卒 業 論 文

基本的なテクニックに特化した解法による
数独の難易度評価

指導教員: 下薗 真一　准教授

九州工業大学情報工学部
知能情報工学科

2021 年度

182C1029

江口 冬和

		指導教員	下薗 真一　准教授
学生番号	182C1029	氏　　名	江口 冬和
論文題目	基本的なテクニックに特化した解法による 数独の難易度評価		

1　はじめに

数独は世界的に有名なペンシルパズルの 1 つである. 数独の解き方は難易度が高い問題ほどより多くの解き方を駆使する必要がある. 一般的には簡単な解き方から使っていき, それで解き進められない場合により難しい解き方を使うが, 人が数独を解く場合は簡単な解き方が使える場面を見落とすこともあるだろう. そのような場面で見落としがなければ, 簡単な解き方だけで問題を解くことは出来るのではないか.

本研究では簡単な解き方の繰り返しでどこまでパズルを解くことが出来るのか, また答えにたどり着くことが出来なければ, どのような工夫をすれば最後まで解くことが出来るか, という着眼点から, いくつかの数独の解法について考察し, 計算実験でアイディアの検証を行う.

2　数独の概要

9 × 9 の正方形の枠内に 1 から 9 までの数字を入れるパズルで, 縦方向の 9 マスの行, 横方向の 9 マスの列, 9 × 9 の盤面を 3 × 3 ごとに区切ったときに出来る部分のグループ, それぞれの中に同じ数字が複数入ってはいけない.

3　基本テクニック

3.1　エリア確定法

与えられた盤面の各エリアの空きマスの数に注目し, それぞれのエリアの中で空きマスが 1 つだけである場合, 他のマスで使われている数字から空きマスの数字を確定する方法である.

3.2　可能性確定法

与えられた盤面の空きマスに入る可能性のある数字の数に注目する. 空きマスが含まれるエリアを調べることで空きマスに入る可能性がある数字を特定することが出来る. この方法はエリア確定法とは異なり, ある空きマスを含むエリアが複数あることを利用する.

3.3　幅優先探索

幅優先探索はグラフの探索に用いられるごく一般的なアルゴリズムである. アルゴリズムは特定のノードから探索を開始し, そこから隣接する未訪問のノード全てを探索する. そしてこれらの探索したノードのそれぞれに対して同様のことを繰り返す. 始点から距離が近い順に探索をしていくため, 探索するノードをキューを使って管理する.

4　実験と評価

6 つの難易度の問題を 3 問ずつ用意し実験を行った. エリア確定法のみを使った場合はどの問題も 1 マスも埋めることが出来なかった. 2 つの基本テクニックを使った場合は, 入門と初級の問題はおおよそ最後まで解くことが出来た. 中級と上級の問題は途中まで埋めることが出来るものもあったが最後まで解くことは出来なかった. 難問と超難問の問題はいずれも最後まで解くことが出来なかった.

そのため, 基本テクニックでは進められない盤面では仮定をおいてパズルを進めることで行き詰まりを解消した. 使用した問題例では, 基本テクニックの繰り返しで最後まで問題を解くために見つける必要がある仮定は高々2 つであることが分かった. 仮定をおいた場合, 仮定をおく回数によって解くまでの時間が変わり, 最大で 924 秒で解くことが出来た.

5　おわりに

基本テクニックの繰り返しで簡単な問題は解くことが出来たが, 予想通り難しい問題は解くことが出来なかった. そのため, 幅優先探索を加え, 基本テクニックが使えない盤面に仮定をおいて進めるように使用した. すると探索木が爆発的に大きくなり, 仮定をおいた回数が多ければ解くまでの時間が増えることが分かった.

今後の課題は, 基本テクニックについてのみの検証だけでなく, より難しい解き方を増やしていき, どのレベルの解き方まで習得すればどの問題でも最後まで解くことが出来るかの検証や, いかにして基本テクニックが使える盤面になるように仮定をおくかの考察が挙げられる.

参考文献

[1] Algoful - アルゴリズム初心者向けの基礎と入門 - https://algoful.com/Archive/Algorithm/BFS.

[2] レッツ！ナンプレ, https://si-coding.net/ ホームページ制作・コーディング代行 IT-Links

[3] 見やすい解きやすいナンプレ, 2021 年 09 月 20 日, インテルフィン

[4] 数独の初期ヒント最小個数は 17, それ未満では解けないと数学者が結論, Gigazine, 2012 年 01 月 09 日

目 次

第1章　はじめに

数独は世界的に有名なペンシルパズルの１つである. 数独の解き方は総当たり法や消去法など様々あり, 難易度が高い問題ほどより多くの解き方を駆使する必要がある. 一般的には簡単な解き方から使っていき, それで解き進められない場合により難しい解き方を使う. ここでの簡単な解き方とは数独を解くにあたって第一選択肢として考えられる基本的な解き方である. ただ人が数独を解く場合は簡単な解き方が使える場面を見落とすこともあるだろう. そのような場面で見落としがなければ簡単な解き方だけで問題を解くことは出来るのではないか. もし簡単な方法だけでどのような難易度の問題も解くことが出来るのであれば, 数独をやったことがない人でも取り組みやすくなる. また, わざわざ難しい解き方を習得する必要がなくなる.

本研究では簡単な解き方の繰り返しでどこまでパズルを解くことが出来るのか, またもしそれで答えにたどり着くことが出来なければ, どのような工夫をすれば最後まで解くことが出来るかという着眼点から, いくつかの数独の解法について考察し, 総当たり法と組み合わせるなどの工夫でソルバーを作成し, 計算実験でアイディアの検証を行う.

第2章　数独とは

以下では数独の説明, 及び定義を行う. 以下で使用する “盤面” とはパズルの表のことである.

2.1　数独の定義

9 × 9 の正方形の枠内に 1 から 9 までの数字を入れるパズルで, 縦方向の 9 マスの行, 横方向の 9 マスの列, 9 × 9 の盤面を 3 × 3 ごとに区切ったときに出来る部分のグループ, それぞれの中に同じ数字が複数入ってはいけない. 以後, この同じ数字が入ってはいけない各行, 各列, 各グループのことをエリアと呼ぶこととする. 数独のパズルの図を図 2.1 に示す.

