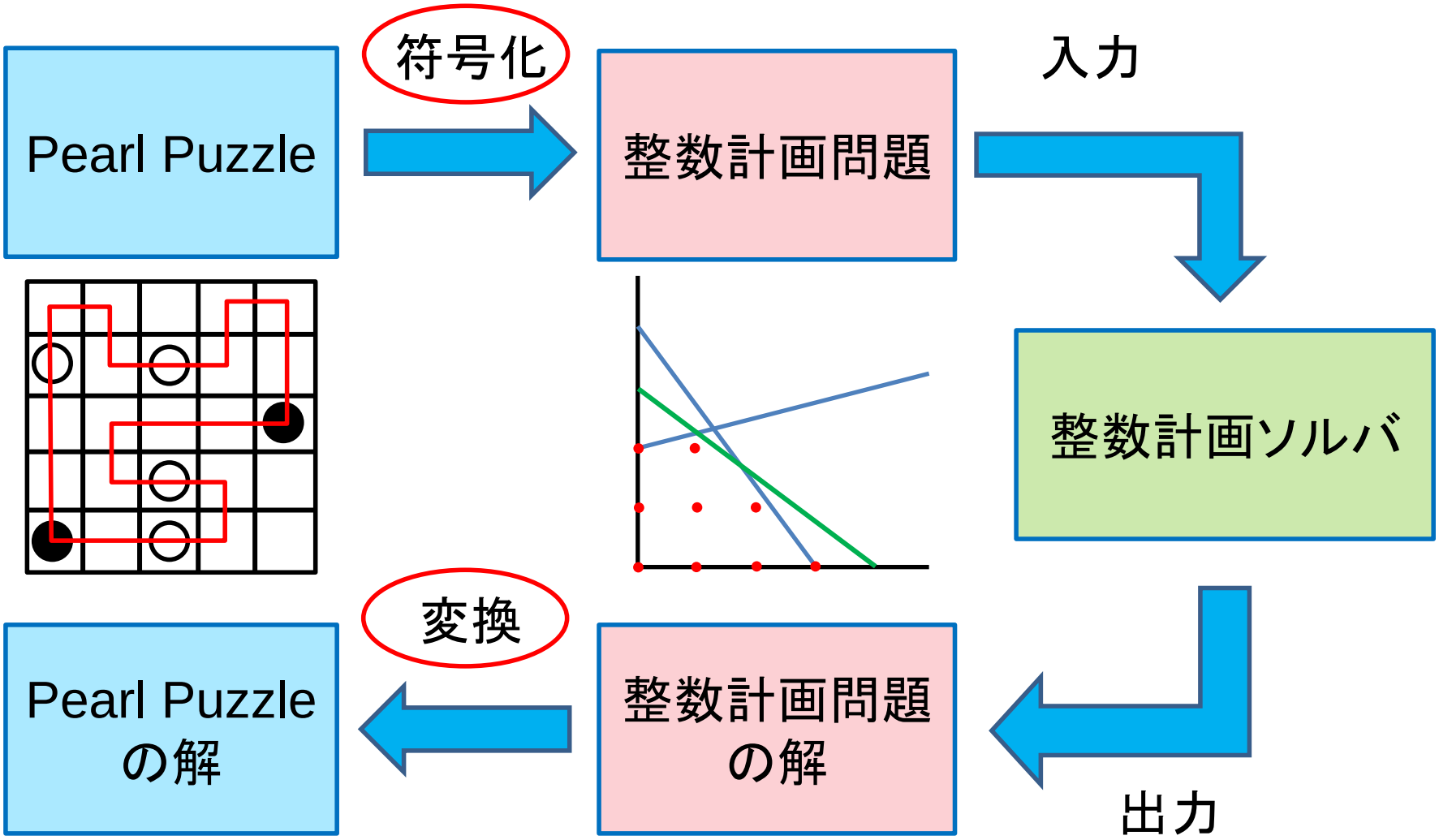




研究背景：整数計画法を用いた解法



研究背景：整数計画法を用いた解法



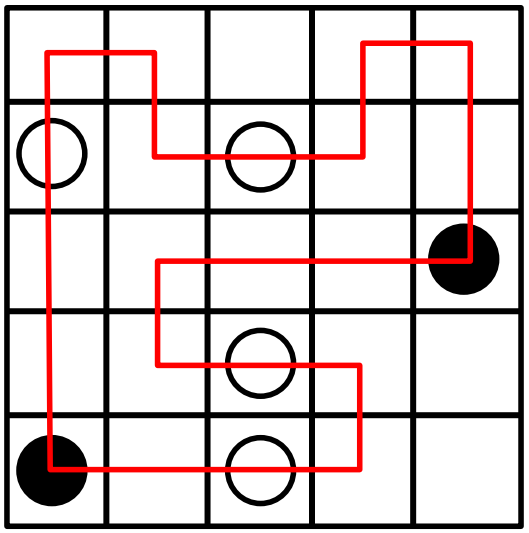
研究背景: Pearl Puzzleの計算複雑さ

- Pearl PuzzleのNP完全性 [Erich Friedman, 2002]
- Pearl PuzzleのASP完全性 [浴本, 2016] New!
  - 各点の次数が3以下の平面グラフでのハミルトン閉路問題から還元

