平成 23 年度

学士学位論文

数独の少数ヒント問題の生成に関する研究

Study on Generation of Sudoku Problems with Fewer Clues

1120254　　那須　律政

指導教員　　松崎　公紀

2012 年 3 月 1 日

高知工科大学 情報システム工学科

要 旨

数独の少数ヒント問題の生成に関する研究

那須　律政

数独は，世界的に人気のあるペンシルパズルの一種である．また，数学やゲーム研究，コンピュータ工学では研究対象として注目されている．

数独において研究されている問題に，最少ヒント問題がある．これは，唯一解を持つ問題を構成できる最少のヒント数はいくつであるかという問題である．本研究では，数独の問題を計算機に生成させる際の生成手法について着目し，ヒント数の少ない問題を生成するための方針について，実際に問題生成プログラムを作成し実験を行った．

まず，数独の問題を入力すると自動で解探索を行い，その問題の解が唯一解であるか，複数解であるか，解なしであるかを判定するプログラムを作成した．また，そのプログラムの評価実験を行い，9×9 の数独に関して実用上高速に解探索が可能であることを確認した．

次に，一定の方針にしたがって数独の問題を生成し，解探索を行うプログラムを作成した．そして，問題の生成実験を行い，完全ランダムな問題生成との比較を行った．その結果，完全ランダムでは生成できた問題の最少ヒント数が 21 であったのに対して，方針に従って問題を生成するプログラムでは，ヒント数 18 の問題を生成することに成功した．

キーワード　数独　最少ヒント問題　候補数字　問題生成

Abstract

Study on Generation of Sudoku Problems with Fewer Clues

Norimasa NASU

Sudoku is puzzle a kind of pencil puzzle, and popular on global. And more, Sudoku is target of Research to Mathmatics, game programing and computer science. In Sudoku, the problem of the minimum clue has been researched. This problems focus is the smallest clues that can be unique answer of problem in sudoku. Focus of this research is method of generate for problem of sudoku to computer, and made program for generate of sudoku. Focus of this research is method of generate for problem of sudoku to computer, and made program for generate of sudoku.

First, made program for search answer of sudoku problem. This program searching answer, and judging answer number of problem. Result of the test, program to solve the problem so fast, that passes the test.

Next, made and tested program for generate of sudoku. Result of the test, the successful generation of 18 clues problem.

key words Sudoku, Minimum clues of sudoku, Generate of Sudoku problems.

目次

図目次

表目次

第 1 章

はじめに

1.1 研究背景と研究目的

数独とは，ペンシルパズルの一種である [1].『数独』という名称は株式会社ニコリが商標登録しているため，他社の雑誌や書籍では『ナンバープレイス』と呼ばれることがある．数独は世界的に人気のあるパズルであり，海外でも『Sudoku』や『Number place』という名で親しまれている．2006 年にはイタリアで世界大会が開催された．また，数学やゲーム研究，コンピュータ工学の世界では研究対象としても注目されている [2] [3] [4]．数独を含むパズル研究を行う目的は，問題を解くだけでなく，問題を生成する，パズルの性質を明らかにするなど，様々な目的がある [1]．これまでに行なわれた数独の研究例では，推論規則 [5]，難易度の推定 [2]，制約充足問題としての検討 [3]，解盤面の数え上げ [4] などがある．これらの研究からパズルの性質を解明し，数理的な知見を得ることがパズル研究を行う意義である．

数独においてかねてより研究・議論されてきた問題のひとつとして，最少ヒント問題がある．最少ヒント問題とは，唯一解を持つ問題を構成できる最少のヒント数はいくつであるかという問題である．最少ヒント問題については，先行研究により一定の成果が挙がっている．9×9 の数独の最少ヒント数は，McGuire によって 17 個であることが証明された [6]．Hitting-Set Algorithm を用いて問題を単純化した上でヒント数 16 の問題を解き，700 万 CPU 時間をかけて唯一解を持つ問題が無いことを確認した [7]．16×16 の最少ヒント数は未だに証明されていないが，2011 年 4 月 9 日時点で白川俊博氏によってヒント数 56 個の問題が発見されている [7].

9 × 9 の問題を解くために 700 万 CPU 時間を要していることから，問題の規模が大きい 16 × 16 では，同じ方法での証明は困難である．そのため，別の方法によるアプローチを考える必要がある．そこで本研究では，計算機を用いて数独の問題生成を行う際の生成手法に着目し，ヒント数の少ない問題を得るための方針について議論する．ヒント数の少ない問題を効率的に生成できる生成手法を通じて，問題が持つ性質についての知見を得ることで最少ヒント問題を解決するための足がかりとなることを目標とする．

1.2　関連研究

数独における関連研究としては，立石らによって数独の問題を論理式に変換し，制約充足問題として解くというアプローチが行なわれている [3]．また，井上らによって 9 × 9 の問題がとり得るすべての解盤面の数え上げが行われている [4]．この研究では，数独の問題の対称性を利用し解の探索空間を削減した．また，数独の問題を解くために用いる解法を推論し，問題の難易度を自動的に正しく判定することを目的とした研究が，松原によって行なわれている [5].

