ビームサーチを用いたヒント数 17 の数独パズルの効率的な生成について

長尾 卓[1,a)]　山本 修身[2,b)]

概要：本論文は最小個数 17 個の要素で構成される数独パズルのヒントを効率的に見つけ出すアルゴリズムを提案する．本アルゴリズムでは空集合から出発して 1 つずつ「最適」なヒント要素をヒント集合に添加しながら，なるべく個数の少ない数独のヒント集合を構成する．ここで「最適」なヒント要素とは，ヒント集合に添加することによって解集合のサイズがなるべく小さくなるようなヒント要素のことである．このような「最適」なヒント要素を探索するためにヒント集合のサイズが小さいときは，メトロポリス法に基づくマルコフ連鎖を構成し，それを用いて解を等確率にサンプリングし，ヒント集合が大きくなるとバックトラック法により解をすべて列挙してヒント要素を見つける．このような方法では必ず最小のヒント数 17 のヒント集合が得られるとは限らず，本アルゴリズムでは [2] の方法を拡張しビームサーチを用いることで 17 個の要素で構成されるヒントが生成される確率を [2] の 6%や [7] の 52%から 83%まで向上させた．

Efficient Generation of Sudoku Puzzles with Seventeen Clues Using Beam Search

Suguru Nagao[1,a)]　Osami Yamamoto[2,b)]

Abstract: The present paper describes an algorithm for generating a hint set of the Sudoku puzzle efficiently which consists of seventeen hint elements, where seventeen is known to be the least number of hint elements for the puzzle. Starting with the empty set, the algorithm constructs hint sets with small number of elements adding ‘optimal’ hint elements one by one, where an ‘optimal’ hint element is defined as an element that reduces number of solutions of the puzzle a lot adding the element. In order to find such an ‘optimal‘ solution, when the hint set is small, the algorithm constructs a Markov chain based on the Metroporis method and samples solutions of the puzzle with equal probabilities. On the other hand, when the hint set gets large, the algorithm enumerates the solutions of the puzzle and find an optimal hint element. The algorithm finds a hint set of size 17, which is known as the least number of Sudoku hints, probabilistically. Expanding [2] with the beam search algorithm, our algorithm improved the generation ratio of Sudoku hints of size 17 to 83%, where the ratios of [2] and [7] were 6% and 52% respectively.

1. はじめに

数独パズルは，あらかじめ与えられたいくつかの要素（これをヒントと呼ぶ）を元にして 1 から 9 の数字をこれ以外の 9×9 の枠内のマスに入れていくパズルである．ただし，すべての縦列，横列，および 3×3 ブロック内では数字が重複してはいけない．図 1 に数独パズルの例を示す．このパズルのヒントは丸で囲われた要素である．プレイヤはヒントから縦列，横列，3×3 ブロック内（それぞれ図 1 に青い列，ピンクの列，黄色いブロックのように示した形状の範囲）で数字の重複がないようにヒント以外の場所の数字を選択しなければならない．また，数独パズルではヒントから決定できる解は１通りでなければならいという制

1 名城大学大学院理工学研究科情報工学専攻
Department of Information Engineering, School of Science and Engineering, Meijo University
2 名城大学情報工学部情報工学科
Department of Information Engineering, Faculty of Information Engineering, Meijo University
a) 223426015@ccmailg.meijo-u.ac.jp
223426015@ccalumni.meijo-u.ac.jp
b) osami@meijo-u.ac.jp

©2022 Information Processing Society of Japan

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

図 1 17 個のヒントによる数独パズルの問題とその解．丸で囲われた要素は問題として与えられるヒントであり，それらに基づいてその他の場所の要素を決定する．

限がある．図 1 に示したヒントから導かれる解は 1 通りであり，このヒントによるパズルは正しい数独パズルであると言える．一般に正しい数独パズルのヒント数は 17 以上であることが知られている [1]．

本論文で用いる方法は先行研究である古川らの方法 [2], [3] に基づいている．この方法では，ヒントとして要素の集合 $H = \{h_1, h_2, \ldots, h_k\}$ が与えられたとき，このヒントの解集合を $S(H)$ として，$|S(H \cup \{h_{k+1}\})|$ がより小さくなるような h_{k+1} を探して，ヒントの集合 H に添加する．$|S(H)| = 1$ になるまでこの操作を繰り返すことによってヒントの少ない数独パズルを求めている．ただし，このような動作を繰り返したとしてもヒント数が最小の 17 となる問題が得られるとは限らない．添加するヒントは後述するように確率的に求めているので，ある確率でヒント数 17 の問題が得られることになる．したがって，どの程度の確率でヒント数 17 の問題を得るかということが問題となる．本論文ではより高い確率でヒント数 17 の問題を得る方法について模索した．

ヒントのそれぞれの要素は数独パズルの盤面の位置の情報とそこに置かれている数字から成る．$|S(H \cup \{h_{k+1}\})|$ が最小となる h_{k+1} を見つけるにはすべての位置 p とその位置において可能な数字 v のそれぞれについて，$S(H \cup \{(p, v)\})$ を実際にバックトラック (BT)[4] などの手法で数え上げれば良い．しかし，これは現実には効率が悪いためヒント集合 H の解 $\mathcal{P} \in S(H)$ をすべて BT で列挙し $\mathcal{P}$ のヒント以外の位置 p とそこでの数字 $1 \leq v \leq 9$ の組 (p, v) の統計をとればよい．すなわち，それぞれの位置 p で数字 v が出現した頻度 $freq[p, v]$ を数えて，それらのうち 1 以上で最小となる (p, v) を見つけて h_{k+1} とすれば良い．