Pearl Puzzle

NP完全

問題の効率的な解法が困難



ASP完全

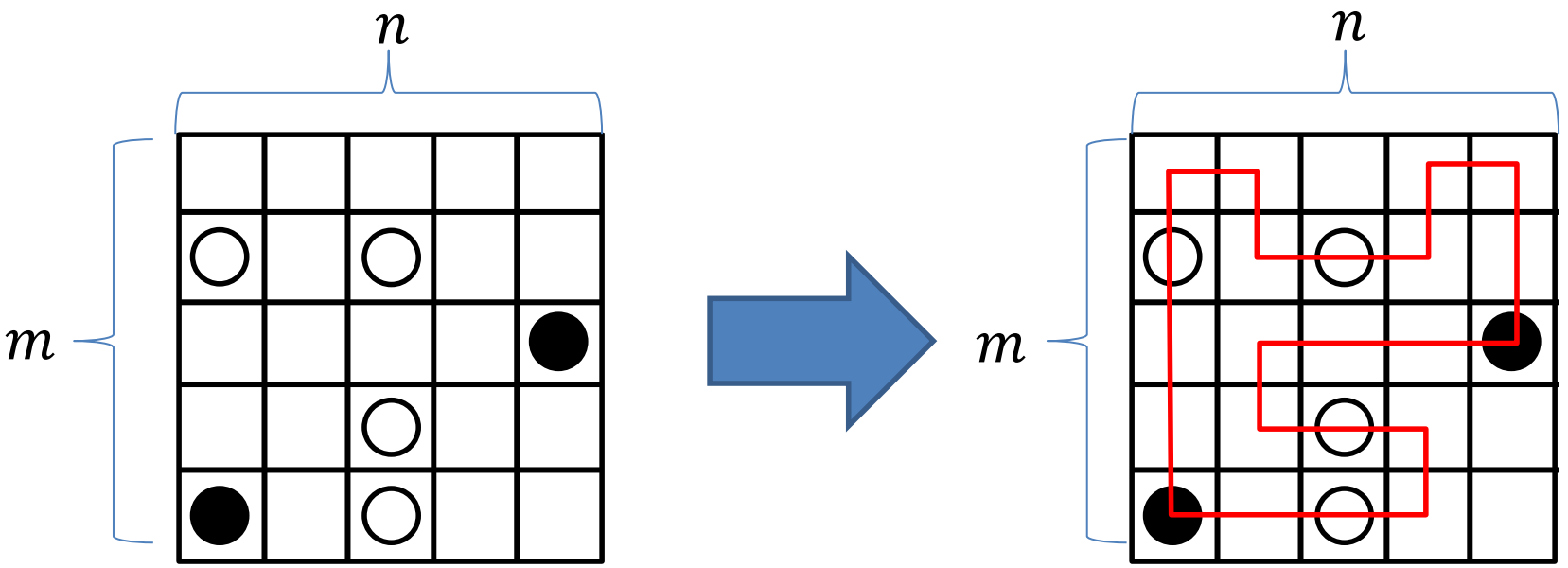
想定解以外の解の効率的な発見が困難

Pearl Puzzle（ましゅ）

- 株式会社ニコリ提供のペンシルパズル

- 入力：○と●が複数配置された$m \times n$の盤面

- 出力：ルールを満たしすべての○と●上を通る1つの輪

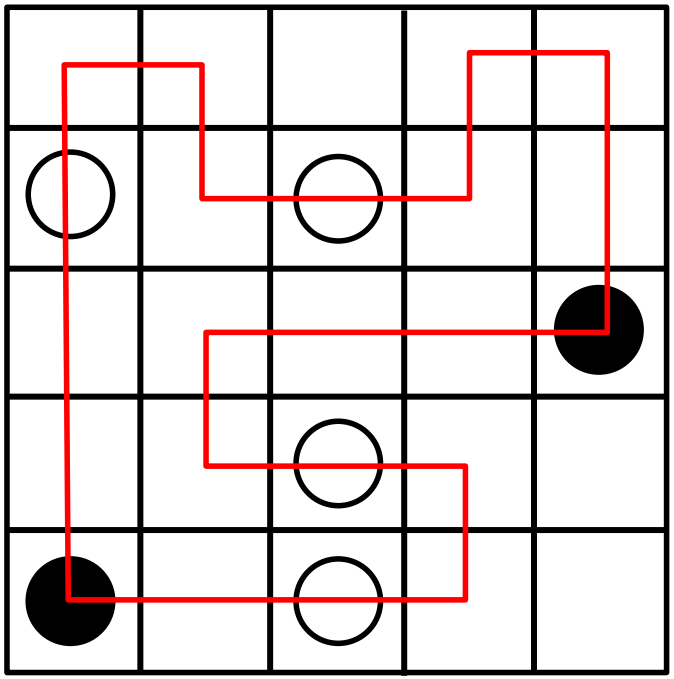


Pearl Puzzleのルール

交差，枝分かれ，
複数の輪は×

○上は直進

両隣のマスで片方は
曲がる



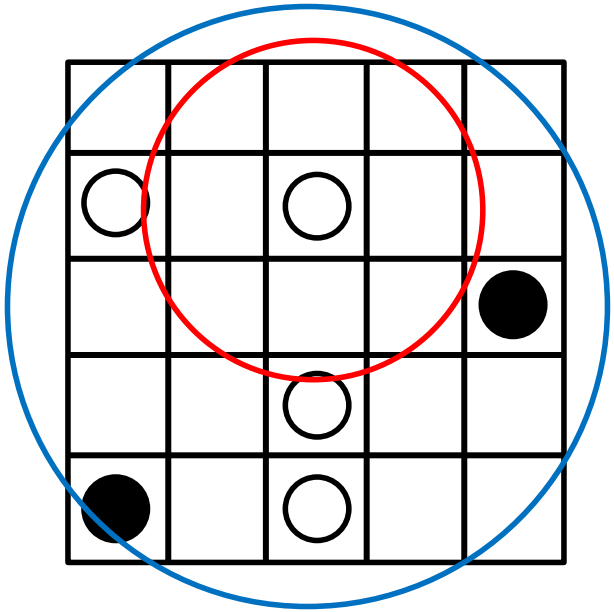
●上は曲がる

両隣のマスは直進

Pearl Puzzleの定式化

- 各ルールを定式化する
- 規模が大きくなる「複数輪の禁止」制約に注目

〇のルールに
必要な情報



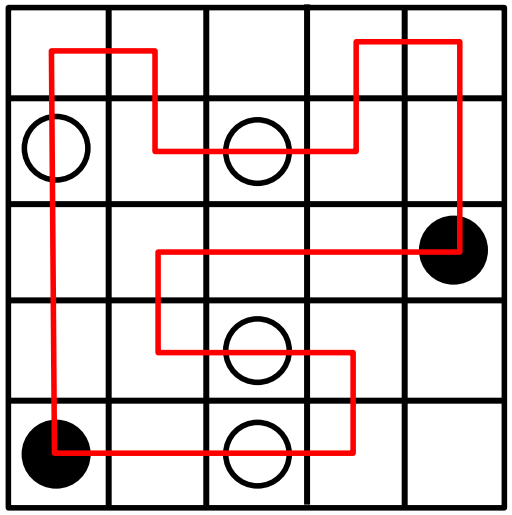
「複数輪の禁止」に
必要な情報

Pearl Puzzleの定式化

Slitherlinkの高速解法
[石濱，久野，2013]

