信州大学工学部

学士論文

# 数独の消失通信路に対する<br>復号誤り特性の測定について

指導教員　西新 幹彦 准教授

学科　電気電子工学科
学籍番号　07T2074D
氏名　比田井 亮

2012 年 3 月 30 日

# 目次

# 1　はじめに

パズルの一つとして数独[*1]というものがある．数独は，かつてより親しまれてきたペンシルパズルの一種であり，空欄のマスに制約を満たす数字を書き込んでいくパズルである．このことから数独はもともとナンバープレースと呼ばれていた．ナンバープレースは 1970 年代から存在していたが，2005 年にイギリスで「Sudoku」の名で大流行したことをきっかけとして昨今では世界中で Sudoku の名で親しまれており，2006 年にはイタリアのルッカで第 1 回世界大会が開催された [1]．また，数独には一つのスタンダードなルールが存在するが，マスの数や数字の制約を変更することによってさまざまなバリエーションをいくらでも作ることができる．

数独はパズルとしてだけでなく，学問的な取り扱いも盛んに行われている．基本的な点としては，数独を解く問題は問題の大きさに関して NP 完全であることが知られている [2]．解の数に関する研究としては，数独の全ての解の数え上げ [3] とその番号付け [4] や，本質的に異なる解の数え上げ [5] とその番号付け [6]，解の数を確率的に推定する研究 [7] などがある．他に，物性物理学からのアプローチとして，一般化された数独の解の総数の平均場模型による評価 [8] や，数学からのアプローチとして，グレブナ基底を用いて問題を方程式として表現することによって代数的に数独を解く研究 [9] が行われている．

本研究では，パズルとして数独を解く過程を，通信における受信語から符号語を復号する過程とみなし，消失通信路に対する数独の復号誤りを調べることを目標とする．本論文では，数独の復号誤り特性を調べる方法について議論する．

# 2　数独のルール

本研究ではスタンダードなルールの数独を取り扱う．スタンダードな数独では，図 1 にあるような 9 × 9 の 81 マスを用いる．81 個のマスは 9 個の 3 × 3 ブロックに分けられる．マスにはあらかじめ何ヶ所か数字が記入されており，プレーヤーは以下の制約を満たすようにすべての空欄に 1〜9 の数字を入れていく．

- 一つのマスには 1 から 9 までのどれか一つの数字が入る
- どの行にも 1 から 9 までの数字が 1 回ずつ現れる
- どの列にも 1 から 9 までの数字が 1 回ずつ現れる
- 3 × 3 のブロックの中も 1 から 9 までの数字が 1 回ずつ現れる

[*1] 「数独」は株式会社ニコリの登録商標である．

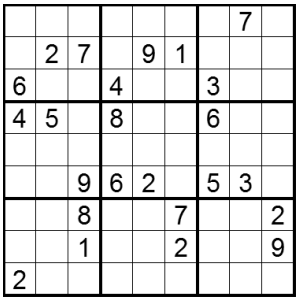

図 1　数独の例

すべてのマスに制約を満たした数字が入った状態を**解**という．あらかじめ記入されている数字はヒントとも呼ばれる．言うまでもなくヒントによって問題の難易度が決定される．ヒントの数が少なければ問題は難しくなる傾向にあるが，しかしその傾向は厳密ではない．ヒントによっては複数の解が存在する場合があるが，そのような問題は「不良問題」と言われる．通常の数独の問題は解が唯一つ存在するようにヒントが用意されており，プレーヤーもそれを暗黙の仮定として解を推定する．

ところで，数独の難易度は本質的には主観的なものである．しかし，客観的な尺度が無いわけではなく，マスに数字を入れるいくつかの定石が知られており，数字を入れられるマスを見つけ出すのにかかる手間は，定石によって明らかな違いがある．ちなみに，定石を使わなくても秩序立てて試行錯誤を繰り返せば必ず解にたどり着くことができるし，複数の解がある場合もすべての解を列挙することができる．

問題の難易度とは別に，愛好家は問題の面白さも重要視する．例えば，解にたどり着くためにどのような定石をどのような順番で適用するのか，ということが問題の面白さに影響する．「面白い問題」を自動生成するための方法がアミューズメントの分野では研究されている [10].

