Corral Puzzleの整数計画法による解法と評価

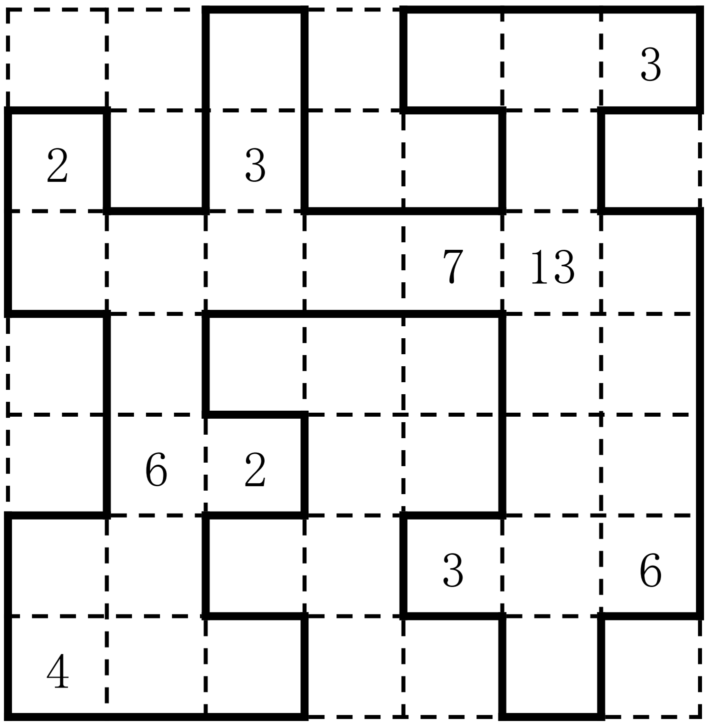
第11回組合せゲーム･パズル研究集会

2016年3月7日（月）

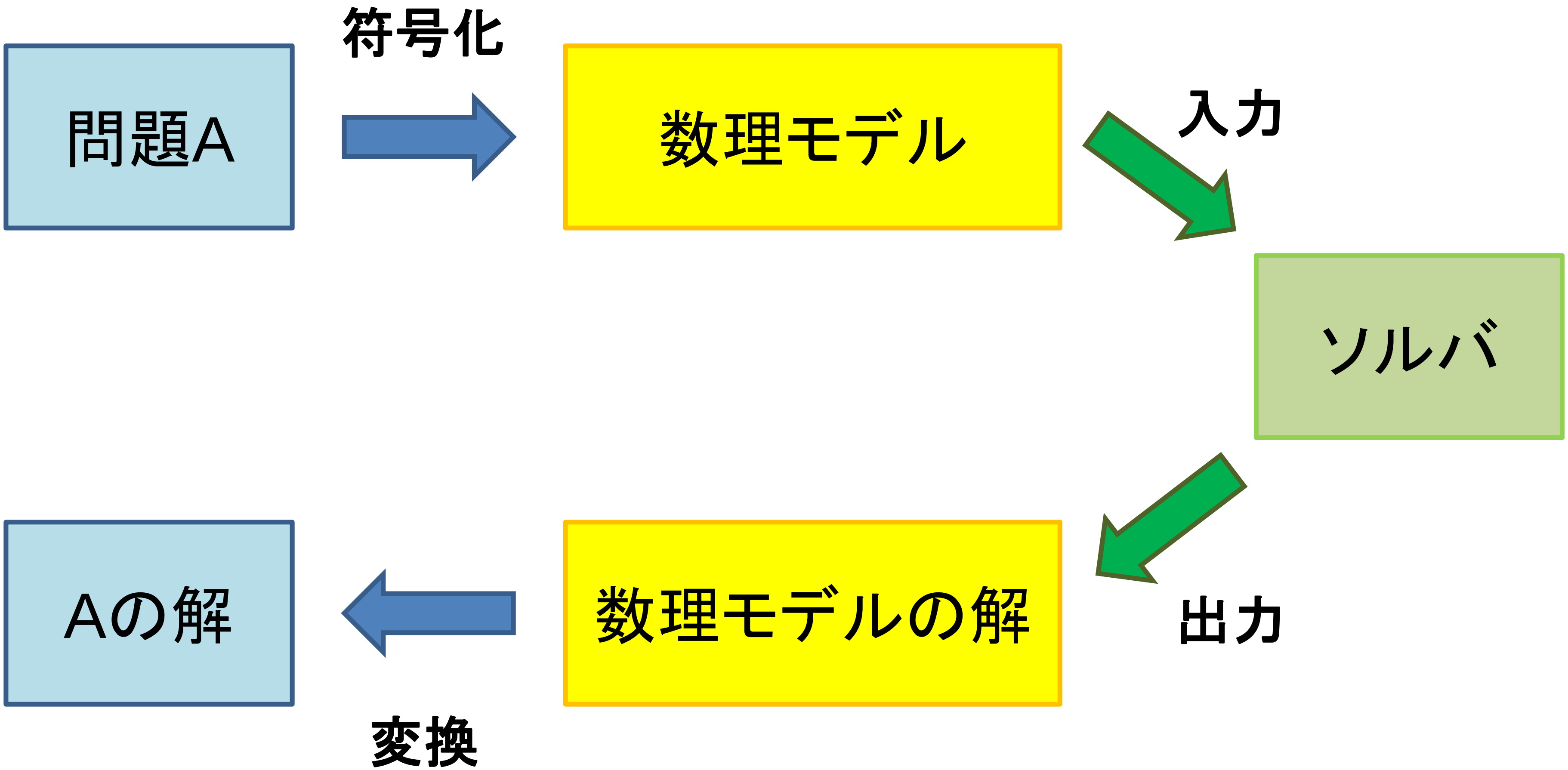
大阪電気通信大学
〇弘中 健太　鈴木 裕章　上嶋 章宏

発表の流れ

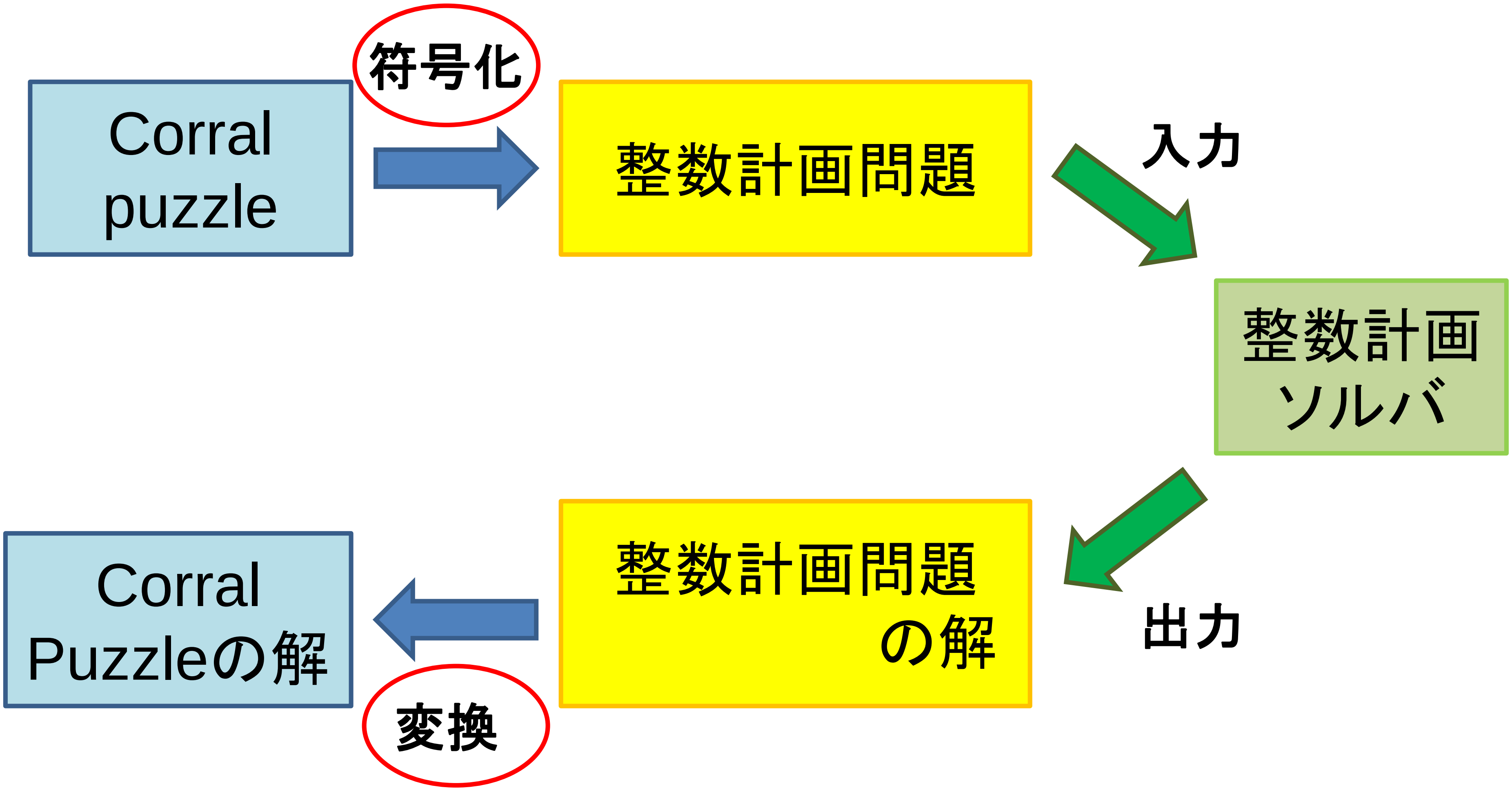
- 研究の背景
  - 整数計画法と先行研究
- Corral Puzzle
  - ルールと定義
- 定式化
  - 2種類の閉路性の定式化
- 評価
  - 計測結果と考察
- まとめと今後の課題



研究背景:整数計画法



研究背景：整数計画法


Corral puzzle
符号化
整数計画問題
入力
整数計画ソルバ
出力
整数計画問題の解
変換
Corral Puzzleの解

研究背景：整数計画法

整数計画法

スケジューリング問題

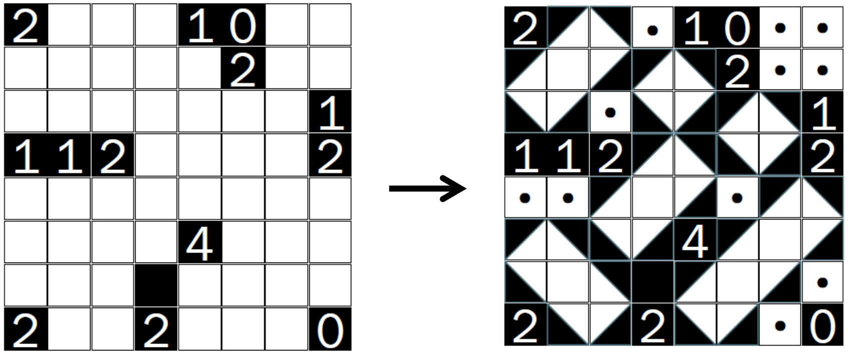
巡回セールスマン問題

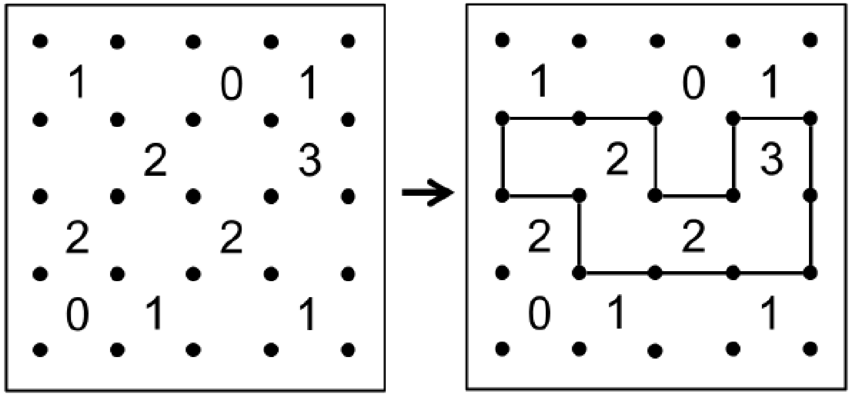
ペンシルパズル

etc

シャカシャカ [岡本, 2013]

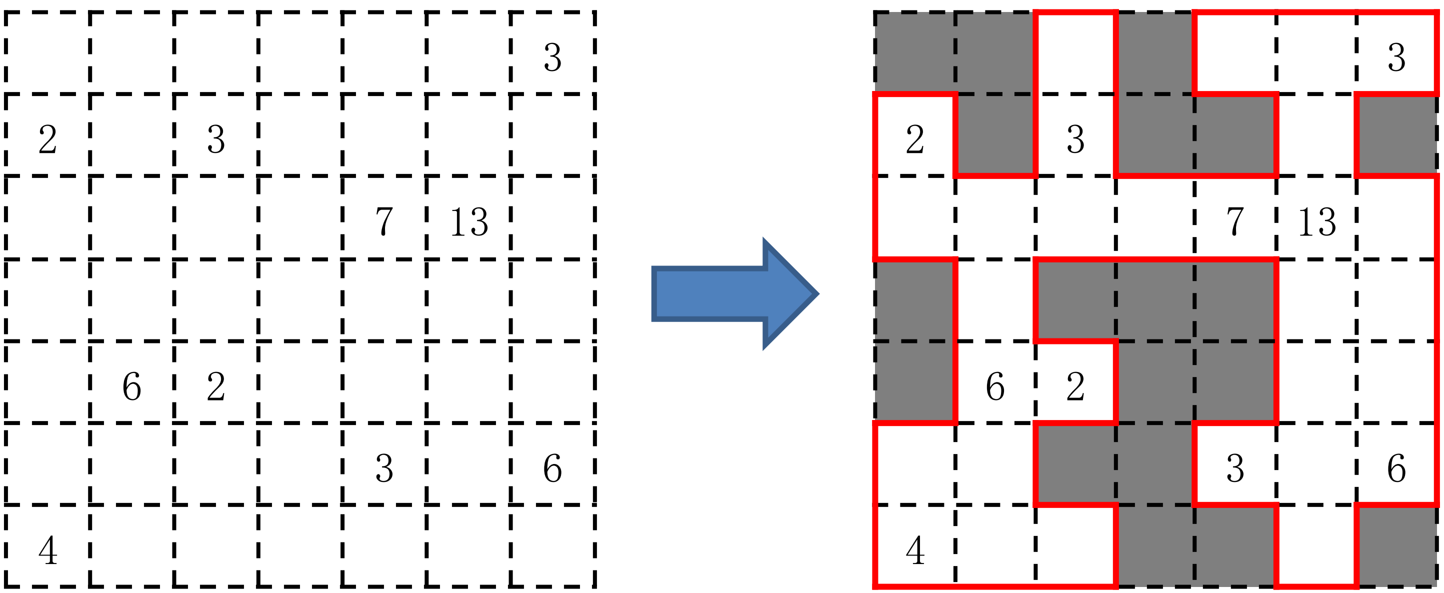
Slitherlink [石濱, 久野, 2013]