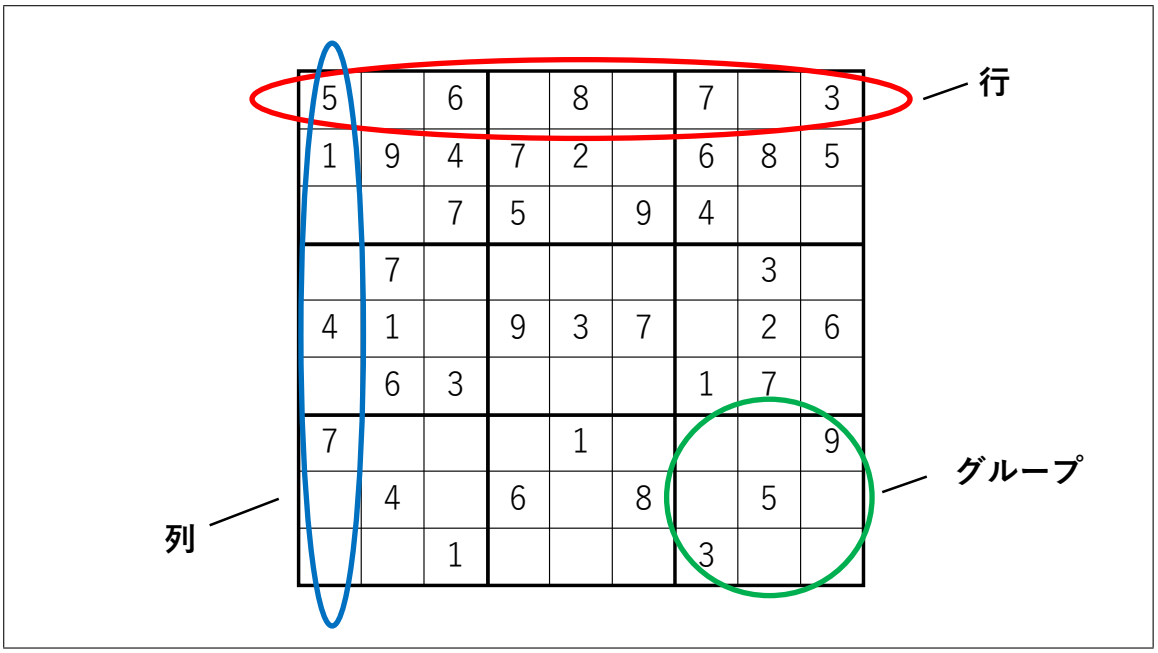

図 2.1 数独の図

数独の問題例は, 空きマスを 0 で表現することで, 以下のように整数 0, 1, ⋯ , 9 を要素とする長さ 9 の列 R の 9 個組 Q で表される.

$$Q = (R_0, R_1, ..., R_8),$$

$$R_i = (e_{i\ 0}, e_{i\ 1}, ..., e_{i\ 8}),$$

$$e_{i\ j} \in 0, 1, ..., 9.$$

各行は $0 \leq r < 9$ について, $(Q[r][0], Q[r][1], ..., Q[r][8]) = R_i$ と書ける.
各列は $0 \leq c < 9$ について, $(Q[0][c], Q[1][c], ..., Q[8][c]) = C_i$ となる.

各グループは $0 \leq g < 9$ それぞれについて, $g^{'} = 27 \cdot \frac{g}{3} + 3(g \bmod 3)$ としたとき,

$$(Q[g^{'}][g^{'}], Q[g^{'}][g^{'}+1], Q[g^{'}][g^{'}+2],$$

$$Q[g^{'}+1][g^{'}], Q[g^{'}+1][g^{'}+1], Q[g^{'}+1][g^{'}+2],$$

$$Q[g^{'}+2][g^{'}], Q[g^{'}+2][g^{'}+1], Q[g^{'}+2][g^{'}+2]) = G_i$$

となる.

以上の 27 つの 9 個組全てにおいて, 0 を除き 1 から 9 までの数字が高々1 回出現することが数独のルールとなる.

問題例の定義

整数 0, 1, ..., 9 の 9 個の列の 9 個組のならび

$$Q \in (\{0, 1, ..., 9\}^9)^9$$

で, 数独のルールを満たすもの.

解の定義

整数 1, 2, ..., 9 のいずれかの 9 個の列の 9 個組のならび

$$A \in (\{1, 2, ..., 9\}^9)^9$$

で, 任意の $0 \leq i < 9, 0 \leq j < 9$ に対して, $Q[i][j] > 0$ のとき $Q[i][j] = A[i][j]$ であり, かつ数独のルールを満たすもの.

第3章　用いた手法の説明

まず, プログラムでも用いている 2 つの簡単でよく知られた解法を説明する. 次に, これらの簡単な解法で行き詰まった場合に総当たり法を用いるために幅優先探索を説明する.

3.1　エリア確定法

与えられた盤面の各エリアの空きマスの数に注目し, それぞれのエリアの中で空きマスが 1 つだけである場合, 他のマスで使われている数字から空きマスの数字を確定する方法である. エリア内にすでに存在する数字は制約によって空きマスに入れることは出来ないため, それらを除いた残り 1 つが空きマスに入ることが決定する.

これを盤面すべてのエリアを調べ, 空きマスが 1 つだけのエリアを完成させることで盤面を解き進める. エリア確定法の使用場面を図 3.1 に示す.