しかし，H の要素数が小さい場合，$S(H)$ の要素数は極端に大きくなり，バックトラックによる列挙は不可能となる．そこで本研究ではメトロポリス法 [5] によってマルコフ連鎖を構成し，それを用いて等確率に $S(H)$ の要素を取り出して，$freq[p, v]$ を近似する．$S(H)$ の要素を見つけ出すとき，構成したマルコフ連鎖の遷移確率を徐々に変化させていくことから本論文ではこのような計算方法をシミュレイテッドアニーリング (SA)（焼き鈍し）[6] と呼んでいる．この部分の詳細については第 2 節で説明する．

このようにして解の個数が低いと思われるヒント要素を添加しながら H を大きくしていき，$|S(H)| = 1$ となれば数独パズルのヒントが得られる．しかし，普通にこのアルゴリズムを実行するだけでは，要素数 17 となるヒントが得られる確率はそれほど高くない．本研究では高い確率でヒント数 17 の数独パズルを生成するためのアルゴリズムについて考える．[3] ではヒント数 17 の数独パズルの生成割合は 6% であったが，[7] では 50% 程度まで引き上げることができている．これまでの研究 [10]，[7] から，最急降下法で添加したヒント要素が最も良いヒント要素とは限らないため，本研究ではさらにビームサーチ (BS)[8] を用いることにより，より柔軟に解の個数を減らすことを試みた．

2. ヒント生成アルゴリズム

本節では本論文で述べるアルゴリズムの基本部分から説明する．前節で述べたように，本論文はそれまでに構成したヒント H の解集合 $S(H)$ を列挙または確率的方法で解析することによって $H^{(0)} = \emptyset$ からスタートして「最適」なヒント要素 h_{i+1} を見つけ出して

$$H^{(i+1)} = H^{(i)} \cup \{h_{i+1}\} \tag{1}$$

という更新するプロセスを $|S(H^{(r)})| = 1$ となるまで繰り返す．これは数独パズルのヒントとなるが $r = 17$ となっていれば本論文での目的のヒントとなる．しかし，$r > 17$ となることも多い．ただし，「最適」なヒント要素とはそのヒント要素を付加することによって解集合の個数がより減るような要素のことである．このような要素を見つけるためには以下の性質を用いる．

性質 1. 数独パズルのヒント H に含まれない位置 p で数字 v となるヒント要素 $h = (p, v)$ を付加したときの解集合 $S(H \cup \{h\})$ は

$$S(H)|_{p,v} = \{\mathcal{P} \in S(H) \mid \mathcal{P}[p] = v\} \tag{2}$$

に一致する．ただし，$\mathcal{P}[p]$ は数独パズルの解 $\mathcal{P}$ の位置 p での数字を表すものとする．

証明. ほぼ自明である．$\mathcal{P} \in S(H \cup \{h\})$ と仮定すれば，$\mathcal{P}[p] = v$ なので $\mathcal{P} \in S(H)|_{p,v}$ であり，逆に $\mathcal{P} \in S(H)|_{p,v}$ と仮定すれば，$\mathcal{P}$ は $H \cup \{h\}$ の解になっているので $\mathcal{P} \in S(H \cup \{h\})$ である． □

この性質より，前節で述べたように，あるヒントが与えられたとき，すべての $\mathcal{P} \in S(H)$ について H に含まれるヒントの位置以外に関して $freq[p, v]$ の統計をとり，最も

©2022 Information Processing Society of Japan

出現回数が少ない (p, v) の組を見つけ出せば，それが最適なヒント要素であることになる．H の要素数が比較的大きく，したがって $S(H)$ が比較的小さくなり解が列挙できる場合には，前節で述べたように，BT を用いて正確に最適なヒント要素を見つけ出すことができる．

これに対して，H の要素数が比較的小さくて $S(H)$ が大きくなる場合には実際に BT を用いることができない．この場合には以下に示すようなメトロポリス法 [5] による確率的なサンプリングを用いることで与えられたヒント H の解集合 $S(H)$ の要素を等確率に取り出してヒント要素に関する統計をとり，最適なヒントを見つけ出す．そのために，まず解集合を含む数独パズルの状態を定義する．ここで用いる状態集合は 9×9 の枠のそれぞれの行に 1 から 9 までの数字が 1 つずつ入っているような状態から成る集合である．図 2 にこのような状態のうちの 1 つを示す．このような状態を集めた集合を $\overline{S}(H)$ と書こう．

さらに，この状態から別の状態への確率的な遷移を定義することで，$\overline{S}(H)$ 上のマルコフ連鎖を定義する．ある状態から別の状態へ遷移させる場合には，図 2 に示すように，まず 9 つの行の中から 1 つの行を等確率に選択する．さらに選択した行の中からヒント以外の 2 つの数字を等確率に選択し，それらを入れ替えて遷移先の要素とする（図 2 参照）．ヒント以外の要素が 1 つしか存在しないというケースも考えられるが，その場合，その要素も変更することができないので，その要素をヒントと考えても同じになる．このように定義すると，状態 x から状態 y へ遷移する確率を $Q(x|y)$ と書けば，$Q(x|y) = Q(y|x)$ が成り立つ．また，遷移を複数回繰り返すことにより任意の状態へ遷移させることができる．ここで，このマルコフ連鎖 M を用いて任意の状態 $s \in \overline{S}(H)$ からの遷移が次のように定義される別のマルコフ連鎖 M' を定義する．$\overline{S}(H)$ から非負実数の集合への関数 f が与えられているとする．このとき，M' の遷移を

(1) 状態 s から M を用いて確率的に遷移先 s' を決める．
(2) $f(s) < f(s')$ の場合には，次の状態は s' とする．
(3) $f(s) \geq f(s')$ の場合には，次の状態は確率 $p = f(s')/f(s)$ で s' とし，確率 $1-p$ で s とする．

