スリザーリンク解答システムと問題作成システム

白井 裕己，五十嵐 力，但馬 康宏，小谷 善行
東京農工大学

概 要

近年，ペンシルパズルというタイプのパズルが注目され，多くの人が遊んでいる．ペンシルパズルというのは図示された問題に対して答えを徐々に書き込んでいくことによって解いていくパズルのことであり，数独，ぬりかべ，カックロなど，多くのペンシルパズルが存在する．中には，世界的に人気のあるものもある．本稿では，このペンシルパズルの一つであるスリザーリンクについて述べる．一般的にペンシルパズルは制約充足問題であり，スリザーリンクも制約充足問題である．この制約を利用した探索によるスリザーリンクの解答アルゴリズムと，解答アルゴリズムを用いた問題作成システムにおいて，それぞれのアルゴリズムとそれらを用いたプログラムの実行結果について示した．

Solving and Making Problems of *Slither Link*

Hiroki Shirai, Chikara Igarashi, Yasuhiro Tajima, Yoshiyuki Kotani
Tokyo University of Agriculture and Technology

Abstract

Recently, many people are interested in the puzzles of a type of *pencil puzzle* and they play the puzzles. *Pencil puzzle* is puzzle solved by writing gradually the answer in the problem and there are many *pencil puzzles*, for example, *Sudoku*, *Nurikabe*, *Kakuro*. Some of them are popular worldwide. In this paper, we explain about *Slither Link* which is a kind of *pencil puzzle*. Most of *pencil puzzles* are explained by Constraint Satisfaction Problem (CSP) and *Slither Link* is also explained by CSP. We show algorithm for solving *Slither Link* by search that uses the constraint, algorithm for making *slither link* that uses the algorithm and results of programs that use them.

1. はじめに

本稿では，スリザーリンク解答プログラム，問題作成プログラムについて述べる．スリザーリンクとは株式会社ニコリのオリジナルのペンシルパズルである．まず，スリザーリンク解答プログラムについて述べ，次に，それを用いたスリザーリンク問題作成プログラムを作る手法について述べる．

2. スリザーリンクについて

2.1 スリザーリンクのルール

スリザーリンクとは，図１のようにいくつかのマスに 0～3 の数字が書かれた盤面に対し，次のルールに従って線を引いていくパズルである．なお、盤面のサイズについての規定はないが、形は長方形であり，m×nマスという形になる．また，今回扱う問題では解は一つしかないものとする．

(1) 点と点を縦横に繋ぎ，全体で一つのループを作る

(2) マスに書かれている数字は，そのマスの周り 4 辺のうち線が引かれる辺の数を表す（数字のないマスの周りには何本の線が引かれるか分からない）

(3) 線は枝分かれしたり交差したりしない

これらのルールによって図 1 を解いた結果を図 2 に示す.

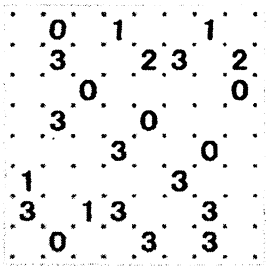

図 1 スリザーリンクの問題例 ([1]より)

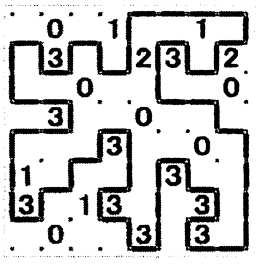

図 2 図 1 の問題の解答 ([1]より)

2.2 関連研究

スリザーリンクについては, [2]により, 解の有無判定問題の NP 完全性が証明された. また, [3]ではスリザーリンクを整数計画問題として解く方法が示されている. 本稿では, スリザーリンクをマス単位の探索により高速に解くこと及びスリザーリンクの問題を作成することを目的とする. 同じサイズの問題では辺の数よりもマスの数の方が少ないので, マス単位で扱うことにより, 辺の単位で解いていく一般的な解き方よりも探索箇所が少なくなる.