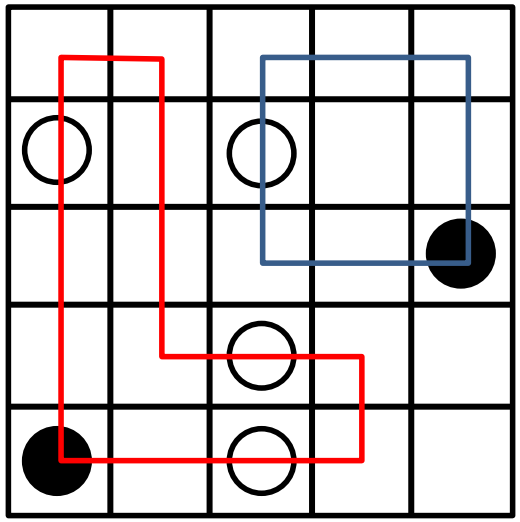
2つの既存手法

番号付により
1つの輪を与える手法

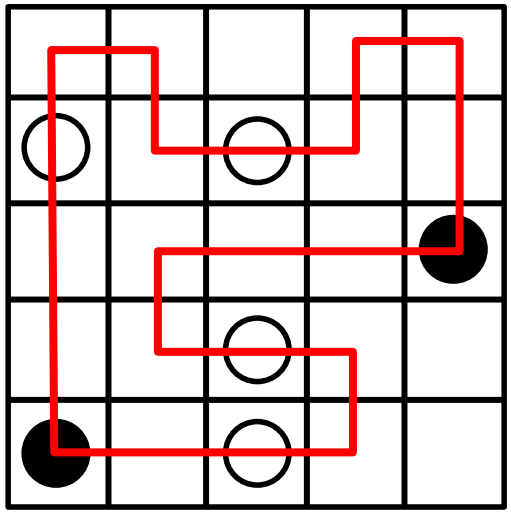


提案手法

暫定解の情報から
制約式を追加する手法

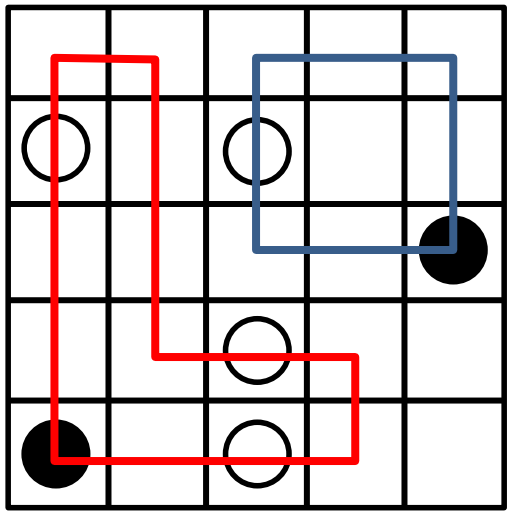


Pearl Puzzleの定式化：番号付



2	3	0	7	8
1	4	5	6	9
22	13	12	11	10
21	14	15	16	0
20	19	18	17	0

出発点から昇順になるように構成

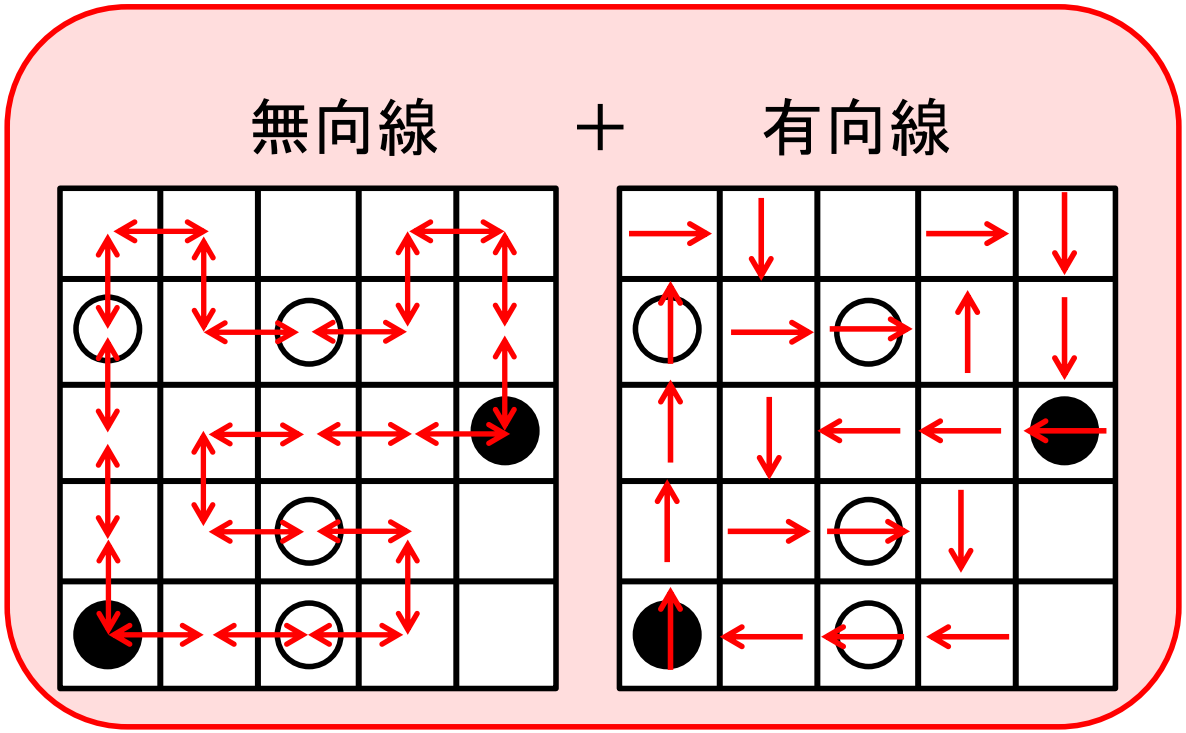


2	3	15	16	17
1	4	22	0	18
14	5	21	20	19
13	6	7	8	0
12	11	10	9	0

制約に反している

Pearl Puzzleの定式化：番号付

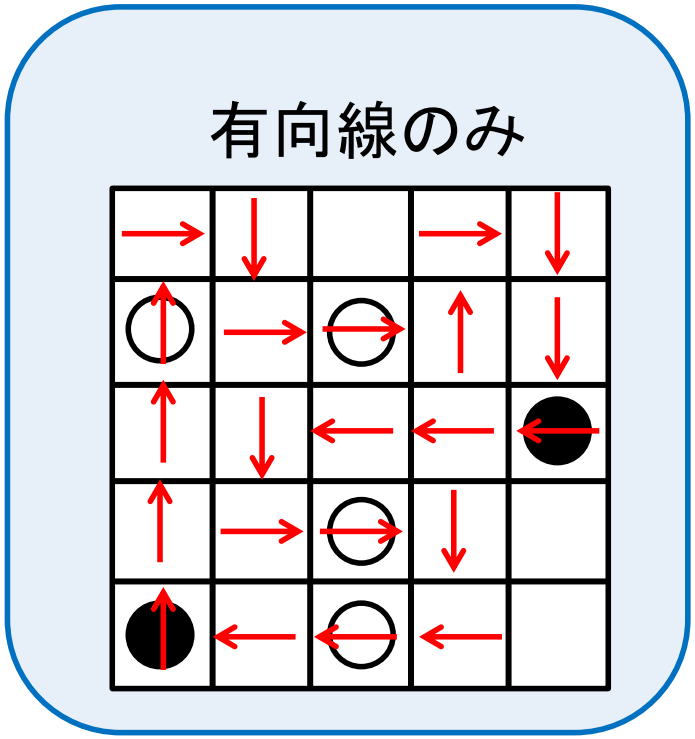
- 2つの既存手法を参考に構築
- 線が通過する方向を変数で表現



http://www21.tok2.com/home/kainaga11/glpkPearl/GLPKpearl.htm

Pearl Puzzleの定式化：番号付

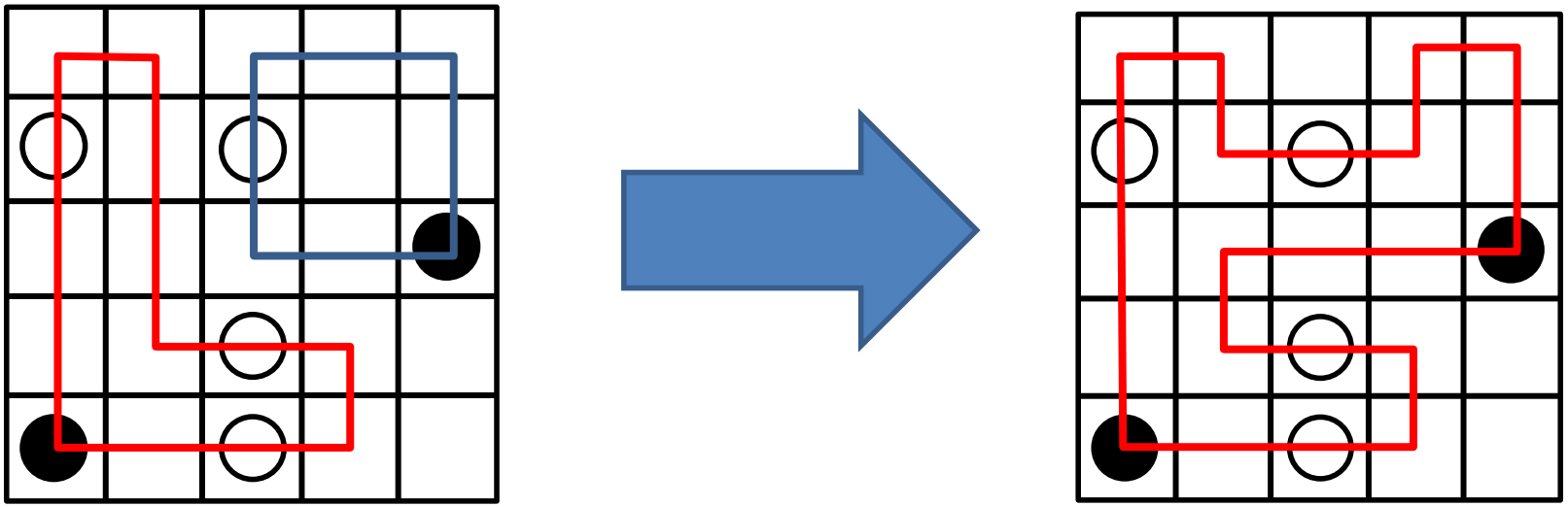
- 2つの既存手法を参考に構築
- 線が通過する方向を変数で表現



http://wavefrontip.la.coocan.jp/ましゅ.pdf

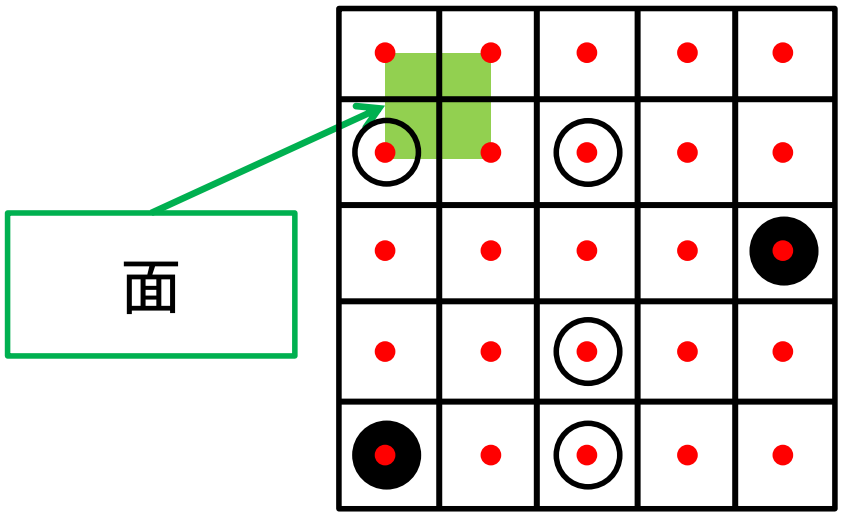
提案手法

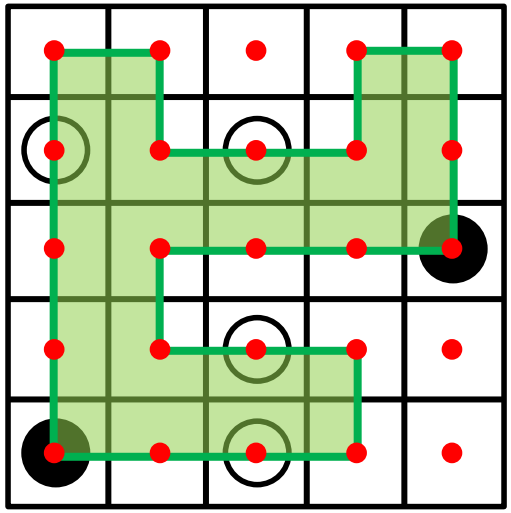
- 暫定解から制約式を追加する手法
- 複数の輪の解から同じ輪を生成しない制約を追加
- 輪が1つになるまで繰り返す



提案手法

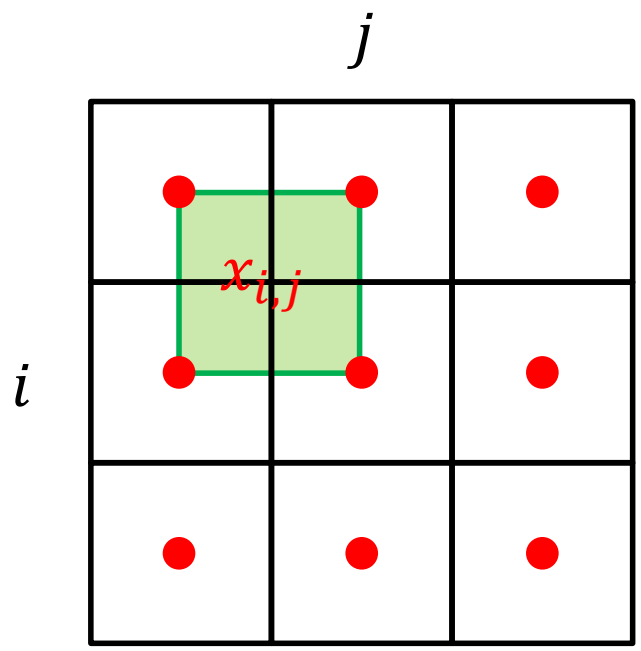
- Slitherlinkの先行研究に基づき構築 [石濱，久野，2013]
- 線ではなく面に注目
- 面が輪の内側か外側かを定める