と決めると，以下の性質が成り立つ．

性質 2 ([5])**.** あるマルコフ連鎖 M（これを提案分布と呼ぶ）が与えられているとする．M は前述の遷移確率の対称性 $Q(y|x) = Q(x|y)$ が成り立ち，何回かの遷移によって任意の状態間で遷移することが可能であると仮定する．このとき，与えられた非負実数値関数 f を用いて上記のようにして構成されたマルコフ連鎖 M' が定常状態において状態 $x \in \overline{S}(H)$ を生成する確率は $f(x)/\sum_{z \in \overline{S}(H)} f(z)$ である．

この性質を用いれば，任意の f について M' を構成することによって $\overline{S}(H)$ の要素が f の値に比例する確率で出現するような状態の列を作り出すことができる．

ここでは $\overline{S}(H)$ の部分集合である解集合 $S(H)$ の要素を生成させたいので，うまく f を定義することで M' によって $S(H)$ の要素が高確率で生成されるようにする．そこで，それぞれの $\overline{S}(H)$ の要素 s について数独パズルの解として不適格であるかの度合いを以下のように定義する：

$$E(s) = \sum_{j=1}^{9} B_j(s) + \sum_{k=1}^{9} C_k(s). \tag{3}$$

ただし，$B_j(s)$ は状態 s の j 番目のブロックの中で $1, \ldots, 9$ の数字のうち出現しないものの個数であり，$C_k(s)$ は s の k 番目の列の中で $1, \ldots, 9$ の数字のうち出現しないものの個数であるとする．このように定義すれば，$E(s) = 0$ ならば $s \in S(H)$ であると言える．そこで，

$$f(s) = \exp(-\beta E(s)) \tag{4}$$

と定義し，これを利用する．ここで β は適当な正数であり，β を大きくすると $E(s)$ が大きくなったとき $f(s)$ が急激に小さくなる．すなわち，マルコフ連鎖 M' では $E(s)$ が大きな状態 s は出現しづらくなる．逆に $s \in S(H)$ の場合には，$E(s) = 0$ なので β の値に関係なく $f(s) = 1$ となる．β が大きくなると相対的に $S(H)$ の要素が出現する確率が高くなることになる．

このように定義すると解集合の要素についてはどれも出現確率が同じになる．一般にこのようにして作られたマルコフ連鎖はある程度の回数，状態遷移を行わないと定常状態にならないことが知られている．そのため一定回数状態遷移させた後 $E(s) = 0$ となる要素が出現すると s は $S(H)$ から等確率に選択された解であると考えられる．結局，このような要素を見つけることは，マルコフ連鎖を用いて数独パズルを解いていることに他ならない．

ここまで説明した確率的方法で解のサンプリングを行う

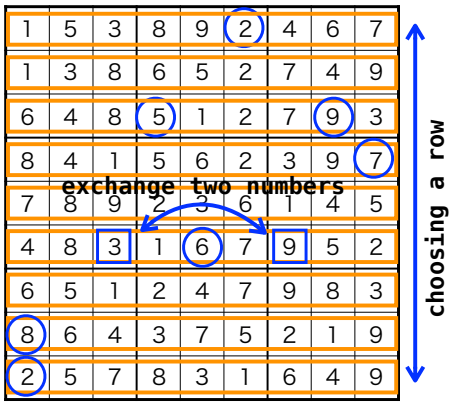

図 2 メトロポリス法で等確率に解集合の要素を生成するために定義する状態集合の要素の例．この要素から別の要素への遷移は，行の等確率な選択とそれに続く行内のヒント要素（丸のついた要素）以外の 2 つの要素の等確率な選択および入れ替えによって定義される．

©2022 Information Processing Society of Japan

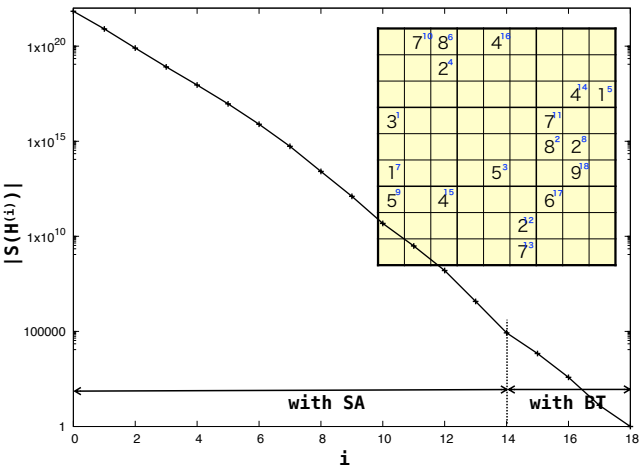

図 3 Number of solutions $S(H^{(i)})$ of hint subsets $H^{(i)}$ for the puzzle shown in the upper left area of the graph. The hint set of the puzzle consists of eightteen hint elemnets.

場合, BT のようにすべての解を列挙するわけではないので, 解のサンプリング回数によってはヒント要素としては意味があるのにサンプリングした解には出現しないヒント要素が存在する可能性がある．これについては，$H^{(i)}$ にある 1 つのヒント要素 h_{i+1} を添加することで解集合の大きさが α 倍 $(0 < \alpha < 1)$ となるとすると, $\alpha = |S(H^{(i+1)})|/|S(H^{(i)})|$ であり，$S(H^{(i)})$ の要素をランダムにサンプリングするとすれば，h_{i+1} を含む解が出てくる確率は α であるので N 回解をサンプリングするとすれば h_{i+1} の出現する回数の分布は 2 項分布 $\mathrm{B}(n, \alpha)$ に従う．この分布の平均と標準偏差はそれぞれ $\mu = n\alpha$, $\sigma = \sqrt{n\alpha(1-\alpha)}$ となる．α は経験的に小さくても数十分の 1 程度なので，$N = 1000$ 程度であれば平均値は 10 以上であり，標準偏差は平均の平方根なので，出現回数が 0 回である確率は極めて小さいということが言える．そのため，ここでは，SA による解のサンプリングで出現しないヒント要素については特に考慮せずアルゴリズムを実行している．