## 3　数独による通信システム

本研究では，消失通信路を数独によって符号化したときの性能評価を目標とする．想定するシステムは次の通りである．数独の解を符号語として用いる．符号語は数独のすべての解の中から等確率に選ばれる．一つの符号語に対し，定常独立な消失通信路が 81 回使われて受信語が復号器に届く．つまり，1 マスに対して 1 回通信路がつかわれ，消失シンボルは空白のマスとして受信される．受信語に対し，消失しなかった数字をヒントとして解が求められる．解が一意であれば誤りなく復号されたことを意味し，複数の解が存在すればそれらはそれぞれ等確率で正しい．

このシステムの基本的な性質を確認する．数独の解の総数 $N_0$ は

$$N_0 = 6670903752021072936960 \approx 6.67 \times 10^{21}$$

個であることが知られている [3] ので，符号語数は $N_0$ である．また，符号化率は $\frac{1}{81}\log_9 N_0 \approx 0.282$ である．消失確率 $\epsilon$ の消失通信路の通信路容量 $C$ が $C = 1 - \epsilon$ であることから，数独と同じ符号化率の符号を用いたときに，任意に小さい復号誤りを達成できる消失確率の上限（シャノン限界）は $\epsilon \approx 0.718$ である (図 2)．本研究の興味のひとつは，数独の復号誤り特性をシャノン限界と比較することである．

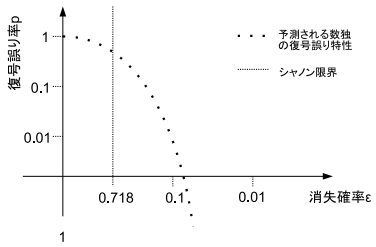


図 2　復号誤り特性

## 4　復号誤り特性の測定

一般に，受信語に対して，消失シンボルの数と復号誤りの間には明確な関係はない．しかし，だからといって $N_0$ 個の解に対してあらゆる消失パターンを試すには膨大な時間がかかる．本研究では符号語を等確率で発生させることによってモンテカルロ法で復号誤り特性を測定する．符号語を効率よく等確率で発生させることは簡単ではない．例えば，81 個のマスにランダムに数字を入れていき，解としての制約を満たさないものをあとから排除する愚直な方法では，解を選ぶ確率は $3.39 \times 10^{-56}$ であり，著しく効率が悪い．始めから制約を満たすものを選択することが重要である．

従来研究を組み合わせてこれを実現するには，1 から $N_0$ までの番号をランダムに発生させ，解の番号付け [4] にしたがって解を選び出せばよい．実はこの手続きを洗練させれば，「番号」の概念を経由することなく，解を効率よく等確率で発生させることができる．これが本論文で提案するアルゴリズムである．またそのような理由から，提案アルゴリズムは解の総数を求めるアルゴリズムを内包している．つまり，誤り特性に関する部分を取り除くと解の総数を求めるアルゴリズムとなる．

なお，復号誤り特性は数独の問題としての難易度や解法のアルゴリズムとは関係がない．しかし，測定を効率よく行うために，効率のよい数独の解法が必要である．特に，試行錯誤は著

しく効率が悪いので，知られている定石を用いることが必要となる．定石の詳細は本研究の本質ではないので，本論文では数独の解法アルゴリズムについては議論しない．

## 5　ランダムな解の生成

解を効率よく発生させるためには，解全体を階層的に適切な部分集合に分解することが重要である．部分集合のサイズを求めることによって，各部分集合が選ばれる確率が定まる．言い換えれば，解を葉とする木構造を導入することで，根から葉に至る経路をランダムに選べるようにするのが，提案法の基本的な考え方である．

