モンテカルロ木探索手法に基づく数独の多数ヒント盤面の生成

松崎　公紀[1*]　那須　律政[2]

(受領日：2013年5月10日)

[1] 高知工科大学情報学群
〒782-8502　高知県香美市土佐山田町宮ノ口185

[2] 高知工科大学大学院工学研究科
〒782-8502　高知県香美市土佐山田町宮ノ口185

* E-mail: matsuzaki.kiminori@kochi-tech.ac.jp

要約：数独は，世界的に人気のあるペンシルパズルであり，コンピュータ科学における研究対象としてもよく用いられるものである．本論文は，どのヒントを取り除いても唯一解とならないような数独の盤面におけるヒント数の最大値がいくつであるかを問う「数独の最多ヒント問題」に関連するものである．特定の性質を持つような数独の問題を生成することは，盤面の種類が多い（探索空間が広い）ことと，その性質を表すことが容易でないことから難しい．そこで，碁などのゲームプログラミング分野において近年研究が進められているモンテカルロ木探索手法をこの問題に適用して，多数のヒントからなる盤面を効率良く求めるアルゴリズムとその実装を与える．特に，モンテカルロ木探索では，乱数を用いて終局盤面までプレイするプレイアウトの品質が重要であるため，その点についていくつか手法を提案し実験を行った．その結果，これまでに知られているヒント数35には至らないものの，ヒント数33の問題を生成することに成功した．

1. はじめに

数独は，世界的に人気のあるペンシルパズルである．また，人が解くパズルとしてだけでなく，コンピュータ科学における研究対象としてもよく用いられる．例えば，数独の盤面の性質から難易度を評価する研究[1,2,3,4]や，数独そのものの性質に関する研究[5]のほか，汎用的なアルゴリズムやソルバの応用[6,7,8,9]などにも用いられている．

本論文は，数独の最多ヒント問題に関係するものである．数独の最多ヒント問題とは，解盤面が唯一であるような盤面であり，かつ，その盤面に含まれるどのヒントを取り除いても解が複数となってしまうような盤面を構成できるヒントの数の最大値を求めるものである．この問題は，比較的良く調べられている最少ヒント問題[10]とは逆の問題であり，Games and Puzzles Competitions on Computers（GPCC）の2013年の問題として採用されている[11]．

多数ヒント盤面を生成することはそれほど容易ではない．困難さの理由は大きく次の2つである．1つ目の理由は，数独盤面の数（探索空間）が非常に大きいことである．例えば，9×9の盤面にヒントをランダムに30個配置することを考える．問題として成り立たないものや対称性を除去せずに考えると，盤面はおよそ10^{62}通りにもなる．2つ目の理由は，ヒント数が多くなる盤面が持つ特徴が分からないことである．

この問題に対して本研究では，ゲームプログラミング分野で研究されてきたモンテカルロ木探索手法[12]を適用する．モンテカルロ木探索は，乱数を用いたシミュレーション（これをプレイアウトと呼ぶ）によって近似解を求めるモンテカルロ法をゲーム木探索に組み合わせたものである．モンテカルロ木探索の特長は，探索空間が膨大であるが適切な評価関数が作りにくいゲームにおいて有効であることであり，コンピュータ囲碁において勝率を劇的に

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

図 1. 数独における問題の一例

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

図 2. 図 1 の問題の解盤面

向上させている[13]．上に述べたように，数独の多数ヒント盤面の生成にあたっても同様の状況が成り立つと考え，モンテカルロ木探索を数独の多数ヒント盤面の生成に適用することを考えた．

本論文では，モンテカルロ木探索を利用して数独の多数ヒント盤面を生成するアルゴリズムを提案・実装し，実験によりその効果を調査する．特にプレイアウトの質を向上させることがモンテカルロ木探索の性能に強く影響することが知られているため[13]，本論文ではプレイアウトの改良を中心に述べる．実験の結果，これまでに知られている最多のヒント数 35 の問題を生成することはできなかったが，最終的にはモンテカルロ木探索によりヒント数 33 の問題の生成にまで成功した．