ここまで本節で示したアルゴリズムの実行例を示す．普通に実行することによりある程度ヒント数の少ない数独パズルを生成することができるが，最小のヒント数 17 個の数独パズルになるとは限らない．図 3 にそれぞれのヒント要素を添加した時点での解集合の個数 $|S(H^{(i)})|$ をマルコフ連鎖モンテカルロ法で求めたグラフを示す．本論文では詳述しないが．[9] と類似の方法によって本節で示したマルコフ連鎖を用いて解集合の個数を見積もることが可能である．ここで示す例はグラフ中に示した数独パズルの問題を用いており，ヒントの右肩の数字が添加した順番を表す．ヒント数は 18 である．縦軸は対数軸となっており，$|S(H^{(i)})|$ は i と共にほぼ一定割合で減少してきて 18 個目のヒント要素を添加して $|S(H^{(18)})| = 1$ となっているのがわかる．この減り方を見る限り 18 個のヒント要素で解の個数が 1 まで落ちるか 17 個で解の個数が 1 まで落ちるかはかなり微妙であり，全体を通した解の個数の減り具合に依存すると思われる．

3. アルゴリズムの改良

3.1 アルゴリズムの概要

第 1 節の最後で述べたように $H^{(i)}$ から「最適」なヒント要素を添加して $H^{(i+1)}$ を生成することで常にヒント数 17 の問題が得られるわけではない．ここで問題となるのはヒント数 17 の問題が生成できる割合を高くすることである．

そのために本研究ではビームサーチ（BS）を用いて，解の個数が少ない w 個の $H^{(i)}$ を保持しながら，柔軟に解の個数が減るようなヒント要素をそれぞれの $H^{(i)}$ に添加していく．BS は木構造上に根からスタートする w 本のパスを考えて，それらのパスから同時に伸ばしながら最適なノードを探索するヒューリスティックな探索アルゴリズムである [8]．本研究での BS において i 個のヒント要素を添加した段階のヒントの組を以下のように表現する：

$$B^{(i)} = \left(H_1^{(i)}, H_2^{(i)}, \ldots, H_w^{(i)}\right) \tag{5}$$

本研究が提案するヒント生成アルゴリズムは 2 つの部分に分かれる．最初の部分では SA を用いて最適なヒント要素を見つけ出し $H^{(i)}$ へ添加する．その後，ヒント数が多くなった時点では BT を用いて最適なヒント要素を添加していく．ここでは最初 SA を用いてヒント要素を添加しながら $H^{(2)}$ から $H^{(s)}$ までを生成する．その後，$H^{(s)}$ から $H^{(r)}$ までを BT を用いて生成する．ただし．$H^{(s+1)}$ において一定の条件が成り立っていない場合には，このヒント生成プロセスは破棄して新たな生成プロセスを再スタートさせる．本研究では，$w = 4$, $s = 13$ としている．$w = 4$ とした理由については 3.4 節で説明する．$B^{(17)}$ の中の w 個のヒント集合のうち解集合の要素数が 1 となっているものがあれば，ヒント数 17 の問題が得られたことになる．

まず，$B^{(0)} = (\emptyset, \ldots, \emptyset)$, $B^{(1)} = \left(\{\tilde{h}_1\}, \{\tilde{h}_2\}, \ldots, \{\tilde{h}_w\}\right)$ とする．ただし．$\tilde{h}_j$ はランダムに選んだ 1 つ目のヒント要素とする．

提案するヒント生成アルゴリズムの流れを以下に示す．

(1) $B^{(2)}, \cdots, B^{(s)}$ については以下のように生成する：

- (1-1) $B^{(i)}$ のそれぞれのヒント集合 $H_j^{(i)}$ から SA を用いて解をサンプリングし，統計情報 $freq[j, p, v]$ を作る．このとき，それぞれのヒント集合の持つ解の個数 $|S(H_j^{(i)})|$ は一般に異なるが，3.2 節で説明するように $H_j^{(i)}$ について同じ回数 SA で解のサンプリングを試みることによってこの差異を補正することができる．
- (1-2) $freq[1, p, v], \cdots, freq[w, p, v]$ の中で個数の少ないヒント要素 $[j, p, v]$ を w 個を選択する．ただし，同一のヒント集合 j から選択されるヒント要素の数は，高々 2 個とする．

©2022 Information Processing Society of Japan

(1-3) これらのヒント要素を，それぞれのヒント集合 $H_j^{(i)}$ に添加して $H^{(i+1)}$ を作り，これを集めて $B^{(i+1)}$ とする．

(2) $B^{(s+1)}, \cdots, B^{(r)}$ の生成についても基本は (1) と同様であるが，$H^{(s)}$ の解集合は大きく BT で計算するのは効率が良くないのに対して，$H^{(s+1)}$ の解集合の大きさはそれに比べて小さくなるので BT を用いることができる．また，$B^{(s+1)}$ のそれぞれのヒント集合 $H_j^{(s+1)}$ が $|S(H_j^{(s+1)})| < \mathcal{R}$ とならない場合は，$B^{(1)}$ から再度生成しなおす．ただし，ここでは $\mathcal{R} = 160,000$ としている．$|S(H^{(r)})| = 1$ となるヒント集合 $H^{(r)}$ を生成した時点で，一連の処理が停止する．これについては 3.2 節で説明する．