説明のため，各マスには図 3 のように記号を付ける．$(s_{i1}, s_{i2}, \ldots, s_{i9})$ を第 $i$ 行と呼び，$(s_{1j}, s_{2j}, \ldots, s_{9j})$ を第 $j$ 列と呼ぶ．また，9 個の $3 \times 3$ ブロックにも図 4 のように記号を付ける．

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| s11 | s12 | s13 | s14 | s15 | s16 | s17 | s18 | s19 |
| s21 | s22 | s23 | s24 | s25 | s26 | s27 | s28 | s29 |
| s31 | s32 | s33 | s34 | s35 | s36 | s37 | s38 | s39 |
| s41 | s42 | s43 | s44 | s45 | s46 | s47 | s48 | s49 |
| s51 | s52 | s53 | s54 | s55 | s56 | s57 | s58 | s59 |
| s61 | s62 | s63 | s64 | s65 | s66 | s67 | s68 | s69 |
| s71 | s72 | s73 | s74 | s75 | s76 | s77 | s78 | s79 |
| s81 | s82 | s83 | s84 | s85 | s86 | s87 | s88 | s89 |
| s91 | s92 | s93 | s94 | s95 | s96 | s97 | s98 | s99 |

図 3　マスの番号付け

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

図 4　ブロックの番号付け

## 5.1 解の絞り込み

数独の解全体を $S_0$ とおく．$S_0$ に属する解のうち，ブロック 1 が図 5 のようになる解を $S_1$ とおく．すると，$S_1$ に属する適当な解に対して，数字を置換することによって $S_0$ に属する任意の解を作ることができる．すなわち，$S_0$ を 9! 個の同サイズの部分集合に分解することができる．また数字の置換によって復号誤りは変化しないことに注意すれば，$S_1$ に属する解を等確率に発生させればよいことが分かる．

次に，$S_1$ に属する解に対して，$s_{14} < s_{15} < s_{16}$ となっている解全体を $S_2$ とおく．$S_2$ に属する解の第 4, 5, 6 列を入れ替えることによって $S_1$ に属する任意の解を作れることに注意しよう．したがって $S_1$ は $S_2$ とサイズの等しい 6 個の部分集合に分割できる．さらに $S_2$ の解のうち，第 7, 8, 9 列についても同様に $s_{17} < s_{18} < s_{19}$ となっている解全体を $S_3$ とおくと，$S_2$ は $S_3$ とサイズの等しい 6 個の部分集合に分割できることがわかる．ここで，$S_3$ の解を $s_{14} < s_{17}$ を満たすものとそうでないものが同数あることに注意して，満たすものを $S_4$ とおく．解の列を入れ替えても，定常独立消失通信路に対する誤り特性は変わらないことから，$S_4$ に属する解を等確率に発生させればよいことが分かる．

列の入れ替えと同様に，行の入れ替えを考えると，$S_4$ に属する解のうち，$s_{41} < s_{51} < s_{61}$, $s_{71} < s_{81} < s_{91}$, $s_{41} < s_{71}$ を満たす解全体を $S_5$ とおいて，$S_5$ に属する解を等確率に発生させればよいことが分かる．ちなみにここまでの議論で，$N_0 = |S_0| = 9! \times 72^2 \times |S_5|$ が成り立っている．

## 5.2 解の分類

ここまでで発生させる解を $S_5$ にまで絞り込んできた．これから $S_5$ を部分集合に細かく分解するが，各部分集合の誤り特性は一般には等しくない．しかし，以降のように注意深く調べれば，いくつかの部分集合はサイズと誤り特性が等しいことがわかる．これから行う $S_5$ の分解とは，ブロック 1, 2, 3, 4, 7 の値が等しいものによる同値類分解である．どのような同値類に分かれるのか，構造と数を確認する．