本論文の構成は以下の通りである．第 2 節では数独およびその最多ヒント問題についての導入を行う．第 3 節ではモンテカルロ木探索のアルゴリズムを復習する．第 4 節では，数独の多数ヒント問題を生成するアルゴリズムとその実装における工夫を述べ，第 5 節で実験の結果をまとめる．第 6 節で関連研究を述べ，第 7 節でまとめと今後の課題を述べる．

2. 数独とその最少ヒント問題

数独とは，図 1 のようにマス目で区切られた正方形の盤面にいくつかの数字があらかじめ配置されたものから始め，マスに数字を書き込んでいき，図 2 のようにすべてのマスに数字を入れるパズルである．本論文では，最も一般的な 9 × 9 の数独のみを扱う．まず，数独の盤面に関係する用語を定義する．

ヒント 盤面にあらかじめ配置されている数字をヒントと呼ぶ．

ブロック 数独の盤面のうち，太線で区切られた 3 × 3 の領域をブロックと呼ぶ．

数独では，マスに入る数は次の条件（ルール）を満たす必要がある．

条件 1 1 つのマスには 1 つの数字が入る．

条件 2a 各行には 1〜9 の数字が 1 つずつ入る．

条件 2b 各列には 1〜9 の数字が 1 つずつ入る．

条件 2c 各ブロックには 1〜9 の数字が 1 つずつ入る．

これらのルールにしたがって，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．

解盤面 上記の数独のルールを満たすようにすべてのマスに数字が入ったその盤面を解盤面と呼ぶ．

唯一解 初期盤面のヒントをすべて含む解盤面が唯一であるとき，その初期盤面は唯一解を持つと呼ぶ．

複数解 初期盤面のヒントをすべて含む解盤面が複数存在するとき，その初期盤面は複数解を持つと呼ぶ．

解なし 初期盤面のヒントをすべて含む解盤面が存在しないとき，その初期盤面は解なしと呼ぶ．

人が数独の問題を解く際には，上記の数独のルールとそれによって生まれる規則性を用いる．ルールから導かれる規則性は解法とも呼ばれる．人が数独を解く際には，それぞれの空マスに，そのマスに入りうる数の候補を記すことがある．それらは，候補数字もしくはペンシルマークと呼ばれる．

候補数字 空マスにおいて，上記の数独のルールもしくはその他の解法により取り得ないと判定された数を除いた，解の候補となる数字を候補数字と呼ぶ．

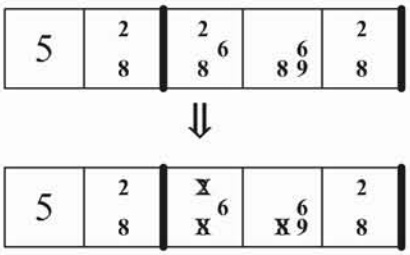

図 3. 2-Naked の例: 5 の右のマスと右端のマスについて，候補数字が 2 と 8 のみである．したがって，この行の他のマスには 2 と 8 が解として入ることはありえない．

本論文中にて用いる解法を以下に示す．

Cell-Unique あるマス x について，x が持つ候補数字が a の 1 つだけであるとする．このとき，x の数字を a に確定する．これは，上記の条件 1 に対応する解法である．

Line/Block-Unique ある行，列，ブロックについて，数字 a が入り得るマスが x の 1 つだけであるとする．このとき，x の数字を a に確定する．これは，上記の条件 2a，2b，2c に対応する解法である．

***k*-Naked** ある行，列，ブロックについて，k 個のマス $x_1, x_2, \ldots, x_k$ が持つ候補数字が k 個の数 $a_1, a_2, \ldots, a_k$ の部分集合のみからなるとする．このとき，その行，列，ブロックの $x_1, x_2, \ldots, x_k$ 以外のマスの候補数字から，$a_1, a_2, \ldots, a_k$ を消す．

***k*-Hidden** ある行，列，，ブロックについて，k 個の数 $a_1, a_2, \ldots, a_k$ のいずれかを候補数字として持つマスが $x_1, x_2, \ldots, x_k$ の k マスのみであるとする．このとき，$x_1, x_2, \ldots, x_k$ の候補数字から，$a_1, a_2, \ldots, a_k$ 以外の候補数字を消す．