研究背景:計算複雑さ

- Corral PuzzleのNP完全性 [E. Friedman, 2002]
  - 問題の効率的な解法がない
- Corral PuzzleのASP完全性 [上野, 2015]
  - 想定解以外の解の発見が困難

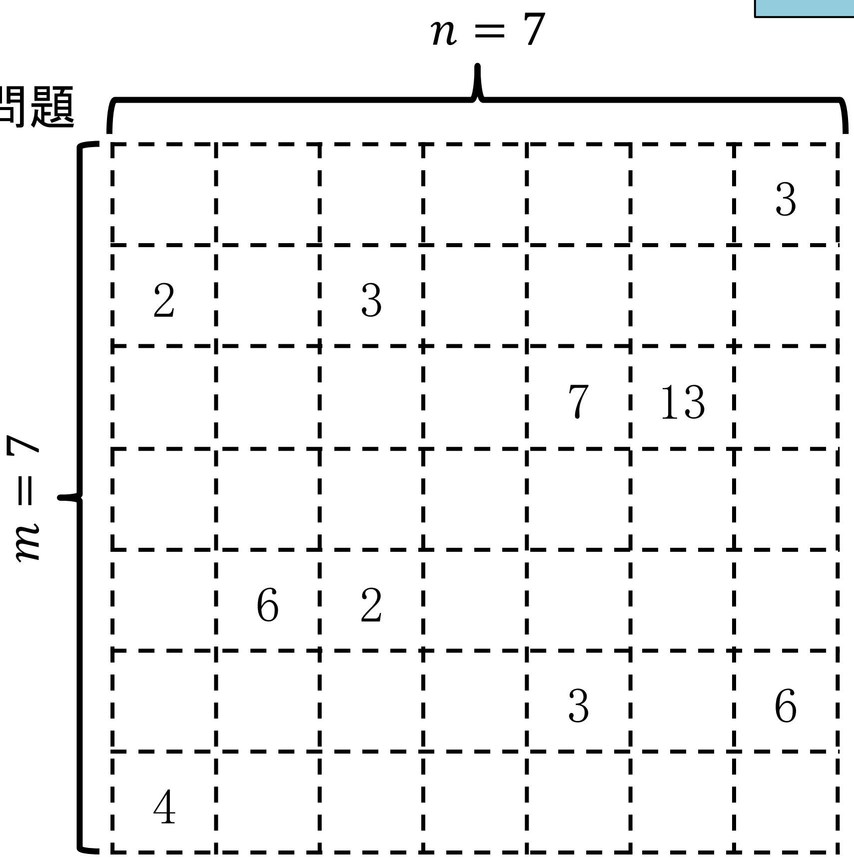


ルール

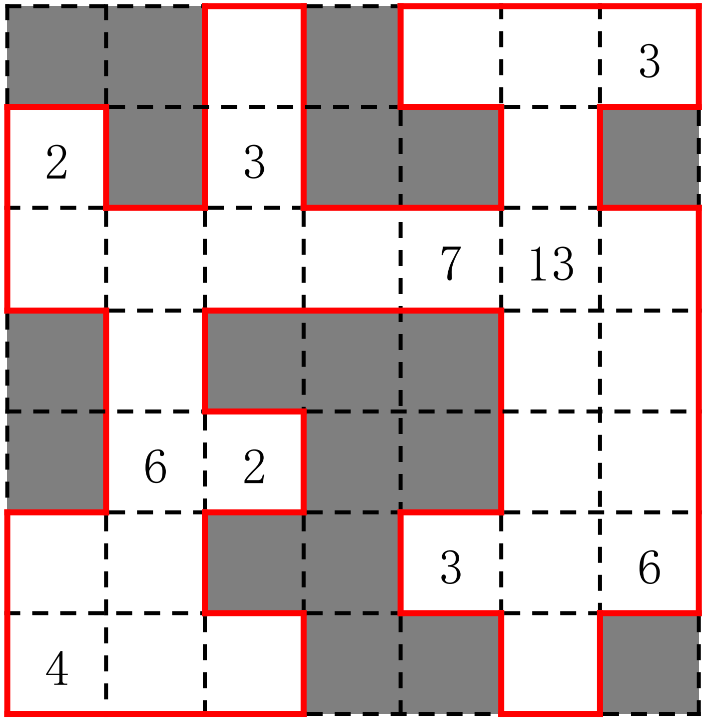
盤面サイズは$m \times n$

①点線に線を引き1つの単純閉路を作る
②数字マスは閉路まで数字分マスが並ぶ
③数字マスはループ線の内側に入る
④線は交差，枝分かれしない

問題



答え

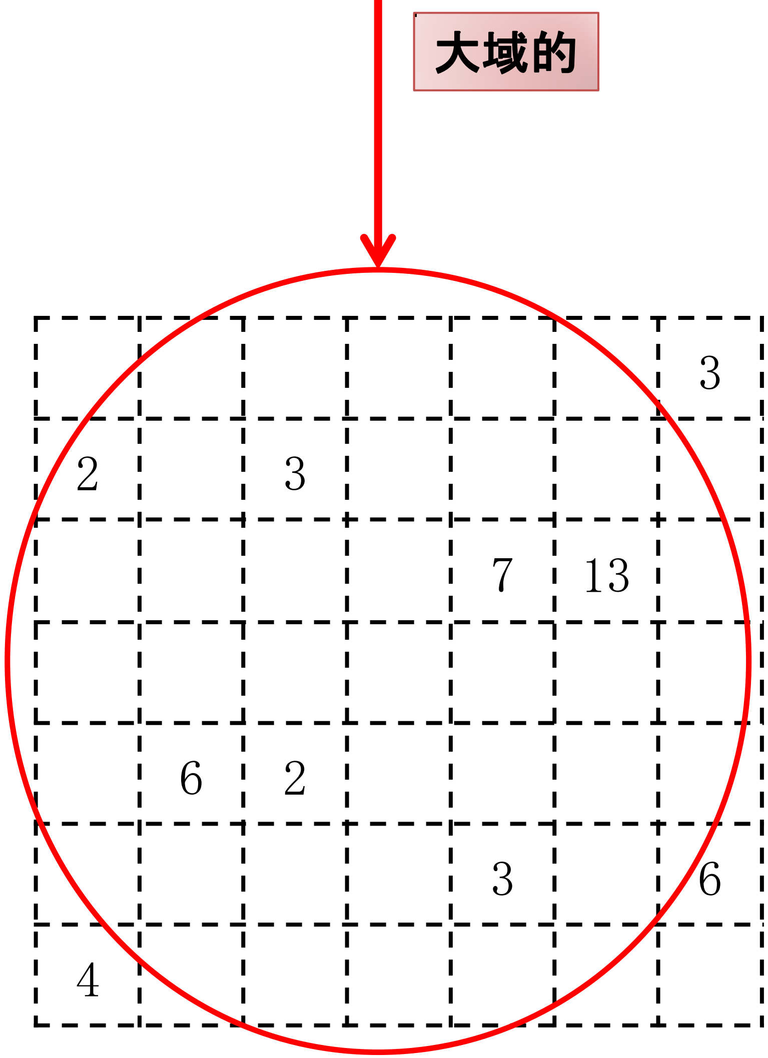


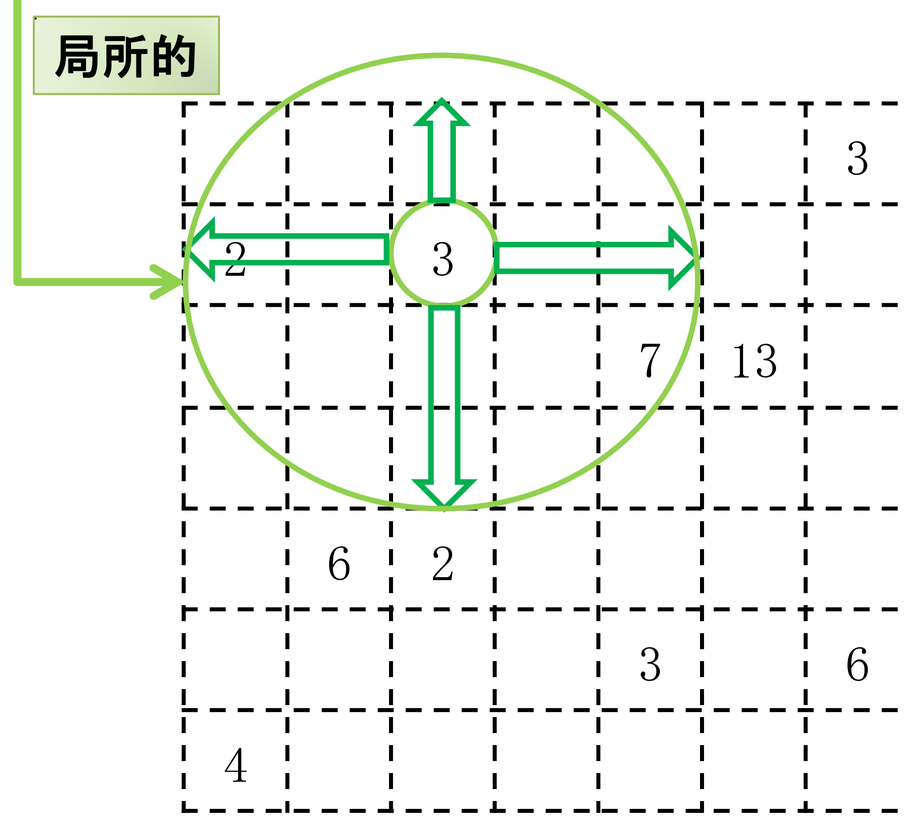
①点線に線を引き1つの単純閉路を作る

②数字マスは閉路まで数字分マスが並ぶ

③数字マスはループ線の内側に入る

④線は交差，枝分かれしない





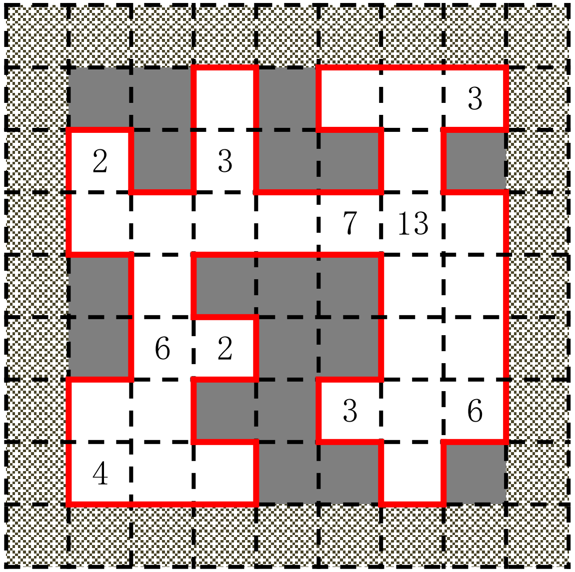
定式化の方策

- 各マスが閉路の内側か外側かを2種類の変数で表現
- 内側と外側のマスの境界線を閉路とする

内側のマスを白マス: $b(i,j)$
外側のマスを黒マス: $w(i,j)$

入力盤面に対し，1周り分マスを用意する

外枠と呼び，黒マスである

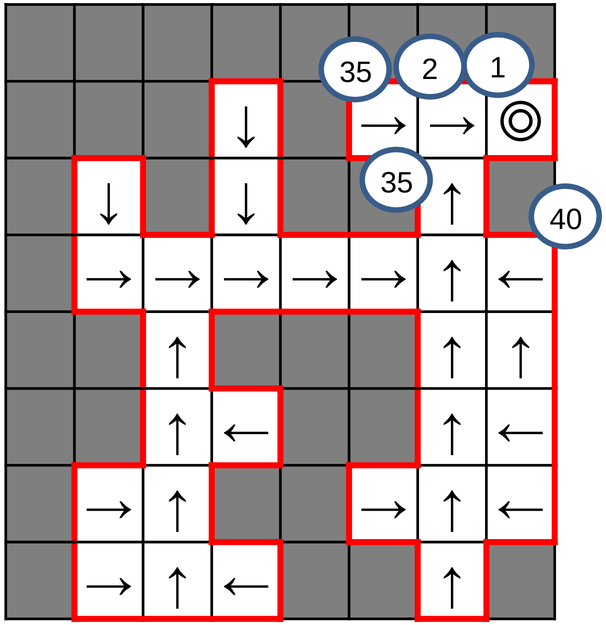


定式化

Slitherlinkパズルの解法[石濱，久野，2013]

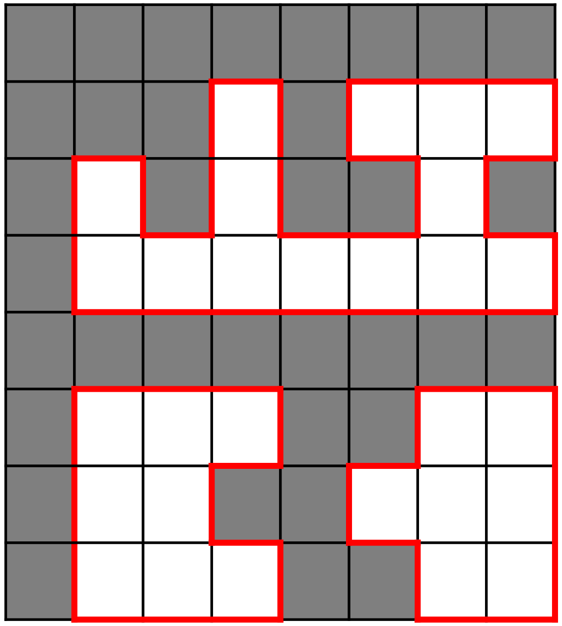
閉路性を保証する制約を有する

矢印と数値を与える



外部的に制約を加える

ある閉路の生成を禁止

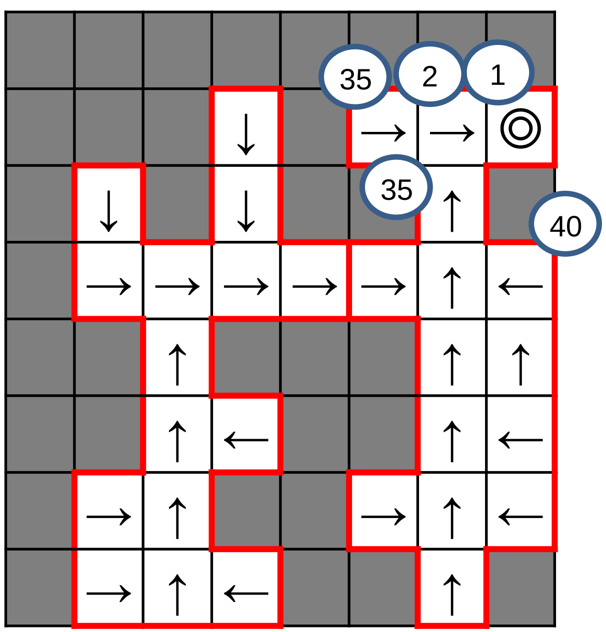


定式化

Slitherlinkパズルの解法[石濱, 久野, 2013]