1.3　本論文の構成

本論文の構成は以下のようになっている．第 2 章では，数独の基本ルールと数独を解く際の解法について説明する．第 3 章では数独を自動で解くプログラムの作成，第 4 章では数独の問題を生成するプログラムの作成について述べ，実験をおこなった結果とそれに対する考察を行っている．そして，第 5 章で本論文についてまとめ，今後の展望を述べている．

1.4　本研究における実験環境

本研究では，計算機を用いた評価実験を複数行った．本研究で行われた実験は，すべて以下の環境の計算機を使用していることをあらかじめ示しておく．

1.4　本研究における実験環境

- プロセッサ：Intel(R) Core(TM) i3 540 @ 3.07GHz
- メモリ：4GB
- OS:Windows7 Home Premium 32bit

第 2 章

数独

2.1　数独の基本ルール

数独とは，マス目で区切られた正方形の盤面に数字を書き込んでいくパズルである．ここでは，最も一般的な 9 × 9 の数独を例に挙げて説明する．問題として，図 2.1 のような盤面が与えられる．盤面は 3 × 3 ごとに太線で区切られており，この区切られた領域をブロックという．また，盤面にはあらかじめ数字が配置されており，この数字をヒントという．

与えられた盤面に対して，以下の規則にしたがって 1〜9 の数字を入れていく．

規則 1　1 つのマスには 1 つの数字が入る

規則 2　すべての横列と縦列において，1 つの列には 1〜9 の数字が 1 つずつ入る

規則 3　すべてのブロックにおいて，1 つのブロックには 1〜9 の数字が 1 つずつ入る

この規則にしたがって，すべてのマスに数字を入れることができれば完成となる．図 2.1 の問題の解は，図 2.2 となる．図 2.2 のように，すべてのマスに数字がはいった盤面を解盤面という．

2.2　数独の解法

人間が数独の問題を解く際には，第 2.1 節で述べた基本ルールとそれによって生まれる規則性を用いて，各マスに入り得る数字の候補を絞り込んでいく作業を行う．各マスに入る数字の候補を，候補数字という．本節では解法の中でも頻繁に用いられる解法について，9 × 9 の数独を例に挙げて説明する．

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

図 2.1　数独における問題の一例

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

図 2.2　図 2.1 の問題の解盤面

2.2.1　基本的な解法

まず最初に，数独の基本ルールに沿った解法である，Line Unique，Block Unique，Cell Unique について説明する．

Line Unique は，基本ルールの 1 つである “すべての横列と縦列において，1 つの列には 1〜9 の数字が 1 つずつ入る” というルールに沿った解法である．Line Unique のルールを，以下のように定義する．

解法：Line Unique

ある横列あるいは縦列において，任意の数字 x が入る可能性のあるマスがいずれか 1 マスだけであるとき，そのマスは数字 x に確定する．

図 2.3 の盤面を例に説明する．y1 列に注目する．(x1,y2) に 1 が入っているため，同じブロック内の (x1,y1)，(x2,y1)，(x3,y1) には 1 が入らないことがわかる．同様に，(x9,y3) の 1 によって (x7,y1)，(x8,y1)，(x9,y1) にも 1 が入らない．また，(x4,y4) に 1 があるため，同じ縦列の (x4,y1) には 1 が入らない．同様に，(x6,y7) の 1 によって (x6,y1) にも 1 が入らない．

y1 列について，1 が入らないマスに斜線を引くと図 2.4 のようになる．図 2.4 を見ると，y1 列の中で 1 を入れることができるマスは (x5,y1) だけとなる．よって，(x5,y1) は 1 に確

	x1	x2	x3	x4	x5	x6	x7	x8	x9
y1									
y2	1								
y3									1
y4				1					
y5									
y6									
y7						1			
y8									
y9									

図 2.3　Line Unique が成立する盤面の例

	x1	x2	x3	x4	x5	x6	x7	x8	x9
y1									
y2	1								
y3									1
y4				1					
y5									
y6									
y7						1			
y8									
y9									

図 2.4　図 2.3 の盤面において 1 が入らないマスに斜線を引いた盤面

定する．

Block Unique は，基本ルールの 1 つである “すべてのブロックにおいて，1 つのブロックには 1〜9 の数字が 1 つずつ入る” というルールに沿った解法である．Block Unique のルールは，以下のとおりである．

解法：Block Unique

あるブロックにおいて，任意の数字 x が入る可能性のあるマスがいずれか 1 マスだけであるとき，そのマスは数字 x に確定する．

図 2.5 の盤面を例に説明する．x4 から x6, y4 から x6 の中央のブロックに注目する．(x5,y2) に 3 があるため，同じ縦列の (x5,y4),(x5,y5),(x5,y6) には 3 が入らないことがわかる．同様に，(x6,y9) に 3 があるため，(x6,y4),(x6,y5),(x6,y6) には 3 が入らない．また，(x7,y5) に 3 があるため，同じ横列の (x4,y5),(x5,y5),(x6,y5) には 3 が入らない．同様に，(x2,y6) に 3 があるため，(x4,y6),(x5,y6),(x6,y6) には 3 が入らない．