本節の次項以降では提案するヒント生成アルゴリズムのそれぞれの部分の詳細について述べる．

3.2 $H_j^{(i)}$ からの SA による解のサンプリングについて

本アルゴリズムでの SA では遷移回数の上限を $K = 11,000$ にしている．解が得られる前に遷移が K を超えた場合は解のサンプリングに失敗したことになる．逆温度 β は 0 からスタートし，遷移 100 回ごとに $\beta \leftarrow \beta + \Delta$ として増加させている．ただし，β が上限 $\beta_{\max}$ に達した場合には，それ以上大きくすることはしない．$H^{(i)}$ に対する Δ と $\beta_{\max}$ は

$$(\Delta, \beta_{\max}) = \begin{cases} (0.0875,\ 3.5), & (i < 11) \\ (0.1,\ 4.0) & (i \geq 11) \end{cases} \tag{6}$$

とする．

3.3 $B^{(i)}$ からの SA によるサンプリングによる $B^{(i+1)}$ の構成について

本アルゴリズムでは BS を用いていることから，性質 1 に基づいてそれぞれのヒント集合から解をサンプリングし，その結果を $freq[j,p,v]$ に登録してそれぞれのヒント候補の出現回数を数え上げる．その際，ヒント集合が複数存在することから元のヒント集合の解集合の大きさが異なる場合，それぞれのヒント集合から一定回数解をサンプリングして得た統計を同等に扱うことは正しくない．

我々がそれぞれの段階で添加すべきヒント要素は解集合の大きさをなるべく小さくするものであるが，もともと多くの解集合を持つようなヒント集合では割合としてはより $freq[j,p,v]$ が小さくならないと実際の解集合の個数が同等に減少しない．そのため，それぞれのヒント集合に関する統計を平等に扱うには元の解集合の要素数 N_j を掛けて $N_j freq[j,p,v]$ で比較すれば良い（図 4 上図参照）．

しかし，それぞれのヒント集合の要素数を見積もることは容易ではない．そこで，本アルゴリズムでは以下のような仮定をすることで，それぞれのヒント集合について同じ

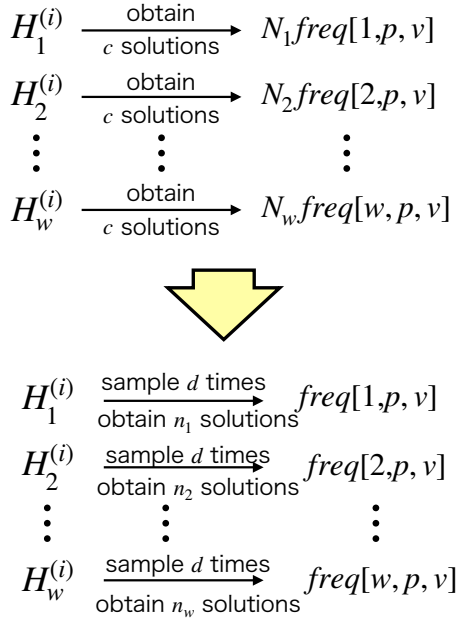

図 4 ビームサーチにおけるそれぞれのヒント集合でのサンプリングの考え方．

回数解をサンプリングすることを試みる（すべての試行で解が得られるとは限らない）ことで，結果的に元のヒント集合の解集合の大きさに比例する係数が掛けられると考える．すなわち，本アルゴリズムでは，ヒント集合が与えられたとき，その解集合の大きさは解のサンプリングの成否の割合に比例すると仮定している：

仮定 1. $B^{(i)} = \left(H_1^{(i)}, H_2^{(i)}, \ldots H_w^{(i)}\right)$ のそれぞれのヒント集合 $H_j^{(i)}$ についてその解集合の要素数が N_j とすれば，

$$N_1 : N_2 : \cdots : N_w \fallingdotseq n_1 : n_2 : \cdots : n_w. \tag{7}$$

ただし，n_j はそれぞれのヒント集合 $H_j^{(i)}$ について生成したマルコフ連鎖を用いて一定回数状態遷移を繰り返したとき，上限の繰り返し回数までに数独パズルの解が得られた回数とする．

この仮定により，図 4 下図に示すように，それぞれのヒント集合に関する統計を同数の試行を行うことで調整することができる．

図 5 に生成された大きさ 14 の色々なヒント集合 $H^{(14)}$ について BT によって得られた解集合の大きさと，マルコフ連鎖によってそのヒント集合から解が得られた割合をプロットしたものを示す．ヒント数 14 は BT と SA が両方計算可能な限界の値となる．これを見るとあまりはっきりはしないが，比例関係が成り立っているように見える．ヒント数がもっと小さい領域においては，もっとはっきりと比例関係が成り立つと考えられる．

なお，i が大きくなるにつれて $|S(H^{(i)})|$ が減り，解をサンプリングしづらくなるため，以下のように，サンプリングの試行回数 t_i は i に概ね比例して大きくなるように

©2022 Information Processing Society of Japan

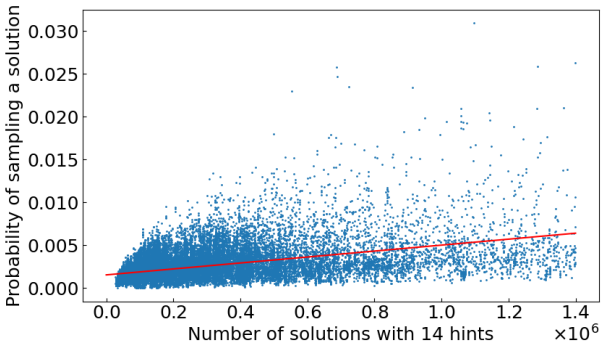

図 5 本アルゴリズムによって生成された 50,000 個のヒント数 14 のヒント集合 $H^{(14)}$ について，解の個数 $|S(H^{(14)})|$ と SA を用いてそれから解が得られた割合をプロットした散布図.