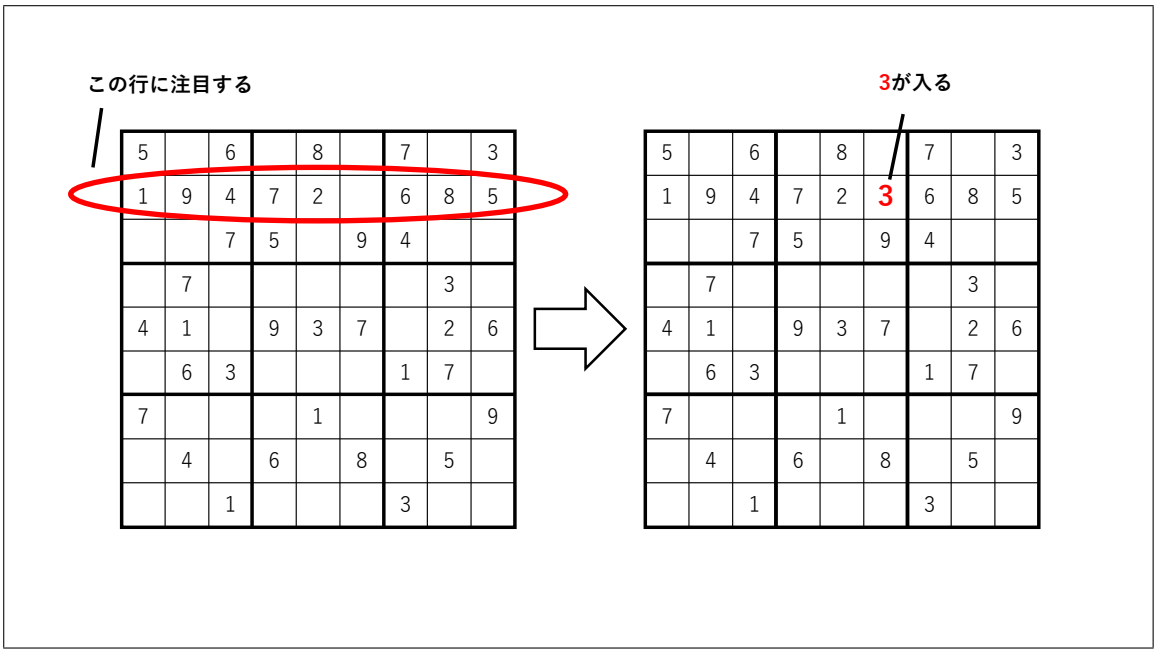

図 3.1 エリア確定法の使用場面

3.2 可能性確定法

空きマスの可能性を調べ, その可能性が 1 つだけであれば, その数字がマスに入る. この操作を全ての空きマスに対して行い盤面を解き進めていく方法. 与えられた盤面の空きマスに入る可能性のある数字の数に注目し, 空きマスが含まれるエリアを調べることで空きマスに入る可能性がある数字を特定することが出来る.

この方法はエリア確定法とは異なり, あるエリアに空きマスが複数あることを利用する. エリア内に存在する数字は制約によって空きマスに入れることが出来ないため, 1 から 9 の中でそれらを除いた数字が空きマスに入る “可能性” となる.
可能性確定法の使用場面を図 3.2 に示す.

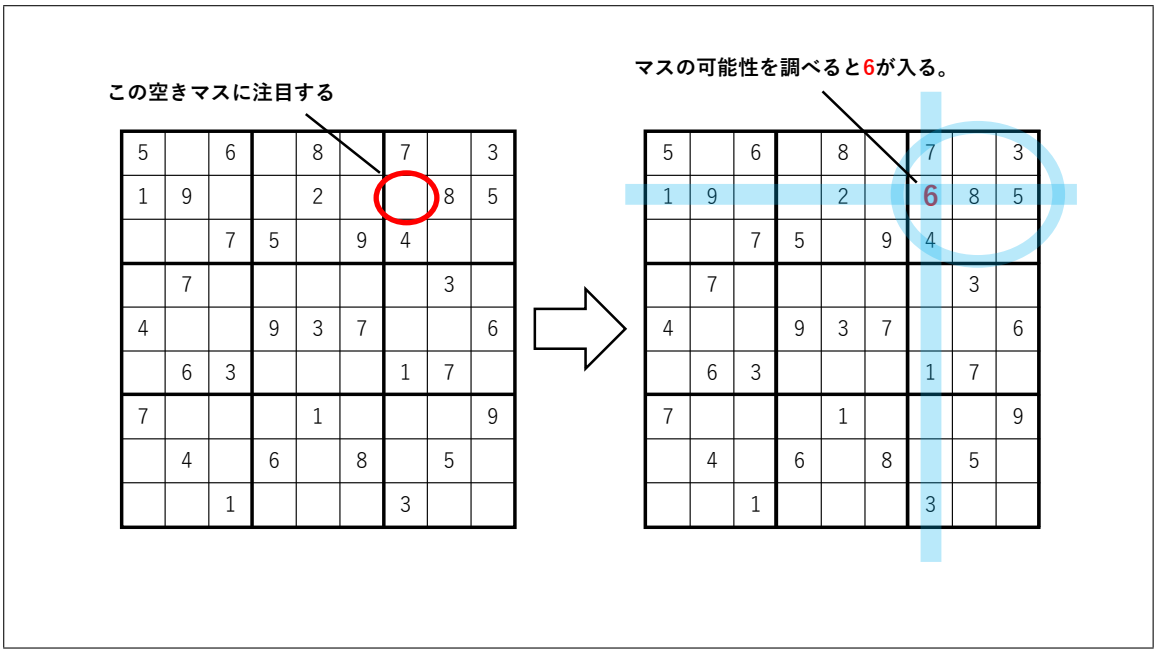

図 3.2 可能性確定法の使用場面

3.3 基本テクニックの利点と欠点

基本テクニックは数独のルールから明らかで, 誰でも簡単に理解出来るものである. つまり数独を解く手段の第一選択肢としてふさわしい. また, 他の空きマスに入るの可能性のある数字を記憶せずに使うことが出来る. つまり記憶領域の負担が少ない.

ただ利点だけではなく欠点もある. それはパズルの問題例の難易度によるが, 人が数独を解く場合に基本テクニックの繰り返しでは解にたどり着けないことが多いことである. 本研究ではコンピュータで解く場合にも同様のことが起こるかを確かめる.

基本テクニックの繰り返しで行き詰まった場合は, 幅優先探索, 数独でいう「総当たり法」を使うことで解消する. 総当たり法は空きマスに対して全ての数字をいれることで, 考えうる全てのパターンを確かめ, 正解までたどり着く方法である. 総当たり法も基本テクニックと同様に誰でも簡単に理解できる数独の解法の１つで

あるが, これだけを使って数独を解くことはコンピュータを用いても現実的ではない. そこで基本テクニックが使えなくなった場面に対して使うことで現実的に使えるような工夫をした. この工夫によって「仮定をおく」という方法が実現出来た. 「仮定をおく」とは基本テクニックの繰り返しで行き詰まった盤面で, 空きマスをに制約を満たす数字を入れる操作で, この数字は入れた時点では正解か不正解か分からないため, 盤面を進めるために仮においている数字である. 仮定をおいて進めた結果, ある空きマスに対して制約を満たす数字見つからない場合が出てくる. この場合を不正解と呼ぶ. 不正解に到達した場合, バックトラックをする必要がある.