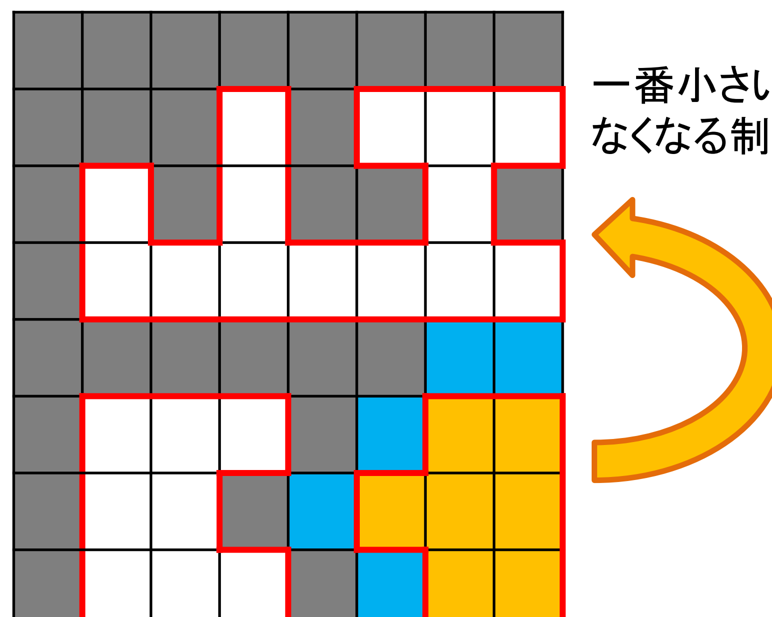
閉路性を保証する制約を有する

矢印と数値を与える



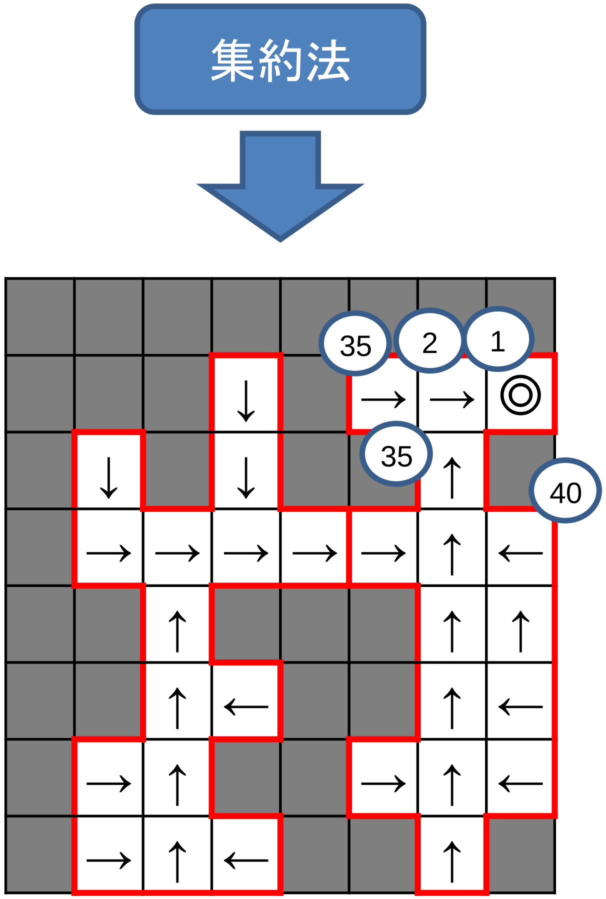
外部的に制約を加える

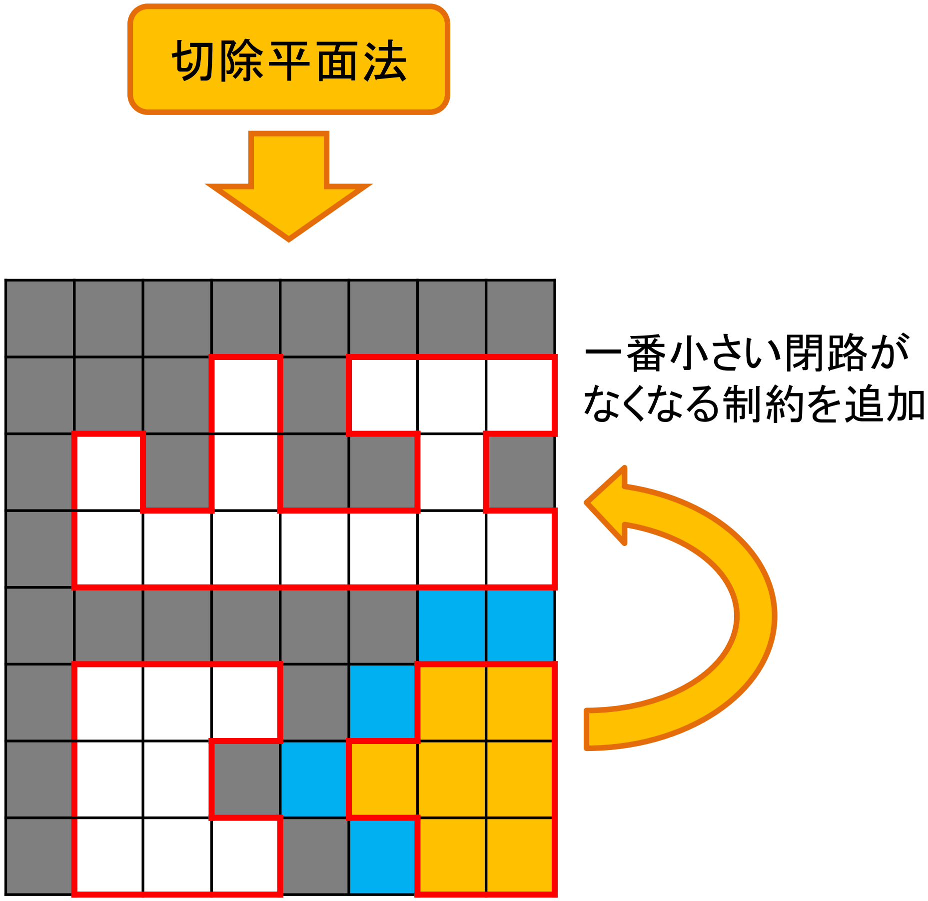
ある閉路の生成を禁止



一番小さい閉路がなくなる制約を追加

定式化

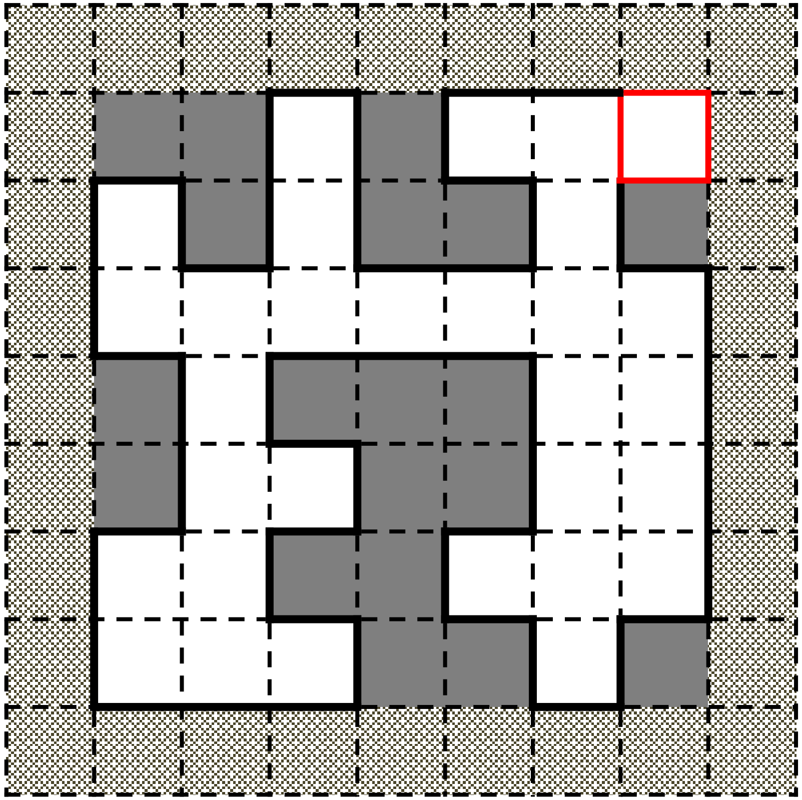


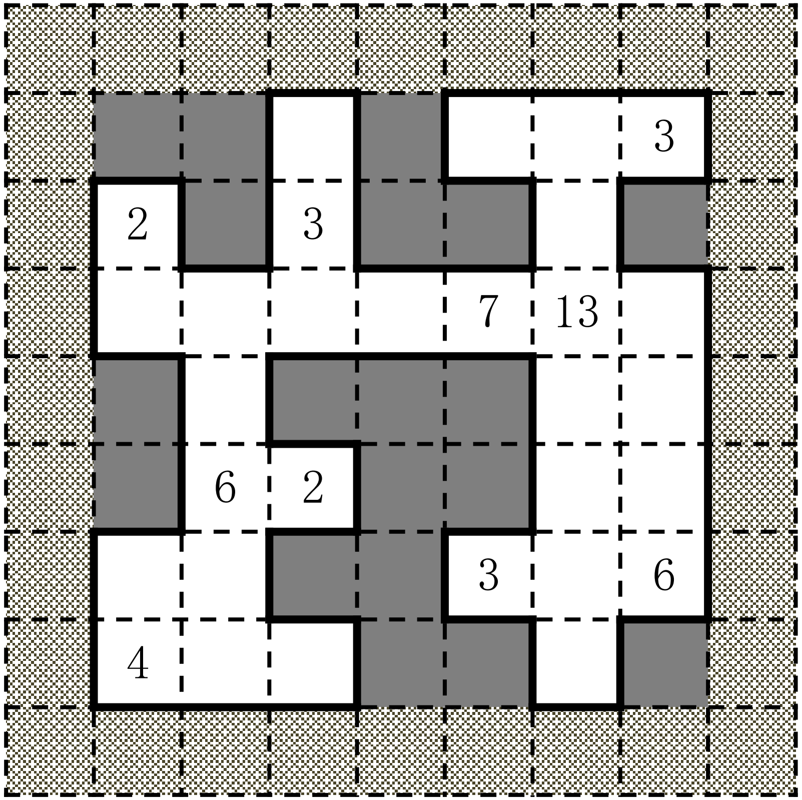


集約法：白マスver

特別なマスを1つ選ぶ

特別なマスは矢印を持たず小さい値を固定して与える

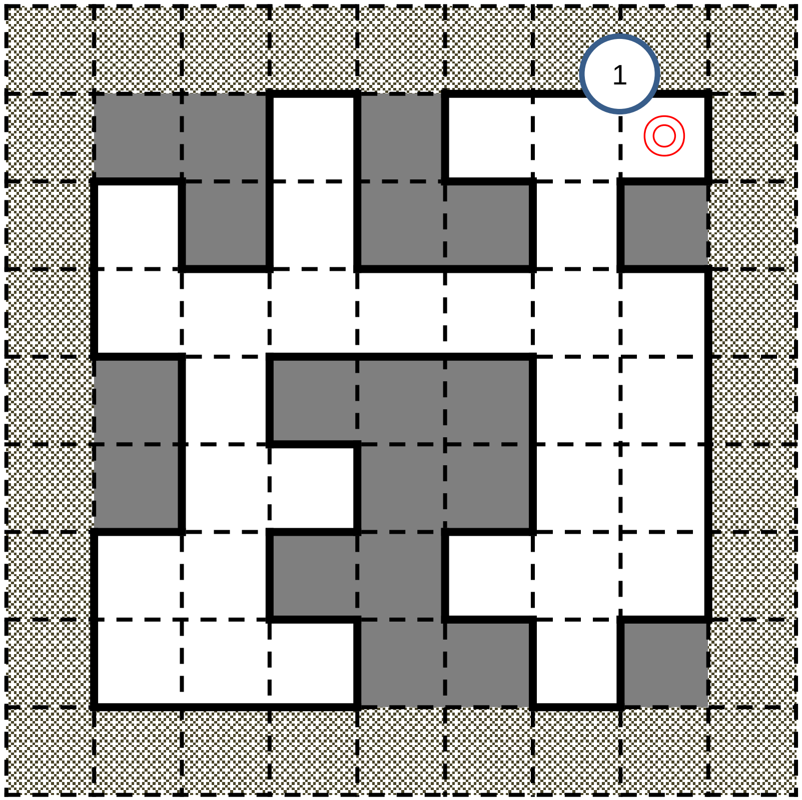


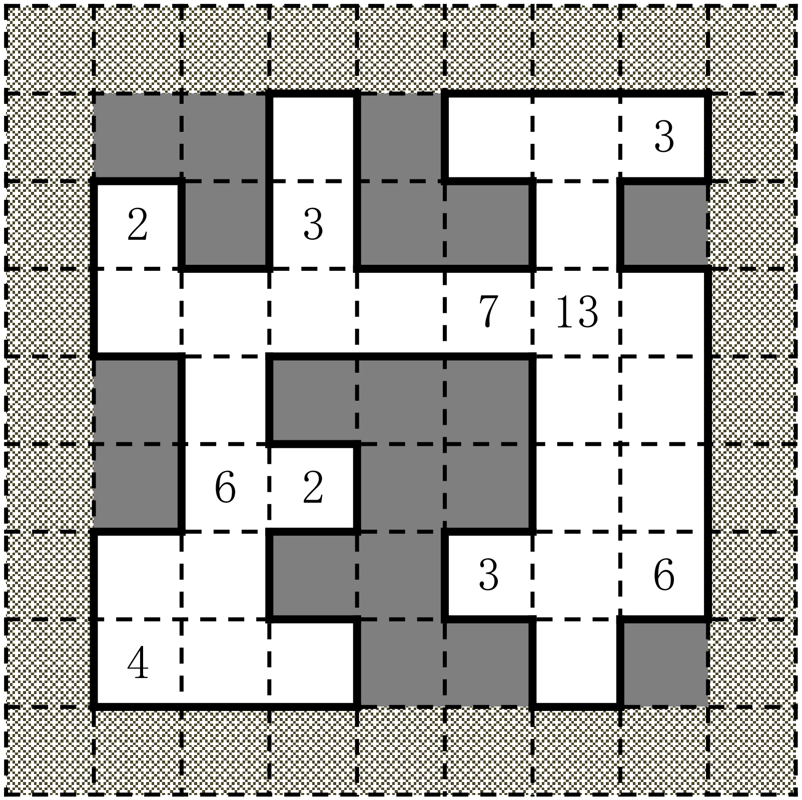


集約法：白マスver

特別なマスを1つ選ぶ

特別なマスは矢印を持たず小さい値を固定して与える

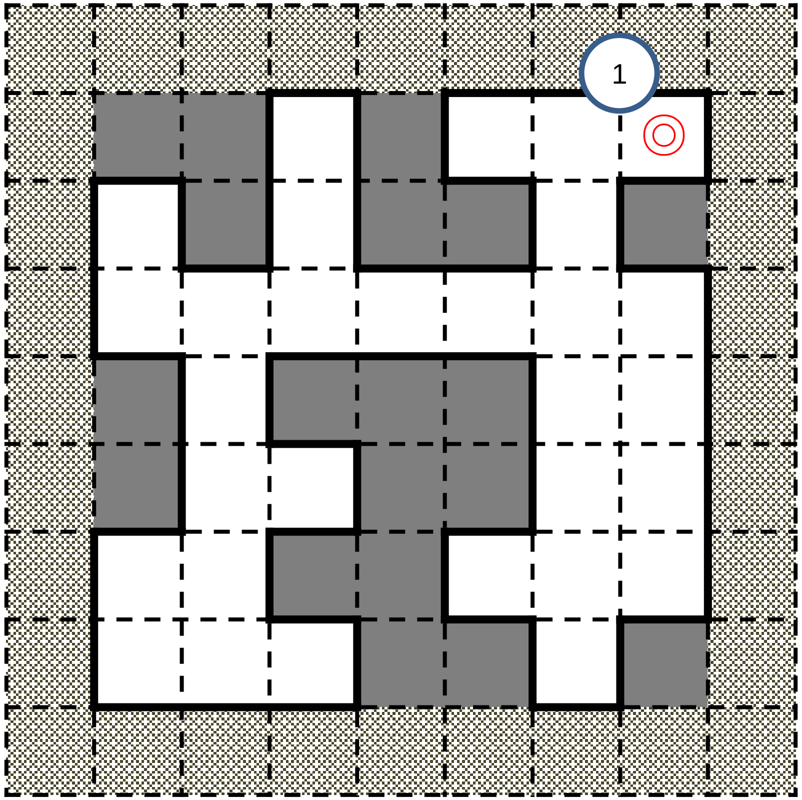


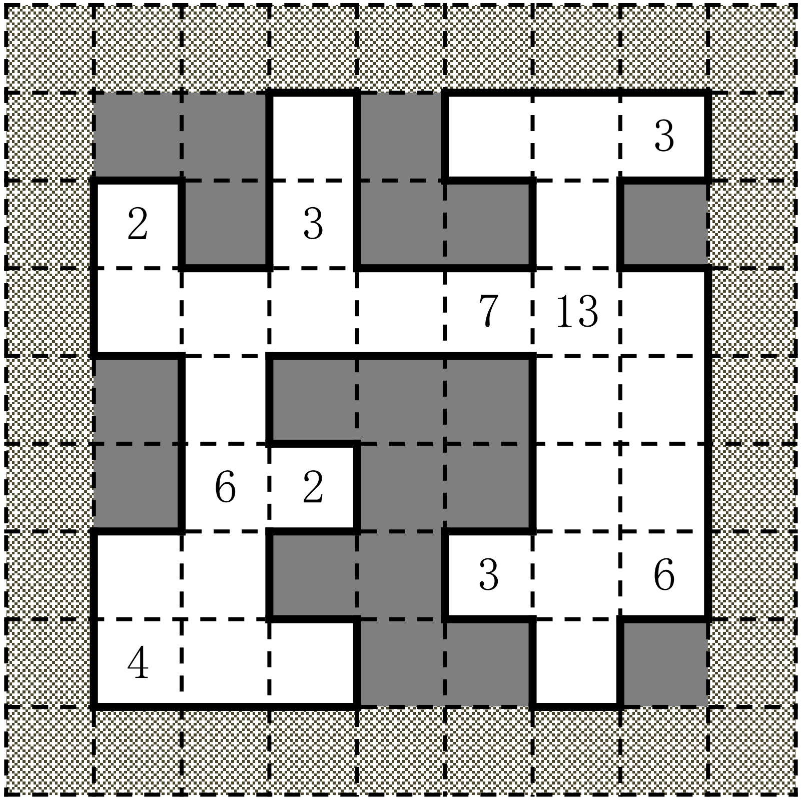


集約法：白マスver