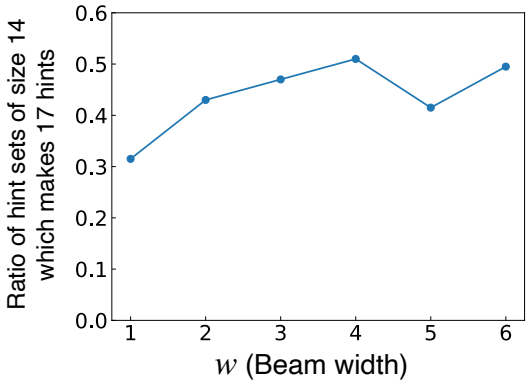

図 6 ビーム幅 w と確率 $\Pr(|S(H_1^{(14)})| < \mathcal{R})$ の関係.

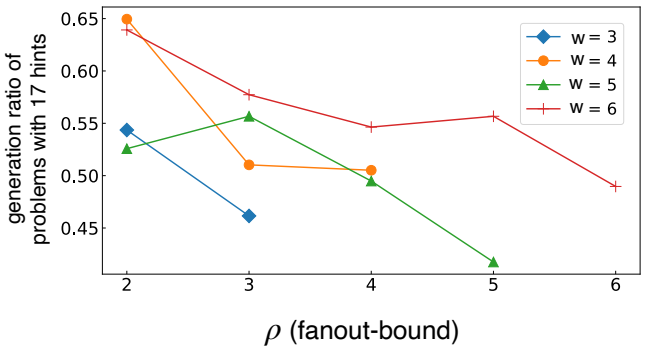

図 7 ビーム幅 w と 1 つの状態から派生する状態の個数の制限 ρ を決めたときの $B^{(14)}$ における確率 $P(w,\rho) = \Pr(|S(H_1^{(14)})| < \mathcal{R})$ の変化.

設定した：$t_1 = 750$, $t_2 = 750$, $t_3 = 1,000$, $t_4 = 1,000$, $t_5 = 1,500$, $t_6 = 2,500$, $t_7 = 4,500$, $t_8 = 6,000$, $t_9 = 8,000$, $t_{10} = 11,000$, $t_{11} = 13,000$, $t_{12} = 15,000$, $t_{13} = 10,000$. また，サンプリングの試行はそれぞれのマルコフ連鎖上での遷移回数が 11,000 回を超えた時点で解が得られていなければ中止する.

$i < 12$ の場合，上記の回数サンプリングを試行することで解を 400〜1,000 個ほど得ることができる．i が大きくなると得られる解の個数が減ってしまい，統計 $freq[j,p,v]$ でカウントされないヒントが出現する可能性が出てくる．一方，$i > 12$ を満たす $H_j^{(i)}$ については BT を用いて $|S(H^{(i)} \cup h'_{i+1})| > 0$ となるヒントの候補 h'_{i+1} を効率よく列挙することができる．そのため，サンプリングした解に出現しない $freq[j,p,v]$ に対しても，添加するヒント要素の候補として扱うことが可能になる．これについては，3.6 節で詳述する.

3.4 ビーム幅について

本アルゴリズムでは以下に示す理由からビーム幅は $w = 4$ としている．BS を用いるとき解の個数をより柔軟に減少させるビーム幅を調べてみる．ビーム幅 $w = 1, \ldots, 6$ について $B^{(14)}$ を生成し，$|S(H_1^{(14)})| < \mathcal{R}$ となる確率を図 6 に示す．ただし，$H_1^{(14)}$ は $B^{(14)}$ のヒント集合の中で解集合の大きさが最小のものであるとする．この図より，$|S(H_1^{(14)})| < \mathcal{R}$ となる $H_1^{(14)}$ を生成する確率は，ビーム幅 $w = 1$ で最も低くなり，$w = 2, 3, 4$ の順番に確率が高くなっていき，$w = 4$ のとき最大の確率となっている.

$B^{(i)}$ $(i = 1, \ldots, 12)$ から $B^{(i+1)}$ を生成するとき，解の個数が減少しずらい $H_j^{(i)}$ から更新された w 個の $H_{j'}^{(i+1)}$ で $B^{(i+1)}$ が構成され $B^{(i+1)}$ から $B^{(14)}$ まで生成すると $B^{(14)}$ のそれぞれの $H_j^{(14)}$ の解の個数が $\mathcal{R}$ 個を下回らなくなるケースが存在する可能性がある．また，解の個数が減少しずらい $H_j^{(i)}$ の判定法は見つかっておらず，このような $B^{(14)}$ を生成することを避けるために，$B^{(i)}$ の複数の $H_j^{(i)}$ から $H_{j'}^{(i+1)}$ を生成して $B^{(i+1)}$ を得るように制約を設定することを考える．ここでは，$B^{(i)}$ から $B^{(i+1)}$ を生成する際に，1 つの $H_j^{(i)}$ から更新されて得た状態 $H_{j'}^{(i+1)}$ の個数が高々 ρ 個となるという制約を設けた．このような制約をつけて $B^{(14)}$ を生成したときの確率 $P(w,\rho) = \Pr(|S(H_1^{(14)})| < \mathcal{R})$ を図 7 に示す．w を一定にすると $\rho = 2$ で $P(w,\rho)$ が最大となるケースがほとんどである．また，ρ が大きくなるにつれ，$P(w,\rho)$ は減少する傾向にある．図の中での $P(w,\rho)$ 最大値は $(w,\rho) = (4,2)$ で与えられる．本研究ではこの結果と前段落の結果を合わせて $(w,\rho) = (4,2)$ とした.