この動作のためには, 基本テクニックのみを使って解く場合に記憶の負担が少ないという利点を失うことになるが, 解き進めるためにはやむを得ないといえる.

3.4 幅優先探索

幅優先探索はグラフの探索に用いられるごく一般的なアルゴリズムである. アルゴリズムは特定のノードから探索を開始し, そこから隣接する未訪問のノード全てを探索する. そしてこれらの探索で見つけたノードのそれぞれに対して同様のことを繰り返す. 始点から距離が近い順に探索をしていくため, 探索するノードをキューを使って管理する [1]. 幅優先探索の疑似コードは以下のようになる.

1. 始点のノードを探索待ちキューに追加する.
2. 探索待ちキューにノードがあれば取り出す. なければ探索終了.
3. 取り出したノードが目的ノードであれば探索終了.
4. 取り出したノードに隣接するノードのうち, 未探索のノードを探索待ちキューに追加する.
5. 2 の処理に戻る.

数独における幅優先探索では木に出現しうるノードすべてを走査する. 与えられた問題例を根ノードとし, 各ノードが表す盤面の空きマスを 1 つ埋めた状態をそ

の子ノードとする. 本研究では子ノードが未探索かどうかの判定は行わない. 探索では各ノードに対して, 作成可能な子ノードすべてを探索対象にすることで盤面を埋めていき, 最終的にすべてのマスに数字が入って, かつ制約を満たすノードを見つければ探索は終了となる.

“盤面を埋める” は幅優先探索の過程で空きマスに数字を入れて行くことを指す. 結果的に間違った数字を埋める操作に対しても使う. “盤面を解き進める” は正解ノードへのパスを進む数字を盤面に入れることを指す.

幅優先探索や深さ優先探索などの全探索のみを使用する方法では問題に対してすべての数字のパターンを入れることになるため, 数独では総当たり法と呼ばれている. 総当たり法のアイデアは最もシンプルだが, 計算量が非常に大きくなるため, これのみを使って解くことは現実的ではない. そこで今回は, 基本テクニックでは解き進めることが出来ない盤面を進めるためにのみ総当たり法を用いる. なお, 深さ優先探索を使わない理由は, 探索の対象の木の出来るだけ浅い場所で解にたどり着く探索をしたいからである. これは「人間が簡単に見つける」に相当する.

ただし, 幅優先探索を純粋に総当たりとして用いると空きマスに数字を入れる段階で制約を満たさないものまで入れてしまい, 無駄な探索の枝が生じるので, 制約を満たす数字のみ入れることで計算量を減らし使いやすくする工夫をした. これは仮定をおくことにあたり, 仮定をおいてパズルを進め, 不正解であればバックトラックをすることを仮定法と呼ぶ. 探索の中での不正解の盤面は, それは葉ノードとなる.

幅優先探索の様子を図 3.3 に示す.

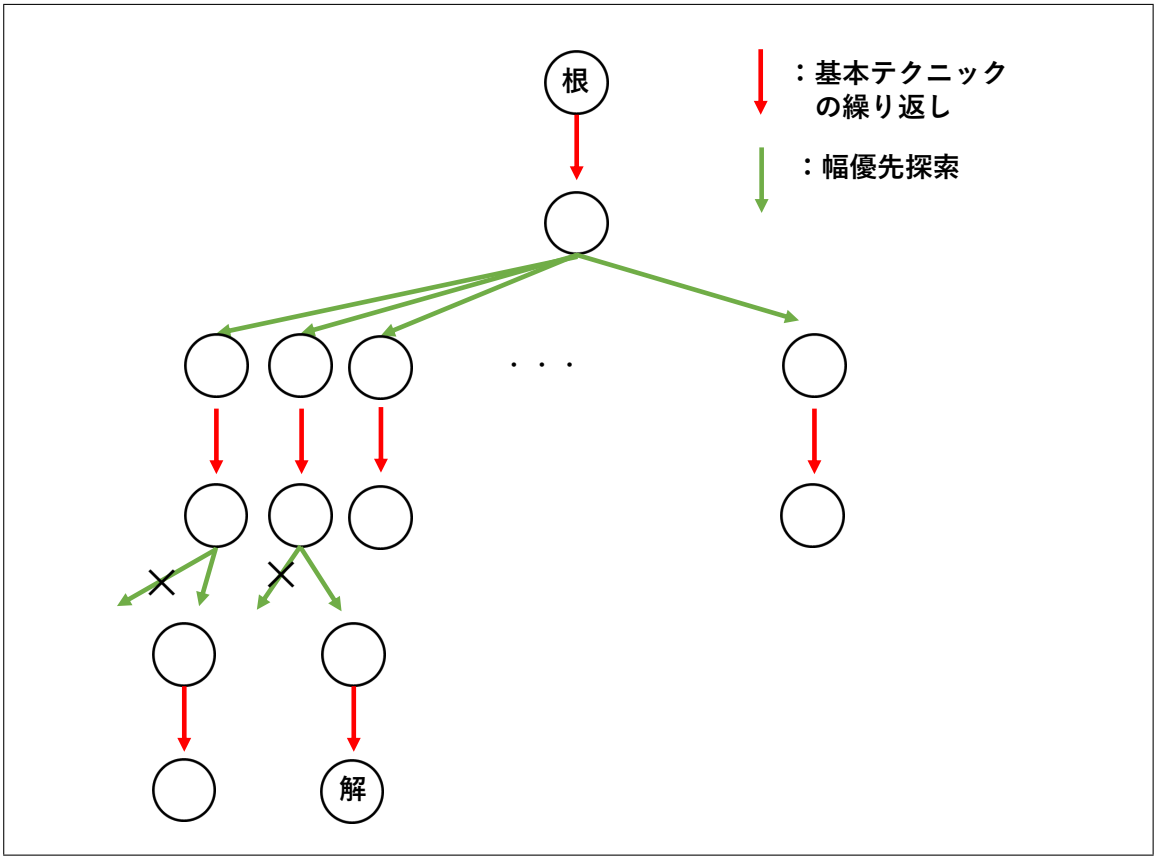

図 3.3 幅優先探索の図

第4章　作成したプログラムの概要

本研究のプログラムは Python3 で書いている.