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

図 5　固定された第 1 ブロック

$S_5$ に属する解のうち，$(s_{14}, s_{15}, s_{16}, s_{17}, s_{18}, s_{19})$ が取りうる値は

$$(4,5,6,7,8,9) \tag{1}$$
$$(4,5,7,6,8,9) \tag{2}$$
$$(4,5,8,6,7,9) \tag{3}$$
$$(4,5,9,6,7,8) \tag{4}$$
$$(4,6,7,5,8,9) \tag{5}$$
$$(4,6,8,5,7,9) \tag{6}$$
$$(4,6,9,5,7,8) \tag{7}$$
$$(4,7,8,5,6,9) \tag{8}$$
$$(4,7,9,5,6,8) \tag{9}$$
$$(4,8,9,5,6,7) \tag{10}$$

である．すなわち 10 種類の値を取りうるが，最初のひとつとそれ以外では性質が異なる．

まず，(1) の場合を考える．このとき，$(s_{24}, s_{25}, s_{26})$ は $\{7,8,9\}$ の順列である．これを $\{s_{24}, s_{25}, s_{26}\} = \{7,8,9\}$ と書く．同様に考えて結果をまとめると

$$\{s_{24}, s_{25}, s_{26}\} = \{7,8,9\}$$
$$\{s_{34}, s_{35}, s_{36}\} = \{1,2,3\}$$
$$\{s_{27}, s_{28}, s_{29}\} = \{1,2,3\}$$
$$\{s_{37}, s_{38}, s_{39}\} = \{7,8,9\}$$

となる (図 6)．これ以外の値は取ることができない．したがって $S_5$ の同値類のうち $(s_{14}, s_{15}, s_{16}, s_{17}, s_{18}, s_{19}) = (4,5,6,7,8,9)$ となるものの数は $3! \times 3! \times 3! \times 3! = 1296$ 個である．

次に (2) の場合を考える．このとき他のマスの取りうる値は

$$\{s_{24}, s_{25}, s_{26}\} = \{8,9,a\}$$
$$\{s_{34}, s_{35}, s_{36}\} = \{6,b,c\}$$
$$\{s_{27}, s_{28}, s_{29}\} = \{7,b,c\}$$
$$\{s_{37}, s_{38}, s_{39}\} = \{4,5,a\}$$

となる（図 7)．ただし $(a,b,c)$ は $(1,2,3)$, $(2,3,1)$, $(3,1,2)$ のいずれかである．したがって

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 4 | 5 | 6 | {7,8,9} | | | {1,2,3} | | |
| 7 | 8 | 9 | {1,2,3} | | | {4,5,6} | | |

図 6　上 1 行を固定したときの組み合わせ 1

| 1 | 2 | 3 | 4 | 5 | 7 | 6 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|
| 4 | 5 | 6 | {8,9,a} | | | {7,b,c} | | |
| 7 | 8 | 9 | {6,b,c} | | | {4,5,a} | | |

図 7　上 1 行を固定したときの組み合わせ 2

$S_5$ の同値類のうち
$(s_{14}, s_{15}, s_{16}, s_{17}, s_{18}, s_{19}) = (4, 5, 7, 6, 8, 9)$ となるものの数は $3 \times 3! \times 3! \times 3! \times 3! = 3888$ 個である．

(3) から (10) の場合も (2) の場合と同様で，それぞれの同値類は 3888 個存在する．したがって，ブロック 1, 2, 3 による $S_5$ の同値類は 36288 個存在する．さらにブロック 4, 7 も加えて同値類分解を細かくすると，ブロック 1, 2, 3, 4, 7 の値が等しいものによる $S_5$ の同値類は $36288 \times 36288 = 13146706944$ 個存在する．これらの同値類の中には，数と誤り特性が等しいものが存在する．それらを計算機によって予め探索することによって，$S_5$ のごく一部の解だけから復号誤りを測定することができる．