3.5 限界 $\mathcal{R}$ の決定方法

提案するアルゴリズムにおいては，$|S(H^{(14)})|$ が大きいと最終的にヒント数 17 の問題生成ができる確率が低くなってしまう．そこで，$B^{(14)}$ を生成したときに，ある限界 $\mathcal{R}$ を設けて，$|S(H^{(14)})| \geq \mathcal{R}$ となるヒント集合が存在する場合はそれ以上計算せずに，新たに最初からヒントを生成するようにしている.

この限界として最適な値を得るために，ヒントを作り出す工程で出現した $H^{(14)}$ の 1,089 個のヒントについてその解集合の大きさを BT によって計算した．変数 X を $[0, 7 \times 10^5]$ の範囲で動かしたとき，解集合の大きさが X 以下であるヒント集合のうち 17 個のヒントが得られた割合を図 8 に示す.

©2022 Information Processing Society of Japan

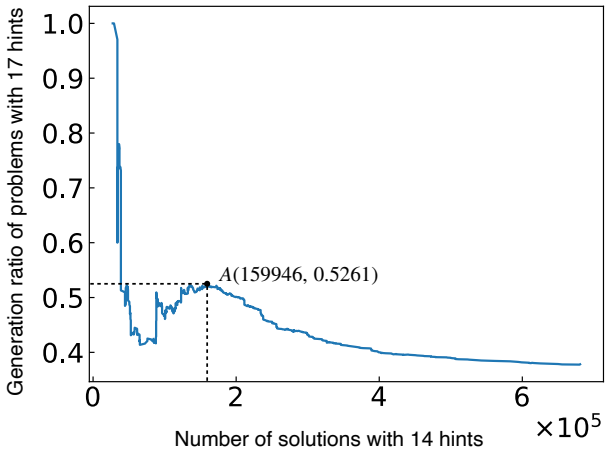

図 8 1,089 個の $H^{(14)}$ について，それぞれ解集合の大きさ $|S(H^{(14)})|$ を計算し，この大きさが X 以下であるヒントのうち，最終的に 17 個の数独問題が得られた割合をプロットしたグラフ．このグラフは最初の部分のノイズを除外すると A の位置で最大値をとっている．

この図を見ると A の位置で 17 個のヒントが得られる割合が最大になる．このことから本アルゴリズムでは $\mathcal{R} = 1.6 \times 10^5$ と設定した．

再度 $B^{(1)}$ からヒントを生成することで，ヒント数 17 の問題生成確率を高めている．提案するアルゴリズムでは，1 つの数独パズルの問題を生成するまでに $B^{(1)}$ からの生成処理を行った回数の平均値は約 2.3 回であった．

3.6 $B^{(14)}$ の生成方法

BT を用いて安定して解集合を列挙することができるのはヒント数 14 以降である．しかし，それぞれのヒント候補 (p, v) が 14 個目のヒントになり得るかどうかは BT で判別することができる．そのため，本アルゴリズムでは $B^{(13)}$ から $B^{(14)}$ を生成する際，候補となるヒントを BT で列挙し，それとは別に SA で解をサンプリングしている．この 2 つの結果を照合することにより，最適なヒントの候補を漏らすことなく列挙し，統計 $freq[j, p, v]$ をとることができる．実際 $H^{(13)}$ では SA での解のサンプリングの効率はあまりよくなく，SA のみではサンプリングされずにヒントの候補とならないものが存在しやすくなる．ここで列挙されたヒントの候補のうち，統計 $freq[j, p, v]$ の少ない 50 個の $[j, p, v]$ をそれぞれのヒント集合 $H_j^{(13)}$ に添加して，改めて BT によって解集合の大きさを求めて，それに基づいて $B^{(14)}$ を構成している．

また，$B^{(14)}$ の生成に際しては，ビーム幅を広げて $w = 4$ から $w = 8$ にしている．

3.7 ヒント 15, 16, 17 個目をまとめて $H^{(14)}$ に添加する

最後に，$B^{(14)}$ から $B^{(15)}$ と $B^{(16)}$ を順番に生成せずに一足飛びに $B^{(17)}$ を生成することを考える．このような探索をすることで，ヒント数 17 のヒント集合が構成できる場合には，必ずそのヒント集合を手に入れることが可能となる．$B^{(14)}$ までヒント集合が大きくなると，探索すべき空間は相当に小さくなり，このような計算が可能となる．

ここでは，以下のような手順で計算を行う：まず，$B^{(14)}$ のそれぞれの $H_j^{(14)}$ に対して，K_p を数独パズルの 81 個の位置から 3 箇所を選択した組の集合として，添加する 3 つのヒント要素の組の集合を

$$Q = \{[(p_1, v_1), (p_2, v_2), (p_3, v_3)] \mid p_1, p_2, p_3 \in K_\mathrm{p},\ 1 \le v_1, v_2, v_3 \le 9\} \quad (8)$$

とする．Q のそれぞれの要素を用いて仮の 17 個のヒント集合 $H'^{(17)} = H_j^{(14)} \cup \{(p_1, v_1), (p_2, v_2), (p_3, v_3)\}$ を生成する．そのヒント集合 $H'^{(17)}$ に対して BT を用いて解の個数を計算し，解の個数が 1 以上かつ最も少ない w 個の $H'^{(17)}$ を $B^{(17)}$ の要素とする．第 4 節に示す実行環境下では $B^{(14)}$ から $B^{(17)}$ を生成する時間は 6〜12 分であった．

4. アルゴリズムの性能比較

図 9 に [2], [3], [7], [10]，および本研究のヒント数別生成割合を示す．ヒント数 17 の問題生成確率は，先行研究では約 6%〜約 50% であるのに対し，本研究では約 83% まで向上させることができた．また，1 つの問題の平均生成時間は，先行研究で約 90 分〜105 分であったのに対して本研究では約 75 分であった[*1]．