中央のブロックについて，3 が入らないマスに斜線を引くと図 2.6 のようになる．図 2.4 を見ると，中央のブロックの中で 3 を入れることができるマスは (x4,y4) だけとなる．よって，(x4,y4) は 3 に確定する．

	x1	x2	x3	x4	x5	x6	x7	x8	x9
y1									
y2					3				
y3									
y4									
y5							3		
y6		3							
y7									
y8									
y9						3			

図 2.5　Block Unique が成立する盤面の例

	x1	x2	x3	x4	x5	x6	x7	x8	x9
y1									
y2					3				
y3									
y4									
y5							3		
y6		3							
y7									
y8									
y9						3			

図 2.6　図 2.5 の盤面において 3 が入らないマスに斜線を引いた盤面

Cell Unique は，数独の基本ルールである “1 つの横列，縦列，ブロックには，1〜9 までの数字が 1 つずつ入る” というルールに沿った解法である．Cell Unique のルールは，以下のとおりである．

解法：Cell Unique

あるマスに入り得る候補数字が数字 x だけになったとき，そのマスに入る数字は x に確定する．

図 2.7 の盤面を例に説明する．(x5,y1) のマスに着目すると，同じブロックに 1,3,4,8,9 がある．そのため，(x5,y1) にはこれらの数字は入らないことになる．また y1 列には，上記の数字のほかに 2,7 が存在する．さらに x5 列には，6 が存在する．そのため，(x5,y1) にはこれらの数字は入らない．この時点で (x5,y1) の候補数字は，5 だけとなる．したがって，(x5,y1) は 5 に確定する．

	x1	x2	x3	x4	x5	x6	x7	x8	x9
y1		2		1		3		7	
y2				8		4			
y3						9			
y4									
y5					6				
y6									
y7									
y8									
y9									

図 2.7　Cell Unique が成立する盤面の例

2.2.2　中級解法

次に，中級レベル以上の問題で用いられる，Naked Pair と Hidden Pair について説明する．Naked Pair と Hidden Pair は，どちらも列あるいはブロックにおける候補数字の組に注目し，候補数字を絞る解法である．

Naked Pair のルールは以下の通りである．

解法：Naked Pair

ある縦列または横列，あるいはブロックにおいて，n 個のマスに着目した時，着目したマスに入る数字が n 種類に限定される場合，他のマスにはその数字は入らない．

図 2.8 の盤面を例に，n=2 の場合について説明する．図 2.8 は，x1 から x3，y1 から y3 のブロックを拡大した盤面である．(x1,y1) に 7，(x1,y2) に 8 が入っており，他のマスの小さい数字はそのマスの候補数字を表わしている．(x2,y2) と (x3,y2) の 2 つのマスに注目すると，この 2 つのマスに入る数字は 5 または 6 しか入らないことが解る．この場合，(x2,y2) と (x3,y2) 以外のマスに 5 あるいは 6 が入ると，注目したマスに入れる数字が無くなるため，このブロックの他のマスには 5 と 6 は入らないことになる．この解法を適用すると，図 2.9 のように候補数字を絞ることができる．