## 5.3　等価な同値類の探索

以降，数と誤り特性が等しい解の集合を**等価**であると言うことにする．

ブロック 1, 2, 3, 4, 7 の値が等しいものによる $S_5$ の同値類分解を $\mathcal{S}_6$ とおく．同値類分解 $\mathcal{S}_6$ は次のような構造を持っている．ブロック 1, 2, 3 の値による $S_5$ の同値類分解を $\mathcal{S}_7$ とおく．よって $\mathcal{S}_7$ に属する同値類は互いに素で $\bigcup_{S \in \mathcal{S}_7} S = S_5$ である．一方，ブロック 1, 4, 7 の値による $S_5$ の同値類分解を $\mathcal{S}_8$ とおく．すると，$\mathcal{S}_6$ に属する任意の同値類 $T$ は適当な $T_1 \in \mathcal{S}_7$, $T_2 \in \mathcal{S}_8$ を用いて $T = T_1 \cap T_2$ と書ける．実際，先ほどの $36288 \times 36288 = 13146706944$ の計算はこの事実に基づいている．

いま，改めて $T_1, T_2 \in \mathcal{S}_7$ を考え，$T_1$ と $T_2$ は等価だとする．すると，任意の $T \in \mathcal{S}_8$ に対して $T_1 \cap T$ と $T_2 \cap T$ は等価である．したがって，$\mathcal{S}_6$ の代わりに $\mathcal{S}_7$ の中で等価な組を見つければよい．さらに，$\mathcal{S}_8$ の構造は $\mathcal{S}_7$ の構造と同型なので，両者の間の適切な全単射を見つければ，$\mathcal{S}_8$ の中で等価な組を見つける必要はない．

以上のようにして，$\mathcal{S}_6$ の中の等価でない部分集合についてのみ，数と誤り特性を調べればよいことが分かる．これが本論文で提案する誤り特性の測定方法である．

提案法を実現するためには $\mathcal{S}_7$ の中で等価な集合を見つける必要がある．等価な集合を（事前に）すべて見つけ出す方法は知られていないが，以下の方法がある [4][6].

### 5.3.1 行や列の入れ替えによる変換

どんな解に列や行の入れ替えをしてもその結果はやはりひとつの解であり，かつ復号誤りももとのものと等しい．この性質を用いてひとつの同値類 $S \in \mathcal{S}_7$ から別の等価な同値類に変換する手順がある．

**行交換** 集合 $S \in \mathcal{S}_7$ を考える．各 $t \in S$ に対して，第 1, 2, 3 行に置換を施した解を $t'$ とおく（図 8）．すると，$t'$ は $S_1$ の要素ではなくなる．しかし，マスの数字に関して適切な置換を施せば $S_1$ の要素となる．これを $t''$ とおく．すると，$t''$ は $S_5$ に属していないかも知れないが，第 4, 5, 6 列の入れ替えと第 7, 8, 9 列の入れ替え，そして必要ならばブロック 2, 5, 8 とブロック 3, 6, 9 をそれぞれ交換することによって $s$ を作り，$s_{14} < s_{15} < s_{16}$, $s_{17} < s_{18} < s_{19}$, $s_{14} < s_{17}$ とすることができる．行に関しても同様に入れ替えたものを $s'$ とすれば $s' \in S_5$ となる．以降，解 $t'$ から $s' \in S_5$ を導くこの操作を**標準化**と呼ぶことにする．さらに，$s'$ は $S$ と行の置換のみに依存して定まるある $S' \in \mathcal{S}_7$ に対して $s' \in S'$ となる．このとき $S$ と $S'$ は等価である．行変換を用いた探索の結果，$\mathcal{S}_7$ に属する 36288 個の同値類のうち，互いに等価でない同値類は高々 6467 個であることが分かった．

**ブロック交換** 集合 $S \in \mathcal{S}_7$ を考える．各 $t \in S$ に対して，ブロック 1, 4, 7 とブロック 2, 5, 8 とブロック 3, 6, 9 を単位として入れ替えた解を $t'$ とおく（図 9）．$t'$ を標準化したものを $s'$ とおくと，$s'$ は $S$ とブロックの入れ替えのみに依存して定まるある $S' \in \mathcal{S}_7$ に対して $s' \in S'$ となる．このとき $S$ と $S'$ は等価である．ブロック交換を用いた探索の結果，互いに等価でない同値類は高々 16700 個であることが分かった．