ただし，先行研究では SA での状態の評価値の計算をその都度盤面全体について計算していたが，本研究では値が変化した部分だけに注目して計算しているため，この部分でかなりの効率化が実現できている．そのため，本アルゴリズムがアルゴリズムとしてより効率的になっているかどうかは検証できていない．

また，生成した数独パズルのヒントは並びとしては一致しないが，ブロックの入れ替えや回転などによって，実質的に同じヒントになってしまう場合がある．提案するアルゴリズムで生成したヒント数 17 のヒント集合については，390 個に対して 263 種類の異なるヒントが生成されていた．一方，ビーム幅を 1 とした場合，同様に生成されたヒント数 17 のヒント集合 390 個に対して 305 個の異なるヒントが生成されていた．ビームサーチを用いると若干同じ問題が出来やすくなる傾向があると考えられる．

5. 結論および今後の課題

本論文ではヒントの少ない数独パズルを確率的に作り出す方法について述べた．特に数独パズルの最小ヒント数は 17 であることが知られているため，ヒント数 17 の数独パズルを効率的に生成するアルゴリズムを提案した．本アル

[*1] 実行環境は以下の通りである： OS：Linux Ubuntu 16.04.7, CPU：Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40GHz，メモリ：64G，コンパイラ：gcc5.4.0.

©2022 Information Processing Society of Japan

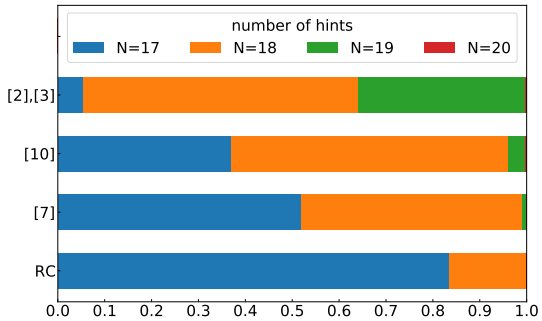

図 9 先行研究 [2], [3], [7], [10] と本研究（RC）のヒント数 N ごとの問題生成割合.

ゴリズムでは，先行研究 [10] と同様に空集合からスタートしてヒントの候補を探し出し，最適なヒント要素をヒント集合に添加していく．本アルゴリズムではヒント数 17 の数独パズルを生成できる割合をなるべく高くすることを目指した．最適なヒント要素を見つけ出すために，メトロポリス法に基づくシミュレイテッドアニーリングを用いたり，バックトラックを用いている．また，幅広くヒント集合を探索するために，ビームサーチの手法を取り入れた．本アルゴリズムではビーム幅を $w = 4$ して，常に 4 つのヒント集合を並列に持つことでより幅広くヒント集合を探すことができている．さらに，第 3 節で述べたいくつかの工夫により効率化を図っている．その結果，ヒント数 17 の数独パズルの生成割合を約 83%まで高めることができた．

本論で述べた種々の工夫が実際にどの程度アルゴリズムの性能に貢献しているかどうかについてはまだ検証できていない．これについては今後検討していく必要がある．また，シミュレイテッドアニーリングで生成する状態数の上限などについても今後改めて精査する必要があると考えられる．

本アルゴリズムでは生成されたヒント集合の質については特に問題にしていない．しかし，本質的に同じようなヒント集合が多数生成されているとすれば，アルゴリズムの品質に関わる問題となる．そこで生成されるヒント集合がどの程度バラエティに富んだものになっているかということについて今後検討する必要がある．前節の最後で述べたようにビーム幅 1 と 4 で比べた場合，ビーム幅 1 の方が若干バラエティに富んだヒント集合が生成されていることが確認できた．今後の課題として，ヒント数 17 の問題生成確率は下げずに本質的に多くのヒント集合を生成するアルゴリズムについて検討する必要がある．特に，我々は $\mathcal{R}$ の値の設定によってヒント集合の質が変わってくるという感触を持っており，これについては今後の課題である．

参考文献

[1] G. McGuire, B. Tugemann, and G. Civario: There is no 16-clue Sudoku: Solving the Sudoku minimum number of clues problem via hitting set enumeration. *Experimental Mathematics*, 23:2, pp. 190-217 (2014).

[2] 古川湧，山本修身：MCMC 法によるヒントの少ない数独問題の生成．情報処理学会全国大会，6K-06 (2021).

[3] 古川 湧：ヒントの少ない数独パズルの生成に関する研究．2020 年度名城大学大学院理工学研究科修士論文 (2021).

[4] 藤原 暁宏：アルゴリズムとデータ構造．第 2 版，情報工学レクチャーシリーズ，森北出版（2016）.

[5] Berg, B.: Markov Chain Monte Carlo Simulations and Their Statistical Analysis. World Scientific Publishing Co. Pte. Ltd. (2004).

[6] 池田 智悟，窪田 耕明：シミュレーテッドアニーリング概説．`http://mikilab.doshisha.ac.jp/dia/monthly/monthly00/20000415/ikeuchi_kubota.pdf`，参照 2022-01-06 (2000).

[7] 長尾卓，山本修身：ヒント数 17 の数独パズルの MCMC を利用した生成について．電子情報通信学会 2022 年総合大会，ジュニア&学生ポスターセッション，ISS-A-022 (2022).

[8] 長谷洋斗，川原純，笠原正治：解の多様性を考慮したビームサーチと局所探索法によるフロンティア法を高速化するための変数順序付け．第 112 回人工知能基本問題研究会，pp.30-35 （2020）.

[9] Pinn, K., Wieczerkowski, C.: Number of magic squares from parallel tempering Monte Carlo. *International Journal of Modern Physics C*, Vol. 09, No. 04, pp. 541–546 (1998).

[10] 長尾卓，山本修身：ヒントの少ない数独パズルの効率的な生成について．情報処理学会第 84 回全国大会，5S-02 (2022).

©2022 Information Processing Society of Japan