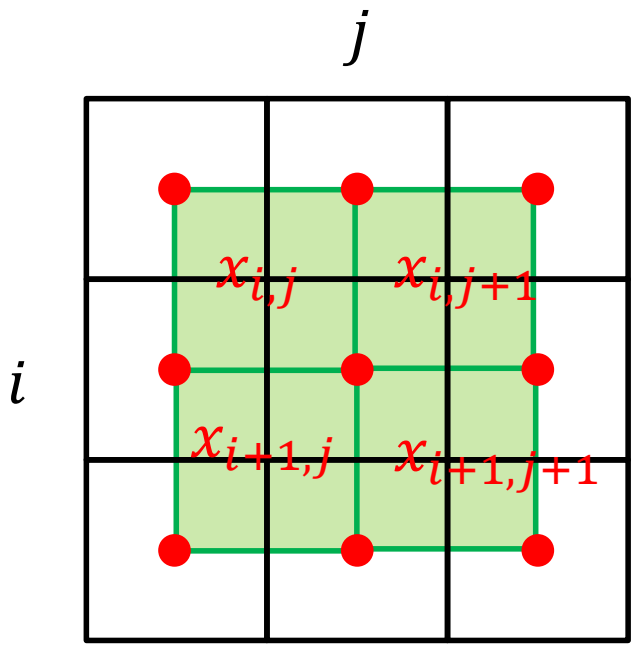




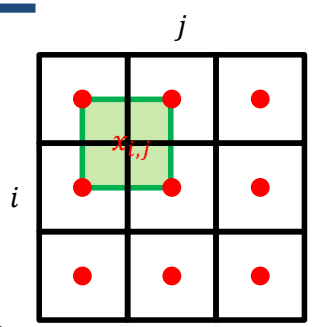
提案手法：変数

- 元盤面のマスの左上面をx_{ij}
- $x_{i,j} \in \{0,1\}$



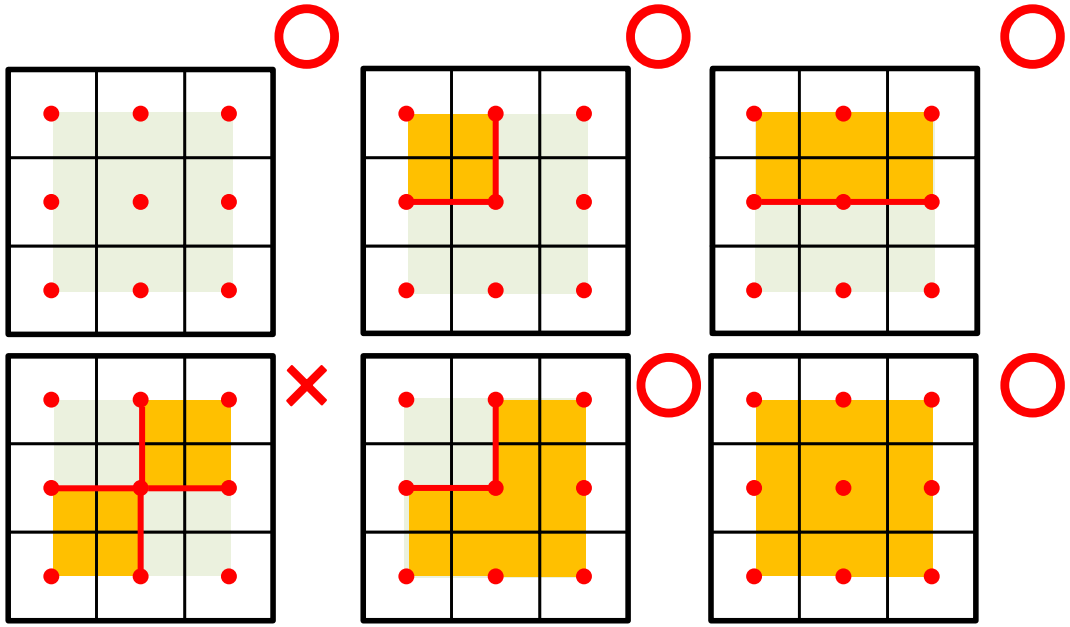


提案手法：枝分かれ･交差の禁止



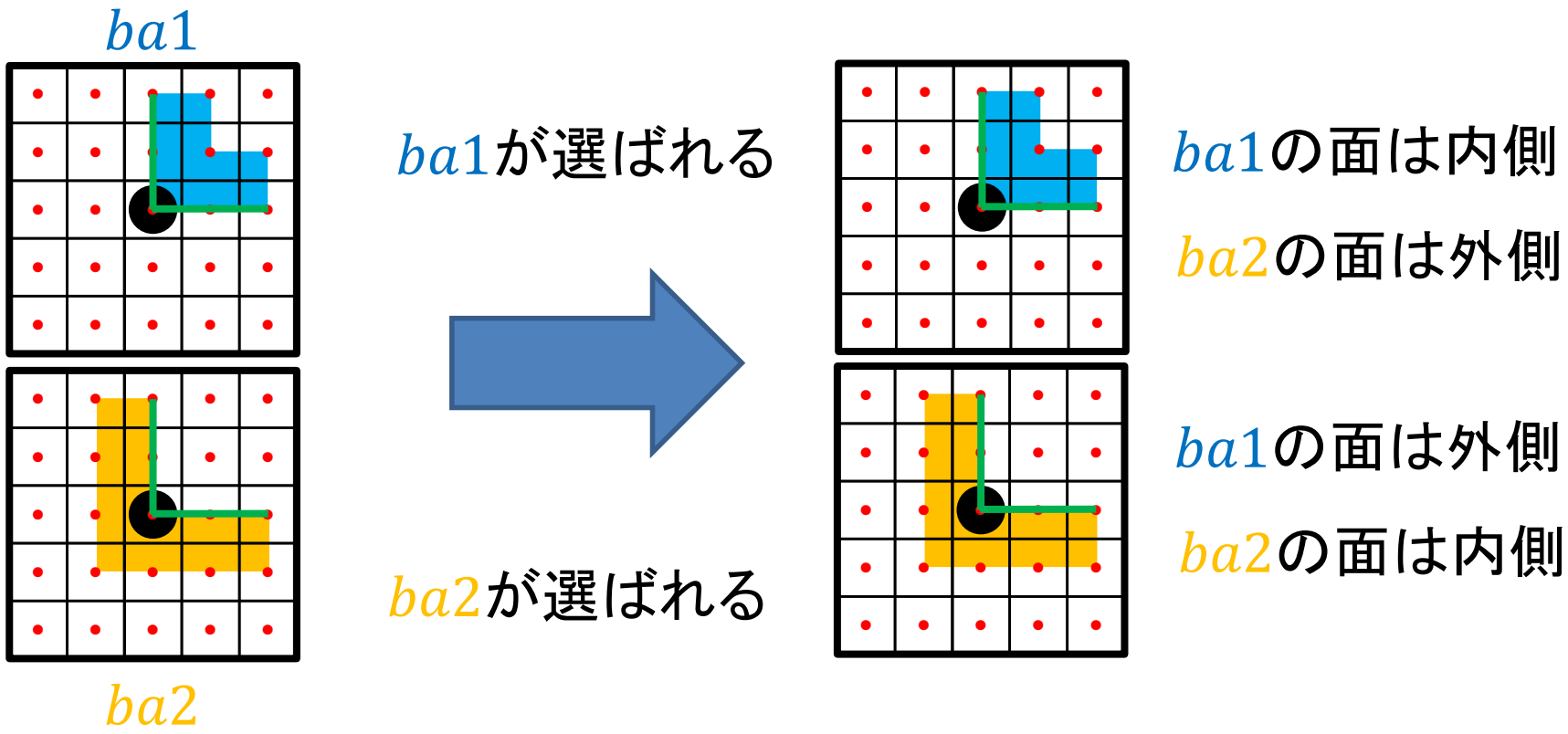
- 点を共有する4面の組合せは6つ
- 線が交差する組合せは除外

$$x_{i,j} + (1 - x_{i+1,j}) + (1 - x_{i,j+1}) + x_{i+1,j+1} \leq 3$$
$$x_{i,j} + (1 - x_{i+1,j}) + (1 - x_{i,j+1}) + x_{i+1,j+1} \geq 1$$