3. スリザーリンク解答プログラム

3.1 塗り分けルール

解答プログラムでは, 塗り分けルール ([4]参照) によりスリザーリンクを解いていく. 塗り分けルールとは, 出来上がるループの内側になるか外側になるかでマスごとに塗り分けていく方法である. つまり, 次のルールによって全てのマスを「内側」か「外側」に塗り分けることになる.

(1) 全体で「内側」のマスと「外側」のマスが一塊ずつになる (盤面の周りは「外側」のマスで覆われているとする)

(2) マスに書かれている数字は, そのマスと隣接する 4 マスのうち, そのマスと反対側になるマスの数を表す

ここでは分かりやすいように色を使って説明する.

3.1.1 盤面の構造

マスを塗り分けていくときは「内側」のマスを赤 1, 「外側」のマスを赤 2 というようにして塗り分けていく. 解いていくときの盤面は, 実際の問題の盤面より 1 回り大きなものにし, その 1 周分は予め「外側」つまり赤 2 で塗っておく. さらに, それ以外のマスは, 青 1, 黄 1 など, それぞれ異なる色で全て塗っておく. つまり, 問題を解いていくときの盤面の初期状態は図 3 のようになる (以下, 図中で同じ形のマスは同じ色, 異なる形のマスは異なる色, 実線のマスは色 1, 点線のマスは色 2 を表す). また, 色によるマスの表し方であるが, ある色 1 のマスと同じ色 1 のマスは同じ側のマス, 同じ色 2 のマスは反対側のマスを表す. 例えば, 青 1 のマスと青 1 のマスは同じ側, 青 1 のマスと青 2 のマスは反対側となる.

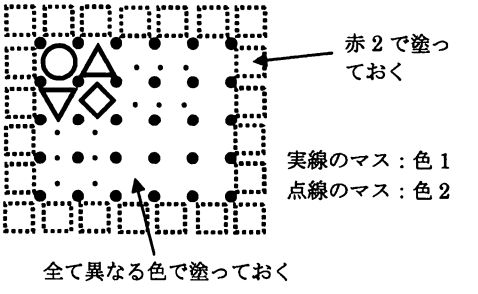

図 3 問題を解くための盤面の初期状態

3.1.2 色の塗り替え

問題を解いていって，このマスとこのマスは同じ側あるいは反対側になるということが分かったら，これらのマスをどちらか一方の色（どちらかが赤のマスの場合は赤）で統一する．例えば，緑 2 のマスと紫 2 のマスが反対側になるということが分かったら，紫 2 のマスを全て緑 1，紫 1 のマスを全て緑 2 に塗り替える．これらのマスの塗り替えの様子を図 4 に示す．このように色の塗り替えを行っていって最終的に全てのマスが赤 1，赤 2 だけになったとき，それが解である．

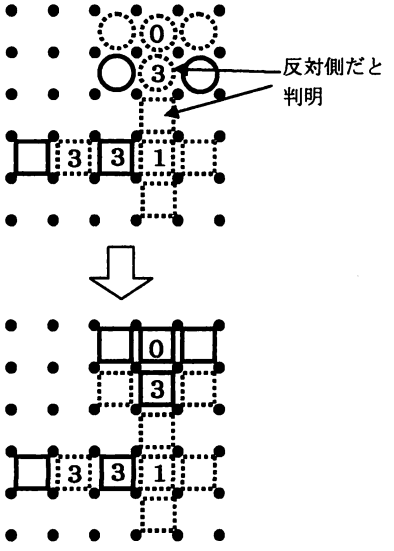

図 4　マスの塗り替えの例

3.2 塗り替え探索

バックトラック探索によって上述した塗り替えを行って解いていくが，単純なバックトラック探索では問題のサイズが大きいと探索空間が非常に大きくなってしまうので，バックトラック探索に確定探索を組み込んで解いていく．ここで，確定探索とは，その時点で答えを確定できる箇所を全て確定する，つまり，分岐せずに探索を進められるところを探索するという方法である．探索においては，答えを確定できる限り確定探索を行い，確定できないときにバックトラック探索を行う．塗り替え探索におけるスリザーリンクの探索木は基本的に二分木になるが，この確定探索を組み込むことにより，図 5 のように分岐数 1 のノードができるので，探索空間が小さくなる．また，確定探索では答えを確定する際にルールに違反していないかのチェックもしており，ここで違反が見つかったら矛盾と判断しバックトラックする．なお，この探索は全解探索となっており，仮に解が複数存在する場合はそれら全てを求める．