特別なマスを基準に矢印と数値を与える

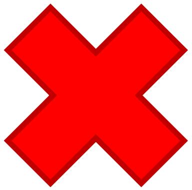
隣接するマスで**向かい合う矢印**を置いてはいけない
矢印の向いてる先に**自分と違う色のマス**があってはいけない
矢印の向いてる先の隣接するマスの**数値が小さい**



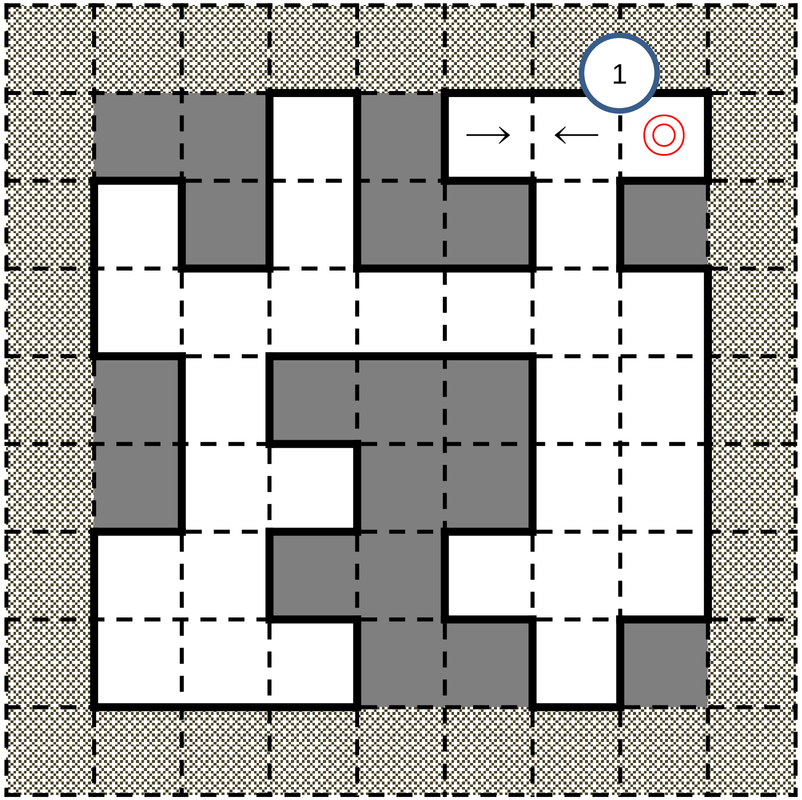


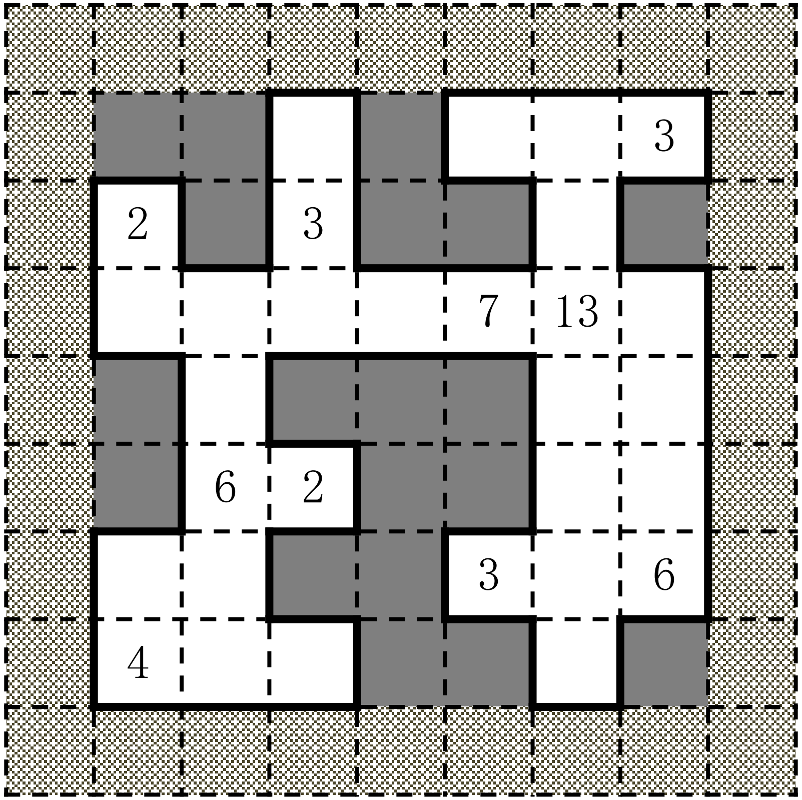
集約法：白マスver

特別なマスを基準に矢印と数値を与える



隣接するマスで**向かい合う矢印**を置いてはいけない
矢印の向いてる先に**自分と違う色のマス**があってはいけない
矢印の向いてる先の隣接するマスの**数値が小さい**

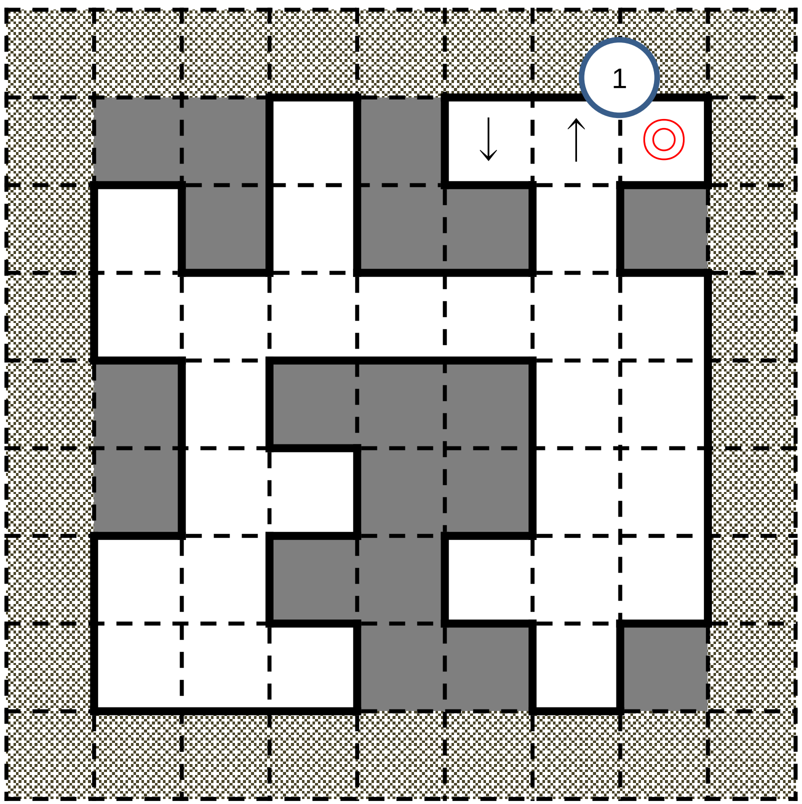


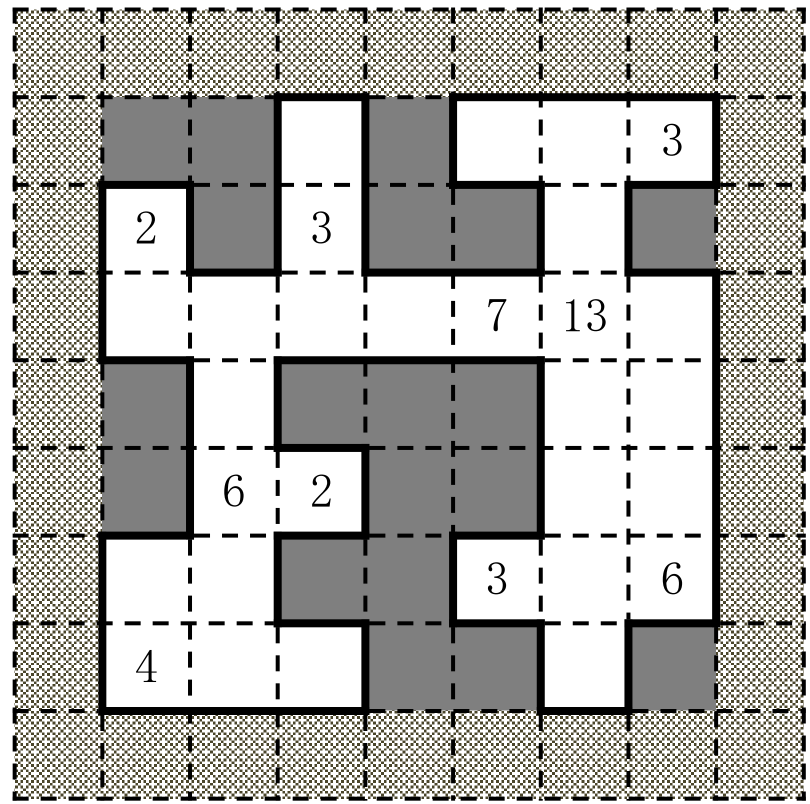


集約法：白マスver

特別なマスを基準に矢印と数値を与える

隣接するマスで**向かい合う矢印**を置いてはいけない
矢印の向いてる先に**自分と違う色のマス**があってはいけない
矢印の向いてる先の隣接するマスの**数値が小さい**