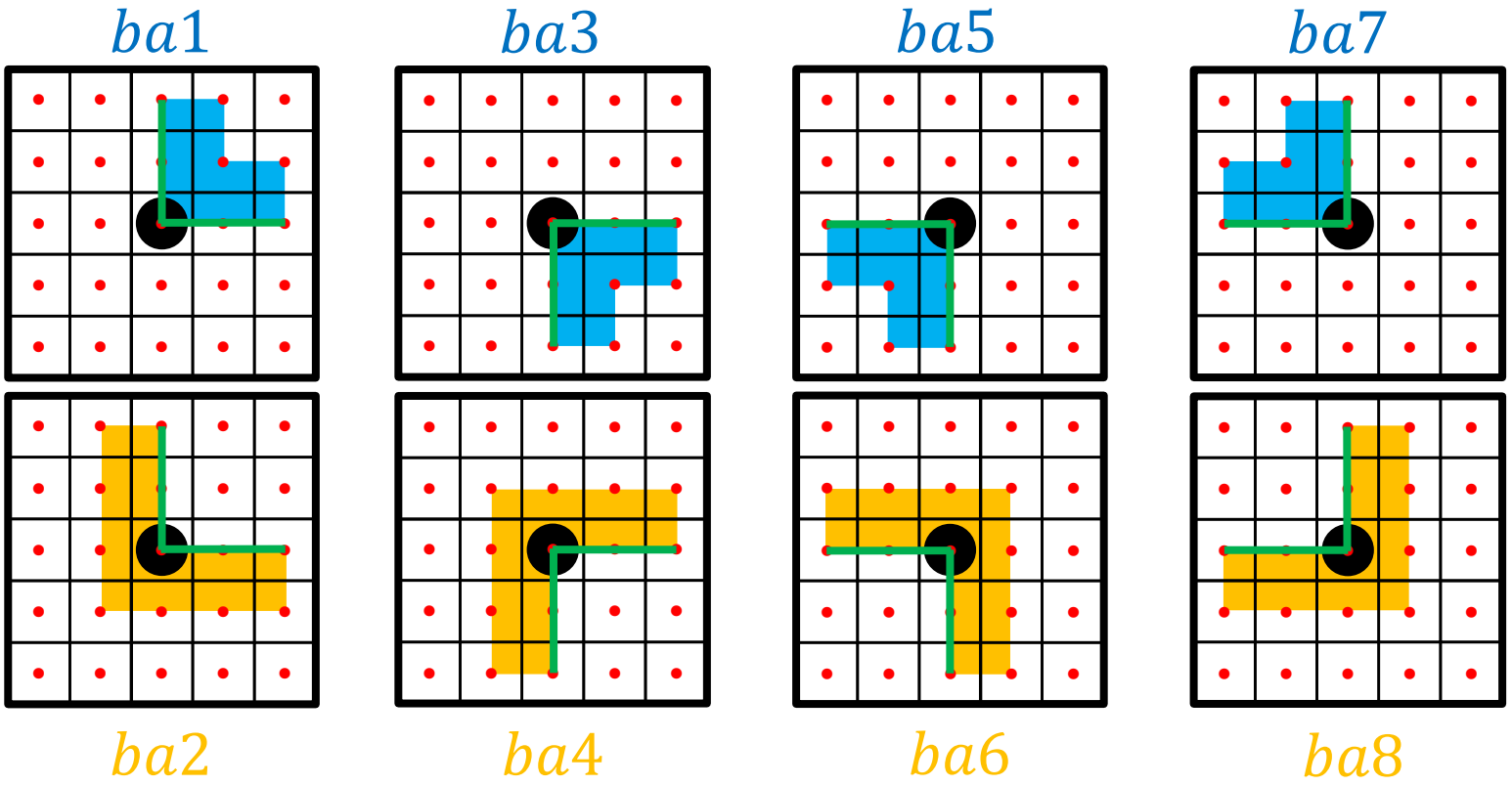
提案手法：●のルール

- 線の引き方に対応する面が内側かを選ぶ変数



提案手法：●のルール

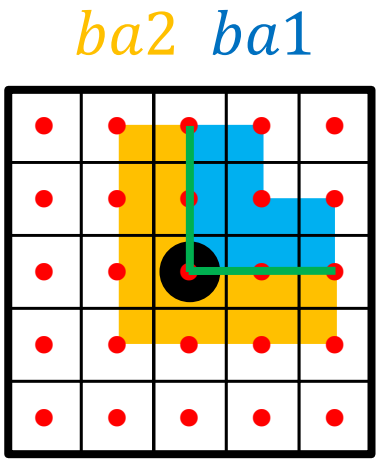
- 線の引き方に対応する面が内側かを選ぶ変数
- すべての変数から1つを選ぶ



制約式 $ba1 + ba2 + ba3 + ba4 + ba5 + ba6 + ba7 + ba8 = 1$

提案手法：●のルール

- 選ばれた変数の面は内側，反対側の変数の面は外側



$$3 \times ba1 - x_{i-1,j+1} - x_{i,j+1} - x_{i,j+2} \leq 0$$

$ba1 = 1$：青の3マスを内側にする

$$3 \times ba2 + x_{i-1,j+1} + x_{i,j+1} + x_{i,j+2} \leq 3$$

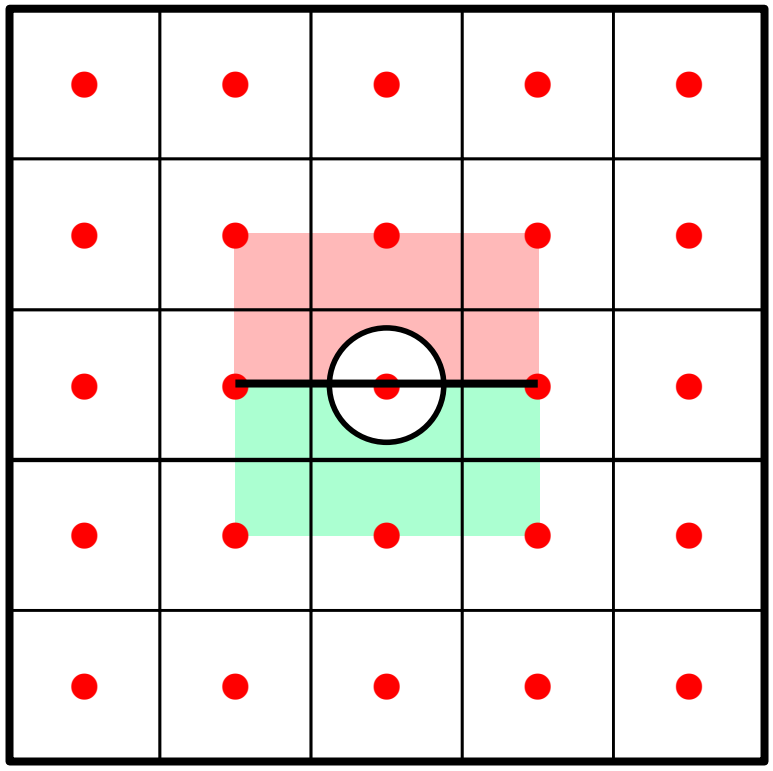
$ba2 = 1$：青の3マスを外側にする

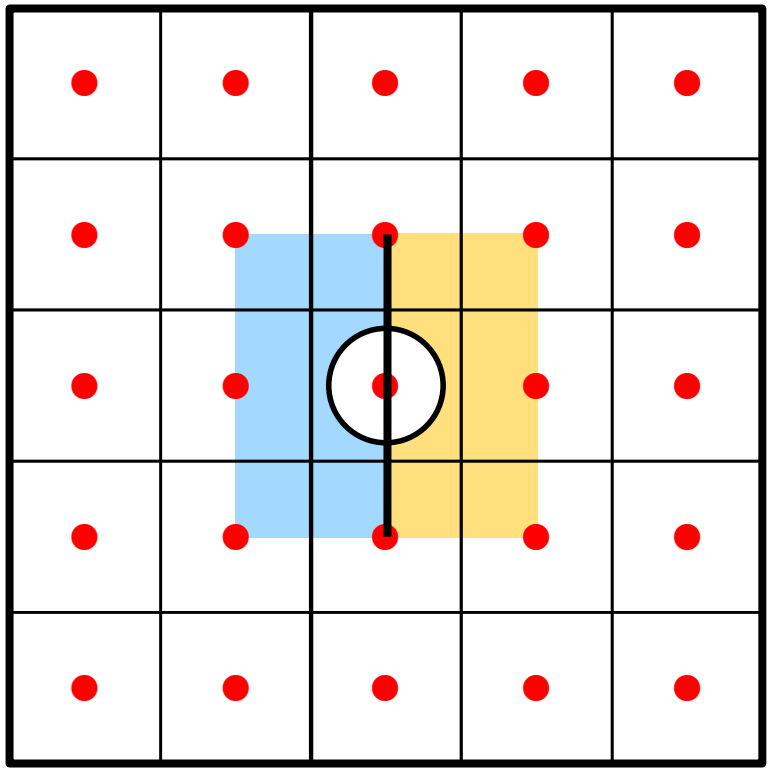
$$5 \times ba2 - x_{i,j} - x_{i-1,j} - x_{i+1,j} - x_{i+1,j+1} - x_{i+1,j+2} \leq 0$$

$$5 \times ba1 + x_{i,j} + x_{i-1,j} + x_{i+1,j} + x_{i+1,j+1} + x_{i+1,j+2} \leq 5$$