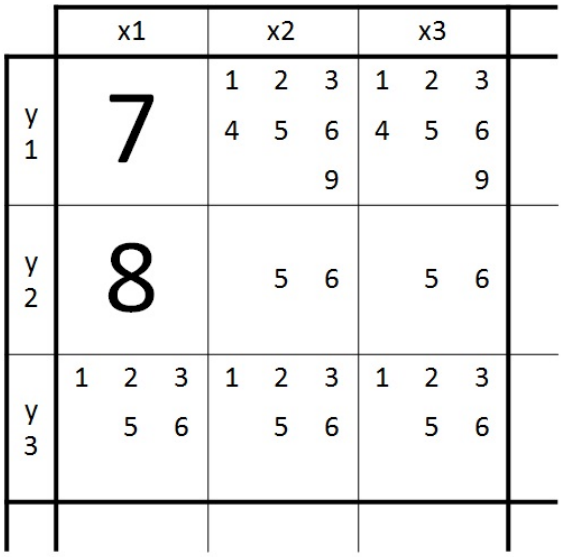


図 2.8　Naked Pair が成立する盤面の例

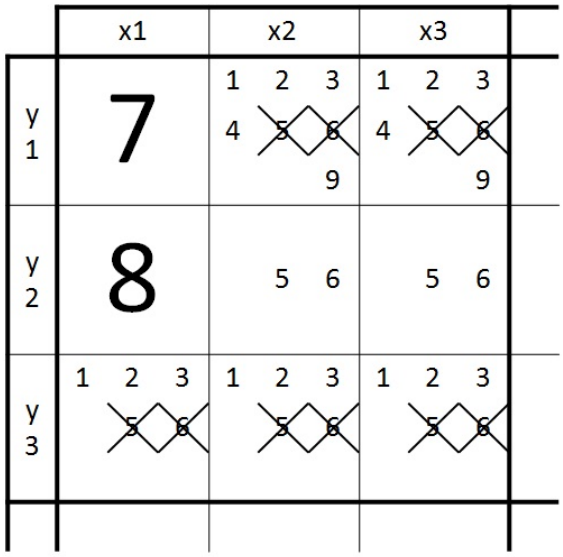


図 2.9　図 2.8 の盤面に Naked Pair を適用した盤面

Hidden Pair のルールは以下の通りである．

解法：Hidden Pair

ある縦列または横列，あるいはブロックにおいて，n 種類の数字に着目したとき，その数字が入るマスが n 個に限られる場合，そのマスには着目した数字以外は入らない．

図 2.10 の盤面を例に，n=2 の場合について説明する．(x2,y1) と (x3,y1) に着目すると，このブロック内で 4 と 9 が入るのはこの 2 つのマスだけであることがわかる．(x2,y1) と (x3,y1) に他の数字を入れると，ブロック内に 4 または 9 を入れるマスが無くなるため，この 2 つのマスには 4 と 9 以外は入らないことになる．この解法を適用すると，図 2.11 のように候補数字を絞ることができる．

2.2.3　上級解法

ここでは，上級問題で使われる X-Wing について説明する．

X-Wing のルールは，以下の通りである．

解法：X-Wing(1)

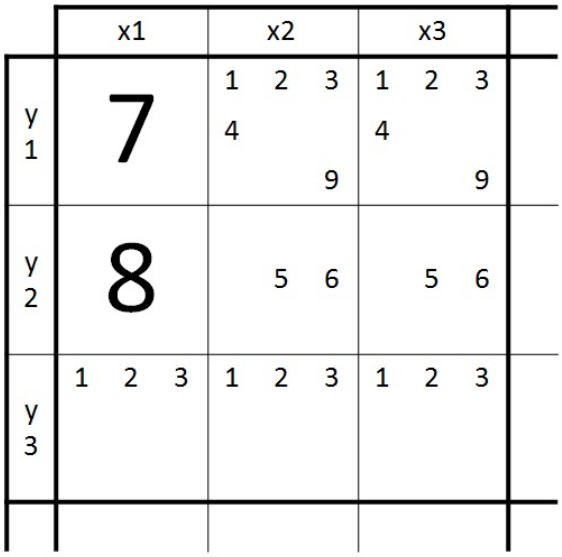


図 2.10　Hidden Pair が成立する盤面の例

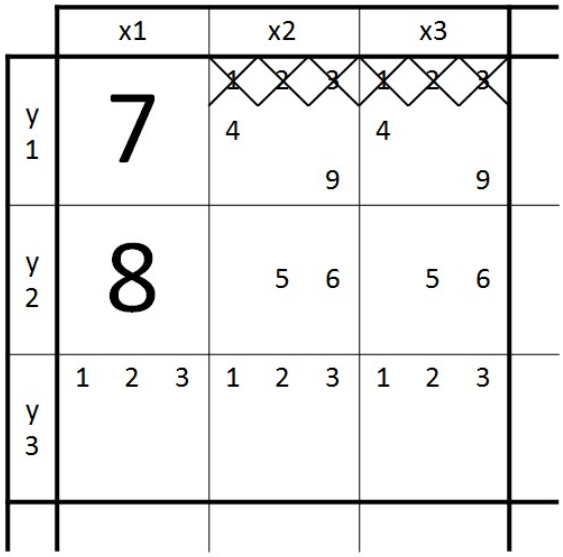


図 2.11　図 2.10 の盤面に Hidden Pair を適用した盤面

ある n 列の横列に数字 x が入る可能性のあるマスがそれぞれ n 箇所ずつしかない場合で、その n 箇所のマスがともに同じ縦列に位置しているとき、その縦列の他のマスには数字 x は入らない

解法：X-Wing(2)

ある n 列の縦列に数字 x が入る可能性のあるマスがそれぞれ n 箇所ずつしかない場合で、その n 箇所のマスがともに同じ縦列に位置しているとき、その縦列の他のマスには数字 x は入らない

X-Wing という名称は $n = 2$ の場合にのみ用いられている名称であり，$n = 3$ の場合は Sword fish，$n = 4$ の場合は Jelly fish という名称が用いられる．n の数が異なるだけで操作は同じであるため，本論文では X-Wing に統一する．

図 2.12 の盤面を例に，$n = 2$ の場合について説明する．x があるマスが，数字 x を候補数字として持つマスとする．y2 列と y6 列に注目すると，どちらも x3 列と x6 列にだけ x があることがわかる．この場合，この 2 列に数字 x が入る組み合わせは，(x3,y2) と (x6,y6)，もしくは (x6,y2) と (x3,y6) の組み合わせのどちらかに限定される．どちらの組み合わせであっても，x3 列と x6 列の他のマスには x が入ることはないため，候補数字から除くことが

	x1	x2	x3	x4	x5	x6	x7	x8	x9
y1	x		x	x				x	
y2			x			x			
y3		x		x		x			
y4									
y5		x		x	x		x		
y6			x			x			
y7	x	x	x		x	x			
y8			x						
y9			x			x	x		

図 2.12　X-Wing が成立する盤面の例

	x1	x2	x3	x4	x5	x6	x7	x8	x9
y1	x		x	x				x	
y2			x			x			
y3		x		x		x			
y4									
y5		x		x	x		x		
y6			x			x			
y7	x	x	x		x	x			
y8			x						
y9			x			x	x		

