情報処理学会論文誌　Vol.59　No.1　1–8 (Jan. 2018)

数独に対する最も簡単な解法探索による難易度判定付きソルバー

鹿屋 直大[†1]　斎藤 寿樹[†1]

受付日 2016年3月4日，再受付日 2015年7月16日 / 2015年11月20日，
採録日 2016年8月1日

概要：近年, 自作の問題を発表できるサイトの公開などにより, パズルを作った時に自分でその難易度を判断する場面が増えてきた. 難易度の判定には,1 つのパズルに様々ある解法の中で最も簡単なものを探す必要がある. しかし, 人間が行うとそれには全ての解法について同じ問題を解き直す必要があるのでかなり時間がかかり, 正確さも保障できない. よって, 問題を入力として与えると難易度の目安が確認できるソルバーがあると便利である. 本研究では, 数独の問題を与えると, 解の一意性を判定でき, 一番簡単な解法を選んだ場合に難易度の目安となる点数を計算するアルゴリズムを作成した.

キーワード：未定

How to Prepare Your Paper for IPSJ Journal (ipsj.cls version 2.01)

Naohiro Kanoya[†1]　Toshiki Saito[†1]

Received: March 4, 2016, Revised: July 16, 2015/November 20, 2015,
Accepted: August 1, 2016

Abstract: In recent years, with the opening of websites where people can publish their own puzzles, there are more and more situations where people have to judge the difficulty level of a puzzle they have created themselves. To judge the difficulty level of a puzzle, it is necessary to find the easiest solution among the various solutions to a puzzle. However, this requires a human to re-solve the same problem for all the solutions, which takes a lot of time and accuracy cannot be guaranteed. Therefore, it is useful to have a solver that can check the difficulty level of a puzzle when given a problem as input. In this study, we created an algorithm that, given a Sudoku problem, can determine the uniqueness of the answer and calculate a score that serves as an indication of the difficulty level when the easiest solution is chosen.

Keywords: IPSJ Journal, LaTeX, style files, “Dos and Don’ts” list

1. はじめに

多くのパズルは解が一意であることが重要とされる．パズルの作り手は自作の問題を作成した時に, その解の一意性を確認する必要がある．それに加え, 近年では自作の問題を発表できるサイトの公開などにより, パズルを作った時に自分でその難易度を判断する場面が増えてきた?. しかしこれらを人間が手で行う場合, 自分で作った問題をもう一度自分で解き直さなければならないため, 時間がかかる. 特に難易度の判定に時間がかかる. 多くのパズルは, パズルを解くためのテクニックである「手筋」を使う順番の違いなどによって解法が様々にある. そのようなパズルの難易度を判定する場合, その解法の中で一番簡単なものを見つけ, その解法から難易度を決定することが望まれる時がある. しかしそれを人間が行うには全ての解法に対して同じ問題を解いて難易度を確認しなければならない. それにはかなり時間がかかり, また人の手で行っているため正確

[1] 情報処理学会
IPSJ, Chiyoda, Tokyo 101–0062, Japan
[†1] 現在，九州工業大学
Presently with Kyusyu Institute University

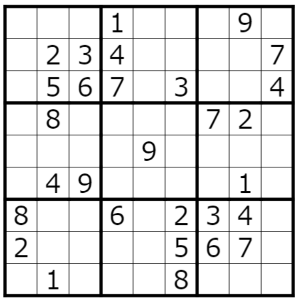

図 1　数独の例
Fig. 1　Example of sudoku

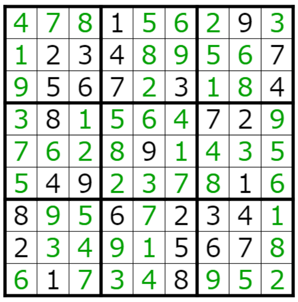

図 2　数独の例の答え
Fig. 2　Example of sudoku - Answer

に行うのも難しい. よって, 問題を入力として与えると, 一意性の確認と難易度の目安の出力ができるソルバーがあると便利である. 先行研究に手筋を見つけるのに探索した領域の数により数独の難易度を数値化するような研究があった?. しかしそれは盤面に片っ端から手筋を適用しており, 多数ある解法の中で一番簡単なものを特定していないため本来人が感じる難易度とは異なる可能性がある. よって本研究では, 数独の問題を与えると, 解の一意性を判定でき, 一番簡単な解法を探索してその解法に対する難易度の目安となる点数を計算するアルゴリズムを作成した.

2.　準備

2.1　数独について

数独とは, 株式会社ニコリのペンシルパズルである. 同じルールではあるが, 別の名前で別の会社がこのパズルを取り扱うこともある. 数独のルールは以下の通りである?.