提案手法：〇のルール

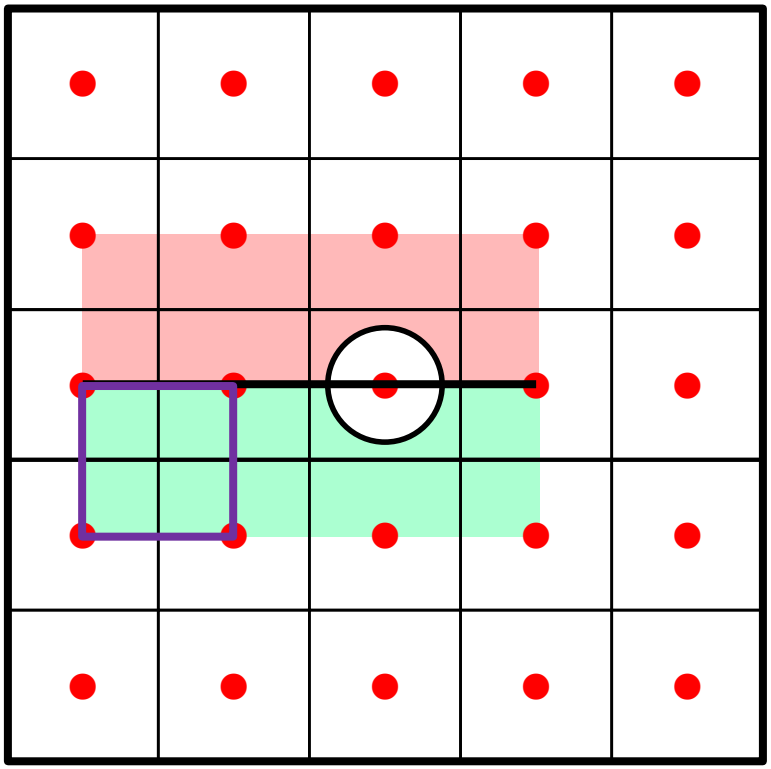
- ●と同じく内側になる面を決める

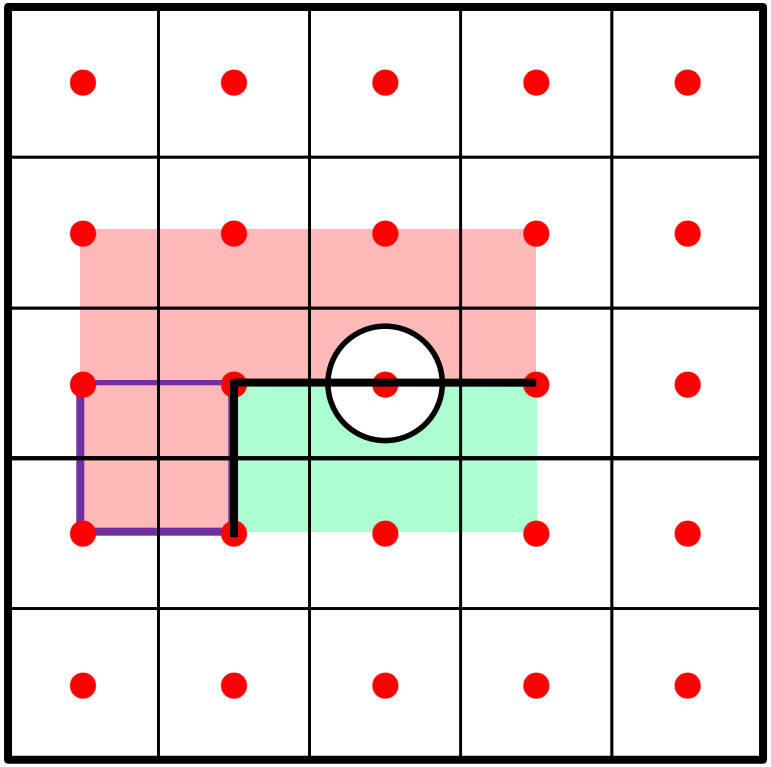




提案手法：〇のルール

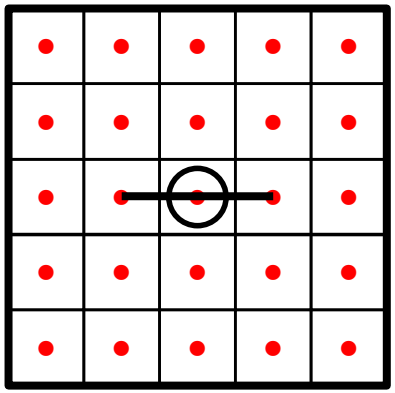
- ●と同じく内側になる面を決める
- 線の引き方によって変わる面がある





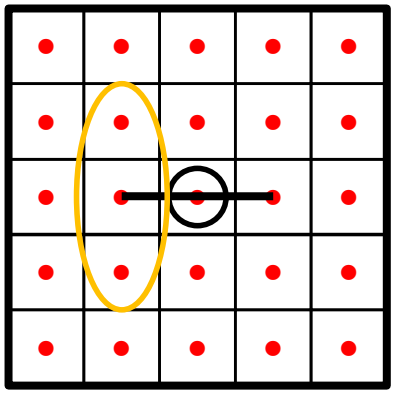
提案手法：〇のルール

- 線の引き方に関する変数を用意する



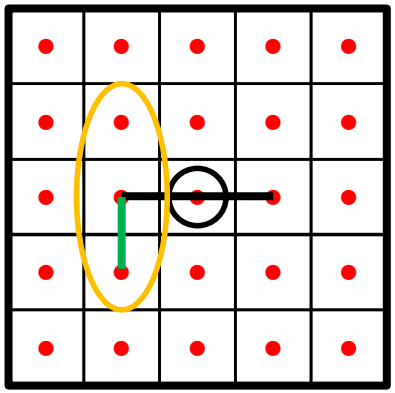
提案手法：〇のルール

- 線の引き方に関する変数を用意する



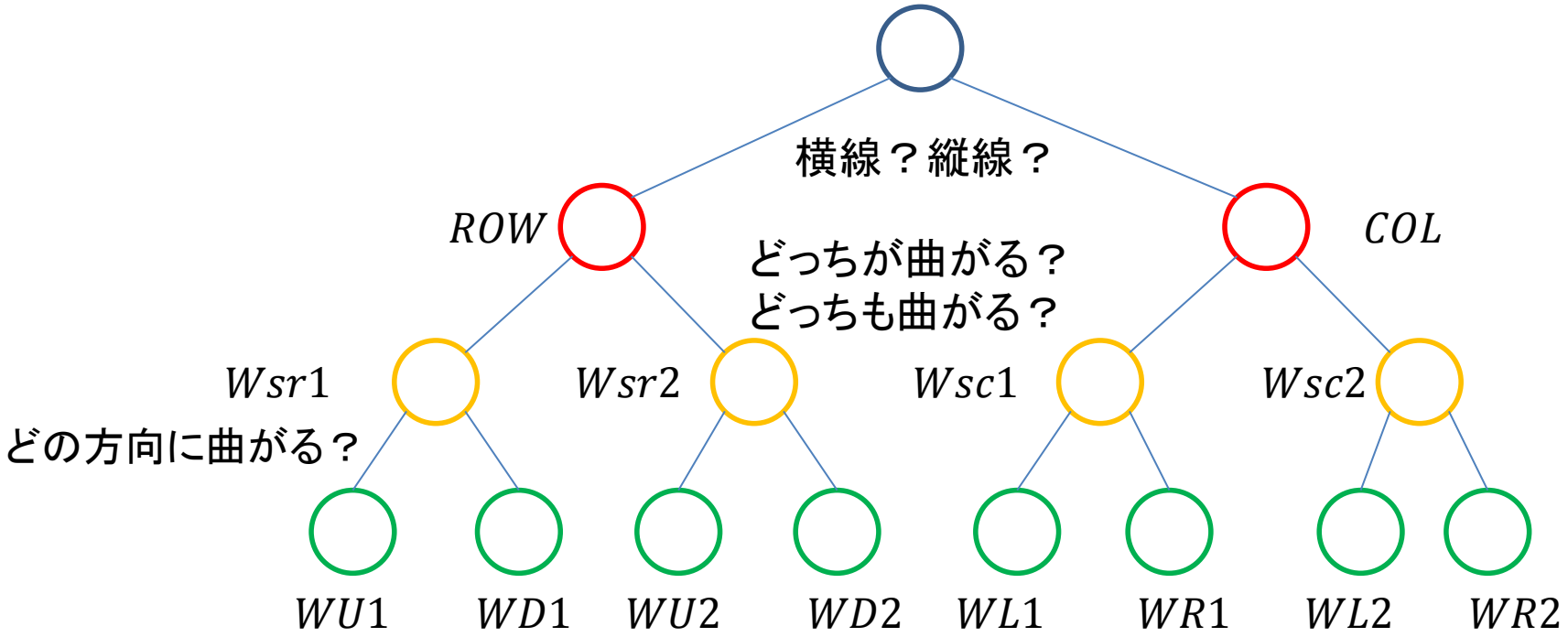
提案手法：〇のルール

- 線の引き方に関する変数を用意する



提案手法：〇のルール

- 線の引き方に関する変数を用意する



提案手法：〇のルール

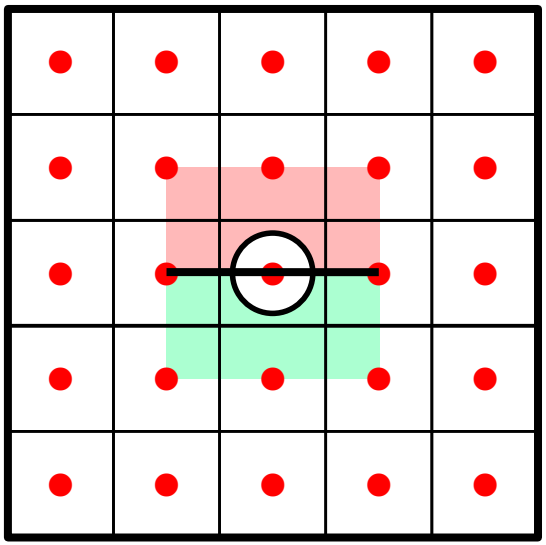
横線か縦線か
$ROW + COL = 1$

内側になる面を選択
$w1 + w2 + w3 + w4 = 1$

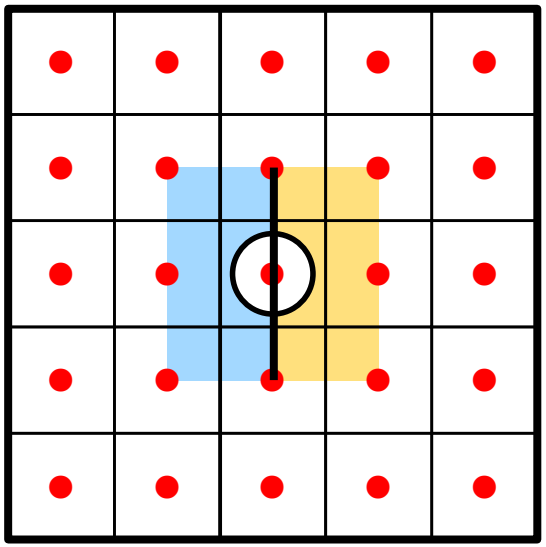
横線の面を選択
$-ROW + w1 + w2 = 0$

縦線の面を選択
$-COL + w3 + w4 = 0$

$w1$
$w2$



$w3$ $w4$



提案手法：〇のルール

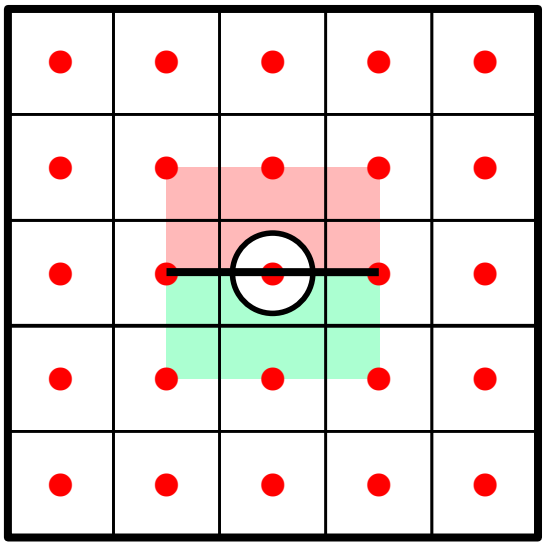
$$2 \times w1 - x_{i,j} - x_{i,j+1} \leq 0$$
$$2 \times w2 + x_{i,j} + x_{i,j+1} \leq 2$$

$$2 \times w3 - x_{i,j} - x_{i+1,j} \leq 0$$
$$2 \times w4 + x_{i,j} + x_{i+1,j} \leq 2$$

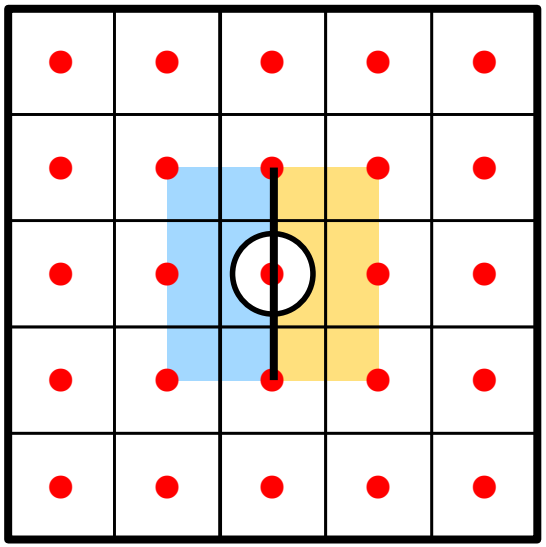
$$2 \times w2 - x_{i+1,j} - x_{i+1,j+1} \leq 0$$
$$2 \times w1 + x_{i+1,j} + x_{i+1,j+1} \leq 2$$

$$2 \times w4 - x_{i,j+1} - x_{i+1,j+1} \leq 0$$
$$2 \times w3 + x_{i,j+1} + x_{i+1,j+1} \leq 2$$

$w1$
$w2$



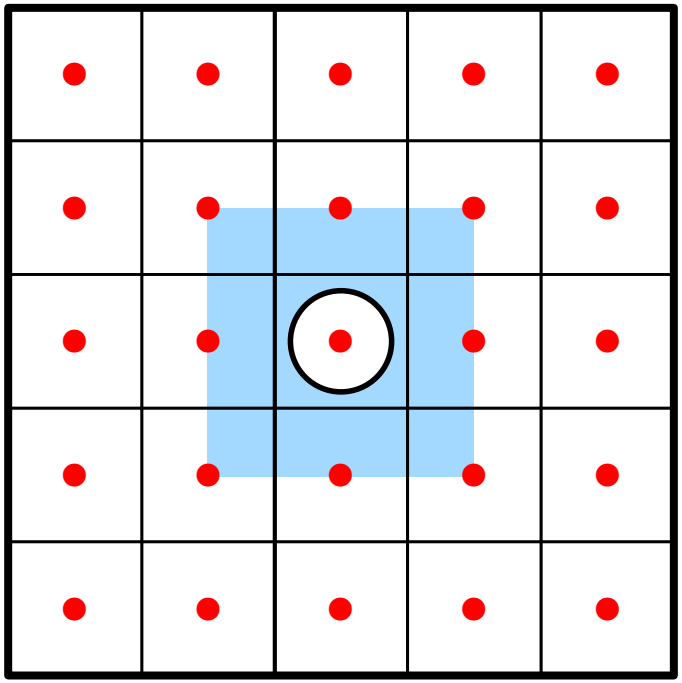
$w3$ $w4$



提案手法：〇のルール

白丸の周り4面の内2面は内側

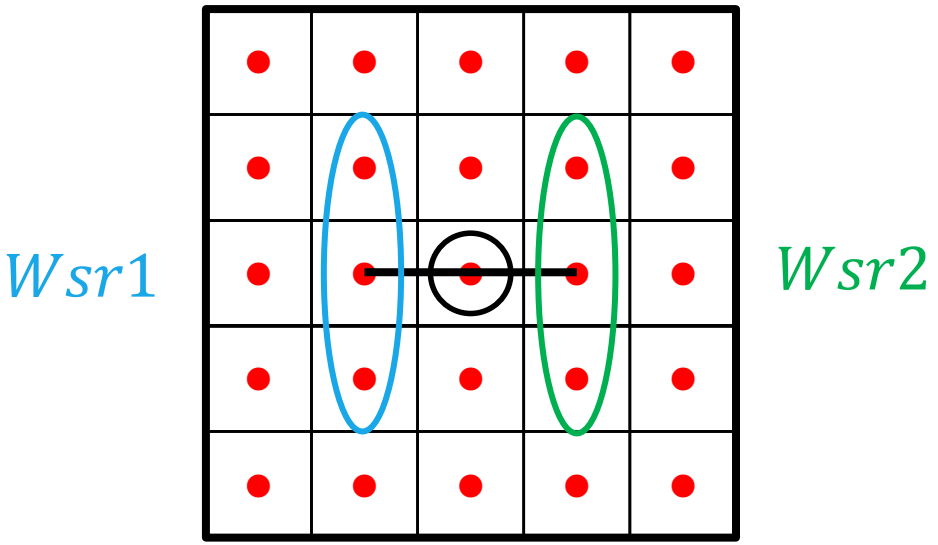
$$x_{i,j} + x_{i,j+1} + x_{i+1,j} + x_{i+1,j+1} = 2$$