行変換とブロック変換の両方を用いた探索の結果，互いに等価でない同値類は高々 4751 個であることが分かった．


図 8　行交換の例


図 9　ブロック交換の例

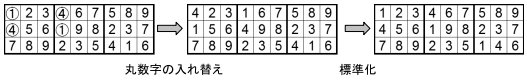


図 10　2 × 2 交換の例

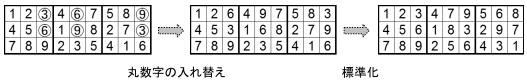


図 11　3 × 2 交換の例

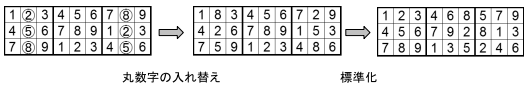


図 12　2 × 3 交換の例

### 5.3.2　マスの入れ替えによる変換

解によっては，特定のマスの数字を入れ替えてもやはりひとつの解になる場合がある．この場合も復号誤りはもとのものと等しい．数字を入れ替えられる場所を見つけられればその変換により別の等価な同値類を得ることができる．

**2 × 2 交換**　集合 $S \in \mathcal{S}_7$ を考える．ある $i_1, i_2, j_1, j_2$ $(1 \le i_1, i_2 \le 3, 1 \le j_1, j_2 \le 9)$ が存在して，任意の $s \in S$ に対して $s_{i_1 j_1} = s_{i_2 j_2}$, $s_{i_1 j_2} = s_{i_2 j_1}$ だとする．このとき，$s_{i_1 j_1}$ と $s_{i_1 j_2}$，$s_{i_2 j_1}$ と $s_{i_2 j_2}$ をそれぞれ入れ替えたものもひとつの解となる（図 10）．この解を標準化したものを $s'$ とおくと，ある $S' \in \mathcal{S}_7$ に対して，$t$ によらず，$s' \in S'$ が成り立つ．このとき $S$ と $S'$ は等価である．2 × 2 交換を用いた探索の結果，互いに等価でない同値類は高々 14615 個であることがわかった．

***k* × 2 交換**　2 × 2 交換を一般化し，$k$ 個の列と 2 個の行を選び，選ばれたマスの各行が同じ数字からなっている場合も，標準化によって等価な同値類を見つけることができる．これを $k$ × 2 交換と呼ぶ (図 11)．さらに 2 個の列と 3 個の行を選び，2 × 3 交換も考えることができる（図 12)．

3 × 2 交換を用いた探索の結果，互いに等価でない同値類は高々 22338 個であることがわかった．4 × 2 交換を用いた探索の結果，互いに等価でない同値類は高々 33616 個であることがわかった．2 × 3 交換を用いた探索の結果，互いに等価でない同値類は高々 30822 個であることがわかった．

***k* × 2 × 3 交換**　ブロック 1, 2, 3 から列をそれぞれ一つずつ選ぶ．各ブロックの選ばれた列をなす 3 つのマスから 2 つずつマスを選ぶ．ただし，ブロック 1, 2, 3 全体として，各行から 2

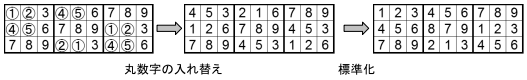


図 13　2 × 2 × 3 交換の例

つのマスが選ばれるようにする．各行の選ばれたマスに入っている数字の組み合わせが 3 つとも同じであれば，各ブロック内で選ばれたマスの数字をそれぞれ同時に入れ替えてもそれはひとつの解となる．この解を標準化することによって等価な同値類を見つけることができる．これを 1 × 2 × 3 交換と呼ぶ．一般化によって $k \times 2 \times 3$ 交換を考えることもできる（図 13）．ところが実際のところ，1 × 2 × 3 交換は異なる同値類を作らないので意味がない．3 × 2 × 3 交換は [4] によって提案されている．