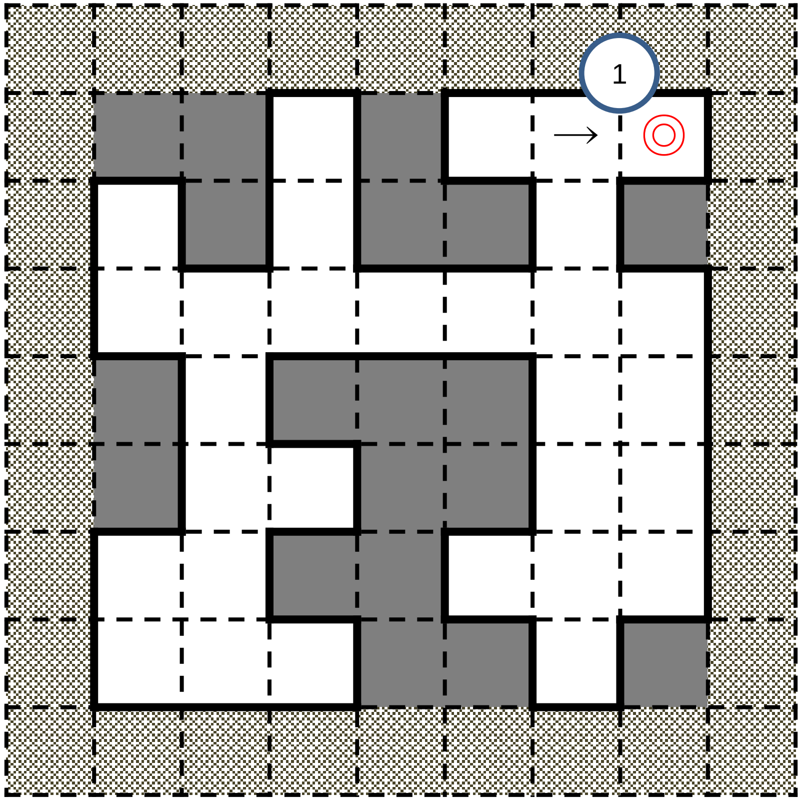


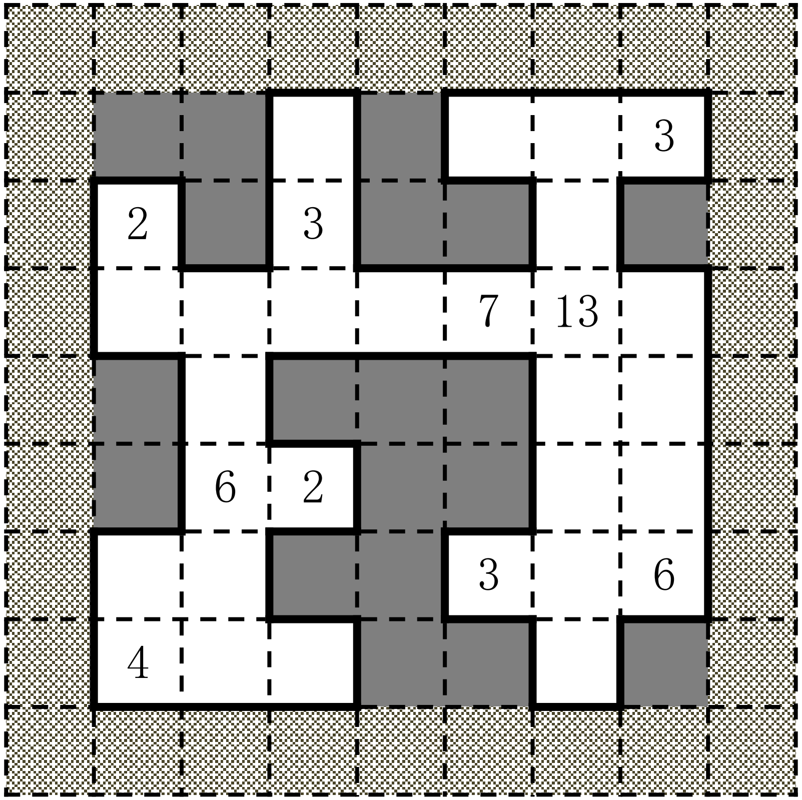


集約法：白マスver

特別なマスを基準に矢印と数値を与える

隣接するマスで**向かい合う矢印**を置いてはいけない
矢印の向いてる先に**自分と違う色のマス**があってはいけない
矢印の向いてる先の隣接するマスの**数値が小さい**

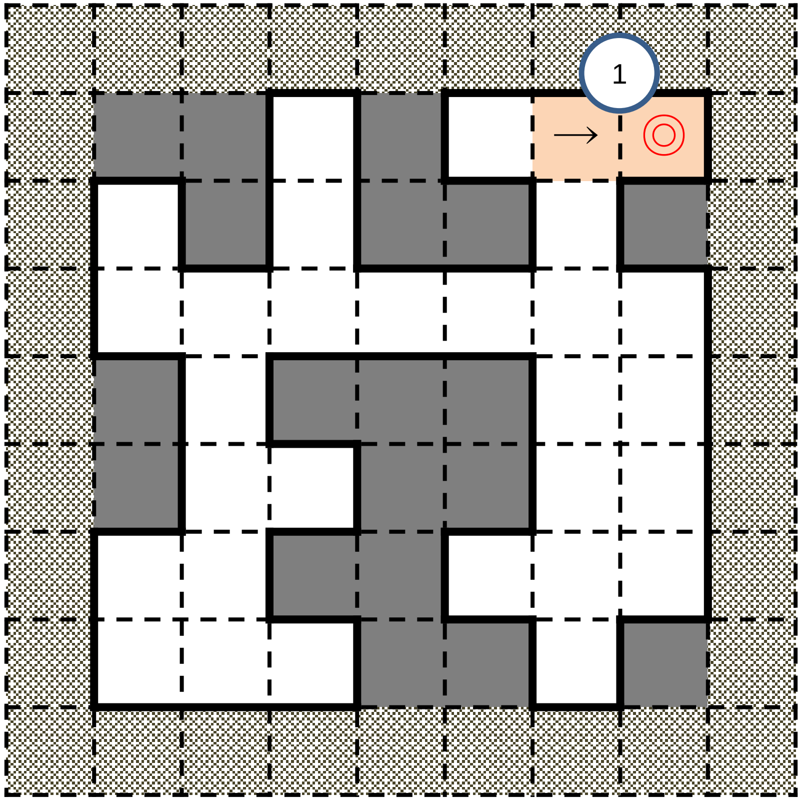


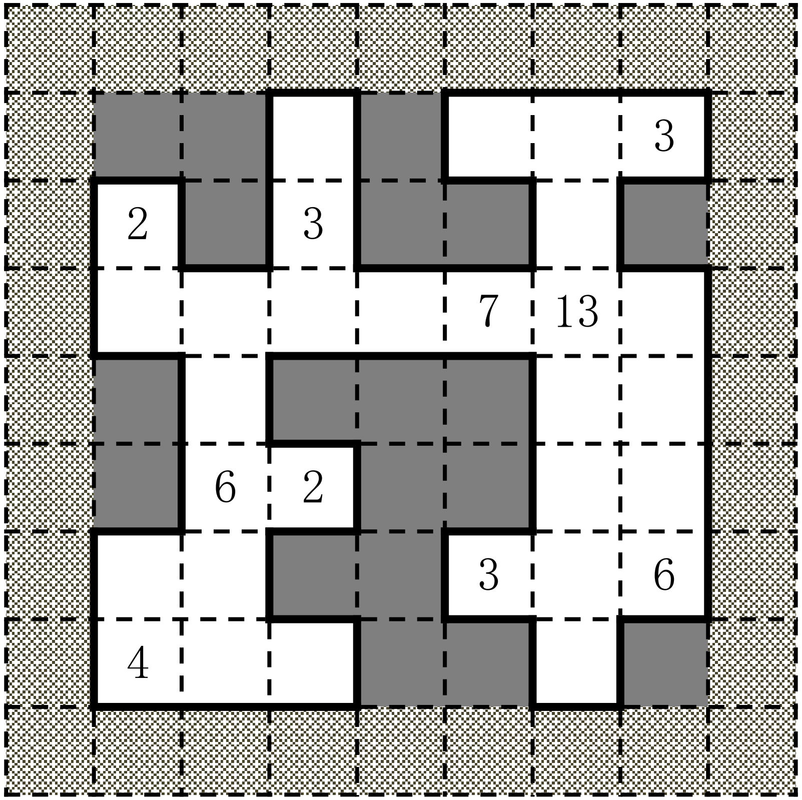


集約法：白マスver

特別なマスを基準に矢印と数値を与える

隣接するマスで**向かい合う矢印**を置いてはいけない
矢印の向いてる先に**自分と違う色のマス**があってはいけない
矢印の向いてる先の隣接するマスの**数値が小さい**

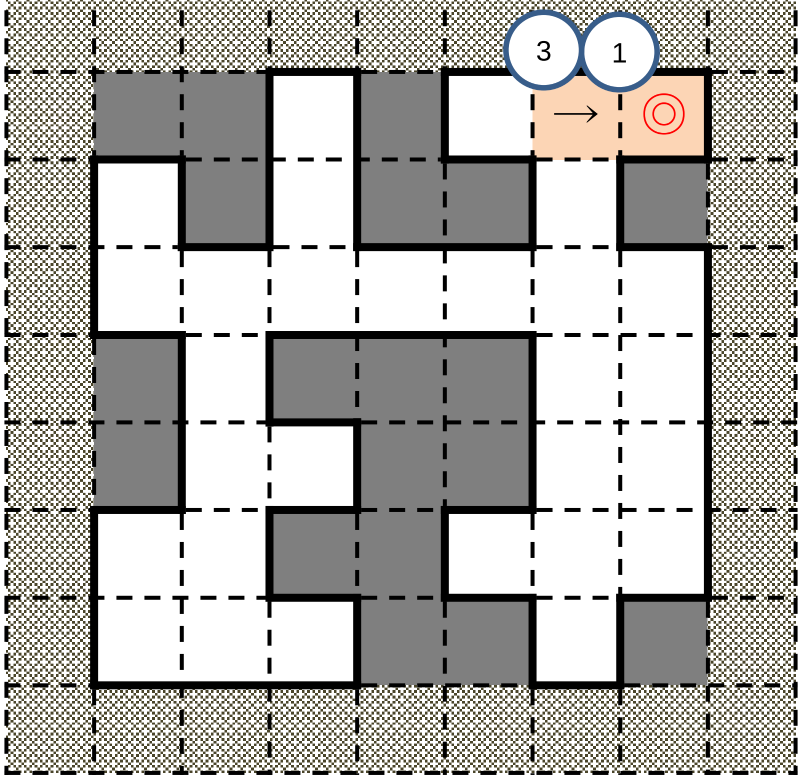


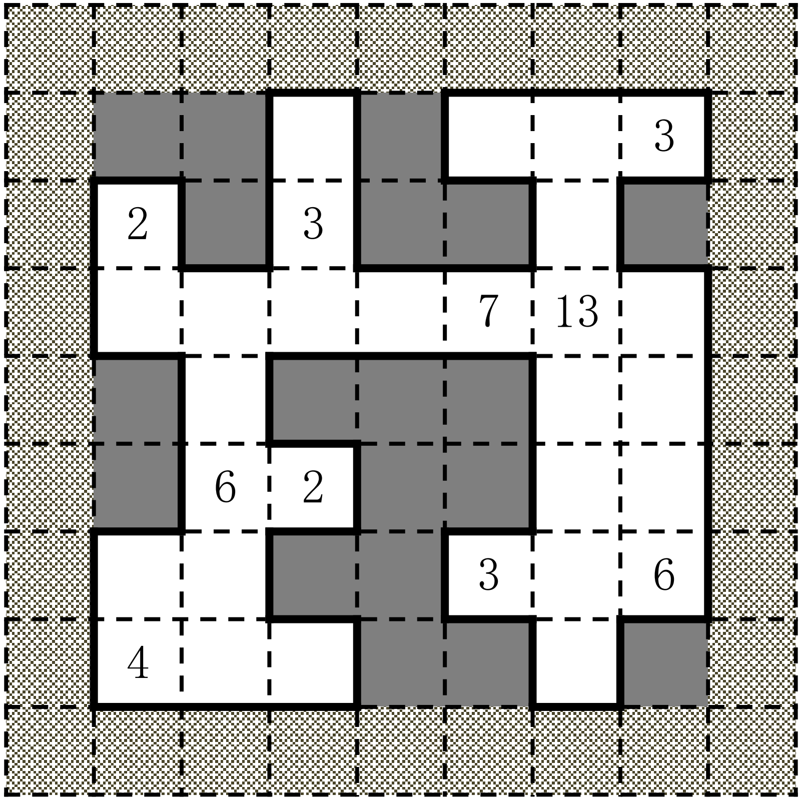


集約法：白マスver

特別なマスを基準に矢印と数値を与える

隣接するマスで**向かい合う矢印**を置いてはいけない
矢印の向いてる先に**自分と違う色のマス**があってはいけない
矢印の向いてる先の隣接するマスの**数値が小さい**

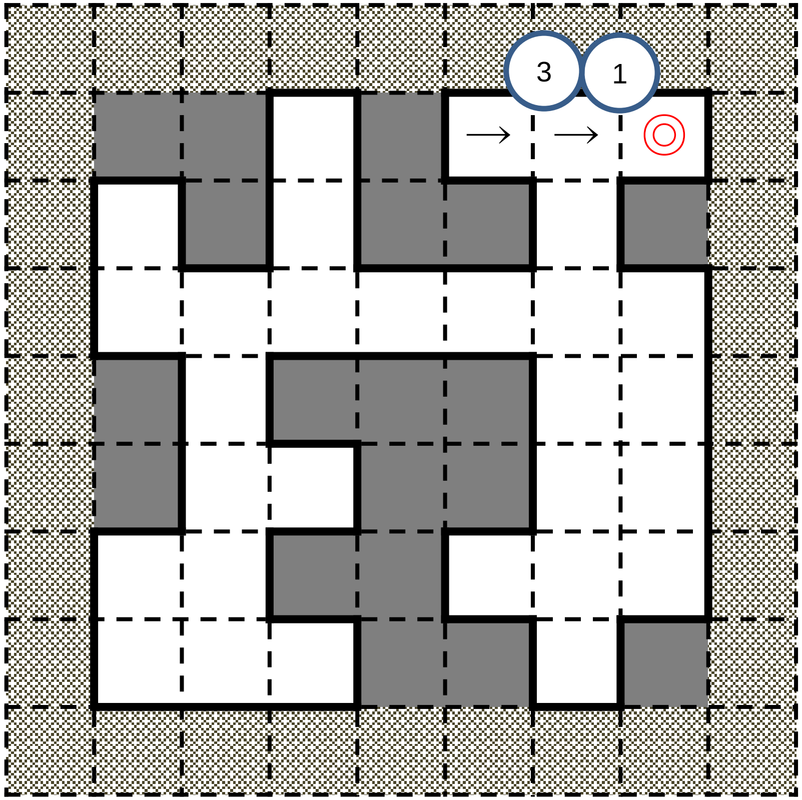


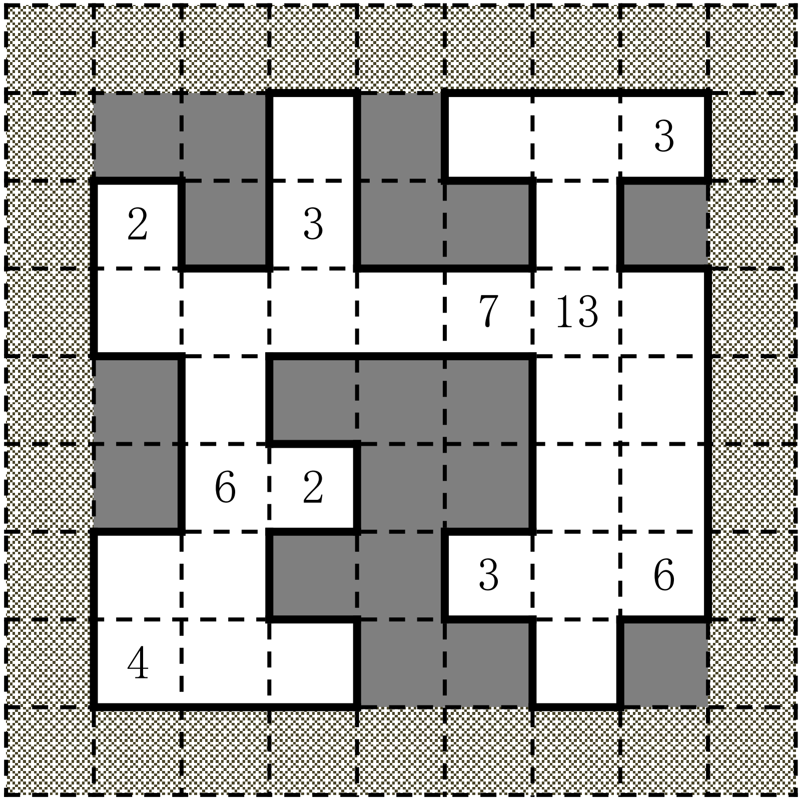


集約法：白マスver

特別なマスを基準に矢印と数値を与える

隣接するマスで**向かい合う矢印**を置いてはいけない
矢印の向いてる先に**自分と違う色のマス**があってはいけない
矢印の向いてる先の隣接するマスの**数値が小さい**