図 2.13　図 2.12 の盤面に X-Wing を適用した盤面

できる．X-Wing を適用して，候補数字を絞った盤面を図 2.13 に示す．

2.3　数独の問題生成におけるルール

数独の問題を作成する場合，以下のような制約がある．

制約 1　1 つの問題に対して解はただ 1 つでなければならない

制約 2　試行錯誤を必須としてはいけない

2 の試行錯誤とは，“あるマスにある数字が入ると仮定して解き進める” ことを指す．試行錯誤を用いれば，盤面の全探索が可能になるため理論上すべての問題を解くことができる．しかし，“盤面から規則性を見つけ出し解を求める” というパズル性が失われることや，探索空間が膨大であるため人間には解くことが困難であることから，人間が解く数独の問題については，試行錯誤が必須な問題は禁止されている．

本研究では計算機に問題を解かせることを前提としているため，ルール 2 は考えずルール 1 のみを遵守することとする．

第 3 章

数独自動解答プログラム

3.1 数独自動解答プログラムの作成

ある問題が唯一解であるかどうかの判定を行うためには，実際にその問題に対して解の探索を行う必要がある．そこで，数独の問題を入力すると，自動で問題を解き，その問題の解盤面の数が唯一解であるか，複数解であるか，あるいは解盤面を持たないかを判定するプログラムを Java 言語を用いて作成した．実際に問題を解く部分には，節 2.1 で述べたルールを実装し，このルールを用いて問題を解くように作成した．また，それらのルールに加えて，解法で解けずに行き詰った場合の処理として，再帰処理による仮置きを実装した．仮置きとは，解を 1 つ以上持つ問題に対してバックトラックを用いて解く解法である．任意の空きマスに，そのマスの候補数字から 1 つを入れ，それが正しいと仮定して問題を解き進める．解き進める途中で矛盾が発生して問題が解けなかった場合，仮置きした数字が間違いであることがわかるので，その数字を候補数字から除くことができる．

作成した自動解答プログラムの擬似コードを以下に示す.

数独自動解答プログラムの擬似コード

```
Solve メソッド（盤面，候補数字テーブル）{
  現在の解盤面と候補数字テーブルを一時保管する
  for(無限ループ){
    候補数字テーブルを更新する

    盤面に X-Wing を適用する
    盤面に Pair Naked, Pair Hidden を適用する

    盤面に Line Unique を適用する
    盤面に Block Unique を適用する
    盤面に Cell Unique を適用する

    if(問題が解けた){
      if(新しい解盤面である){
        解の数+1;
      }
    return;
    }
    if(矛盾がある){
      盤面と候補数字テーブルを一時保管したものに戻す
      return;
    }

    if(盤面と候補数字がループ先頭から変化していない){
      break;
    }
  }

  空いているマスのうち，候補数字の数が最少のマス（x,y）を探す

  for(i = 1; i <= 9; i++){
    if(マス（x,y）の候補数字に i が含まれている){
      マス（x,y）に i を入れる
      Solve メソッド呼び出し（盤面，候補数字テーブル）
      if(発見した解盤面が 2 個以上){
        return;
      }
    }
  }

  盤面と候補数字を一時保管したものに戻す
  return;
}
```

表 3.1　9 × 9 の問題の解探索にかかった時間

	計算時間 [ミリ秒]
最小	15
最大	249
幾何平均	29.7

3.2　自動解答プログラムの評価実験

作成した自動解答プログラムについて，9 × 9 の問題と 16 × 16 の問題が，実用上十分に高速な時間で解けるかどうかを確認するための実験を行った．9 × 9 の問題については，Web 上の数独問題集 [9] から上級レベルの問題を 30 問無作為に選択し，解を求めるまでの時間を測定した．16 × 16 の問題については，GPCC2011 内の，16 × 16 最少ヒント問題の解答 [7] に掲載されている，ヒント数 56〜60 の問題を使用した．

9 × 9 の問題を入力したときの，実験結果を表 3.1 に示す．また，問題ごとの計算時間の累積度数分布を図 3.2 に示す．表 3.1 にある通り最大でも 249 ミリ秒，幾何平均では 29.72 ミリ秒という結果だった．また，図 3.2 からわかるように，90%以上の問題が約 70 ミリ秒で解けている．一問あたり 1 秒未満，ミリ秒単位の時間で問題を解くことができたため，唯一解をもつ 9 × 9 の問題については高速に解くことができた．

次に，16 × 16 の問題を入力したときの，実験結果を表 3.2 に示す．16 × 16 に関しては問題による差が大きく，ヒント数 58 の問題については解決に 9 時間 59 分 7 秒を要した．第 4 章で行う問題生成実験では，問題生成と解探索を繰り返す必要がある．1 問に 10 時間かかっている今回のプログラムでは，16 × 16 の問題生成に用いることが困難であるため，さらなる高速化が必要である．

3.2　自動解答プログラムの評価実験

図 3.1　解探索にかかった時間のヒストグラム

表 3.2　16 × 16 の問題の解探索にかかった時間

	計算時間
ヒント数 60	30 秒
ヒント数 59	22 分 56 秒
ヒント数 58	9 時間 59 分 7 秒
ヒント数 57	1 分 24 秒
ヒント数 56	52 秒