- あいているマスに, 1 から 9 までの数字のどれかを入れる.
- タテ列 (9 列ある), ヨコ列 (9 列ある), 太線で囲まれた 3 × 3 のブロック (それぞれ 9 マスあるブロックが 9 つある) のどれにも 1 から 9 までの数字が 1 つずつ入りる.

2.2　数独の手筋について

以下,1 から 9 までの数字が 1 つずつ入る領域（ブロック, 行, 列）を指す言葉として「場所」と示す事がある.

2.2.1　ブロッケン

ブロック内において, ある数字の入る場所が 1 つしかない時にその場所にその数字を入れられるという初級手筋がある. これを, 株式会社ニコリが取り扱っている用語としてブロッケンと呼ぶことにする?. 図 3 の盤面の場合, 左上のブロックにおける 1 の位置が図 4 のように決定する.

2.2.2　補填

ブロック, 行, 列においてその中の 8 マスが数字で埋まったら, 残りの 1 マスもその 8 マスの中に使われていない数字に決定する初級手筋がある. これを補填と呼ぶことにする. 図 5 の盤面の場合, 一番上の行の一番右のマスに入る数字が図 6 のように決定する.

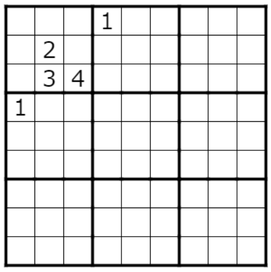

図 3　ブロッケンの例
Fig. 3　Example of blocken

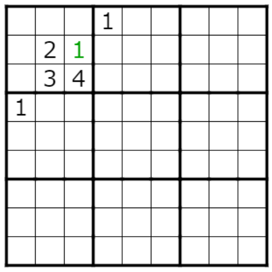

図 4　ブロッケンの例:埋めた後
Fig. 4　Example of blocken: after filling

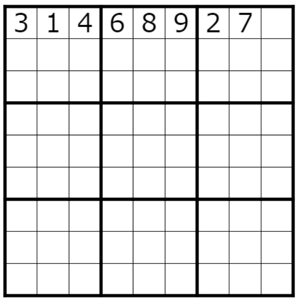

図 5　補填の例
Fig. 5　Example of compensation

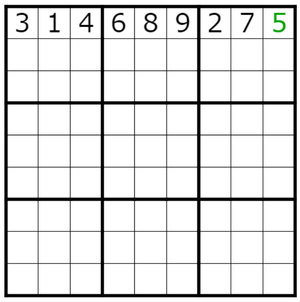

図 6　補填の例:埋めた後
Fig. 6　Example of compensation: after filling

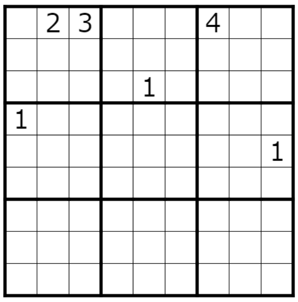

図 7　レッツミーの例
Fig. 7　Example of rettumi

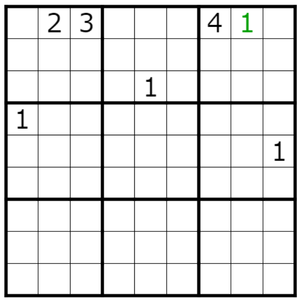

図 8　レッツミーの例:入れた後
Fig. 8　Example of rettumi: after filling

2.2.3　レッツミー

行もしくは列内において, ある数字の入る場所が 1 つしかない時にその場所にその数字を入れられるという中級手筋がある. これを, 株式会社ニコリが取り扱っている用語としてレッツミーと呼ぶことにする?. 図 7 の盤面の場合, 一番上の行における 1 の位置が図 8 のように決定する.

2.2.4　マスミ

あるマスにおいて,8 種類の数字が同じブロック, 行もしくは列にあるため, その 8 種類の数字にない 1 つの数字をそのマスに埋められる中級手筋がある. これを, 株式会社ニコリが取り扱っている用語としてマスミと呼ぶことにする?. 図 9 の盤面の場合, 一番左上のマスが図 10 のように 9 に決定する.