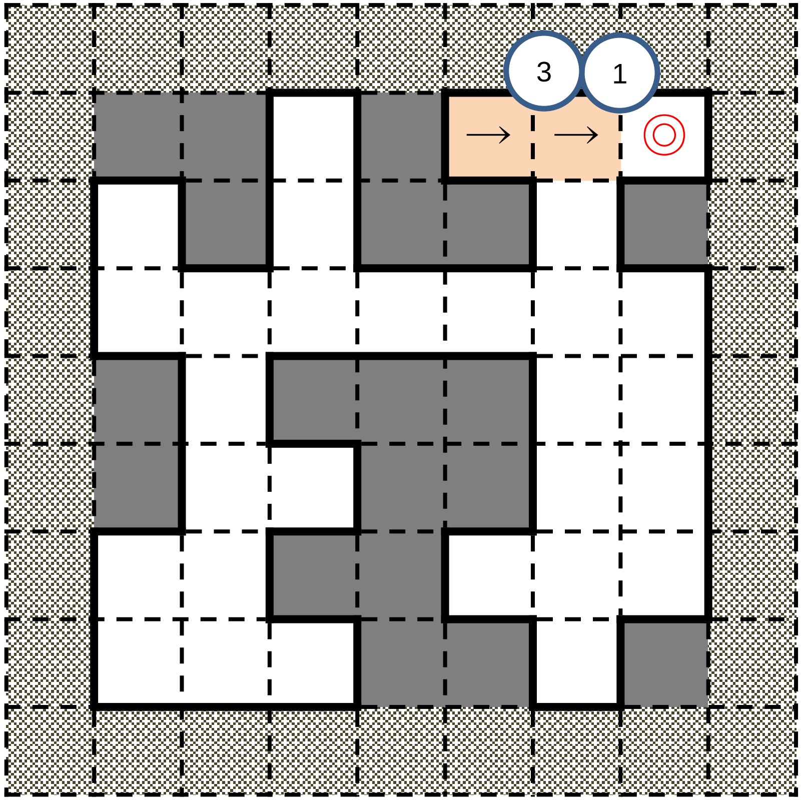


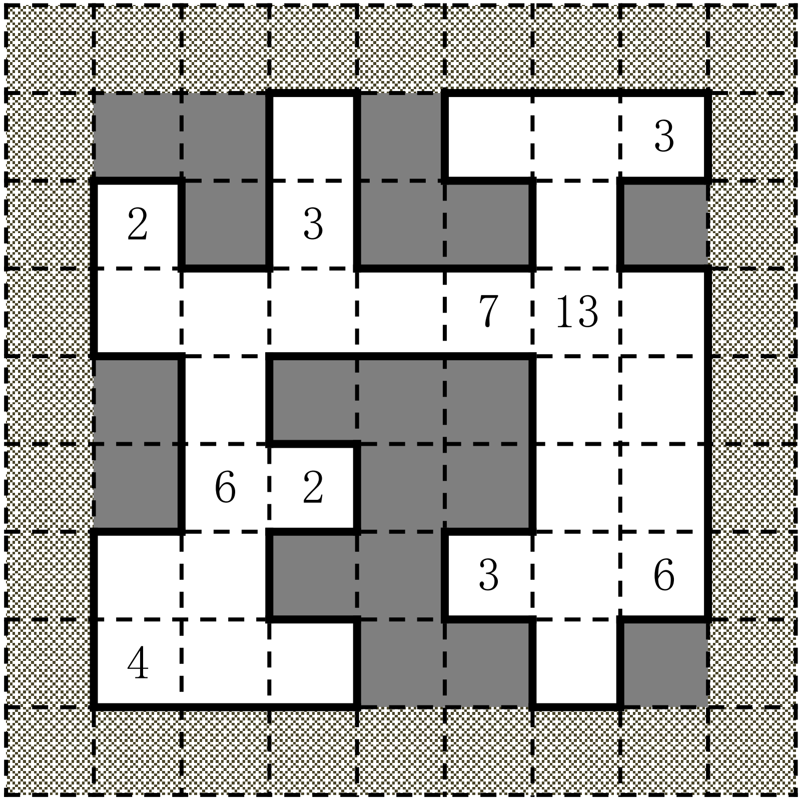


集約法：白マスver

特別なマスを基準に矢印と数値を与える

隣接するマスで**向かい合う矢印**を置いてはいけない
矢印の向いてる先に**自分と違う色のマス**があってはいけない
矢印の向いてる先の隣接するマスの**数値が小さい**

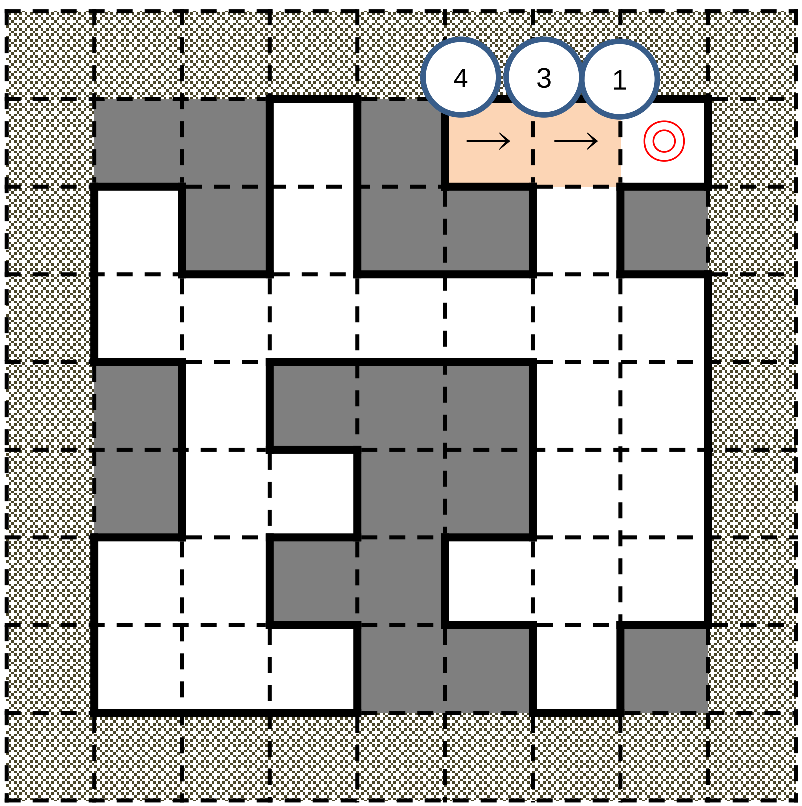


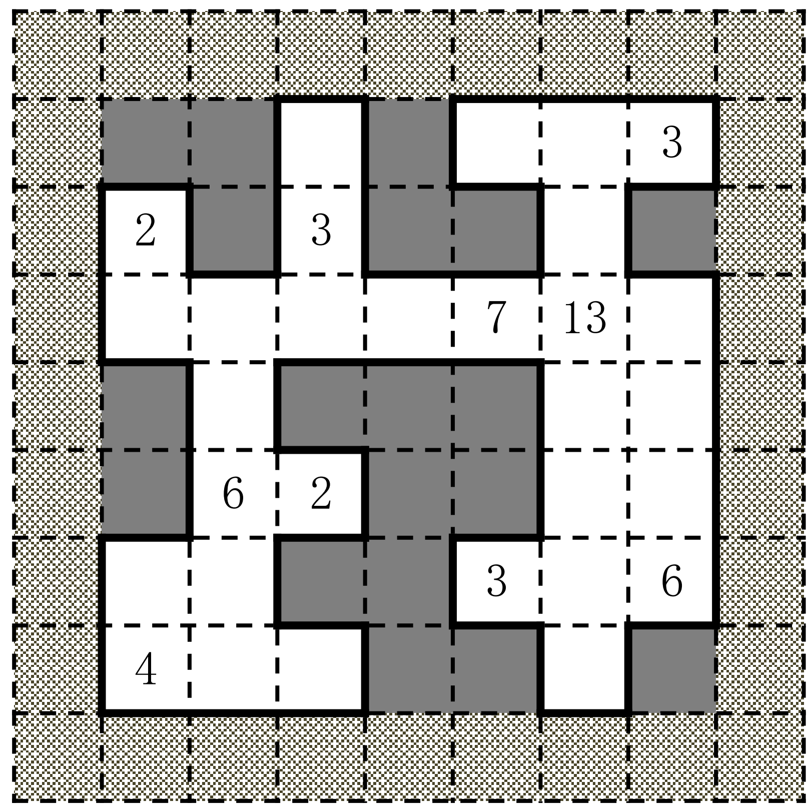


集約法：白マスver

特別なマスを基準に矢印と数値を与える

隣接するマスで**向かい合う矢印**を置いてはいけない
矢印の向いてる先に**自分と違う色のマス**があってはいけない
矢印の向いてる先の隣接するマスの**数値が小さい**

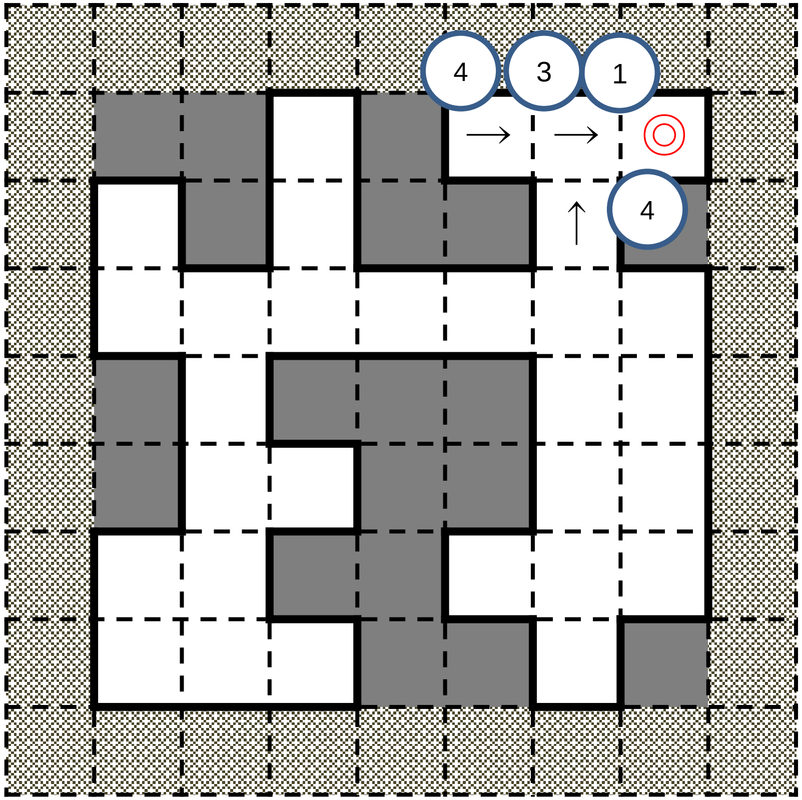


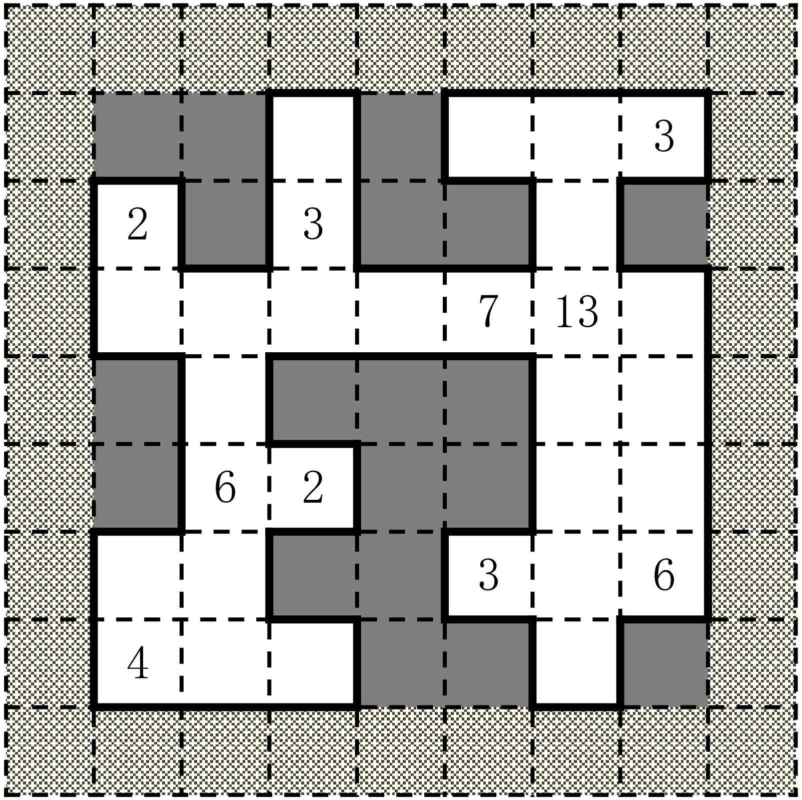


集約法：白マスver

特別なマスを基準に矢印と数値を与える

隣接するマスで**向かい合う矢印**を置いてはいけない
矢印の向いてる先に**自分と違う色のマス**があってはいけない
矢印の向いてる先の隣接するマスの**数値が小さい**