4.1　プログラムの説明

4.1.1　数独のソルバー

3 章で挙げたエリア確定法, 可能性確定法, 幅優先探索を合わせたソルバーを数独のソルバーとし, 以下で説明する. アルゴリズムで出てくる幅優先探索を進めていくなかで注目される盤面を child, 探索されていない child を保存するリストを children とする.

数独のソルバーは以下の手順で動く. なお, technique 関数は, 詳しくは後述するが, 基本テクニックを可能な限り繰り返し適用する関数である.

1. 入力として盤面を与える.
2. 入力の盤面に対して technique 関数を使いパズルを進める. その進めた盤面を幅優先探索の根として children に追加する.
3. 以下をループをする.
   (a) children の先頭から子を 1 つ取り出し child とする.
   (b) technique 関数を使い, child を解き進める.
   (c) 進めた盤面が正解であるかを answer 関数でチェックする.
      i. 盤面が正解であれば探索を終了する.

(d) child に対して仮定法を 1 回使った盤面を生成する。このとき, 最終的に答えではなくなるようなものでもその時点で制約を満たしていれば child の子とする.

(e) 作った子を children に追加する.

(f) child は探索済となるので, children の先頭から削除する.

4.1.2 数独の制約を満たすかチェックする関数 (check 関数)

入れたい数字をある空きマスに入れたとき, 盤面が数独の制約を満たしているかをチェックする関数. 行, 列, グループ, それぞれに対してチェック関数を作成し, それをひとまとめにしたものをエリアのチェック関数とする.
また, この関数を使って制約を満たすか確かめる操作を “チェックする” と呼ぶ.
チェック関数は以下の手順で動く.

1. 入力として盤面, チェックしたい行, チェックしたい列, マスに入れたい数字を与える.
2. 入力した行, 列から調べたいマスをする.
3. そのマスに入れたい数字を入れたとき, 行, 列, グループ, それぞれについて数独の制約を満たすかどうかを真偽で返す.
   (a) すべての場合で制約を満たすとき, True を返す.
   (b) どれか 1 つでも制約を満たしていなければ, False を返す.

チェック関数のサブルーチン

行でのチェック関数

1. 入力として盤面, チェックしたい行, マスに入れたい数字を与える.
2. 調べたい行の全ての列の数字と盤面に入れたい数字を比較する.
   (a) 全ての列に同じ数字がなければ制約を満たすため, True を返す.
   (b) 同じ数字があれば制約を満たさないため, False を返す.

列でのチェック関数

1. 入力として盤面, チェックしたい列, マスに入れたい数字を与える.
2. 調べたい列の全ての行の数字と盤面に入れたい数字を比較する.
   (a) 全ての行に同じ数字がなければ制約を満たすため, True を返す.
   (b) 同じ数字があれば制約を満たさないため, False を返す.

グループでのチェック関数

1. 入力として盤面, チェックしたい行と列, マスに入れたい数字を与える.
2. 1 の行, 列から調べたいマスを特定し, それが所属するグループを調べる.
3. グループ内に入っている数字と盤面に入れたい数字を比較する.
   (a) グループ内に同じ数字がなければ制約を満たすため, True を返す.
   (b) 同じ数字があれば制約を満たさないため, False を返す.

4.1.3　基本テクニック1：エリア確定法

各エリアだけを見て数字を確定する方法. 行, 列, グループそれぞれについてエリア確定法を行う関数を作成し, それらをまとめて全体のエリア確定法とする.

エリア確定法は以下の手順で動く.

1. 入力としてパズルの盤面を与える.
2. 全てのエリアに対してエリア確定法を行い, 1マスでも盤面を解き進めることが出来れば, もう一度エリア確定法を行う. これを盤面を解き進めることが出来なくなるまで繰り返す.

エリア確定法のサブルーチン

行でのエリア確定法

1. 入力として盤面, 調べたい行, マスに入れたい数字を与える.
2. 与えられた行の空きマスの数をカウントする. 空きマスは盤面上では0で入っているため, 0の数をカウントする.
   (a) 空きマスが1つであれば, 与えられた盤面, 調べたい行, 各列, マスに入れたい数字についてチェック関数を使う.
      i. 制約を満たす場合, 入れたい数字をマスにいれ, True を返す.
      ii. 制約を満たさない場合, 何も操作をせず False を返す.
   (b) 調べたい行の空きマスが1つでなければ何も操作をせず False を返す.

列でのエリア確定法

1. 入力として盤面, 調べたい列, マスに入れたい数字を与える.
2. 与えられた列の空きマスの数をカウントする.

(a) 空きマスが 1 つであれば, 与えられた盤面, 調べたい列, 各行, マスに入れたい数字についてチェック関数を使う.

i. 制約を満たす場合, 入れたい数字をマスにいれ, True を返す.

ii. 制約を満たさない場合, 何も操作をせず False を返す.

(b) 調べたい列の空きマスが 1 つでなければ何も操作をせず False を返す.

グループでのエリア確定法

1. 入力として盤面, 調べたい行, 調べたい列, マスに入れたい数字を与える.
2. 与えられた行, 列から調べたいマスが所属するグループを求める.
3. 2 で求めたグループの空きマスの数をカウントする.

(a) 空きマスが 1 つであれば, そのグループの全てのマスをチェックする.

i. 制約を満たす場合, 入れたい数字をマスにいれ, True を返す.

ii. 制約を満たさない場合, 何も操作をせず False を返す.

(b) 調べたいグループの空きマスが 1 つでなければ, False を返す.

4.1.4 テクニック 2：可能性確定法

あるマスについて, そこに入る可能性がある数字が唯一の場合に確定する方法. マスに入れることが出来る数字を可能性, マスの可能性を保持するリストをマス可能性リスト, 可能性全体をまとめるリストを盤面可能性リストとする.
可能性確定法は以下の手順で動く.