2.2.5　いずれにしても理論

ある数字について, あるブロック内で入れるマス全てが同じ行 (列) である場合, いずれのマスに入っても, その行

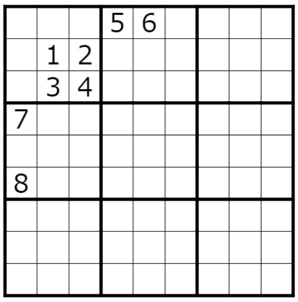

図 9 マスミの例
Fig. 9 Example of masumi

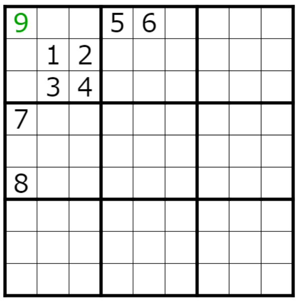

図 10 マスミの例:埋めた後
Fig. 10 Example of masumi: after filling

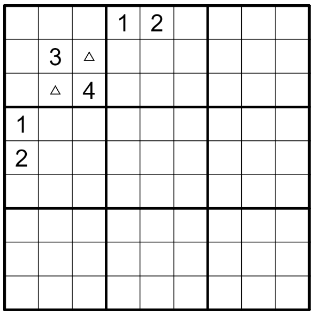

図 13 表予約の例
Fig. 13 Example of front reservation

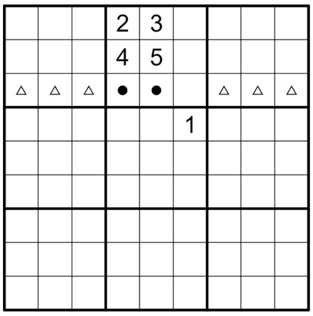

図 11 いずれにしても理論の例:ブロックの場合
Fig. 11 Example of theory”in any case”: block

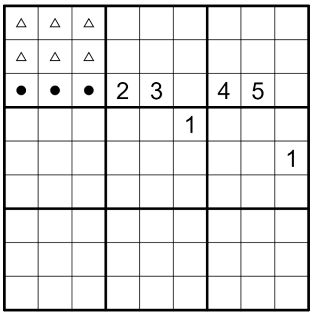

図 12 いずれにしても理論の例:行の場合
Fig. 12 Example of theory”in any case”: row

(列) にある他のブロックに属しているマスで, その数字が入る事はない. 図 11 の場合, 数字が 4 つ入っている中央上のブロックにおいて,1 が入る場所は黒丸で示した 2 つのマスのいずれかであり, どちらも上から 3 行目に位置している. この 2 つのマスのいずれに 1 が入っても, 三角で示したマス, つまり同じ 3 行目で中央上のブロックに属していないマスには 1 は入らない.

また, ある数字がある行 (列) 内で入れるマス全てが同じブロックである場合, いずれのマスに入っても, そのブロックにある他の行（列）に入っているマスでその数字が入る事はない. 図 12 の場合, 上から 3 行目において,1 が入る場所は黒丸で示した 3 つのマスのいずれかであり, どちらも一番左上のブロックに属している. この 3 つのマスのいずれに 1 が入っても, 三角で示したマス, つまり同じ左上のブロックに属し上から 3 行目には属さないマスに 1 は入らない.

これらを, 株式会社ニコリが取り扱っている用語として「いずれにしても理論」と呼ぶことにする?.

2.2.6 表予約

n は自然数で $2 \leq n \leq 7$ とする.1 から 9 のうち n 種類の数字に対して, あるブロックもしくは行か列でそれらの数字が入るマスが, 同じ n マスのうちのいくつかのマスである場合, その n マスの全てのマスについてその n 種類の数字以外の数字は入らない. このような手筋を株式会社ニコリが取り扱っている用語の「予約」を使って, 次に説明する手筋と区別するために「表予約」と呼ぶ?. 例えば図 13 では, 左上のブロックにおいて 1,2 が入る可能性のあるマスは三角で表した 2 マスである. このいずれかの三角で示したマスに 1,2 以外の数字が入ると,1,2 が入る事のできるマスはどちらも一方の三角で示したマスのみになる. しかし, 同じマスに 2 つの数字を入れる事はできない. よって, 三角で示したマスに 1,2 以外のマスは入らない.

なお,n 種類の数字とマスについての表予約（以下,n マスの表予約と呼ぶ）を使う場所（1 つのブロック, 行, 列）では,$n + 2$ マス以上の空きマスが必要である.n マスより少ない場合はマスが足りない事が自明である.n マスの場合は空きマスの全マスで表予約が成立するという事になるが, 他の空きマスがないのでどのマスにも消す候補数字がなく, 手筋を使うことはできない.$n + 1$ マスの場合,n マスの表予約があったとすると, 表予約に使われていない残り 1 マスには n 種類の数字が入らなかったという事になる. 空きマスの数よりこの場所で埋められる数字の種類は $n + 1$ 個で, そのうち n 種類の数字がそのマスに入らないとなると残りは 1 種類であるから, マスミを使って入る数字が決定される. よって, この場所の残り空きマスは n マスとなるので,n マスの表予約は使えない.