2 × 2 × 3 交換を用いた探索の結果，互いに等価でない同値類は高々 34420 個であることがわかった．3 × 2 × 3 交換を用いた探索の結果，互いに等価でない同値類は高々 35251 個であることがわかった．

$\mathcal{S}_7$ に属する同値類のうち，以上の方法を用いると，調べるべき同値類は 1375 個となる．したがって，数と誤り特性を調べるべき $\mathcal{S}_6$ に属する同値類は，転置による重複も考慮に入れれば，高々およそ $1375^2 \div 2 = 877812$ 個で済むことになる．

試みに図 14 で表される同値類に属する解の総数を計算機 (3.4GHz CPU) で数えたところ，50 秒ほどで 6280 個であることが分かった．

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

図 14　$\mathcal{S}_6$ に属する同値類の例

## 6　今後について

前章で提案した方法によってランダムに解を生成して消失通信路を通して通信し，数独の解法プログラムを用いて数独を解くことで復号し，発生させた解と復号した解が一致するかどうか確認することで通信が正確に行われているか評価する．消失通信路の消失確率 $\epsilon$ を変化さ

せ，どこまで通信環境が悪化しても通信が正確に行われるか調べる．

本論文では，上列 3 ブロックに対して復号誤り特性の等しい同値類をまとめ，測定する際に課題となる膨大な数の解に対してより効率的に解を発生させる準備を行った．

今後行うべきこととしては，左列 3 ブロックに対して本論文で扱ったアルゴリズムを行い，さらに探索範囲を狭め，その後残りのマスを埋めることで各標準形がどの程度グループ化されているか調べる必要がある．グループの大きさがわかればどの程度の確率で解を発生させればよいかがわかるので，その確率に従ってランダムに解を発生させるプログラムを作成し，提案法を用いて実際に復号誤り特性を測定することが今後残された課題である．

また，本論文では触れていないが，測定の効率を上げるためには数独解法アルゴリズムの高速化も検討する必要がある．

# 7 まとめ

数独は，かつてより親しまれてきたペンシルパズルの一種であり，空欄のマスに制約を満たす数字を書き込んでいくパズルである．一般的に普及している $3 \times 3$ の数独に注目し，パズルとして数独を解く過程を，通信における受信語から符号語を復号する過程とみなし，消失通信路に対する数独の復号誤り特性の測定に関して研究を行った．本研究では，数独の通信能力の理論値を求めるとともに測定するに当たって問題となる解の総数に関して，従来研究にある Jarvis らのによる解の数え上げで用いた提案法を多く利用し，誤り特性の等しい同値類の探索を行い，消失通信路で測定すべき解の絞込みを行った．誤り特性を変化させずに行う交換方法から，局所的に交換を行う方法まで行った結果，調べるべき同値類を 1375 個に絞り込んだ．今後の課題としては，更なる探索範囲の縮小と，実際に通信を行うことで数独の誤り訂正能力がどのようなものかを明らかにすることである．

# 謝辞

本研究に当たり，細かく指導してくださった指導教員の西新幹彦准教授に感謝の意を表する．

# 参考文献

[1] World puzzle Federation, `http://www.worldpuzzle.org/`, 2011 年 10 月閲覧.

[2] T. Yato, T. Seta, “Complexity and completeness of finding another solution and its application to puzzles,” IEICE trans. fundamentals, pp.1052-1060, 2003.

[3] Bertram Felgenhauer, Frazer Jarvis, “Enumerating possible Sudoku grids,” `http://www.afjarvis.staff.shef.ac.uk/sudoku/sudoku.pdf`, 2011 年 10 月閲覧.

[4] 戸神星也,「数独の解生成と解に対する番号付け」, 東京工業大学理学部情報科学科卒業論文, 2006.