解法 2-Naked を適用することで候補数字を絞り込む例を図 3 に示す．なお，列やブロックにおける未決定なマスの数を n とすると，k-Hidden は $(n-k)$-Naked と同じ結果となる．したがって，9×9 の数独においては，$k = 2, 3, 4$ について k-Naked と k-Hidden を考えれば十分である．

最後に本論文の目標である，最多ヒント問題を定義する．人が解くパズルの場合には，解盤面を求めるのに試行錯誤を必要としないという暗黙のルールがあるが，試行錯誤の範囲の明確な定めがないので，ここでは唯一解であるかのみを問う．

最多ヒント問題 唯一解である盤面であり，どのヒントを除いても複数解となるような盤面を構成できる最も多いヒント数はいくつか．

```
MCT(root) {
  loop until 終了条件 {
    leaf <- select_downwards(root)
    leaf.n <- leaf.n + 1
    if (expand_cond(leaf)) {
      leaf <- expand(leaf).first_child
    }
    board = playout(leaf.board)
    update_upwards(leaf, getvalue(board))
  }
  return select_best_child(root)
}
```

図 4. モンテカルロ木探索アルゴリズム

本論文では，この最多ヒント問題に関連して，ヒント数の多い問題を効率良く生成するためのアルゴリズムとその実装を示す．

3. モンテカルロ木探索

モンテカルロ木探索は，乱数を用いたシミュレーションにより近似解を求めるモンテカルロ法と，ゲーム木の探索とを組み合わせた手法である．モンテカルロ木探索の重要なアイデアは次の 3 つである．1 つ目は，ある盤面について終局まで乱数を用いてプレイすることにより評価値を計算することである．この終局までの乱数によるプレイは，プレイアウトと呼ばれる．2 つ目は，候補手に対してプレイアウトを行う際に，それまでに得られた評価値をもとに有望そうな候補手に多くのプレイアウトを割り当てることである．3 つ目は，ある候補手に対するプレイアウト回数がある閾値を超えた場合には，その手を展開してプレイアウトの開始点を 1 つ深くすることである．これらのアイデアにより，モンテカルロ木探索では有望そうな部分に対する探索を深く行うことができる．

図 4 にモンテカルロ木探索の擬似コードを示す．モンテカルロ木探索では，終了条件（探索時間もしくは探索回数）に到達するまでプレイアウトを繰り返し，最終的に評価値の最も良い候補手を選択する．繰り返しされる処理は，次の 4 つからなる．

1. それまでのプレイアウト回数と評価値をもとに，根から葉まで降りる（`select_downwards`）．
2. 葉のプレイアウト回数を増やし，その盤面に対するプレイアウト回数が閾値を超えた（`expand_cond`）際にはそのノードを展開する（`expand`）．

3. 選択された葉から，乱数によるプレイアウトを行う（playout）．
4. プレイアウトの結果の盤面から評価値を計算し（getvalue），葉から根まで値を更新する（update_upwards）．

図4において関数として記述した，

- select_downwards:プレイアウト対象の子ノードの選択方法，
- expand_cond：ノード展開の閾値，
- expand：ノードの展開方法，
- playout：プレイアウト方法，
- getvalue：終局盤面の評価値，
- update_upwards：ノードの持つ値の更新方法

をそれぞれ定めることによりモンテカルロ木探索を行うことができる．得られた結果の善し悪し（ゲームプレイヤの場合はその強さ）は，これらのパラメータ関数をどのように定めるかに依存する．