2.2.7 裏予約

n は自然数で $2 \leq n \leq 7$ とする. あるブロックもしくは行か列で, そのうち n マスそれぞれについて入る可能性のある数字が, どれも同じ n 種類の数字のうちのいくつかであった場合, その n マス以外のマスにおいてその n 種類の数字は入らない. このような手筋を株式会社ニコリが取り扱っている用語の「予約」を使って, 表予約と区別するために裏予約と呼ぶ?. なお, 株式会社ニコリが取り扱っている「予約」は, 表予約と裏予約の両方を指しており, 本稿でも表予約と裏予約の両方を指すときは「予約」と呼ぶ事にする. 例えば図 14 では, 一番上の行において 1 マスに 8 と 9 が入っているマスが 2 マスある. この 2 マスにはどちらも 8 か 9 のいずれしか入らない. ここで, 三角で示したマスに 8 が入ると, この 2 マスのいずれも 9 しか入らなくなり, 入る数字は 9 に決定する. しかし, この 2 マスは同じ行にあるので 2 つの数字が同じ行に入る事になり, ルールと矛盾が生じる. 三角で示したマスに 9 が入る場合も同様である.

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

図 14　裏予約の例
Fig. 14　Example of front reverse

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

図 15　井桁理論の例
Fig. 15　Example of theory ”igeta”

したがって, 三角で示したマスには 8 と 9 は入らない.

なお, 裏予約は表予約に言い換える事ができる. 理由を説明する. 裏予約があるブロックもしくは行か列における, 裏予約に使われている以外の空きマスの数を k とする. 裏予約が成立しているマスでは, 少なくともその裏予約がある場所ですでに入っている数字と裏予約に使われている数字を除いた k 種類の数字が入らない. ここで, 裏予約がある場所における, 裏予約に使われている以外の空きマスの数も k 個であるため, その k 種類の数字の入る場所は全て同じ n マスのうちのいくつかのマスとなる. ゆえに,n マスの裏予約は, 最大 k マスの表予約となる.k マス以下の場合もありえる理由は,k 個のマスで,k 種類の数字の中で入る数字が 1 つしかない場所がいくつかあることも考えられ, その場合そのマスはブロッケンかマスミでその数字に決まり, 残りのマスで予約ができる. よって必ずしも k マス丁度で予約が成立している訳ではなく,k よりも少ないマス数で予約が成立している場合もある. なお, 同様にして表予約を裏予約に言い換える事もできる.

このように予約を表予約と裏予約で分けた理由は, 数独を解く際に, 表予約の考え方を使った方が発見しやすい場合と, 裏予約の考え方を使った方が発見しやすい場合があるからである. 例えば図 13 では, 表予約を使うと 2 つの数字と 2 つのマスについて考えるだけでよいが, 裏予約を使うと 5 つの数字と 5 つのマスについて考える必要があり, 見つけづらい. 図 14 では, 裏予約を使うと 2 つの数字と 2 つのマスについて考えるだけでよいが, 表予約を使うと 5 つの数字と 5 つのマスについて考える必要があり, 見つけづらい. ある場所での最大の空きマスが 9 マスの場合, この時 5 マス〜7 マスの表予約が 2 マス〜4 マスの裏予約に対応し,5 マス〜7 マスの裏予約が 2 マス〜4 マスの表予約に対応する. よって, この考え方を使うと表予約も裏予約も 2 マス〜4 マスのものについて考えるだけで全ての予約を発見する事ができる.

なお,n 種類の数字とマスについての裏予約（以下,n マスの裏予約と呼ぶ）を使う場所（ブロック, 行, 列）では,$n+2$ マス以上の空きマスが必要である.n マスより少ない場合はマスが足りない事が自明である.n マスの場合は空きマスの全マスで裏予約が成立するという事になるが, 他の空きマスがないので消す候補数字がなく, 手筋を使うことはできない.$n+1$ マスの場合,n マスの裏予約があったとすると, 裏予約に使われていない残り 1 つの数字は, この n マスには入らなかったという事になる. よって, この数字の入る場所はブロッケンもしくはレッツミーを使って裏予約に使われていない残り 1 マスに入る事が決定される. よって, この場所の残り空きマスは n マスとなるので,n マスの裏予約は使えない.