[5] Ed Russell, Frazer Jarvis, “Sudoku enumeration,” http://www.afjarvis.staff.shef.ac.uk/sudoku/sudgroup.html, 2011 年 10 月閲覧.

[6] 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」, 情報処理学会, 第 71 回全国大会講演論文集, vol.4, pp. 741–742, 2009.

[7] 貞廣泰造,「数独解盤面の近似数え上げ」, 数学セミナー, vol.51, no.3, pp. 26–29, 2012.

[8] 田中利幸, 福島孝治,「数独に対する平均場模型」, 日本物理学会講演概要集, pp. 264, 2010.

[9] 井上秀太郎, 佐藤洋介, 鈴木晃, 鍋島克輔,「ブーリアングレブナ基底を使った数独の解法」, 数理解析研究所講究録第 1666 巻, pp. 1–5, 2009.

[10] 前田一貴, 奥乃博,「数独の問題作成支援システムの設計と開発」, 情報処理学会, 第 70 回全国大会講演論文集, vol.4, pp.799–800, 2008.

# 付録 A　ソースコード

以下のプログラムは 5.3 節において，各交換で同値類を列挙したのち，各標準形における等価な同値類の仕分け，集計に用いたものである．

## A.1　同値類を仕分けするプログラム

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>

char *
makename(char *name, int id)
{
    int a5, a4, a3, a2, a1, a0;

    a0 = id % 6, id /= 6;
    a1 = id % 6, id /= 6;
    a2 = id % 6, id /= 6;
    a3 = id % 6, id /= 6;
    a4 = id % 3, id /= 3;
    a5 = id;
    sprintf(name, "A%d%d%d%d%d%d.dat", a5, a4, a3, a2, a1, a0);

    return(name);
}

int marged;

void
marge(int id)
{
    unsigned short hyojun[40000];
    int n, i;
    char filename[16];
    FILE  *fp;

    makename(filename, id);
    fp = fopen(filename, "rb");
    if (fp == NULL){
        printf("can't open %s\n", filename);
        printf("error in %d\n", __LINE__);
        exit(1);
    }
    if (fread(hyojun, 2, 1, fp) == 1){
        if (hyojun[0] == id){
            fclose(fp);
            marged++;
            return;
        }
    } else {
        fclose(fp);
        return;
    }
    for (n = 1; fread(hyojun + n, 2, 1, fp) == 1; n++){
        ;
    }
    fclose(fp);
    for (i = 0; i < n; i++){
        makename(filename, hyojun[i]);
        fp = fopen(filename, "rb");
        if (fp == NULL){
            printf("can't open %s\n", filename);
```

```
            printf("error in %d\n", __LINE__);
            exit(2);
        }
        if (fread(hyojun + n, 2, 1, fp) == 1){
            if (hyojun[n] == hyojun[i]){
                fclose(fp);

                continue;
            } else {
                n++;
                while (fread(hyojun + n, 2, 1, fp) == 1){
                    n++;
                    if (n > 39000){
                        printf("error in %d\n", __LINE__);
                        exit(1);
                    }
                }
            }
        }
        fclose(fp);
        fp = fopen(filename, "wb");
        if (fwrite(hyojun + i, 2, 1, fp) != 1){ /* 統合済み */
            printf("error in %d\n", __LINE__);
            exit(3);
        }
        fclose(fp);
    }
    makename(filename, id);
    fp = fopen(filename, "wb");
    if (fp == NULL){
        printf("can't open %s\n", filename);
        printf("error in %d\n", __LINE__);
        exit(4);
    }
    if (fwrite(hyojun, 2, n, fp) != n){
        printf("error in %d\n", __LINE__);
        exit(5);
    }
    fclose(fp);
    return;
}



int main()
{
    int id;

    marged = 0;
    for(id = 0; id < 1296; id++){
        marge(id);
    }
    for(id = 3888; id<38880; id++){
        marge(id);
    }
    printf("marged: %d\n", marged);
    return(0);
}
```