以下では，コンピュータ囲碁などで広く用いられているUCT（Upper Confidence Tree）アルゴリズムにおける子ノードの選択方法について述べる．UCTでは，子ノードの選択にUCB1（Upper Confidence Bound）アルゴリズム[14]を用いる．これは，多腕バンディット問題（Multi-armed bandit problem）[15]において，計算量が小さく高い報酬が得られる戦略である．UCB1アルゴリズムでは，まず，各候補に対して次の式(1)で与えられるUCB1値を計算する．式において，$\overline{X_j}$はj番目の候補の評価値の期待値，n_jはj番目の候補のプレイアウト回数，nは全体のプレイアウト回数，cは問題ごとに定める定数である．

$$ucb_j = \overline{X_j} + c\sqrt{\frac{2\log n}{n_j}} \qquad (1)$$

次に，得られたUCB1値のうち最大となるような候補を選択する．このUCB1アルゴリズムを用いることで，有望な候補に多くのプレイアウトを行うことと，評価値が偶然低い場合に対する救済という2つの性質を満たすようにプレイアウトが実行される．

4. モンテカルロ木探索による多数ヒント盤面の生成アルゴリズム

本研究では，モンテカルロ木探索を用いた多数ヒント盤面の生成アルゴリズムを開発した．以下では，モンテカルロ木探索のアルゴリズムのうち，プレイアウトの方法，評価値とそれに基づく子の選択，子の展開方法について説明する．

4.1 プレイアウトの方法

モンテカルロ木探索においては乱数によるプレイアウトを多数回実行するが，その際に，目標に近い結果を多く出す良いプレイアウトを行うことは重要である．数独の盤面生成においては，解なしとなるような置き方を避けつつ，ランダムにヒントを配置していけばいずれ唯一解の盤面を得ることができる．このような完全にランダムな手法では，本研究で求めたい多数ヒントの盤面を得ることが容易ではない．以下，完全ランダムな手法をAと呼ぶこととする．

本研究では，よりヒント数が多く置けるようなヒントの選び方を3つ考案した．プレイアウトは，これらの選び方によって順次ヒントを置いていき，最終的に唯一解である盤面が得られるまで繰り返す．また，解なしとなるような選び方は除外するものとする．

簡易手法 B_k 候補数字の中からk個ランダムに選び，そのうち，それと同じ行・列・ブロックに含まれるその候補数字が最も少ないものを選択する．

完全簡易 C_j 盤面の候補数字すべてについて，それと同じ行・列・ブロックに含まれるその候補数字が最も少ないj個のうちからランダムに選ぶ．

解法適用 D_k 候補数字の中からk個ランダムに選び，それを置いたとして，さらに解法によるマスの数字の決定や候補数字の絞り込みを行って最も候補数字が多く残るものを選択する．

これらの考え方は，候補数字ができるだけ多く残る方ができるだけヒントを多く置くことができるだろうという仮説に基づくものである．簡易手法が最も高速に計算でき，一方，解法適用は最も時間がかかる処理となる．これらの手法によるプレイアウトの品質と時間については，評価実験において述べる．

これらのプレイアウトによって得られた盤面には，取り除いたとしても唯一解のままとなるようなヒントが含まれていることがある．したがって，プレイアウトの結果から，取り除いても唯一解であるようなヒントを消すことを最後に行う必要がある．

4.2 評価値とそれに基づく子の選択

著者らが過去に行った少数ヒント盤面の生成においては，盤面に含まれるヒントの数の最小値をそのまま評価値として用いていた[16]．同様の考え方を適用することも可能であるが，本研究では，盤面に含まれるヒントの数の最大値に加えて，そのような盤面の出現回数を考慮に入れた評価値を設計する．