2.2.8　井桁理論

n は自然数で $2 \leq n \leq 4$ とする. ある n 行で, その n 行全てにおいてある数字の入る場所が n マス以下の時, それぞれのマスが属する列が,n 行全てにおいて同じ n 列のうちのいずれかであった場合, その n 列で, その n 行以外のマスにその数字が入る事はない. これを, 株式会社ニコリが取り扱っている用語として「井桁理論」と呼ぶことにする?. なお, この n 行が全て上から 1〜3 行目のいずれかであるか, または上から 4〜6 行目のいずれかであるか, または上から 7〜9 行目のいずれかである場合, マスに入る数字の候補からその数字を消すのに, いずれにしても理論が先に適用される事になるので, 井桁理論は成立しない. また,n 行で数字が入る n 列について, それが全て左から 1〜3 列目のいずれかであるか, または左から 4〜6 列目のいずれかであるか, 左から 7〜9 列目のいずれかである場合もいずれにしても理論が先に適用されるので, 井桁理論は成立しない. なお, この説明について行と列を入れ替えた場合も成り立つ.

図 15 の例では, 上から 1 行目で 1 が入る場所は左から 2 列目か 8 列目のいずれか 2 マスである. 上から 4 行目で 1 が入る場所も, 左から 2 列目か 8 列目のいずれか 2 マスである. ここで左から 2 列目で上から 1 行目と 4 行目以外の場所に 1 を入れると, 上から 1 行目,4 行目ともに 1 が入る場所は左から 8 マス目のみになってしまい, 同じ列で数字の重複が発生する. よって, 左から 2 列目では, 上から 1 行目と 4 行目以外の場所に 1 は入らない. 同様に, 左から 8 列目でも上から 1 行目と 4 行目以外の場所に 1 は入らない. 具体的には, 井桁理論を適用すると新たに三角のマスに 1 が入らない事が分かる.

3. 手法について

3.1 解法から難易度点数を計算する方法

本研究では数独の問題が与えられると, 一番簡単な解法を探し, その解法から難易度点数を決定している. 本研究では解法を, 使用いた手筋の列挙で表すとしている. その用いた手筋には, その手筋が難しいほど高くなるような正の点数が付けられている. 解法である複数の手筋において, 各手筋の点数を合計した値をその解法の点数としている. このようにして最も簡単な解法から難易度点数を計算している.

3.2 最も簡単な解の探索方法

最も簡単な解の探索では, 分枝限定法を用いている. 各局面は解答された状態である最終局面以外は数独の解答途中の盤面で, 手筋を使用する前の盤面である. その局面で使える全ての手筋を列挙し, それを適用した盤面をそれぞれの手筋ごとに作成する. 適用された盤面が適用する前の盤面を親とした新しい局面となる. これを最終局面に辿りつくか, 途中で枝刈りをされるかまで続ける. また, 各局面はその局面の探索の中で, 現時点におけるその局面から最終局面に至るまで最も難易度点数が低くなるような解法の, その難易度点数を記録している. この記録された点数は探索中に現在の値より難易度点数が小さくなるような解法を発見したらその点数に更新されるようになっている.

また, 手筋の使用する前の盤面から手筋を使用した後の盤面を生成する時に, 使用する前の局面が持っている現時点で最も簡単な解法の難易度点数から, 使用した手筋の難易度点数を引いた数をその局面の最も簡単な解法の初期値として持っておく. これにより, 生成された局面の初期値が 0 以下になった場合, 最も簡単な解法が他にあるという事になる. そこからどのような解法を選んでも最も簡単な解法ではなくなるので, そこから先は探索しないことで枝刈りができる. このようにして探索を行い, 最終的に一番上の局面, つまり初期盤面の局面に記録された難易度点数が, 最も簡単な解法の難易度点数になる.

4. 計算機実験

本研究で 3 節のアルゴリズムを c++で実装した. 計算機のプロセッサは Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz 2.00 GHz を使用した.

株式会社ニコリ社から発行されている「てごろな数独 1」の, 難易度 Medium の問題から問題番号が 4 の倍数である 18 問を抜粋し, その問題に対して作成したプログラムを実行した?. その結果, どの問題も一意の解を求める事ができた. この時, 各問題に対して出力された難易度点数の結果を表 1 に示す. なお, 実行時間は 3 回実行した時の平均である. また, 公式難易度とは「てごろな数独 1」など株式会社ニコリの一部の数独の単行本で各問に対して設定されている,1,2,3,4,5,6,7,8,8+,9,9+,10,10+の 13 段階で設定されている難易度を示す値である. 以下に登場する数独の単行本の問題は全てこの 13 段階の難易度が設定されている.