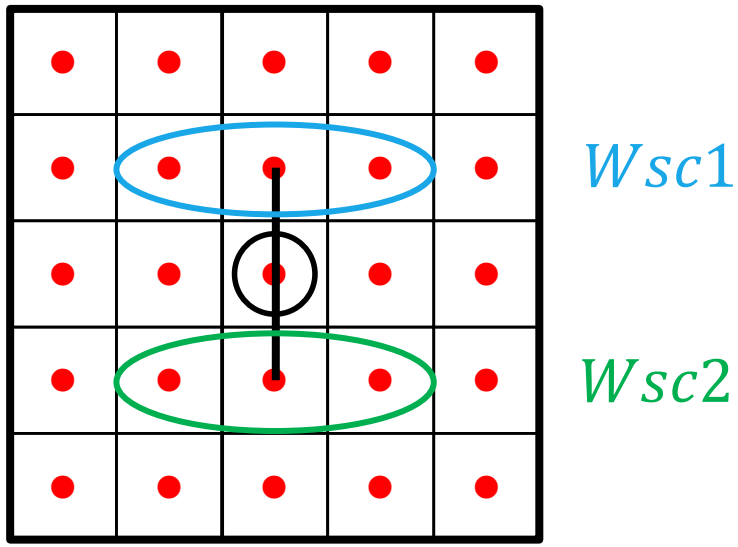


提案手法：〇のルール

横線，縦線それぞれどの箇所が曲がるか

$-ROW + Wsr1 + Wsr2 \geq 0$　　$-COL + Wsc1 + Wsc2 \geq 0$





提案手法：〇のルール

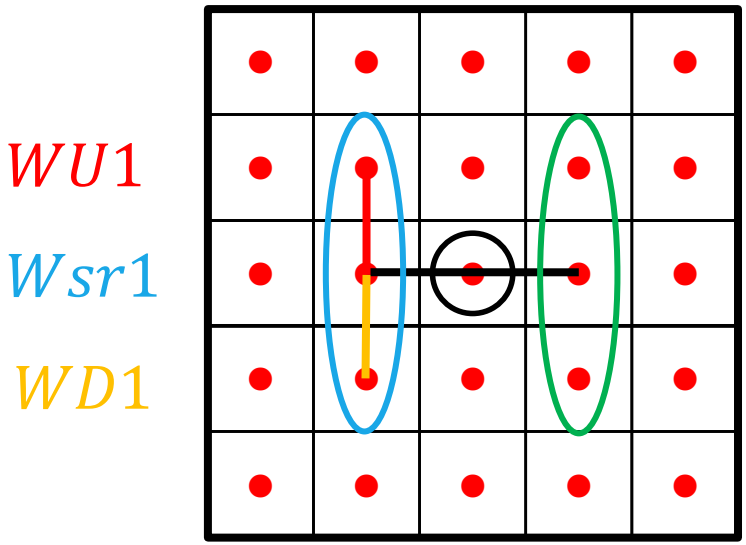
横線，縦線それぞれどの箇所が曲がるか

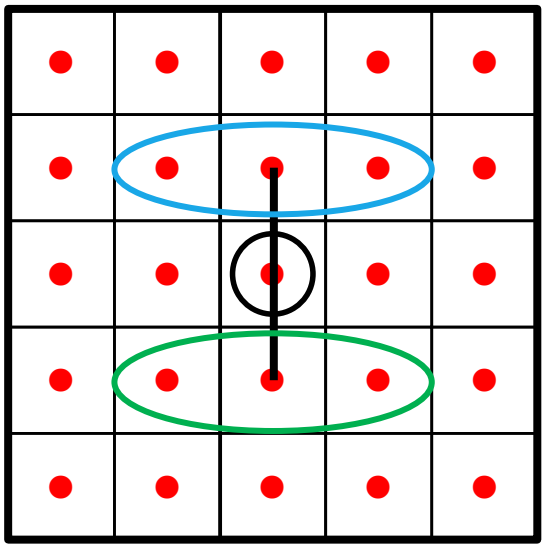
$-ROW + Wsr1 + Wsr2 \geq 0$ $-COL + Wsc1 + Wsc2 \geq 0$

線がどの方向に曲がるか

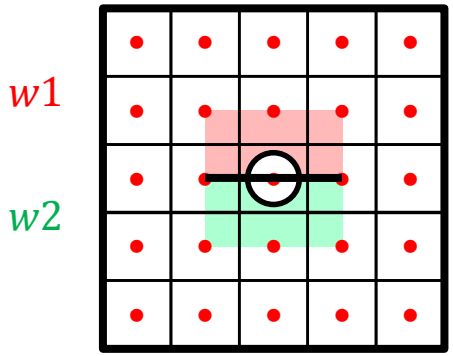
$-Wsr1 + WU1 + WD1 = 0$ $-Wsc1 + WL1 + WR1 = 0$

$-Wsr2 + WU2 + WD2 = 0$ $-Wsc2 + WL2 + WR2 = 0$





提案手法：〇のルール



- 線の引き方によって変わる面がある

曲がらない場合

$-Wsr1 + w2 - x_{i+1,j-1} \leq 0$

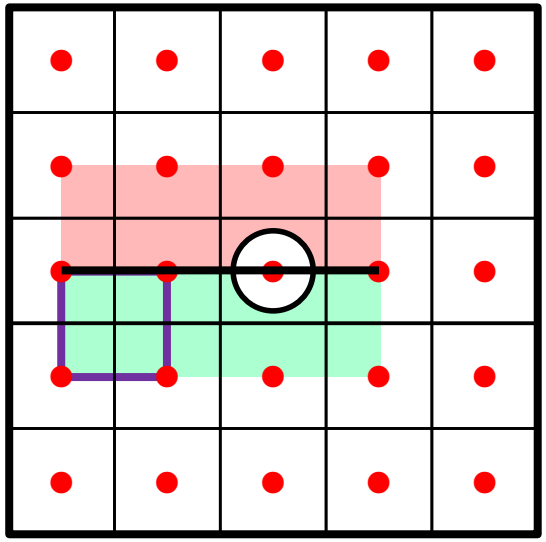
$-Wsr1 + w1 + x_{i+1,j-1} \leq 1$

下に曲がる場合

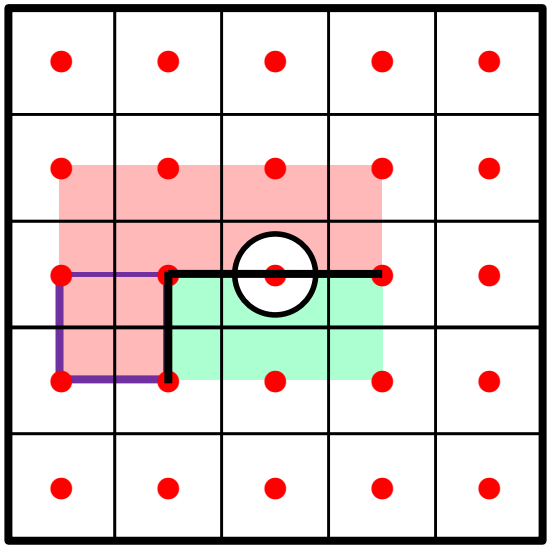
$Wsr1 + WD1 + w1 - x_{i+1,j-1} \leq 2$

$Wsr1 + WD1 + w2 + x_{i+1,j-1} \leq 3$

$Wsr1 = 0$

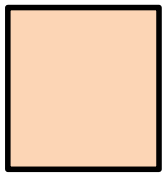


$Wsr1 = 1$ $WD1 = 1$



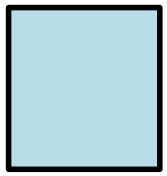
提案手法：複数輪の禁止

- 輪を構成する内側と外側の面を用いて制約式を構成



の集合：L_{out}

	0	0	
0	1	1	0
0	1	1	0
	0	0	



の集合：L_{in}

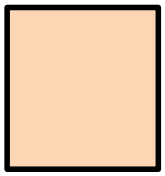
輪を構成する
面の総数

輪を構成する
面の総数−1

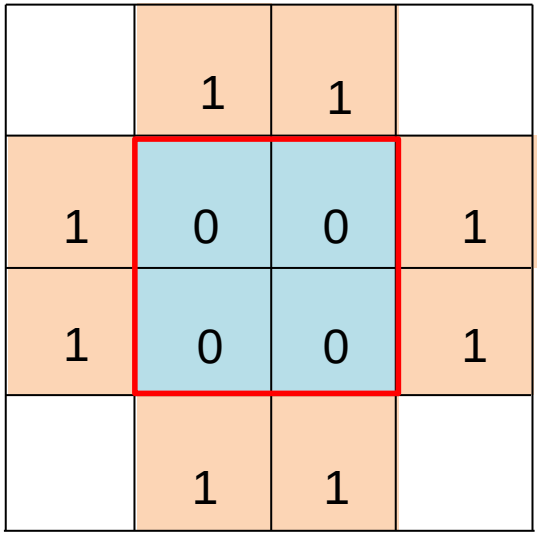
$$\sum_{(i,j)\in L_{in}} x_{i,j} + \sum_{(i,j)\in L_{out}} (1 - x_{i,j}) \leq |L_{in}| + |L_{out}| - 1$$

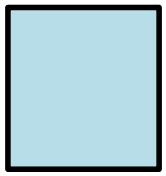
提案手法：複数輪の禁止

- 輪を構成する内側と外側の面を用いて制約式を構成



の集合：L_{out}





の集合：L_{in}

輪を構成する
面の総数

輪を構成する
面の総数−1

$$\sum_{(i,j)\in L_{in}} (1 - x_{i,j}) + \sum_{(i,j)\in L_{out}} x_{i,j} \leq |L_{in}| + |L_{out}| - 1$$