あるノードのプレイアウト回数をn，その親ノードのプレイアウト回数をNとする．また，プレイアウトの結果得られた盤面に含まれるヒント数の

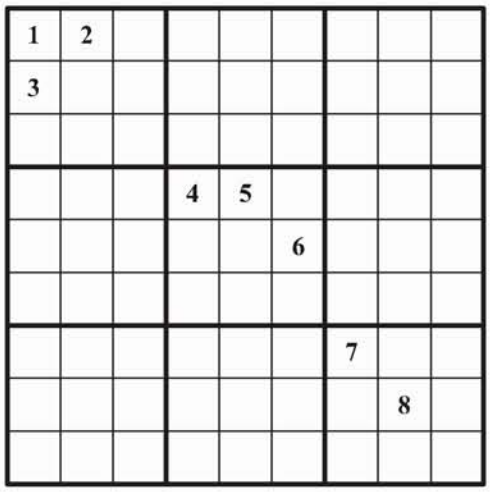

図 5. モンテカルロ木探索の開始盤面

最大を M，そのヒント数からなる盤面が得られた回数を m とする．このとき，そのノードに対する評価値を

$$M + \frac{m}{n} + \sqrt{\frac{2\log N}{n}}$$

とした．ここで，$M+m/n$ の部分が UCB1 における平均に相当し，また UCB1 におけるパラメータ c は 1 とした．中間ノードについても，部分木に含まれる葉におけるプレイアウトをもとに同じ計算を行う．プレイアウトを行う葉を選ぶ `select_downwards` における子の選択では，上記の評価値が最も大きな子を選ぶこととした．

4.3 根ノードと子の展開方法

数独はその盤面および数について対称性があるため，最初のいくつかのヒントの配置については可能な組み合わせが少ない．例えば，最初の 1 つ目のヒントをどこにおいたとしても，正規化を行えば，左上に 1 を置いた盤面と等しくなる．これまでに知られている多数ヒント盤面のヒント数は 35 であるため，少なくとも 1 つの行には 4 つ以上のヒントを持つことが分かり，その行のあるブロック内には 2 つのヒントが必ず入ることになる．このような考え方を使って，一般性をほとんど失うことなく 8 つのヒントを配置した盤面（図 5）をモンテカルロ木探索の根ノードとした．

子の展開については，プレイアウトの回数の閾値を 8 とした．すなわち，8 回のプレイアウトを行ったときにその子を 1 段展開する．展開する際には，プレイアウトと同じ方法によってその子を選択し，子の数は最大で 8 となるようにした．

5. 評価実験

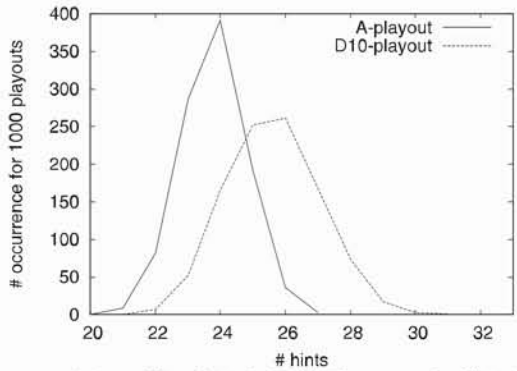

図 6. プレイアウト結果のヒストグラム

5.1 プレイアウトの評価

まず，第 4.1 節で述べたプレイアウトの手法について，それによって得られる盤面のヒント数とその時間の観点で比較を行う．

比較に用いるのは，完全ランダム A，簡易手法 B_{20}，完全簡易 C_{10}，および解法適用 D_5 と D_{10} である．表 1 に開始盤面（図 5）から 2000 回のプレイアウトを行った結果をまとめる．また，図 6 に，完全ランダム A および解法適用 D_{10} について，1000 回のプレイアウトあたりのヒント数のヒストグラムを示す．

この結果より，解法を適用して候補数字の絞り込みを行う D_5 や D_{10} がよりヒント数の多い盤面を生成できていることが分かる．また，プレイアウトのヒストグラムを見ると，生成される盤面はおよそ正規分布に従っていることも見てとれる．

しかし，良いプレイアウトである解法適用には時間オーバーヘッドがある．特に，解法適用 D_{10} では，D_5 に比べて倍，それ以外に比べて 6 倍程度の時間がかかってしまうため，それに反比例するようプレイアウトの回数が減らしたときにも良い結果が得られるかどうかの調査を次に行う．

5.2 モンテカルロ木探索の評価

5 種類のプレイアウト法を用いてモンテカルロ木探索を行った．プレイアウトの時間をおよそ均等にするため，完全ランダム A は 160 万回，簡易手法 B_{20} は 150 万回，完全簡易 C_{10} は 150 万回，および解法適用 D_5 は 80 万回，D_{10} は 40 万回のプレイアウトをそれぞれ行った[1]．これらの設定でモンテカルロ木探索を実行するのにかかった計算時間は，6 コアの Xeon E5645 が 2 つからなる計算機において約 10 時間であった．結果を表 2 に示す．