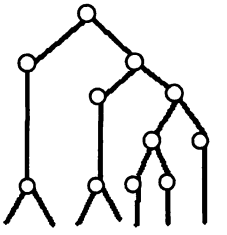

図 5　確定探索を組み込んだ
バックトラック探索

3.2.1 確定探索の内容

確定探索には，次の(1)～(4)に示した数字のマスの条件によるパターンを組み込んだ．

(1)　0 のマス（図 6）

0 のマス及び隣接する四つのマスが全て同じ側になる

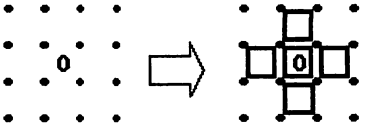

図 6　0 のマスのパターン

(2)　1 のマス（図 7）

a.　1 のマスと，隣接するマスどれか一つが反対側 → 残りの 3 マスは 1 のマスと同じ側になる

b. 1のマスに隣接するどれか二つのマス同士が同じ側 → 1 のマスはこれらのマスと同じ側になり，残りの2マスは反対側同士になる

c. 1のマスに隣接するどれか二つのマス同士が反対側 → 1のマス及び残りの2マスは同じ側になる

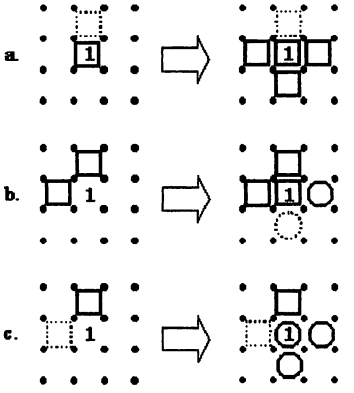

図7　1のマスのパターン

(3)　2のマス（図8）

a. 2のマスに隣接するどれか二つのマス同士が同じ側 → 残りの2マスはそれらのマスと反対側になる

b. 2のマスに隣接するどれか二つのマス同士が反対側 → 残りの2マスは反対側同士になる

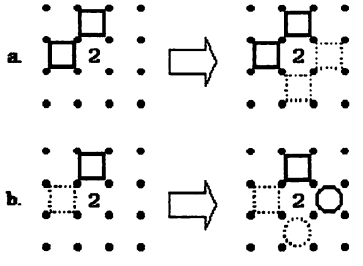

図8　2のマスのパターン

(4)　3のマス（図9）

a. 3のマスと，隣接するマスどれか一つが同じ側 → 残りの3マスは3のマスと反対側になる

b. 3のマスに隣接するどれか二つのマス同士が同じ側 → 3 のマスはこれらのマスと反対側になり，残りの2マスは反対側同士になる

c. 3のマスに隣接するどれか二つのマス同士が反対側 → 3のマスと残りの2マスは反対側同士になる

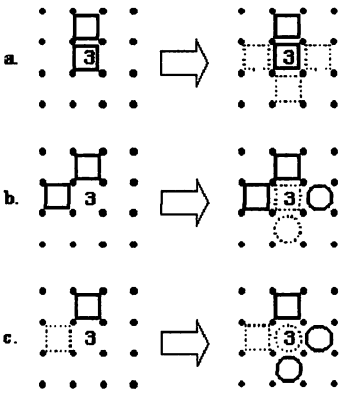

図9　3のマスのパターン

確定探索ではこれらのパターンを適用して問題を解いていくが，その際にこれらのパターンに矛盾している状態を見つけたら，確定探索を打ち切って，違反ありということを示す値を返す．ただし，(1)の 0 のマスのパターンだけは周りのマスの状況に一切関係ないので，実際には確定探索には組み込まず，塗り替え探索の前つまりプログラムの最初に1度だけ行うようにしている．