第 4 章

ヒント数の少ない数独問題の生成プログラム

4.1 完全ランダムな問題生成の評価

まず，完全ランダムに数字を配置して問題を生成するプログラムを作成した．ただし数独のルールに反しないための最低限の処理として，すべての縦列，横列，ブロックにおいて数字が重複する配置は行わないようにした．作成した問題生成プログラムの擬似コードを以下に示す．

ランダムな問題生成プログラムの擬似コード

```
while(盤面に配置されたヒント数 < 指定したヒント数){
  盤面から，ランダムに空きマス (x,y) を選択する
  マス (x,y) に，数字 1〜9 のうち 1 つをランダムに入れる
  if(いずれかの列あるいはブロックにおいて数字が重複している){
    マス (x,y) を空きマスに戻す
  }
}
```

作成したプログラムについて，性能の評価を行うために，問題の生成実験を行った．実装した手法にしたがって，指定したヒント数の問題を 10000 問生成して解探索を行い，問題が持つ解盤面の数と問題解決までの時間を計測した．実験結果を表 4.1 に示す．実験結果を見ると，ヒント数 20 以下の唯一解を持つ問題の生成が全くできていないことがわかる．したがって，完全ランダムな問題生成では，唯一解を持ちヒント数が少ない問題の生成は困難であることがわかった．また，複数解を持つ問題と解を持たない問題の生成数を比較すると，ヒント数 17〜21 のいずれにおいても複数解を持つ問題の方が多く生成されている．

表 4.1　ランダム生成した問題の解探索結果と平均時間

ヒント数	唯一解	複数解	解無し	時間 [ミリ秒]
21	3	5948	4049	133
20	0	6817	3183	242
19	0	7728	2272	788
18	0	8295	1705	1315
17	0	8826	1174	1554

4.2　生成手法を用いた問題生成プログラム

前節で述べたように，完全ランダムな問題生成では，複数解をもつ問題が生成されやすいことがわかった．そこで，できるだけ解の候補が少なくなるような問題の生成手法を検討した．解の候補を少なくするということは，できるだけ盤面の各マスに入る数字の候補を減少させるということになる．つまり，各マスの候補数字の数を減少させることができれば，問題に対する解盤面の数を減少させることができると考え，この方針をもとに問題生成を改良した．

改良したプログラムの擬似コードを以下に示す．

改良した問題生成プログラムの擬似コード

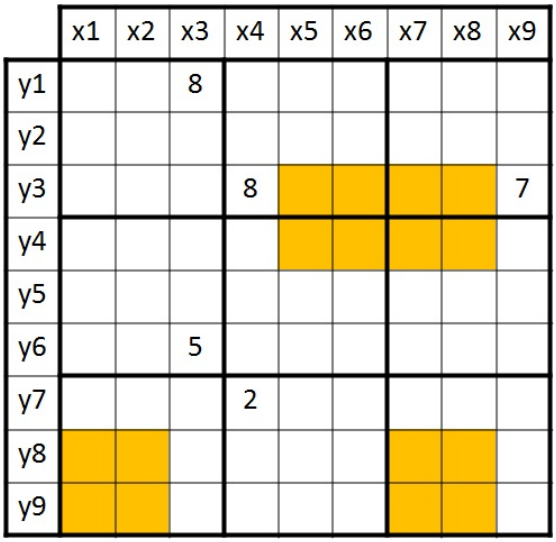
```
while(盤面に配置されたヒント数 < 指定したヒント数){
  min = 32 ビット整数の最大値
  table = 空リスト

  for(y = 0; y < 9; y++){
    for(x = 0; x < 9; x++){

      if(マス (x,y) に数字が入っている){
        continue;
      }

      for(i = 1; i < 9; i++){
        マス (x,y) に数字 i を入れる
        候補数字テーブルを配置しなおす
        now = すべてのマスの候補数字の個数
        if(now >= min){
          if(now > min){
            table を初期化
          }

          table にマス (x,y) と入れた数字 i の組み合わせを保存
        }
      }
      マス (x,y) を空に戻す
    }
  }
  //  table には，全マスの候補数字の個数が最少になる
  // (x,y) と数字 i の組み合わせが保存されている
  table からランダムに 1 つ選びその組み合わせマス (x,y) に数字 i を配置する
}
```

問題生成の例を，図 4.1 の盤面を例に説明する．

図 4.1 の盤面において，すべてのマスの候補数字の合計は 596 個ある．ここに数字を 1 つ配置してすべてのマスの候補数字の個数の合計を最少にするためには，色の濃いマスのいずれかに，1,3,4,6,9 のいずれかの数字を入れることで最少の 567 個になる．この中から，ランダムに 1 つ選択して，数字を配置する．ここでは，(x7,y9) に 4 を置いたとする．数字を配置すると，図 4.2 のような盤面になる．