[1] 表 1 の計算時間にちょうど反比例していない理由は，モンテカルロ木探索のプレイアウト以外の計算が共通して時間がかかるためである．

表 1. プレイアウト単体の評価

手法	最大値	平均値	最小値	標準偏差	時間（秒）
完全ランダム A	27	23.80	20	1.01	48.8
簡易手法 B_{20}	29	24.66	21	1.21	65.8
完全簡易 C_{10}	29	24.70	21	1.20	72.8
解法適用 D_5	29	25.10	21	1.30	203.2
解法適用 D_{10}	31	25.61	21	1.43	394.3

表 2. モンテカルロ木探索の評価

手法	最大値	平均値	H=33	H=32	H=31	H=30	H=29
完全ランダム A	29	23.86	0	0	0	0	11
簡易手法 B_{20}	31	24.69	0	0	6	79	1263
完全簡易 C_{10}	32	24.68	0	1	5	79	1349
解法適用 D_5	32	25.12	0	1	30	458	4251
解法適用 D_{10}	33	25.57	3	13	124	1251	6918

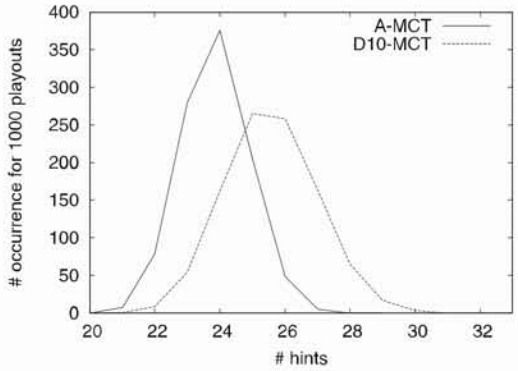

図 7. モンテカルロ木探索におけるプレイアウト結果のヒストグラム

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

図 8. ヒント数 33 の問題の 1 つ

表 1 と表 2 を比べることでいくつかの興味深い結果が見られる．まず，モンテカルロ木探索を行うことで，最大値が大きくなる（2 つないし 3 つ大きくなっている）一方で，平均値についてはあまり変化していないことである．これを確認するため，モンテカルロ木探索について，1000 回あたりのヒント数のヒストグラムを図 7 に示す．この図 7 を図 6 と比較すると，わずかな差はあるもののほぼ同じヒストグラムとなっている．この結果は，著者らの予想から外れた結果であった．

また，計算時間についておよそ同じ時間となるようプレイアウト回数を調整したが，最終的な結果はプレイアウト単体で実行した結果に近いものとなった．このことから，より多数のヒントからなる盤面を生成するためには，プレイアウトのアルゴリズムの改良が重要であることが分かる．

最後に，解法適用 D_{10} によって得られたヒント数 33 の盤面の 1 つを図 8 に示す．

6. 関連研究

6.1 数独に関する様々な研究

数独は，人が解くパズルとして広く楽しまれている．計算機の場合には，一般的な 9 × 9 の数独であれば単純なバックトラックでも簡単に解ける．そのため，計算機を用いて単純に数独を解くのではなく，様々なアルゴリズムやソルバの応用例として，また，数独の問題生成や数独の性質についての研究が広く行われている．

従来の単純なバックトラックとは異なるアルゴリズムやソルバの応用として数独が解けることが示されている．例えば，整数計画法によるもの[7]，二分決定グラフによるもの[9]，SAT ソルバによるもの[6]，GPU を用いた進化計算（遺伝アルゴリズム）によるもの[8] などがある．本研究では，盤面生成にあたり，人が解くときに使う手順をシミュレートする方

法と，SAT ソルバを用いる方法の 2 つを利用した．特に SAT ソルバを利用するにあたっては，Sat4j[17] の addAtMost関数の利用やルールに相当する節の再利用により高速化を行っている．