3.2.2 バックトラック探索の構造

3.2.1 の確定探索を組み込んだバックトラック探索の構造を図 10 に示す．図 10 を見て分かるように，塗り替え探索では「内側」か「外側」かの仮定による塗り替えを行った後，まず確定探索を呼び出し，その時点で確定できるところを全て確定する．このとき，ルールに違反する箇所が検出されたらすぐにバックトラックする．そして，ルール違反がなく確定探索が終了したらバックトラック探索部に移り，まだ

「内側」か「外側」かが分かっていないマス一つをまず「内側」と仮定して探索を進め，その探索が終わったら，次にそのマスを「外側」と仮定して探索を進める．さらにその探索も終了したら，バックトラックする．

```
void 塗り替え探索( color, 内/外 ){
  colorと同じ色のマスを全て内/外に塗り替える;
  /* 確定探索部 */
  if( 確定探索() == ルール違反あり )
    return;
  /* バックトラック探索部 */
  for( 全てのマスについて ){
    if( そのマスが「内側」でも「外側」でもない ){
      塗り替え探索( そのマスの色, 内 );
      塗り替え探索( そのマスの色, 外 );
      return;
    }
  }
  解答の出力;
}
```

図 10　確定探索を組み込んだバックトラック探索の構造

3.3 実行結果

このプログラムで[5]の問題から選んだ 12 問を解いたときの実行時間を表 1 に示す．この 12 問全てにおいて正しく解くことができた．

表 1　解答プログラムの実行時間

サイズ [マス]	実行時間 [s]	ノード数
10×10	0.02	15
10×10	0.02	27
10×10	0.03	289
10×10	0.03	353
18×10	0.03	97
18×10	0.03	365
18×10	0.06	849
18×10	0.43	9677
24×14	0.36	4623
24×14	4.87	80557
24×14	13.4	196869
24×14	72.8	1154669

(CPU:Pentium4 1.90GHz、メモリ:1.00GB)

4. スリザーリンク問題作成プログラム

解答プログラムを用いて，問題作成プログラムを作成した．問題作成プログラムはそれぞれ異なる方法で，数字逐次追加方式と数字逐次削除方式の 2 種類のプログラムを作成した．問題作成プログラムでは解が複数ある場合は解の正確な数が分からなくても複数あるということだけ分かれば良いので，ここでは解答プログラムの部分は全解探索ではなく，解が二つ見つかったらその時点で探索を打ち切るようにしている．

4.1 数字逐次追加方式

4.1.1 数字逐次追加方式のアルゴリズム

数字逐次追加方式は，数字が一つもない盤面に 0〜3 の数字をランダムに選んだマスに 1 マスずつ入れてそれを解くというのを繰り返し，解が一つだけになったら終了するという方法である．ただし，数字を入れたときに解がなくなってしまう場合もあるので，その場合はそのマスに別の数字を入れるようにしている．また，解が複数あるまま全てのマスに数字が埋まってしまう可能性もあるので，その場合は問題を作れないままプログラムを終わらせる．図 11 にこの方式の流れ図を示す．

4.1.2 数字逐次追加方式の特徴

この方法ではマスに数字を入れるところで入れる数字を限定することにより，使う数字を限定した問題を作成することができる．例えば，数字が 1 だけの問題や 2 と 3 だけの問題などを作ることができる．例として，使う数字を限定しないで作成した問題，2 だけに限定して作成した問題をそれぞれ図 12，図 13 に示した．