1. 入力として盤面を与える.
2. 入力の盤面に対して tech2_possibility() を使い, すべてのマスの可能性を調べ, マスごとにリストにする..
3. マス可能性リストの長さを調べ, リストの長さが 1 であれば, その数字を盤面に入れる.

4. もう一度可能性確定法が使えないかを調べる. これは方法を使う前と使った後で埋まったマスが増えたかどうかで決める.

   (a) 盤面に変化があれば更に進めることが出来る可能性があるため, 3 で数字を入れた後の盤面を入力して 1 に戻る.

   (b) 変化がなければ可能性確定法を終了する。

可能性確定法のサブルーチン

全てのマスの可能性を調べる関数　(tech2_possibility())

1. 入力として盤面を与える.
2. 盤面すべてのマスを調べていく. マス可能性リストを左上から順に作成し, それを盤面可能性リストに順に追加していくことでマスの順番に可能性を保存した盤面可能性リストを作成する. すでに盤面に数字が入っているマスは可能性を調べる必要がないため, マス可能性リストは空リストとなる. 空きマスには 1 から 9 までの数字を入れ, 制約を満たすかをチェックする. 制約を満たす数字はそのマスに入る可能性があるため, マス可能性リストに追加する.

4.1.5　2 つの基本テクニックを使う関数　(technique 関数)

1. 入力として盤面を与える.
2. 盤面に対してエリア確定法と可能性確定法の 2 つを使う.
3. 入力として与えられた盤面とテクニック使用後の盤面を比較し, 変化があれば変化後の盤面を入力とし 1 に戻る. これを盤面の変化がなくなるまで行う.

4.1.6　盤面が正解であるかを判定する関数　(answer 関数)

1. 入力として盤面を与える.
2. 盤面すべてのマスの空きマスの数を数える. 空きマスの数が 0 個であればすべてのマスが数字で埋まっている状態となっている. 空きマスを埋める際には check 関数を利用しているため, 制約を満たさない数字が盤面に入っていることはない. つまりマスが全て埋まっていればそれが正解の盤面となる.
   (a) 空きマスの数が 0 個であれば True を返す.
   (b) 空きマスの数が 0 個でなければ False を返す.

4.1.7　プログラム動かすうえで必要な関数

テキストを二次元配列に変換する関数

プログラム実行時に数独の問題例として 81 文字の数字を与える. この 81 文字の数字をプログラム中での数独を表すデータ構造に変換するためにこの関数を使う. 空のリストに入力したテキストを入れていき, 9 文字入れたら盤面のリストに追加する. これを最後まで行い, パズルの盤面を表す 9×9 の二次元リストを作成する.

4.2　工夫した点, 注意点

4.2.1　幅優先探索

幅優先探索では数字を入れる際に child の空きマスの部分を書き換えるが, 1 つの盤面に対して何度も書き換えを行う必要があるため, child のコピーを作成し, コピーを書き換えていくことで不具合なく次の子を作成することが出来る.

4.2.2 エリア確定法

エリア確定法の (a) で, すでに盤面にマスが埋められている部分に対してこの操作が起こらないようにマスに数字を入れる前に空きマスであることを確認する.

4.2.3 可能性確定法の補足

ここで盤面可能性リストにはすべてのマス可能性リストが順番に入っているが, 盤面に入っている数字を指定するときと同じ方法は使えない. 盤面は 2 重リストで value[2][3] のように位置を指定出来るが, 盤面可能性リストは potential[位置][マス可能性リストの中身] のようになっているため, 位置を [9 * y + 1] のように指定する必要がある.

第5章　実験と評価

実験の目的は基本テクニックの繰り返しでどこまでパズルを解くことが出来るかを確かめること．また, それで答えにたどり着くことが出来なければ, どのような工夫をすれば最後まで解くことが出来るか, より早く解くことが出来るかを考察することである．

5.1　使用する問題

入門から超難問までの 6 段階の問題を 3 つずつ, 計 18 問用意した．問題は “レッツ！ナンプレ”[2], “見やすい 解きやすい ナンプレ”[3] のそれぞれから選んだものである．入門から難問までは [2] から 1 問, [3] から 2 問ずつ選んだ．超難問は [2] に問題がなかったため, [3] から 3 題選んだ．[2], [3] の問題の難易度は掲載されている本およびサイトで設定されていた通りである．
使用した問題と難易度を以下に示す．

問題 1, 入門

```
5 6| 8 |7 3
19 | 2 | 85
  7|5 9|4
---+---+---
 7 |   | 3
4  |937|  6
 63|   |17
---+---+---
7  | 1 |  9
 4 |6 8| 5
  1|   |3
```

問題 2, 入門

```
84 | 3 | 92
  2|   |6
 3 |4 6| 8
---+---+---
5  | 8 |  9
1 4| 9 |2 7
 6 |3 4| 1
---+---+---
  7|   |3
 5 | 6 | 2
3 8|5 9|1 6
```

問題 3, 入門

```
 26| 5 |3 7
839| 17| 5
 74| 3 |8
---+---+---
   | 23|7 9
213| 98|  6
9 7|4 5| 23
---+---+---
 91|5  | 38
3  |   |6
768|  2|91
```

問題 4, 初級

```
7  | 5 |  4
 5 |2 7| 1
 1 | 9 | 7
---+---+---
6 1|   |7 5
  7|6 5|4
   |8 4|
---+---+---
53 | 2 | 48
 28|   |65
  6|   |2
```

問題 5, 初級

```
94 | 6 | 51
  1| 7 |3
   |1 5|
---+---+---
 9 |   | 2
31 |259| 64
 2 |   | 1
---+---+---
   |6 3|
  4| 1 |6
53 | 9 | 7
```

問題 6, 初級

```
   |  7| 2
 57|6 9|  4
   |4 1| 98
---+---+---
  1|   | 6
52 |916|8
   |   |9 1
---+---+---
 79| 4 |
31 |89 |4 5
   |   | 1
```

問題 7, 中級

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

問題 8, 中級

```
8  | 95|3
 5 |3  | 4
  9| 2 |  5
---+---+---
 2 |1  |  4
  4|   |2
9  |  3| 8
---+---+---
2  | 6 |7
 7 |  1| 2
  1|78 |  9
```