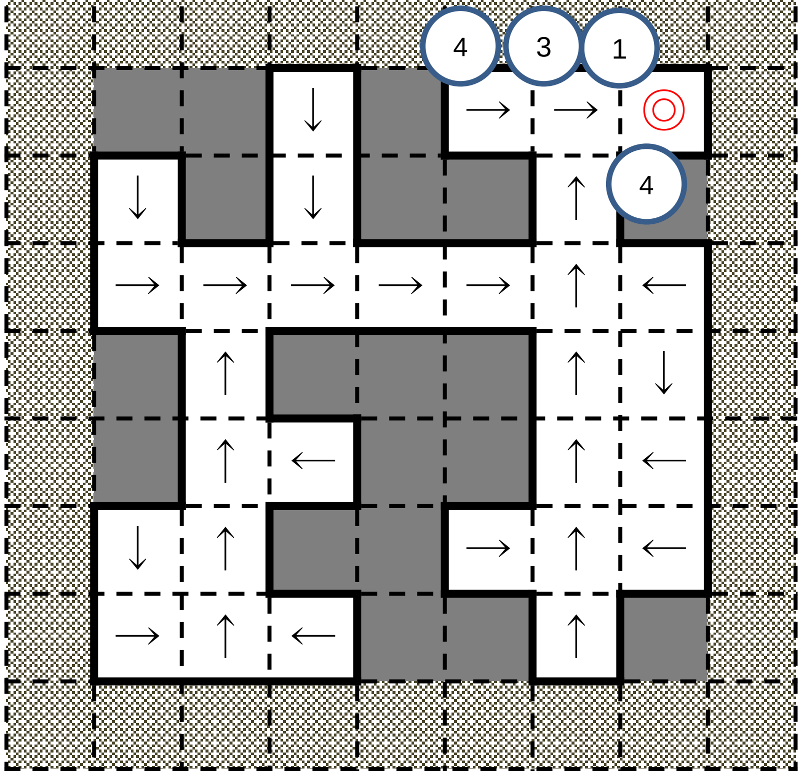


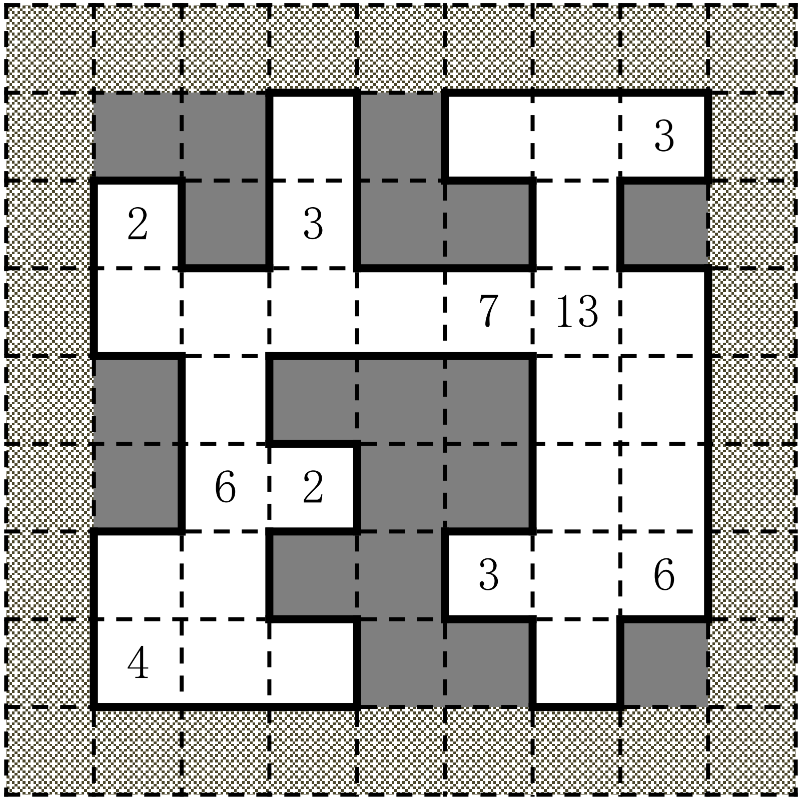


集約法：白マスver

特別なマスを基準に矢印と数値を与える

隣接するマスで**向かい合う矢印**を置いてはいけない
矢印の向いてる先に**自分と違う色のマス**があってはいけない
矢印の向いてる先の隣接するマスの**数値が小さい**