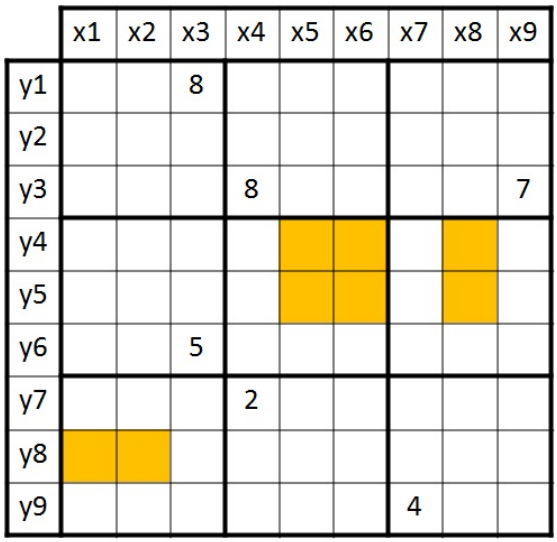
次に，図 4.2 の盤面においてすべてのマスの候補数字の個数の合計を最少にするためには，色の濃いマスのいずれかに，1,3,6,9 のいずれかの数字を入れることで最少の 538 個になる．この中から，ランダムに 1 つ選択して，数字を配置する．(x8,y5) に 3 を置いたとす

4.2　生成手法を用いた問題生成プログラム



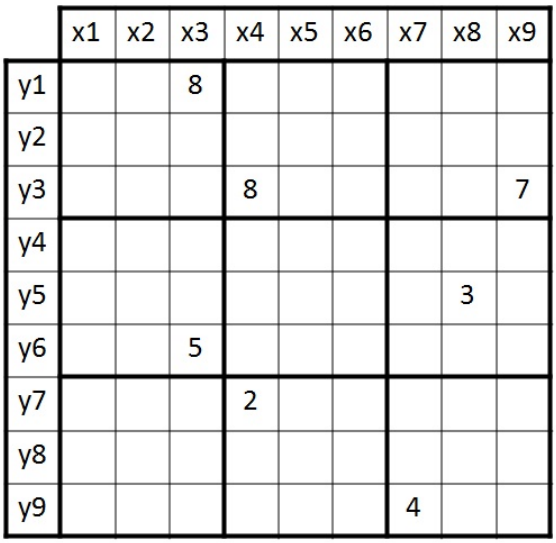
	x1	x2	x3	x4	x5	x6	x7	x8	x9
y1			8						
y2									
y3				8					7
y4									
y5									
y6			5						
y7				2					
y8									
y9									

図 4.1　盤面生成の例



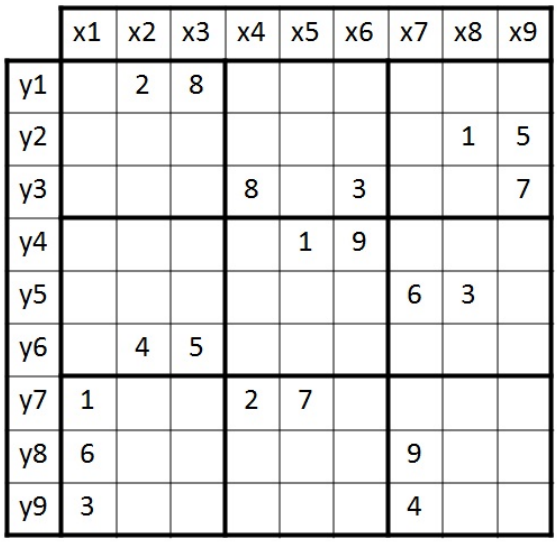
	x1	x2	x3	x4	x5	x6	x7	x8	x9
y1			8						
y2									
y3				8					7
y4									
y5									
y6			5						
y7				2					
y8									
y9							4		

図 4.2　(x7,y9) に 4 を置いた場合の盤面



	x1	x2	x3	x4	x5	x6	x7	x8	x9
y1			8						
y2									
y3				8					7
y4									
y5								3	
y6			5						
y7				2					
y8									
y9							4		

図 4.3　(x8,y5) に 3 を置いた場合の盤面



	x1	x2	x3	x4	x5	x6	x7	x8	x9
y1		2	8						
y2								1	5
y3				8		3			7
y4					1	9			
y5							6	3	
y6		4	5						
y7	1			2	7				
y8	6						9		
y9	3						4		

図 4.4　生成された盤面の一例

ると，図 4.3 のような盤面になる．

このようにして，順に数字を置いていき問題を生成する．ヒント数 20 の問題を生成した時の，生成された盤面の一例を図 4.4 に示す．図 4.4 の盤面における候補数字の総数は，231 個である．

作成した問題生成プログラムについて，性能の評価を行うために，問題の生成実験を行った．実装した手法にしたがって，指定したヒント数の問題を 10000 問生成して解探索を行い，問題が持つ解盤面の数と問題解決までの時間を計測した．

表 4.2　手法を用いて生成した問題の解探索結果と平均時間

ヒント数	唯一解	複数解	解無し	時間 [ミリ秒]
21	82	1386	8532	186
20	61	3624	6315	212
19	32	6995	2973	271
18	7	9064	929	320
17	0	9615	385	323

生成実験の結果を表 4.2 に示す.