4.1.3 数字逐次追加方式の実行結果

5×5 のサイズから 8×8 のサイズまで各サイズ 6 回ずつ実行したところ，6×6 までは全て実行時間が短く，最大で 0.05 秒程であったが，7×6 から 1 分以上経っても実行が終了しない場合が見られるようになった．7×6 では大抵は 6×6 の時とあまり変わらない時間で実行が終了し，1 分以上かかったのも 1 回だけだ

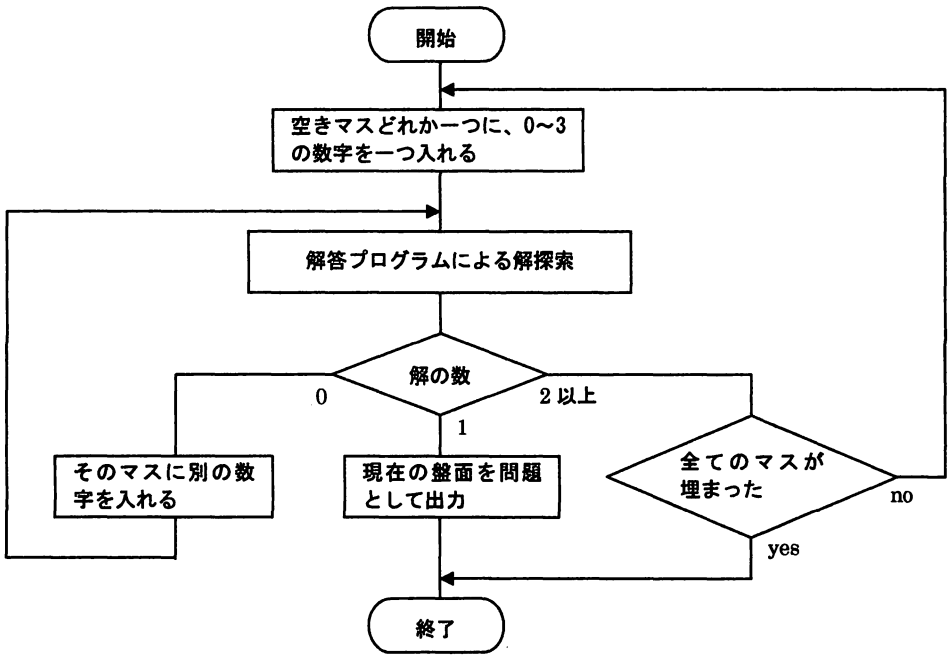

図 11　数字逐次追加方式の流れ図

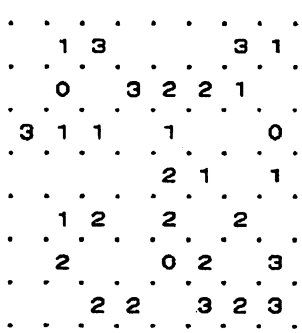

図 12　数字逐次追加方式で作った問題
（数字の限定なし）

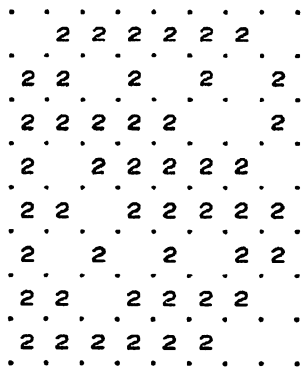

図 13　数字逐次追加方式で作った問題
（使う数字を 2 に限定）

ったが，サイズが大きくなるにつれてその回数が多くなり，8×8 では 1 分以上かかる回数の方が多かった．また，サイズが大きくなると実行時間のばらつきも大きくなり，8×8 でも実行時間が短いものは 0.06 秒程で終了した．これは，どのようにマスに数字が入っていくかで探索時間に差が出るが，サイズが大きくなるほど探索回数が多くなるので，その差が大きな差となって現れるためだと考えられる．

4.2 数字逐次削除方式

4.2.1 数字逐次削除方式のアルゴリズム

数字逐次削除方式は，最初に解答となるループの形を決めてそれに合わせて全てのマスに数字を入れる．そして，ランダムに選んだマスから 1 マスずつ数字を消してそれを解くというのを，消せる数字がなくなるまで繰り返す方