切除平面法

ルール②～④を満たす制約

複数の閉路を許す解

領域の数を調べる

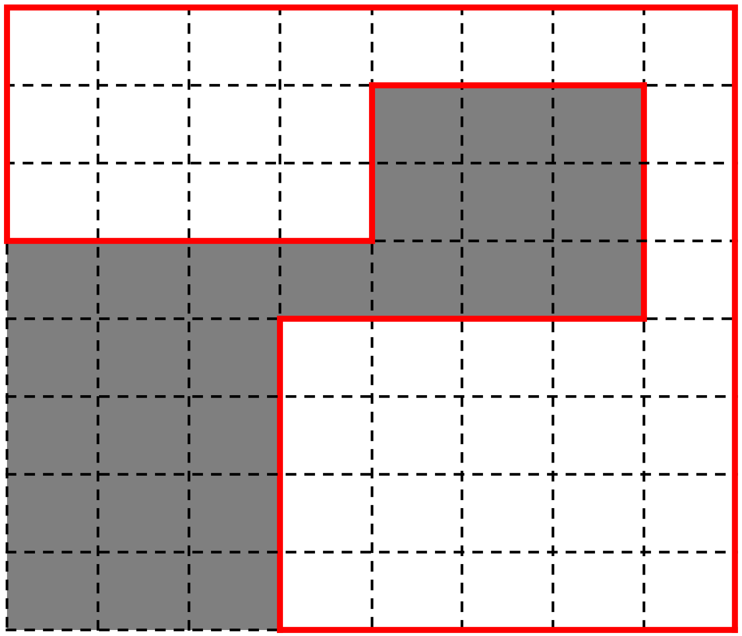
①点線に線を引き1つの単純閉路を作る

②数字マスは閉路まで数字分マスが並ぶ

③数字マスはループ線の内側に入る

④線は交差，枝分かれしない

切除平面法

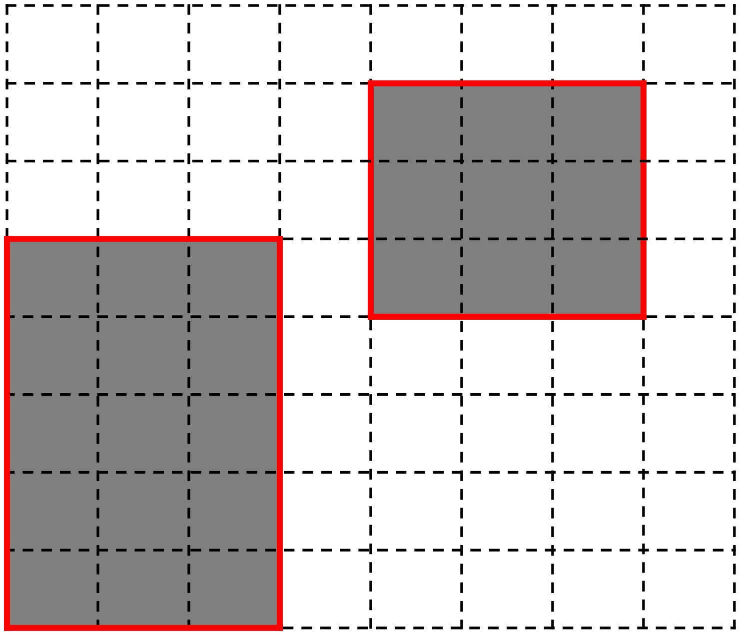


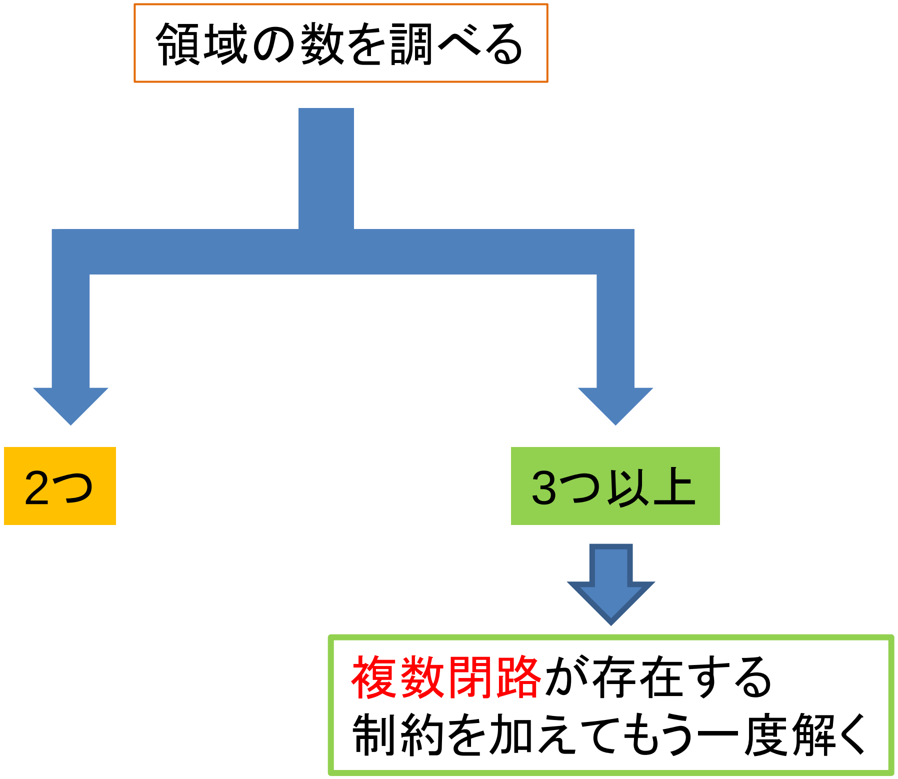
領域の数を調べる

2つ

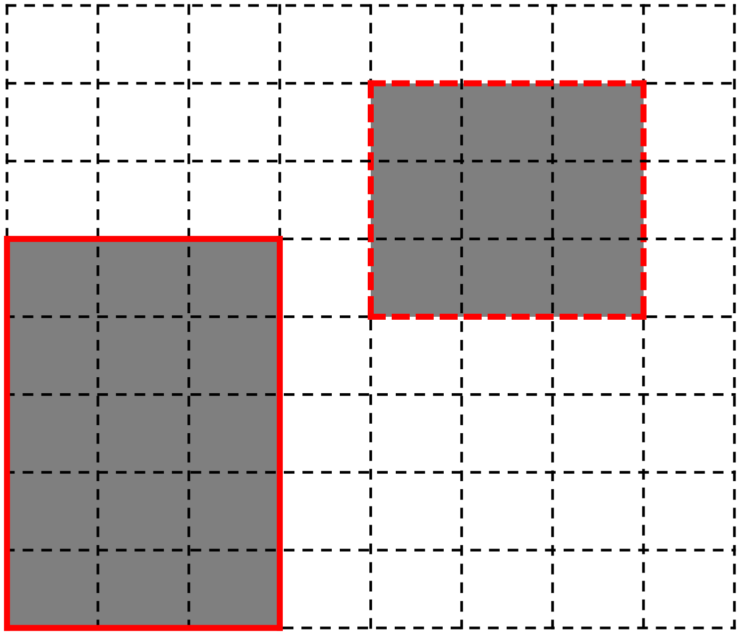
単純閉路になっている
Corral Puzzleの解となっている

切除平面法





切除平面法

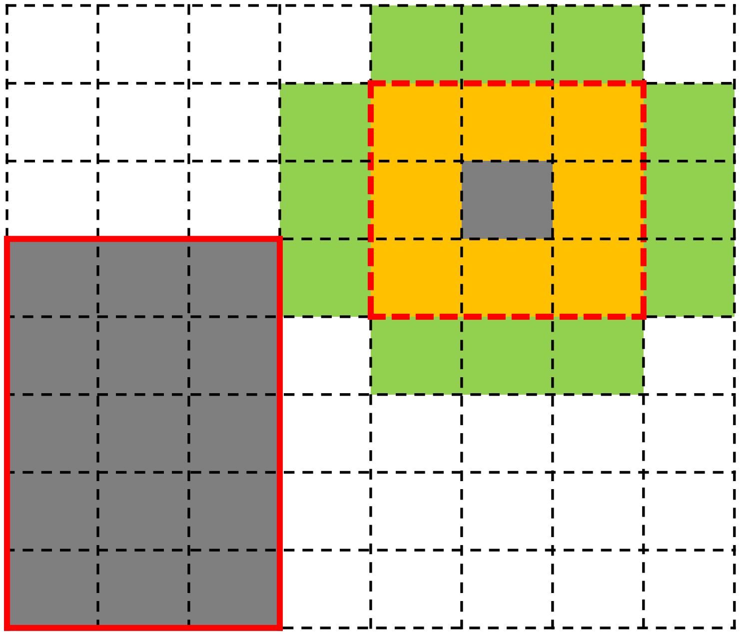


複数の閉路から最小の閉路領域を調べる

↓

一番小さい閉路領域の**閉路**を構成する
内側の輪と**外側の輪**を用いて
閉路の生成を禁止する制約を加える

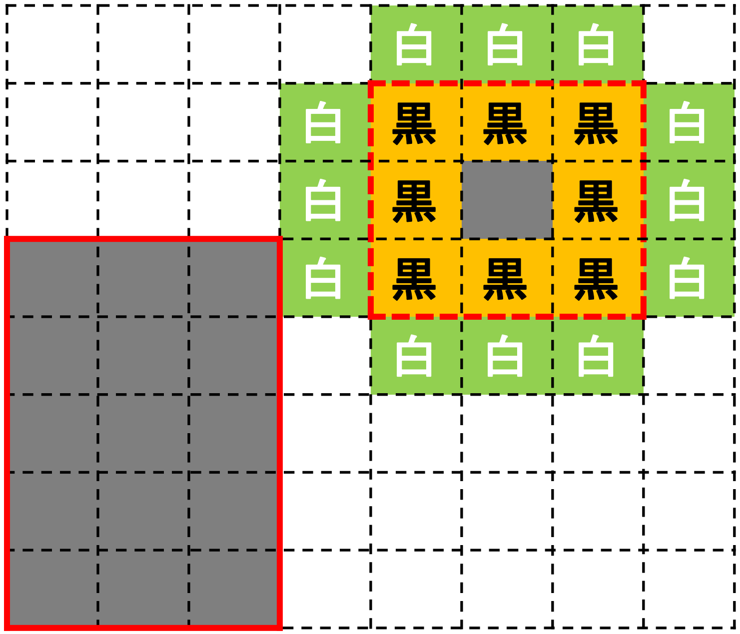
切除平面法



複数の閉路から最小の閉路領域を調べる

一番小さい閉路領域の**閉路**を構成する
内側の輪と**外側の輪**を用いて
閉路の生成を禁止する制約を加える

切除平面法

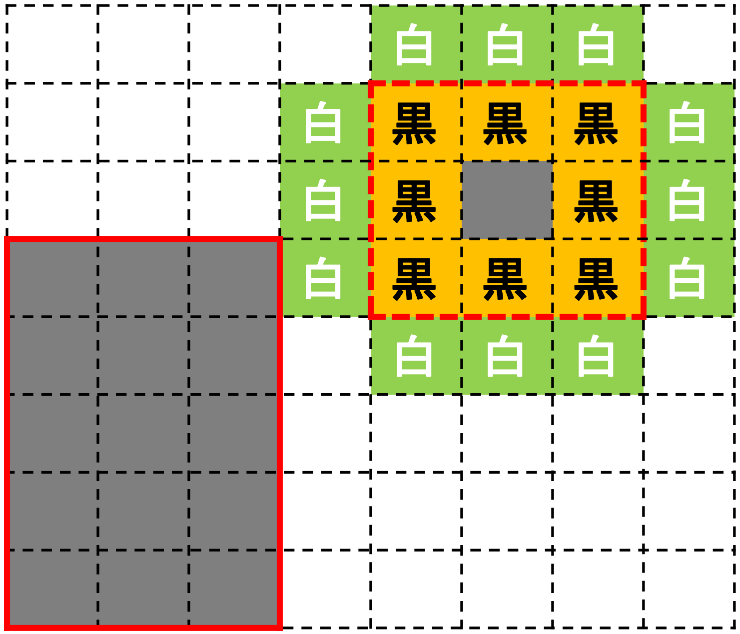


複数の閉路から最小の閉路領域を調べる

一番小さい閉路領域の閉路を構成する
内側の輪と外側の輪を用いて
閉路の生成を禁止する制約を加える

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

切除平面法



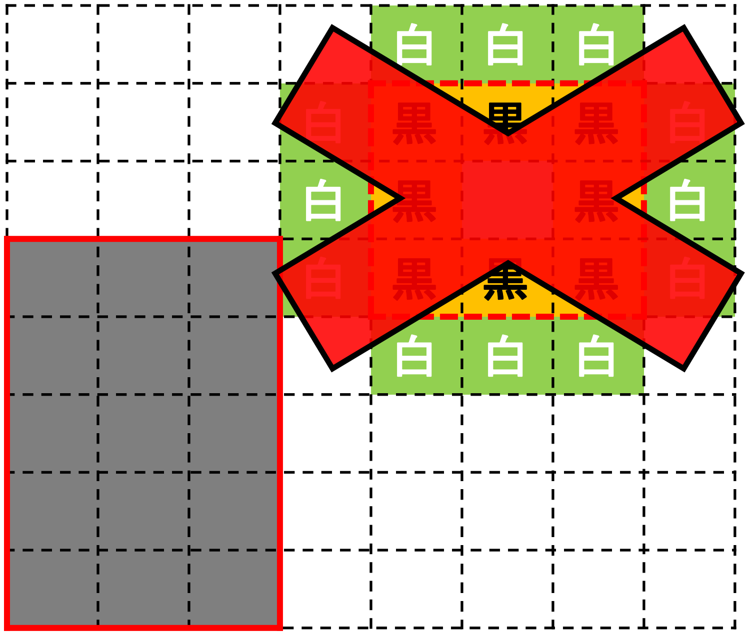
複数の閉路から最小の閉路領域を調べる

一番小さい閉路領域の閉路を構成する
内側の輪と外側の輪を用いて
閉路の生成を禁止する制約を加える

実行可能解が変わり別解が導かれる

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

切除平面法



複数の閉路から最小の閉路領域を調べる

一番小さい閉路領域の閉路を構成する
内側の輪と外側の輪を用いて
閉路の生成を禁止する制約を加える

実行可能解が変わり別解が導かれる

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

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？
  - 集約法 vs. 切除平面法
  - 目的関数の設定による計算効率の違いは？

評価方法

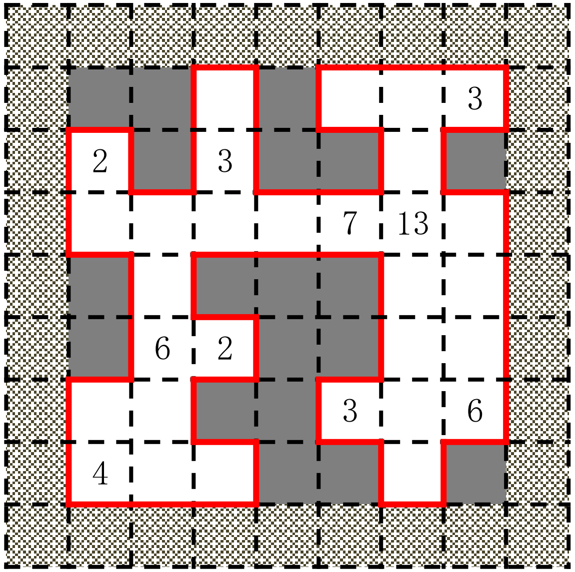
- ニコリパズルの逆襲(10×10盤面，難易度別5題ずつ)を対象
  - ★3×5題，★4×5題，★5×5題の計15問
- 1時間を上限に計測し，3回平均で評価

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）

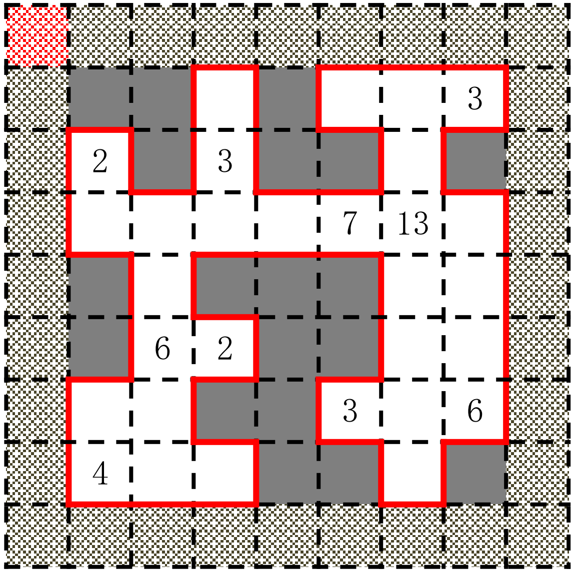
実験的評価(2)：目的関数の設定

- 影響を与えない式
  - $maximize: b(0,0)$
  - 必ず1となる変数の最大化
- 白マスの総数の最大化
  - $maximize: \sum w(i,j)$
  - 解として白マスを最も多くする
- 白マスの総数の最小化
  - $minimize: \sum w(i,j)$
  - 解として白マスを最も少なくする



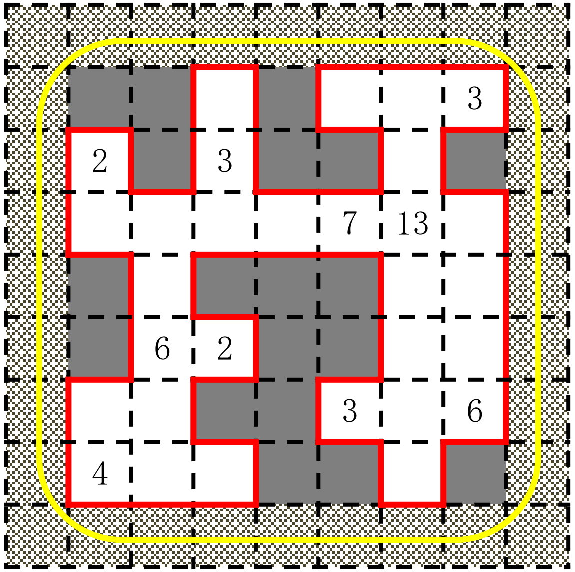
実験的評価(2)： 目的関数の設定

- 影響を与えない式
  - $maximize: b(0,0)$
  - 必ず1となる変数の最大化
- 白マスの総数の最大化
  - $maximize: \sum w(i,j)$
  - 解として白マスを最も多くする
- 白マスの総数の最小化
  - $minimize: \sum w(i,j)$
  - 解として白マスを最も少なくする



実験的評価(2)：目的関数の設定

- 影響を与えない式
  - $maximize: b(0,0)$
  - 必ず1となる変数の最大化
- 白マスの総数の最大化
  - $maximize: \sum w(i,j)$
  - 解として白マスを最も多くする
- 白マスの総数の最小化
  - $minimize: \sum w(i,j)$
  - 解として白マスを最も少なくする



問題番号	数字マス数（最大・最小）	集約法			切除平面法			
		制約数	変数	時間(s)	制約数	変数	時間(s)	反復数
★3_1	14 （4・2）	4,811	1,737	―	1033	585	17.83	75
★3_2	14 （19・2）	5,004	1,826	―	1165	674	20.98	14
★3_3	16 （19・2）	5,006	1,820	989.34	1154	668	0.57	1
★3_4	11 （19・2）	4,882	1,775	―	1090	623	105.66	61
★3_5	12 （19・2）	4,840	1,756	58.20	992	604	1.40	5
★4_1	13 （5・2）	4,780	1,726	―	1076	574	747.76	150
★4_2	16 （6・2）	4,905	1,776	―	1274	624	1982.03	223
★4_3	14 （19・3）	4,965	1,806	20.22	1062	630	2.30	3
★4_4	12 （12・2）	4,879	1,772	―	1070	620	49.01	45
★4_5	14 （19・2）	4,920	1,787	―	1082	635	88.46	16
★5_1	17 （5・2）	4,918	1,779	―	1108	627	14.50	44
★5_2	15 （9・2）	4,843	1,750	―	1018	598	16.86	29
★5_3	19 （4・3）	5,006	1,815	―	1212	663	136.21	60
★5_4	15 （10・2）	4,867	1,762	―	1055	610	9.96	42
★5_5	18 （11・2）	4,959	1,795	―	1110	643	1.67	5

寄与なし

問題番号	数字マス数（最大・最小）	集約法			最大化 切除平面法			
		制約数	変数	時間(s)	制約数	変数	時間(s)	反復数
★3_1	14 （4・2）	4,811	1,737	―	992	585	51.59	34
★3_2	14 （19・2）	5,004	1,826	220.41	1,170	674	35.31	19
★3_3	16 （19・2）	5,006	1,820	4.14	1,155	668	0.58	1
★3_4	11 （19・2）	4,882	1,775	898.58	1,117	623	235.58	88
★3_5	12 （19・2）	4,840	1,756	8.76	996	604	3.00	9
★4_1	13 （5・2）	4,780	1,726	―	944	574	102.69	17
★4_2	16 （6・2）	4,905	1,776	―	1,052	624	―	50
★4_3	14 （19・3）	4,965	1,806	72.12	1,061	630	1.70	2
★4_4	12 （12・2）	4,879	1,772	―	1,081	620	210.80	55
★4_5	14 （19・2）	4,920	1,787	215.91	1,073	635	32.93	6
★5_1	17 （5・2）	4,918	1,779	859.50	1,069	627	30.59	4
★5_2	15 （9・2）	4,843	1,750	―	993	598	3.21	3
★5_3	19 （4・3）	5,006	1,815	―	1,185	663	191.92	32
★5_4	15 （10・2）	4,867	1,762	15.85	1,016	610	0.72	2
★5_5	18 （11・2）	4,959	1,795	15.24	1,107	643	1.03	1

問題番号	数字マス数（最大・最小）	集約法			最小化 切除平面法			
		制約数	変数	時間(s)	制約数	変数	時間(s)	反復数
★3_1	14 （4・2）	4,811	1,737	―	1051	585	37.10	93
★3_2	14 （19・2）	5,004	1,826	186.08	1152	674	1.76	1
★3_3	16 （19・2）	5,006	1,820	3.46	1154	668	0.20	0
★3_4	11 （19・2）	4,882	1,775	―	1041	623	13.79	12
★3_5	12 （19・2）	4,840	1,756	4.47	995	604	2.19	8
★4_1	13 （5・2）	4,780	1,726	―	927	574	―	321
★4_2	16 （6・2）	4,905	1,776	―	1333	624	873.81	281
★4_3	14 （19・3）	4,965	1,806	3.41	1061	630	1.10	2
★4_4	12 （12・2）	4,879	1,772	9.67	1029	620	2.25	3
★4_5	14 （19・2）	4,920	1,787	―	1079	635	11.02	12
★5_1	17 （5・2）	4,918	1,779	―	1140	627	20.01	75
★5_2	15 （9・2）	4,843	1,750	―	1041	598	40.28	51
★5_3	19 （4・3）	5,006	1,815	―	1266	663	692.03	113
★5_4	15 （10・2）	4,867	1,762	―	1052	610	12.44	38
★5_5	18 （11・2）	4,959	1,795	―	1113	643	3.53	7

評価・考察(1): 寄与なし

- 集約法
  - 解けた問題数:3問
  - 解ける問題も少なく, 解けた場合でも時間がかかる
- 切除平面法
  - 解けた問題数:15問
  - 全問解くことができたが, 計算に時間のかかるものがあった
- 考察
  - 問題数 : × 集約法 ◎ 切除平面法
  - 計算時間 : × 集約法 △ 切除平面法

評価･考察(2)： 白マス数の最大化

- 集約法
  - 解けた問題数：9問
  - 3種類の目的関数では解ける問題数が1番多い
- 切除平面法
  - 解けた問題数：14問
  - 1問解けなかったが，寄与しない場合に比べて全体的に計算時間が速く繰り返し回数も少ない
- 考察
  - 問題数　：　○ 集約法　○ 切除平面法
  - 計算時間：　△ 集約法　○ 切除平面法

評価･考察(3)：白マス数の最小化

- 集約法
  - 解けた問題数：5問
  - 他の目的関数と比べた時，解けたものならば計算時間が一番短い
- 切除平面法
  - 解けた問題数：14問
  - 繰り返し回数が多く，計算に時間のかかる問題例があった
- 考察
  - 問題数　：　△ 集約法　○ 切除平面法
  - 計算時間：　○ 集約法　△ 切除平面法

まとめと今後の課題

- まとめ

集約法 vs. 切除平面法

LOSE WIN

  - 目的関数による計算効率

集約法:総和の最大化 ☆ 切除平面法:寄与しない

- 今後の課題
  - 大きい盤面サイズでの検証
  - 集約法の洗練
  - より良い目的関数の提案

13×13 1問
集約法× 切除平面法○

10×18 1問
集約法× 切除平面法○

18×18 1問
集約法× 切除平面法×