表 1 先行研究の中級問題に対する難易度点数

問題	点数 [点]	公式難易度	実行時間 [s]
問題 16	1	4	0.0175
問題 20	1	4	0.0203
問題 24	1	4	0.0201
問題 28	1	4	0.0228
問題 32	1	4	0.0244
問題 36	1	4	0.0203
問題 40	3	5	0.0224
問題 44	2	5	0.0355
問題 48	3	5	0.0425
問題 52	3	5	0.0406
問題 56	3	5	0.0397
問題 60	6	5	0,0423
問題 64	3	6	0,0413
問題 68	4	6	0.0412
問題 72	6	6	0.0404
問題 76	6	6	0.0446
問題 80	10	6	0.0399
問題 84	18	6	0.0407

公式難易度と難易度点数で相関を取った結果, 相関係数は 0.68 となった.　　また, 株式会社ニコリ社から発行されている「難関数独 11」の問題から問題番号が 4 の倍数である 26 問を入力として抜粋した?. それらの問題に対して作成したプログラムを実行した. その結果, どの問題も一意の解を求める事ができた. この時, 各問題に対して出力された難易度点数と実行時間の結果を図に示す. なお, 実行時間は 3 回実行した時の平均である.

　公式難易度と難易度点数で相関を取った結果, 相関係数は 0.90 となった.

また, 数独の難易度を測る研究として, 手筋を探索する回数をもとに難易度の点数を決定する研究が行われている?. 本研究との比較のために, その先行研究に基づいて難易度点数を求めるプログラムを作成した. 比較対象は「てごろな数独 1」「難関数独 11」の本研究で入力した問題とした??. なお先行研究で実装されていないために探索できない手筋を使う必要があるため,「難関数独 11」の問題のうち 104 番のみ結果を取っていない.

本研究で入力した「てごろな数独 1」の問題に対する, 先行結果の難易度点数と実行時間の結果を図 3 に示す. なお, 実行時間は 3 回実行した時の平均である.

公式難易度と難易度点数で相関を取った結果, 相関係数は 0.27 と本研究より低い値となった.

また, 本研究で入力した　「難関数独 11」の 104 番以外の問題に対する, 先行結果の難易度点数と実行時間の結果

表 2 先行研究の上級問題に対する難易度点数

問題	点数 [点]	公式難易度	実行時間 [s]
問題 4	7	7	0.0336
問題 8	7	7	0.0293
問題 12	14	7	0.0367
問題 16	7	7	0.0331
問題 20	17	7	0.0529
問題 24	20	8	0.0329
問題 28	21	8	0.0474
問題 32	7	8	0.0284
問題 36	10	8	0.0606
問題 40	28	8	0.0420
問題 44	30	8+	0.0354
問題 48	30	8+	0.0506
問題 52	40	8+	0.0413
問題 56	30	8+	0.0331
問題 60	60	8+	0.0899
問題 64	40	9	0.0846
問題 68	44	9	0.0364
問題 72	54	9	0.0828
問題 76	65	9	0.1197
問題 80	31	9	0.3216
問題 84	80	9+	0.0534
問題 88	80	9+	0.0459
問題 92	71	9+	0.1571
問題 96	90	10	0.0604
問題 100	90	10	0.0487
問題 104	320	10+	0.0487

表 3 本研究の中級問題に対する難易度点数

問題	難易度点数 [点]	公式難易度	実行時間 [s]
問題 16	41	4	0.0310
問題 20	45	4	0.0448
問題 24	74	4	0.0778
問題 28	72	4	0.0570
問題 32	65	4	0.0511
問題 36	68	4	0.0307
問題 40	66	5	0.0688
問題 44	65	5	0.0230
問題 48	75	5	0.0639
問題 52	69	5	0.0558
問題 56	71	5	0.0519
問題 60	70	5	0.0314
問題 64	61	6	0.0380
問題 68	66	6	0.0856
問題 72	70	6	0.0254
問題 76	62	6	0.0729
問題 80	75	6	0.0735
問題 84	67	6	0.0776

を図 3 に示す. なお, 実行時間は 3 回実行した時の平均である.

公式難易度と難易度点数で相関を取った結果, 相関係数は 0.58 と本研究より低い値になった.

表 4 先行研究の上級問題に対する難易度点数

問題	難易度点数 [点]	公式難易度	実行時間 [s]
問題 4	185	7	0.0406
問題 8	125	7	0.0186
問題 12	584	7	0.0189
問題 16	218	7	0.0174
問題 20	255	7	0.0188
問題 24	220	8	0.0178
問題 28	719	8	0.0186
問題 32	156	8	0.0178
問題 36	173	8	0.0176
問題 40	739	8	0.0185
問題 44	196	8+	0.0171
問題 48	156	8+	0.0171
問題 52	501	8+	0.0184
問題 56	190	8+	0.0166
問題 60	522	8+	0.0183
問題 64	169	9	0.0180
問題 68	615	9	0.0189
問題 72	742	9	0.0179
問題 76	1200	9	0.0194
問題 80	224	9	0.0175
問題 84	212	9+	0.0180
問題 88	502	9+	0.0176
問題 92	950	9+	0.0191
問題 96	2032	10	0.0208
問題 100	2321	10	0.0208

表 5 マスミの点数による相関係数の変化

マスミの点数	相関係数
3	0.80
4	0.77
5	0.73
6	0.70
7	0.68
8	0.65
9	0.63
10	0.62
11	0.60

5. 考察

5.1 中級問題について

ここで, 本研究の結果についてマスミの点数を変えると相関係数がどのように変化するのか調査を行った. その結果を表 2 に示す.