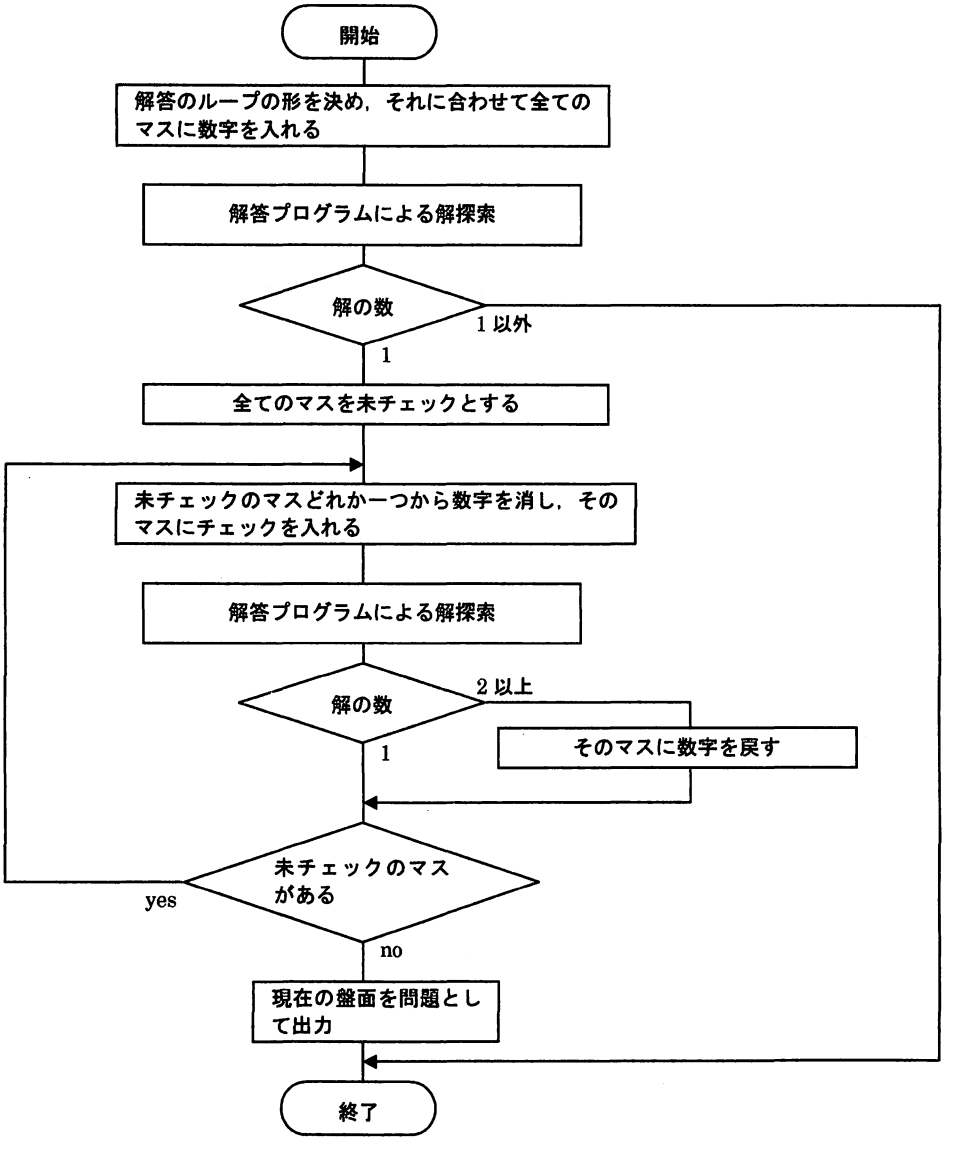

図 14　数字逐次削除方式の流れ図

法である．ここで，消せる数字がないというのは，盤面のどの数字を消しても解が複数になってしまうという状態である．ただし，最初に数字を全部埋めた時点で解が複数あるものや，最初のループの形がルールを満たしておらず解が存在しないものは，問題を作ることができないのですぐに終了する．図 14 にこの方式の流れ図を示す．