表 4.2 にあるとおり，改良した問題生成プログラムでヒント数 18 までの問題生成に成功した．また，表 4.1 と比較すると，ヒント数 19〜21 において複数解を持つ問題の生成数が減少していることと，解なしの問題の生成数が増加していることから，1 問あたりの解盤面数を減少させることには成功していると予想する．問題生成の途中で解探索を行い，解なしになる場所にヒントを置かないようにすれば，唯一解を持つ問題生成の精度がさらに向上すると考える.

また，最少ヒント数であるヒント数 17 の問題の生成には成功していないため，生成手法のさらなる改良が必要である．今回の実験で，盤面の候補数字の個数に注目した生成手法が有効であることがわかった

第 5 章

まとめ

数独における最小ヒント問題への足がかりとして，ヒント数の少ない問題を自動で生成させることを目的として，ヒント数の少ない問題を得るための方針について実験を行った．問題生成を行うにあたって，生成した問題が唯一解であるかどうかを確かめる必要がある．そこでまず，数独の問題を入力として自動で解探索を行い，問題の解が唯一解であるかそうでないかを判定する数独自動解答プログラムを作成した．プログラムには人間が数独を解く際に用いる解法を実装した．作成したプログラムについて性能評価のための実験を行ったところ，9×9 の問題では，最長 249 ミリ秒，幾何平均で 29.72 ミリ秒という結果がでたため，実用上十分に高速に問題を解くことができた．一方 16×16 の問題では，問題によって時間差が大きく 1 問に約 10 時間かかる問題も存在したため，実用上高速に解くことができなかった．そのため，自動解答プログラムの高速化が今後の課題となる．

次に，数独の問題を自動生成するプログラムを作成した．まず，完全にランダムに数字を配置して問題を作成するプログラムを作成して，9×9 の問題の生成実験を行って，唯一解の問題がいくつ生成されるかを実験した．その結果，ヒント数 21 の問題生成に成功したものの，ヒント数 20 以下の問題は生成できなかった．また，生成された問題のうち，複数解を持つ問題の生成率が高い傾向が見られた．

そこで，できるだけ解盤面の候補を少なくするために，各マスの候補数字の数が少なくなるように数字を配置するように問題生成プログラムを改良して，再度問題生成実験を行った．その結果，ヒント数 18 の問題を生成することに成功した．また，複数解の問題の生成数が減少し，解なしの問題の生成数が増加したことから，問題の解盤面の候補を減少させることができたと考えた．今後の目標は，9×9 のヒント数 17 の問題生成と，16×16 の問

題の生成実験を行うことである．9 × 9 のヒント数 17 の問題生成については，今回の実験で，候補数字に注目した問題生成が有効であることが解ったので，これをもとにさらに改良を加える必要がある．16 × 16 の問題の生成実験を行うことについては，自動解答プログラムの動作速度の遅さがネックであるため，自動解答プログラムの高速化を行う必要がある．

謝辞

高知工科大学情報システム工学科，松崎公紀准教授にはプログラムのコーディングやデバッグ，論文の添削など，様々な場面で多くのご助力をいただきました．先生のお力添えあって，なんとか本研究を形にすることができました．心より感謝いたします．また，ご多忙にも関わらず貴重なお時間を割いていただき研究に関して貴重なご意見をいただきました，東京大学大学院情報理工学系研究科，美添一樹助教に心より感謝いたします．

また，本研究の副査をお引き受けくださった高田喜朗准教授，酒井敬一講師に心より感謝いたします．

最後に，本研究だけでなく様々な場面で相談に乗ってくださった，松崎研究室のメンバーおよび吉田研究室の滝優基氏に感謝いたします．

参考文献

[1] 小谷　善行：“ゲーム情報学におけるパズル研究”, pp107–111, 情報処理 , Vol.53, No.2, 通巻 563 号, 一般社団法人 情報処理学会, 2012.

[2] 小場　隆行, 中所　武司：“数独の難易度判定アプリケーションの提案と評価”, 情報処理学会研究報告. GI, [ゲーム情報学] 2011-GI-25(8), 1-6, 2011.

[3] 立石　匡，湊　真一：“二分決定グラフを用いた数独パズルの解探索と列挙”，全国大会講演論文集 第 70 回平成 20 年 (5), ”5-253”-”5-254”, 2008.

[4] 井上　真大，奥乃　博：“本質的に異なる数独解盤面の列挙と番号付け”，全国大会講演論文集 第 71 回平成 21 年 (4), ”4-741”-”4-742”, 2009.

[5] 松原　康夫：“数独の推論規則と難易度に関する考察”，情報処理学会研究報告， EC, エンタテインメントコンピューティング 2006(134), 1-6, 2006.

[6] Gary McGuire : “There is no 16-Clue Sudoku: Solving the Sudoku Minimum Number of Clues Problem”, eprint arXiv:1201.0749, 2012.

[7] 白川　俊博：“数独のヒント最少問題の解答”, http://hp.vector.co.jp/authors/VA003988/gpcc/11p1.htm, 2011.

[8] Felgenhauer and Jarvis : “Enumerating possible Sudoku grids”, 2005.

[9] “数独無料ゲーム - 数独問題集”, http://www.sudokugame.org/, 2012.