これより, マスミの点数が小さくなるほど, 相関係数が 1 に近づいている事が分かる. しかし, 実際にマスミの点数を下げる場合, 例えば 3 まで下げると, レッツミーのみを 3 回使う, 中級問題の中では比較的易しい問題と難しい問題の丁度間にある難易度のような問題と, マスミのみを 1 回使

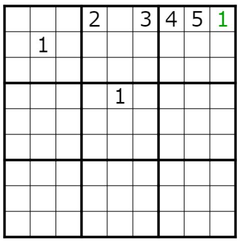

図 16 レッツミーの簡単な例

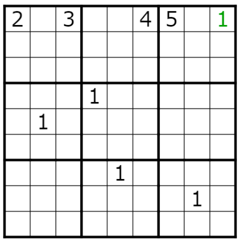

図 17 レッツミーの難しい例

う中級問題の中では比較的難しい問題が, 同じ難易度点数になってしまう. これを避けるには, 使用したレッツミーの回数が少ないほど, マスミの点数を上げるといった対策が考えられる. これにより, レッツミーの使用回数が少ないマスミを使う中級問題の点数は比較的高いままで, 複数回のレッツミーとマスミを使う問題の難易度点数を下げられ, 相関係数が高くなると考えられる.

また, 同じ種類の手筋で, かつ注目する領域が同じで注目する数字の種類もしくはマスの数が同じであっても, 数字の配置や手筋が成立する場所によって, その手筋の発見のしやすさが変わってくることがある. 例えば図??では, 一番上の一番右のマスに 1 が入ることをレッツミーで見つけるには,2 つの 1 を見る必要がある. 一方, 図??では, 一番上の一番右のマスに 1 が入ることをレッツミーで見つけるには,4 つの 1 を見る必要がある. この 2 つのうち後者の方が一般的に見つけるのが難しいと考えられる. このような手筋の見えにくさを数値化して, 同じ手筋でも見えにくさによって違う点数を取るようにすれば, より精度が高い難易度点数判定ソルバーが作成できるかもしれない.

先行研究の結果では問題 24 から 84 の問題がおおよそ 70 前後という結果になった. この先行研究では,1 つの数字に着目し, 初級の手筋で入るブロックを探すといったことを 1 から 9 まで繰り返し行っていて,1 つの数字に対する探索につき 1 加算する方法を取っている. それで埋まらなければ, 残っている空白マスの数を難易度点数に加算して中級の手筋で解く, といった難易度付けをしている. 中級手筋を使うまでに埋まった数字を調べるとおおよそが 15 個前後で, その時点での残りマスが 50 個前後であったため, 先行研究の点数の付け方を踏まえるとこの結果は妥当であろう. また, 問題 16,20 の結果が 41,45 と他と違う値になった理由を調べたところ, 加算した残りの空白マスの数が他の問題より少ない, つまり比較的盤面に数字が埋まったところで中級の手筋を使い出したからであった.

5.2 上級問題について

問題 32 と 36 に着目する. 難易度点数の結果はそれぞれ 7,10 となっており, 他の難易度 8 と比べると低い. 問題 32 の解法についてさらに考察を行うと, 問題を解く過程で中級手筋の中でも難しい手筋であるマスミを使っていることが分かった.. これより, 使っている上級手筋の種類や回数は少なくとも, 途中で難しい中級手筋を使えば上級手筋の種類や回数が同等の問題よりも難しくランク付けされている可能性がある. 今回は難易度が格段に違うと行った理由から, 上級問題での解法分析では中級以下の手筋に難易度点数がないものとして点数を計算している. しかし, 上級手筋だけでなく中級手筋がどのように使われたかも加味すればより精度の高い難易度決定ができるかもしれない.