問題 9, 中級

```
   | 26| 3
9 7|   | 5
6  | 8 |
---+---+---
   |  2|  5
 3 |   |6
   | 9 |47
---+---+---
 5 |  7| 18
   |43 | 9
8  | 1 |
```

問題 10, 上級

```
 2 | 3 | 7
  6| 9 |5
   |1 2|
---+---+---
735|   |126
   | 1 |
   |6 7|
---+---+---
  3| 6 |4
 7 |4 8| 5
4  |   |  8
```

問題 11, 上級

```
 31| 5 |
2  |3  |1
4  |7  | 8
---+---+---
6  |8  |
 83|   |96
   |  5|  3
---+---+---
 6 |  3|  5
  4|  1|  6
   | 6 |73
```

問題 12, 上級

```
8  | 95|3
 5 |3  | 4
  9| 2 |  5
---+---+---
 2 |1  |  4
  4|   |2
9  |  3| 8
---+---+---
2  | 6 |7
 7 |  1| 2
  1|78 |  9
```

問題 13, 難問

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

問題 14, 難問

```
   |31 |  8
  6| 8 |
 9 |6  |1
---+---+---
5 9|   |
74 | 9 | 52
   |   |4 9
---+---+---
  7|  4| 2
   | 2 |6
4  | 69|
```

問題 15, 難問

```
   |63 |
   |   |2
428|7  |  6
---+---+---
1 2|   | 79
 8 |   |3
  9|  3|  8
---+---+---
   |  7| 2
   | 9 |45
 56|   |
```

問題 16, 超難問

```
   2|   | 6
 4  |   | 2
    |5  |
 ---+---+---
    | 84|
 75 |   |4
  3 |   |
 ---+---+---
   8| 61|
    |  2|
    |   |5 7
```

問題 17, 超難問

```
 8 |1   |
   | 7 | 16
61 |8   |
---+---+---
  4|    |7 2
   |9 6|
9 5|    |4
---+---+---
   |  1| 28
45 | 9 |
   |  3| 4
```

問題 18，超難問

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

5.2 使用するソルバー

ソルバー 1. エリア確定法のみ

ソルバー 2. エリア確定法と可能性確定法

ソルバー 3. エリア確定法, 可能性確定法と幅優先探索

1, 2 では解を出すことが出来るか, 出すことが出来なければ何マス埋めることが出来たかを確かめる. 3 では解が出るまでの時間と, 必要仮定数を確かめる. 仮定での中でも, ある仮定をおいた後再度基本テクニックを適用すると最後まで問題を解くことが出来るようになるものを必要仮定と呼ぶ. 例えば, 基本テクニックが使えない盤面に対して, 1 行, 2 列のマスに 4, 3 行, 6 列のマスに 8 を仮定としておくと基本テクニックの繰り返しで解を出すことが出来れば, その 2 つの仮定が必要仮定となる. 必要仮定数は必要仮定の数で, 上記の例では必要仮定が 1 行, 2 列 に 4, 3 行, 6 列 に 8 であるため, 必要仮定数は 2 である.

5.3　実験結果

5.3.1　ソルバー 1 の結果

エリア確定法のみを使用した場合, 問題 1 から 18 までのすべての問題例で, 1 マスも進められなかった.

問題 1, 問題 9, 問題 13 にソルバー 1 を使用し, 進められなくなった時点での盤面を以下に示す.

問題 1

```
5 6| 8 |7 3
19 | 2 | 85
  7|5 9|4
---+---+---
 7 |   | 3
4  |937|  6
 63|   |17
---+---+---
7  | 1 |  9
 4 |6 8| 5
  1|   |3
```

問題 9

```
   | 26| 3
9 7|   | 5
6  | 8 |
---+---+---
   |  2|  5
 3 |   |6
   | 9 |47
---+---+---
 5 |  7| 18
   |43 | 9
8  | 1 |
```

問題 13

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

5.3.2　ソルバー 2 の結果

ソルバー 2 でエリア確定法と可能性確定法の繰り返しを使用した結果を表 5.1 に示す. 表では最後まで解くことが出来たかどうか, 解くことが出来なければ埋めることが出来た空きマスの数を記す.

表 5.1 ソルバー 2 で解くことが出来たかと埋めることが出来た空きマスの数

問題	解が出たか	埋めた空きマスの数
問題 1	○	-
問題 2	○	-
問題 3	○	-
問題 4	○	-
問題 5	×	0 マス
問題 6	○	-
問題 7	×	6 マス
問題 8	×	0 マス
問題 9	×	8 マス
問題 10	×	28 マス
問題 11	×	0 マス
問題 12	×	10 マス
問題 13	×	0 マス
問題 14	×	1 マス
問題 15	×	0 マス
問題 16	×	0 マス
問題 17	×	0 マス
問題 18	×	0 マス

問題 1, 問題 9, 問題 13 にソルバー 2 を使用した結果を以下に示す.

問題 1

```
526|481|793
194|723|685
837|569|412
---+---+---
275|146|938
418|937|526
963|852|174
---+---+---
752|314|869
349|678|251
681|295|347
```

問題 9

```
   | 26| 3
9 7| 4 | 5
6  | 8 | 4
---+---+---
   | 72| 85
 3 | 5 |62
   | 9 |47
---+---+---
 5 | 67| 18
   |43 | 9
8  | 1 | 6
```

問題 13

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

5.3.3 ソルバー 3 の結果

ソルバー 3 ではすべての問題例について正解を求めることが出来た. それぞれの問題例で解が出るまでの時間と, 必要仮定数を以下の表に示す.

表 5.2 ソルバー 3 で解が出るまでの時間と, 必要仮定数

問題	かかった時間 [s]	必要仮定数
問題 1	0.095	0
問題 2	0.102	0
問題 3	0.082	0
問題 4	0.093	0
問題 5	1.364	1
問題 6	0.090	0
問題 7	12.784	2
問題 8	13.556	2
問題 9	12.311	2
問題 10	0.165	1
問題 11	332.539	2
問題 12	2.092	1
問題 13	17.927	2
問題 14	38.800	2
問題 15	66.435	2
問題 16	453.498	2
問題 17	147.038	2
問題 18	923.645	2

また, 問題 1, 問題 9, 問題 13 にソルバー 3 を使用した結果を以下に示す.