実験的評価（1）

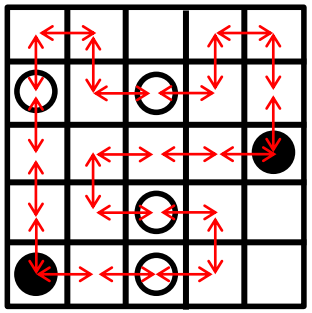
- 通常，人手で解かれる規模の問題例を効率的に解けるか？
  - 2つの既存手法と提案手法の比較

- 評価方法
  - (株)ニコリの問題例を対象
  - 10×10盤面を3問，10×18盤面を3問の計6問
  - 1時間を上限に計測し，3回平均で評価

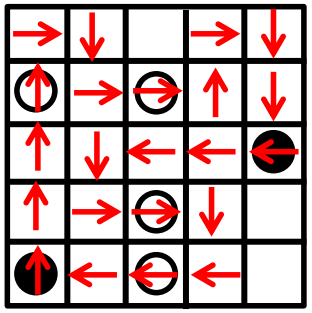
- 実験環境
  - CPU：Intel® Core™ i5-2540M CPU@ 2.60 GHz 2.60 GHz
  - メモリ：4.00 GB
  - OS：Microsoft Windows 7 Professional Edition
  - Solver：GLPK version 4.34 （CPLEX LP形式で記述）

実験的評価(2)：目的関数の設定

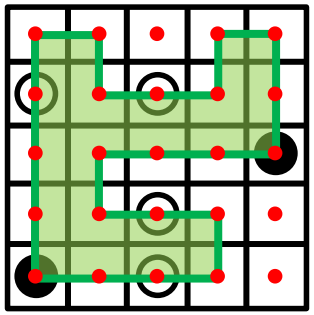
- 無向線+有向線
  - 線の総数を最小化
  - 既存手法と同様の設定



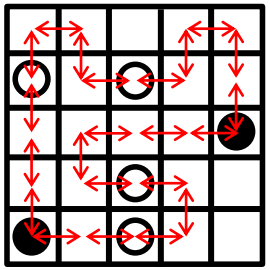
- 有向線のみ
  - 番号の総和を最小化
  - 最小の閉路にするため

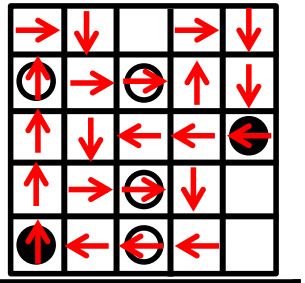


- 提案手法
  - 内側となる面の総数を最大化
  - Slitherlinkの先行研究と同様の設定



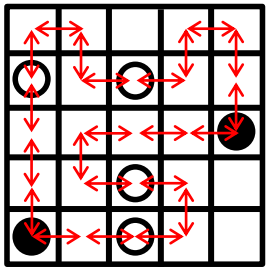
評価：既存手法

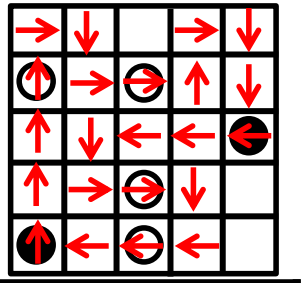




No	サイズ	無向線+有向線			有向線		
		変数	制約式	時間(s)	変数	制約式	時間(s)
1	10 × 10	2,161	2,246	0.911	701	1,366	10.446
2	10 × 10	2,161	2,182	2.365	701	1,257	–
3	10 × 10	2,161	2,173	6.312	701	1,233	–
4	10 × 18	3,601	3,928	0.804	1,179	2,433	–
5	10 × 18	3,601	3,816	2.617	1,179	2,269	–
6	10 × 18	3,601	3,790	–	1,179	2,185	–

評価：既存手法

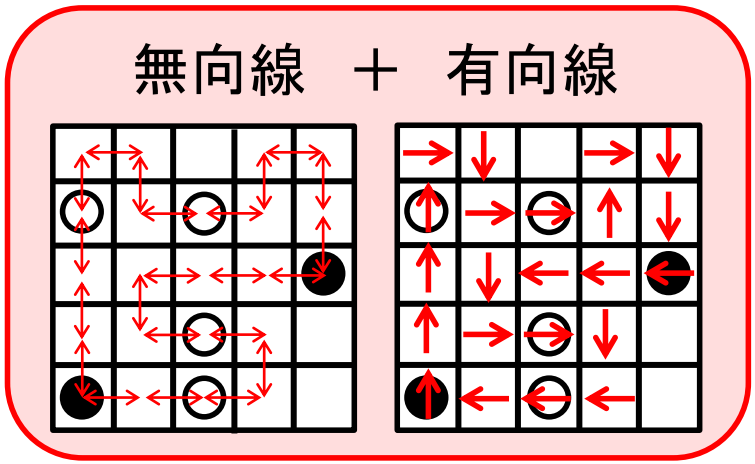


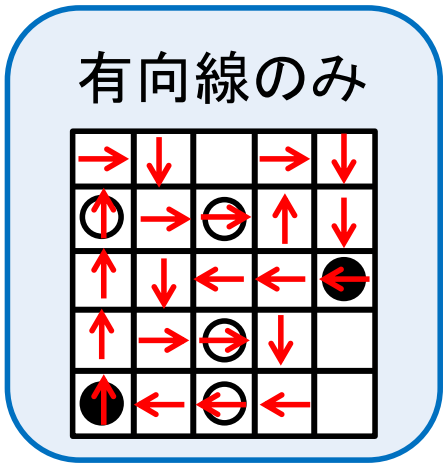


No	サイズ	無向線+有向線			有向線		
		変数	制約式	時間(s)	変数	制約式	時間(s)
1	10 × 10	2,161	2,246	0.911	701	1,366	10.446
2	10 × 10	2,161	2,182	2.365	701	1,257	–
3	10 × 10	2,161	2,173	6.312	701	1,233	–
4	10 × 18	3,601	3,928	0.804	1,179	2,433	–
5	10 × 18	3,601	3,816	2.617	1,179	2,269	–
6	10 × 18	3,601	3,790	–	1,179	2,185	–

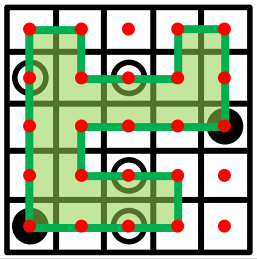
実験結果の考察(1)：既存手法の比較

- 「無向線＋有向線」符号化法
  - 10×10の問題例であれば，効率的に解ける
  - 10×18サイズでは，現実的な時間で解けない問題例も散見
- 「有向線のみ」符号化法
  - 10×10以上の問題例では，ほぼ効率的に解けない
  - 複雑な意味を持たせた変数による，少ない変数数での定式化
  - 制約式数は比較的少ないが，“緩い”定式化のため非効率





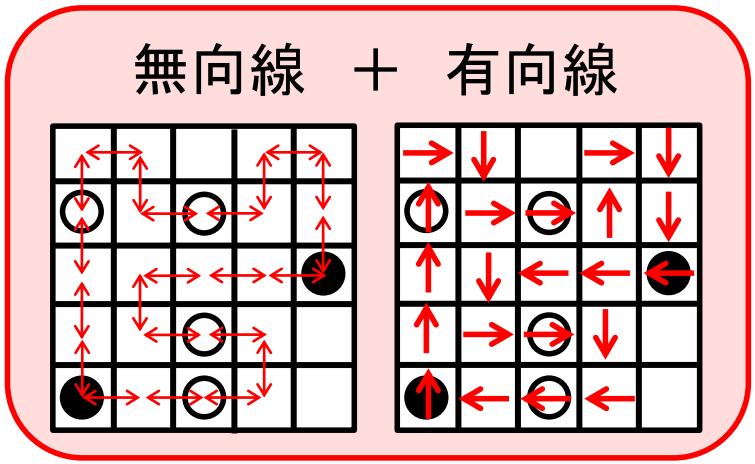
評価：提案手法

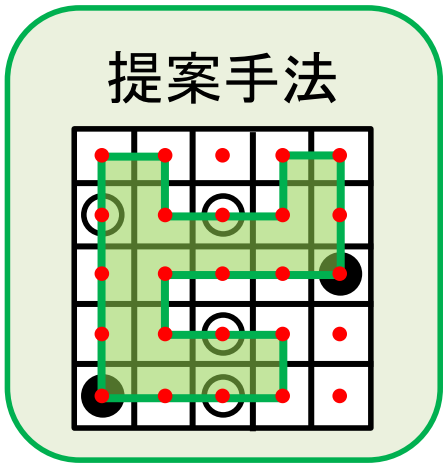


No	サイズ	提案手法			
		変数	制約式	時間(s)	反復数
1	10 × 10	683	1,754	0.650	0
2	10 × 10	497	1,244	0.923	1
3	10 × 10	481	1,210	2.652	2
4	10 × 18	1,183	3,049	0.862	0
5	10 × 18	871	2,182	2.847	0
6	10 × 18	815	2,063	26.624	3

実験結果の考察(2)：既存手法と提案手法

- 提案手法の評価
  - 「無向線＋有向線」符号化法と同程度に優秀
  - 暫定解から制約式を追加する手法の有用性を確認
- 「無向線＋有向線」符号化との比較
  - (制限時間内に)一方のみで解ける問題例が存在
  - 提案手法の方が，同規模の問題例を広範囲にカバー(？)
  - 変数数および制約式数を抑えた定式化になっている





まとめと今後の課題

- まとめ
  - Pearl Puzzleを整数計画問題として記述
    - ✓ 既存手法の修正およびCPLEX LP形式への変換
    - ✓ Slitherlinkでのアイデア[石濱，久野，2013]を適用
  - 実問題に対する実験的評価
    - ✓ 10×10サイズ程度であれば充分高速に解ける
    - ✓ 10×18サイズ規模では手に負えない問題例もある
    - ✓ 「無向線＋有向線」符号化が比較的優秀（？）
- 今後の課題
  - 提案手法で反復回数が多い問題例への対応
  - 目的関数の設定に基づく計算効率の違いを検証