また, 先行研究の結果では同じ難易度でも比較的点数が低い問題と高い問題に分かれている. これらの違いを調べてみると, 例えば, 難易度 7 の問題である問題 4 から 20 のうち,12 番だけが他よりも高い値となっている. これは,12 番以外の難易度 7 の問題は先に探索される 2 マスの表予約と裏予約のみ使って解ける問題であるのに対し,12 番の問題はその後に探索されるいずれにしても理論を使わなければ解けない問題となっているからである. つまり,12 番以外の問題は 1 種類の上級手筋の探索で終わるが,12 番は 2 種類の上級手筋の探索をしなければならないので難易度点数が高くなっている. このように, 同じ難易度でも使う手筋によって盤面を探索する回数に変化がある. 先行研究によると 1 つの手筋に対する 1 回の探索で少なくとも 54 難易度点数が上がるので, これにより難易度点数にばらつきが現れると考えられる.

また, 先行研究では公式難易度 7 と 8+で, 難易度点数の差が変わらない問題がある. これは, 難易度 7 の問題ではブロック内の 2 マスの表予約を使う問題があり, 公式難易度 8+の問題では列内の 2 マスの表予約を使う問題がある. 本の難易度設定ではブロックよりも列にある予約の方が難しいとされているので後者の方が難しいとされている. しかし, 先行研究ではブロックの 2 マス表予約と列の 2 マス予約を同じ難易度と見なし 1 回の探索で同時に探索を行っているので, 難易度点数の差が変わっていないと考えられる.

また, 上級手筋でも手筋の見えにくさというものがある. 例えば, 次のようなブロックでの 3 マスにおける表予約の成立時に顕著に出る. 例えば図 18 と図 19 はどちらもブロック内の 3 マスの表予約が成立している場合である. ここで, 図 18 では表予約を成立させている数字が上から 1 行目と左から 1 列目に固まっているので, この 2 つの列に注目すれば表予約が存在することに気づきやすい. しかし, 図 19 では注目するべき列が上から 1 行目と 2 行目, 左から 1 行目と 2 行目と先ほどの例よりも多い. さらに表予約が成立しているブロックに対して 2 行が効いている数字（ここでは 1）,2 列が効いている数字（ここでは 2）,1 行と 1 列が効いている数字（ここでは 3）がある. どの数字もブロックに対しての数字の効き方が異なるので, この 3 つの数字が表予約を成立させていることに気づくのは前述の 2 列しか見なくてよい表予約よりも一般的に難しいとされている.

よって, 先述したブロックでの 3 マスにおける表予約の

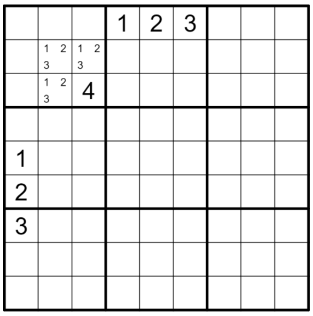

図 18　ブロック 3 マス表予約の簡単な例

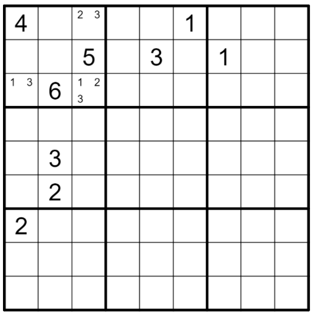

図 19　ブロック 3 マス表予約の難しい例

ように, 手筋が成立しているマスの位置や, 手筋成立に関わっている数字がある行や列を調べて, 手筋に付ける難易度点数を変えるといった改善も考えられる.

6.　おわりに

今回の入力問題では, 相関係数の結果や考察から先行研究よりも本研究の方が公式の難易度に近い難易度の目安を出せることが分かった. しかし, 本研究では入力問題が掲載されている本を出版している会社が, どの手筋をどのような難易度で扱っているかを参考にして製作した. よって, 他の出版社の本では今回とはまた違った結果になると考えられる. 例えば, 先行研究のように探索の回数の方を重点的に見るような指標で公式難易度が付けられている本であれば, 先行研究の方が適切であろう. しかし, 本研究の難易度設定とは違えど, それぞれの手筋に対して初級手筋, 中級手筋など大まかな難易度付けが行われており, それに基づいて公式難易度が設定されているならば, 本研究の手法が有効である可能性は高い. なぜなら, その難易度設定を元に各手筋の点数を決め, 本研究の手法のように解法に点数を付けることで, それを難易度の目安とできるからである. また, 先行研究は決して本研究より悪い点ばかりではない. 数字を探索する時の手間という, 人間が解く時に難しいと感じる部分を点数で表している研究である. 今回作成した最も簡単な解法を見つける手法と, 先行研究の探索の手間暇を数値化する手法, これらの良いバランスを取っている手法が今後出てくれれば, 数独の難易度決定はより安定し, より多くの人が納得できる難易度を付けやすくなるだろう.