問題 1
```
526|481|793
194|723|685
837|569|412
---+---+---
275|146|938
418|937|526
963|852|174
---+---+---
752|314|869
349|678|251
681|295|347
```

問題 9
```
548|726|931
927|143|856
613|589|742
---+---+---
496|372|185
731|854|629
285|691|473
---+---+---
354|967|218
162|438|597
879|215|364
```

問題 13
```
328|694|517
174|325|869
965|718|342
---+---+---
741|289|635
283|567|491
659|143|728
---+---+---
496|871|253
832|956|174
517|432|986
```

5.3.4 結果のまとめ

ソルバー 1, 2 の結果, 基本テクニックの繰り返しのみで解くことが出来るのは入門, 初級までであった. ただ中級, 上級までは途中まで進めることが出来たが最後まで解くことは出来なかった. 難問, 超難問に関しては基本テクニックだけでは進めることも出来なかった.

必ず最後まで解くことが出来るように幅優先探索を加えたソルバー 3 を使用する必要があった. 仮定法を使う場合, 必要仮定数が解を求める時間に大きく影響した. 必要仮定数が少ない問題は早い段階で解き終わるが, 2 回になると問題によってはかなり時間がかかった. また必要仮定を早い段階で見つけることが出来れば正解にたどり着く時間も短くなった. 例えば, 同じ超難問の問題でも問題 16 では, 2 行, 5 列のマスに 7 と 3 行, 8 列のマスに 1 が必要仮定だが, 問題 17 では, 1 行, 5 列のマスに 3 と 4 行, 8 列のマスに 3 が必要仮定で, 問題 17 の方が早く必要仮定が見つかるため, 解き終わる時間がかなり早くなっている.

5.3.5　結果の考察

ソルバー 1, 2 の結果より基本テクニック 1 より 2 の方がより強力であることが分かった．もっと早く, 最後まで問題を解くためには基本テクニックだけでなく, 消去法など他のマスの可能性から数字を確定させていく, 基本テクニックより難易度の高い解法を用いるか, 必要仮定をいかに早く見つけるかを工夫することが必要だと考える．本研究では幅優先探索で順に仮定を調べていったが, 可能性が少ないマスから調べる工夫をするとより早く必要仮定を見つけることが出来ると考える．また, 今回実験に使用した問題例セットでは, 高々2 つの必要仮定を見つけることが出来れば, 基本テクニックの繰り返しだけで問題を解くことが出来たため, 人が数独を解くには十分強力であると考える．バックトラックは記録が必要で, 時間もかかることから人にとっては労力のかかることであり, 何度も行うことは出来ないため多くても 2 回程度が限度だと問題作成者も考えているのではないだろうか．

第6章　一般的な数独の問題の難易度についての考察

まずは問題の空きマスの数が1つの指標となる. 空きマスが少なければ少ないほど, 空きマスに対する数独の制約が厳しくなるため, 基本テクニックが使える可能性が高くなり, 簡単な問題となる. 逆に空きマスが多ければ多いほど, 問題の時点では基本テクニックが使えず, 空きマスの可能性がより多くなるため, より難しい方法で解かなければならない. 今回取り上げた問題も難易度が上がるにつれて空きマスの数も増えていく傾向にある. ただしパズルは基本的に答えが1つに限られるため, 空きマスを極端に増やしすぎることは出来ない. 9×9 の問題に関しては初めに埋められている数字の最小は17個であると知られている [4].

また今回実践したような, 基本テクニックだけで解けるかどうかも難易度に関わる. 実際に実験結果から入門と初級は基本テクニックだけで解けることが分かった. 今回は基本テクニックしか扱っていないが, 中級や上級はこれより少し難易度が高い解き方の繰り返しで解くことが出来るのではないだろうか.

さらに出来る限り解法を使用し盤面が行き詰まったときにどこかに仮定をおいて進めていく方法がとられるが, 仮定をおく回数の多さも難易度に関わると考える. 今回の実験結果でも仮定をおく回数の多さで正解までたどり着く時間も多くなっているため, 一般的な解法で問題を解く場合にも同じことが挙げられると考える. 人がバックトラックを行うにはどの段階で仮定をおいたかを記録しなければならないためより負担は大きくなる.

第7章　おわりに

本研究では数独を基本的な解き方の繰り返しでどこまで解くことが出来るかを調べた. 結果, 2つの基本テクニックを用いて, 入門, 初級レベルは最後まで解くことが出来た. 中級, 上級レベルは途中まで解くことができ, 難問, 超難問は1マスも解くことが出来なかった. 最後までパズルを解くにはより難易度の高いテクニックを用いるか, 仮定をおくことで盤面を埋め, 基本テクニックが使える状態にすることが必要であると考える. そこで今回は仮定をおくことに注目してパズルを解いた. 幅優先探索を加え, 基本テクニックでは解くことが出来ない盤面に対して使用し, 制約を満たす数字のみ入れることで仮定法を実現した . これにより仮定をおいた回数が多ければ解くまでの時間が増えることが分かった. 今後の課題は, 基本テクニックについてのみの検証だけでなく, より難しい解き方を増やしていき, どのレベルの解き方まで習得すればどの問題でも最後まで解くことが出来るかの検証やいかにして基本テクニックが使える盤面になるように仮定をおくかの考察が挙げられる.

謝辞

本研究を進めるにあたり, 多くの助言と御指導を頂いた下薗准教授には心より感謝申し上げます. また, 多くの数独の問題を使用させて頂いたレッツ!ナンプレの運営事業者である IT-Links 社, 見やすい解きやすいナンプレの発行元である株式会社インテルフィンの方々に感謝の言葉を申し上げます.

参考文献

[1] Algoful - アルゴリズム初心者向けの基礎と入門 - ,
https://algoful.com/Archive/Algorithm/BFS.

[2] レッツ！ナンプレ, https://si-coding.net/
ホームページ制作・コーディング代行 IT-Links

[3] 見やすい解きやすいナンプレ, 2021 年 09 月 20 日, インテルフィン

[4] 数独の初期ヒント最小個数は 17, それ未満では解けないと数学者が結論,
https://gigazine.net/news/20120109-sudoku-puzzle-starting-digits-is-17/,
Gigazine, 2012 年 01 月 09 日