数独の問題を解くだけでなく，数独の問題生成や数独そのものの性質を調べる研究も行われている．例えば数独の難易度評価について，人が解く際に用いる手法に着目することで行う手法が多く提案されている[1,2,3,4]．井上らは，盤面の対称性を除いた本質的に異なる数独の解盤面を数え上げを行った[5]．

本研究における多数ヒント盤面の生成は，著者らが過去に行った少数ヒント盤面の生成[16]をもとに実現している．数独の問題として成り立つ最少のヒント数がいくつであるかを求める「数独の最少ヒント問題」については，これまでに複数の研究がある．9 × 9 の数独においては，McGuire らにより，解が唯一となるヒント数 16 の盤面がないことが計算機を用いて調べられた[10]．より大きな数独については，最少のヒント数は分かっていないものの，16 × 16 の数独においてヒント数が 56 である盤面と 25 × 25 の数独においてヒント数が 174 である盤面がそれぞれ白川によって発見されている[18]．

6.2 モンテカルロ木探索

モンテカルロ木探索[12]は，探索空間が広大であり評価関数を設計することが難しいようなゲームにおいて有効なアルゴリズムである．その性質から，コンピュータ囲碁において強いプレイヤを作ることに大きく貢献している[13]．

これまでに，さまざまな種類のゲームに対してモンテカルロ木探索が適用されてきた．そのうち，1 人完全情報ゲームに対してモンテカルロ木探索を適用する研究として，SameGame へ適用することが Schadd ら[19]や Takes ら[20]によって行われた．本研究の数独の多数ヒント盤面生成は，条件を満たす盤面を終了状態，そのときのヒント数を得点とすると 1 人完全情報ゲームとしてもとらえることができる．よって，モンテカルロ木探索の 1 人完全情報ゲームへの応用例の 1 つとしても捉えることができる．

7. まとめ

本論文では，モンテカルロ木探索手法を数独の多数ヒント盤面の生成に適用し，その過程で行った工夫と実験結果を報告した．特に，モンテカルロ木探索手法を適用する上で重要となるプレイアウトの精度向上のため，解法を適用した上で候補数字の少なくなるようなものを優先して選ぶようにした．実験の結果，これまでに得られている多数ヒント盤面よりも少ないヒント数の盤面しか得られなかったものの，ランダムな探索や単純なモンテカルロ法に比べると良い結果を得ることができた．

本論文における実験結果は，著者が予想していたよりもモンテカルロ木探索がうまくはたらいていないことを示している．その理由の一つは，プレイアウトにおける値の分布に比べて，最大値を求めることが非常に極端な場合を取り出そうとしているためではないかと考えている．そのような極端な場合に対するモンテカルロ木探索の性能向上に取り組むことが今後の課題である．

文献

1) 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**, Vol. 2006, No, 134, pp. 1–6, 2006.
2) M. Danburg-Wyld: “How to Score Sudoku.” (URL= http://sudokuplace.com/sudoku_scoring.asp)
3) R. Pelánek: “Difficulty Rating of Sudoku Puzzles by a Computational Model.” In *Proceedings of the Twenty-Fourth International Florida Artificial Intelligence Research Society Conference*, 2011.
4) 土出 智也, 真貝 寿明: “数独パズルの難易度判定—解法ロジックを用いた数値化の提案—.” **大阪工業大学紀要, 理工篇**, Vol. 56, No. 1, pp. 1–18, 2011.
5) 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.” **一般社団法人情報処理学会 全国大会講演論文集**, 平成 21 年, No. 4, pp. “4-741”–“4-742”, 2009.
6) I. Lynce and J. Ouaknine: “Sudoku as a SAT problem.” In *Proceedings of 9th International Symposium on Artificial Intelligence and Mathematics*, 2006.
7) 岡本 吉央: “「整数計画法によるパズル解法」実習報告.” **組合せゲーム・パズルミニプロジェクト第 2 回ミニ研究集会**, 2007.
8) 佐藤 裕二, 長谷川 直広, 佐藤 未来子, 並木 美太郎: “メニーコアプロセッサによる進化計算を用いた数独解法の高速化.” **情報処理学会研究報告**, *HPC*, **ハイパフォーマンスコンピューティング**, Vol. 130, No. 4, pp. 1–7, 2011.
9) 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.” **一般社団法人情報処理**

学会 全国大会講演論文集, 平成 20 年, No. 5, pp. "5-253"–"5-254", 2008.