4.2.2 数字逐次削除方式の特徴

この方法では先に解答のループの形を決めているので，答えが好きな形になる問題を作ることができる．交差のない一筆書きのループという条件の下なので制限はあるが，例えば，答えが何かの図や模様になるといった問題を作ることができる．例として，この方式で作成した問題を図 15 に示す．

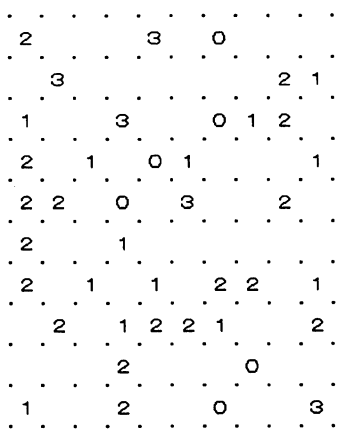

図 15　数字逐次削除方式で作成した問題

4.2.3 数字逐次削除方式の実行結果

数字逐次削除方式では予め解答となるループの形を決めておかなければならないので，[5]の問題のうち 10×10 のサイズの問題の中から 5 問選び，それらの解答をループの形とし，それぞれ 3 回ずつ合計 15 回実行した．その結果，15 回のうち 12 回は実行時間が 10 秒以内あるいは約 10 秒であり，最小のものは約 0.4 秒であった．しかし，実行時間が最大のものは約 68 秒であり，実行時間に大きな差があった．これも数字逐次追加方式と同様に，探索回数が多いことにより，1 回の探索時間の差が大きな差となったためだと考えられる．

5. 考察

スリザーリンク問題作成プログラムであるが，問題のサイズが大きくなると実行時間が長くなってしまうため，数字逐次追加方式，数字逐次削除方式ともに現時点ではまだサイズの小さな問題しか作ることができない．そのため，改良する必要があるが，どちらの作成方法の場合も実行時間のほとんどは解答プログラムが動いている時間だったので，解答プログラムの改良が重要であると言える．

そこで，解答プログラムの改良方法であるが，まずは確定探索の部分である．現時点では確定探索には 3.2.1 で述べた単純なパターンしか組み込んでいないので，もっと色々なパターンを組み込んで確定探索を強化することができる．そうすれば，図 5 に示した探索木の直線部分が長く多くなるので，探索空間がもっと小さくなる．実際に人が解くようにバックトラック無しで解けるようになるのが理想である．

また，バックトラック探索の部分でも改良が考えられる．このプログラムではバックトラック探索で左上のマスから順に未確定のマスを調べて仮定しているが，確定探索が有効になるように仮定するマスを選択できるようにすればより確定探索が効果的になる．

6. おわりに

バックトラック探索を用いているので，理論的にはどんなサイズの問題でも解答，作成することができるが，実行時間が問題である．実際に 5 章で述べたような改良をどれだけできるかが課題であり，できるだけ大きなサイズの問題を解いたり作成したりできるようにしたい．

参考文献

[1] web ニコリ［パズル通信ニコリ］ http://www.nikoli.co.jp/ja/ （2006/10/10 アクセス）

[2] 八登 崇之: スリザーリンクの NP 完全性について，情報処理学会アルゴリズム研究会研究報告，Vol.2000, No.84, pp.25-31 (2000)

[3] 杉村 由花: 整数計画法を用いたスリザーリンクの解法，卒業論文，東京大学 (2005)

[4] ExpF Burrow http://hp.vector.co.jp/authors/VA010341/ （2006/10/10 アクセス）

[5] ニコリ編，ペンシルパズル本 24 スリザーリンク 1，株式会社ニコリ (1992)