10) G. McGuire, B. Tugemann, and G. Civario: "There is no 16-Clue Sudoku: Solving the Sudoku Minimum Number of Clues Problem." *The Computing Research Repository (CoRR)*, abs/1201.0749, 2012.
11) Games and Puzzles Competitions on Computers (GPCC): "GPCC2013 問題." 2013. (URL = http://hp.vector.co.jp/authors/VA003988/gpcc/gpcc13.htm)
12) L. Kocsis and C. Szepesvári: "Bandit Based Monte-Carlo Planning." In *Proceedings of the 17th European Conference on Machine Learning (ECML 2006)*, pp. 282–293, 2006.
13) 美添 一樹: "モンテカルロ木探索：コンピュータ囲碁に革命を起こした新手法." **情報処理**, Vol. 49, No. 6, pp. 686–693, 2008.
14) P. Auer, N. Cesa-Bianchi, and P. Fischer: "Finite-time Analysis of the Multiarmed Bandit Problem." *Machine Learning*, Vol. 47, No. 2–3, pp. 235–256, 2002.
15) 但馬 康宏, 小谷 善行: "k-armed バンデット問題のゲームへの適用における試行回数について." **情報処理学会研究報告**, *AL*, **アルゴリズム研究会報告**, Vol. 2008, No. 49, pp. 65–70, 2008.
16) 那須 律政, 松崎 公紀: "モンテカルロ木探索による数独少数ヒント盤面の生成." **第 *54* 回プログラミング・シンポジウム**, pp. 173–180, 2013.
17) D. Le Berre and A. Parrain: "The Sat4j library, release 2.2." *Journal on Satisfiability, Boolean Modeling and Computation*, Vol. 7, No. 2–3, pp. 59–64, 2010.
18) Games and Puzzles Competitions on Computers (GPCC): "数独のヒント最少問題の解答." 2011. (URL = http://hp.vector.co.jp/authors/VA003988/gpcc/11p1.htm)
19) M. P. D. Schadd, M. H. M. Winands, M. J. W. Tak, and J. W. H. M. Uiterwijk: "Single-player Monte-Carlo tree search for SameGame." *Journal Knowledge-Based Systems*, Vol. 34, pp. 3–11, 2011.
20) F. W. Takes and W. A. Kosters: "Solving SameGame and its Chessboard Variant." In *Proceedings of the 21st Benelux Conference on Artificial Intelligence (BNAIC)*, pp. 249–256, 2009.

(Regular Paper)

Generating Many-Clue Sudoku Problems with Monte-Carlo Tree Search

Kiminori Matsuzaki[1]* **Norimasa Nasu**[2]

(Received: May 10th, 2013)

[1] School of Information, Kochi University of Technology
185 Tosayamadacho-Miyanokuchi, Kami, Kochi, 782-8502, JAPAN

[2] Graduate School of Engineering, Kochi University of Technology
185 Tosayamadacho-Miyanokuchi, Kami, Kochi, 782-8502, JAPAN

* E-mail: matsuzaki.kiminori@kochi-tech.ac.jp

Abstract: "Sudoku" is a famous pencil puzzle and it is often used on the theoretical or practical studies in the computer science. This paper is related to the *maximum number of clues problem*, which is to find the maximum number of clues in the Sudoku board such that no clues can be removed to let the board have a unique answer. It is difficult to generate sudoku boards with some particular characteristics, due to the quite big number of possible sudoku boards and difficulty in specifying the characteristics. In this study, we apply the *Monte-Carlo tree search* technique to generating Sudoku boards with many clues. Here, Monte-Carlo tree search is a technique that has been recently intensively studied in game programming for "Go". We develop three methods to improve the quality of playouts, which is important in monte-calro tree search. As a result, we succeeded to generate some 33-clue sudoku boards.