ZDD を用いたスリザーリンクパズルの難易度評価と問題自動生成アルゴリズムの提案

○船江公希　永田裕一　小野典彦　（徳島大学）

In slitherlink , Proposal of Difficulty evaluation using ZDD and Automatic generation algorithm

∗Kohki Funae, Yuichi Nagata, Norihiko Ono (Tokushima University)

Abstract− In recent years, puzzles have been played by many people and studied for difficulty metric and automatic problem generatior. In this paper, we propose a difficulty metric for a puzzle called slitherlink using ZDD and an automatic problem generation algorithm based on a tabu search, which incorporates the difficulty metric proposed.Furthermore, we report the experimental results and validate the difficulty metric through the questionnaire about the generated puzzles.

Key Words: ZDD, ペンシルパズル, メタ戦略

1　はじめに

パズルは現在, 多くの人々に親しまれている娯楽の一つである. パズルとは論理的な思考を用いてある命題に対し解を導く問いで, 数学を用いるパズルや, 言葉を用いたパズル, ボードゲームを用いたパズル等様々なものが存在する. その一種に, ペンシルパズルがある. ペンシルパズルとは紙に印刷された問題の図に鉛筆で書き込む形式のパズルで, それらを扱う雑誌は数多く存在する. 有名なものには「数独」「ナンクロ」「スリザーリンク」「イラストロジック」などがある. そして近年はこれらのパズル, 特に数独に対しては難易度判定の研究[1)], 問題自動生成[2)] の研究が盛んに行われている.

問題の自動生成手法については現状, その多くが, 対象とするパズルの性質を考慮した生成手法に留まっている. 例えばヒントの数字に制約を持つ問題を生成する手法や, 解答を所望の形に変形した状態で生成する手法などである. パズルの難易度を利用した問題自動生成の研究は未だ着手されておらず, ある問題に対しより難易度の高い問題, より低い問題を自動生成する手法の研究を行う余地があると考えられる. そこで本研究では, ペンシルパズルの一種であるスリザーリンクパズル (以下スリザーリンク) を研究対象とし,2 つの提案を行う. 1 つめに, ヒント適用による解の候補数の減少量とヒントの適用順序, 解に必要のない冗長なヒントの数に着目し, ZDD を用いた問題の難易度評価に関する考察と提案を行う, 次にその難易度に基づくメタヒューリスティックな探索法を用いた, 所望の難易度の問題を自動で生成できるアルゴリズムを提案する.

2　スリザーリンク

スリザーリンクは次のように定義されている. スリザーリンクは, 正方格子点の隣り合う二点間を線分でつなぎ, 一つのサイクル (閉路) を作るパズルである. 以下にルール (制約) を示す.

- 各線は隣り合う点同士を水平線もしくは垂直線で描かなければならない
- 線を交差させる動作, 線に分岐を生じさせる動作をしてはならない
- いくつかの単位正方形（以下マス）を構成する四つの点の中には 0 から 4 までの数字 (以下ヒント) が記入されており, そのマスを構成する 4 辺のうち, ちょうどそのヒントの数字の数だけ線を引かなければならない
- ヒントが記入されていないマスの周囲 4 辺には何本の線を引いても構わない

これらの制約を満たすただ一つのサイクルを導くパズルがスリザーリンクである. 解は必ず一意に定まる.Fig1 にスリザーリンクの問題例及びその解答を示す.

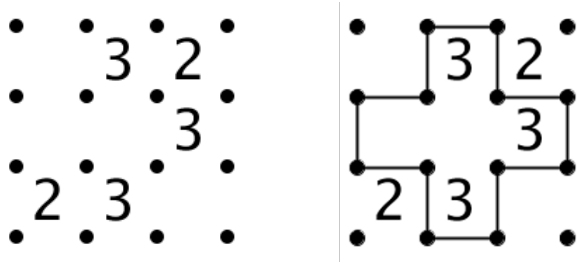

Fig. 1: A sample of Slitherlink

3　ZDD と解答アルゴリズム

パズルを生成する一般的な手法に　(1) パズルの問題を制約に則って生成 (2) 問題解答器を用いて生成された問題の解が一意に定まるかどうかを検証　以上の 2 つの手順を繰り返すものが存在する. 本研究ではゼロサプレス型二分決定グラフ[3)](Zero Suppressed Binary Decision Diagram; ZDD) と呼ばれる, 組み合わせ集合を効率的に表現できるデータ構造 (Fig2) を用いた問題解答器を採用し, 問題の自動生成を行っている. 問題解答器の詳説については[4)] を参照されたい. ノードをスリザーリンクの頂点, エッジをスリザーリンクの解の一部である辺の要素と考え, 与えられたグラフを全体グラフ, 経路やサイクル, 木などの列挙するグラフを部分グラフとすると, スリザーリンクは特定の制約を満たすサイクルである, 部分グラフを列挙する問題と言える.ZDD は膨大な数の組み合わせ集合を効率的に、かつ高速で列挙できることから、スリザーリンクを上記の部分グラフ列挙問題と考えて解く.

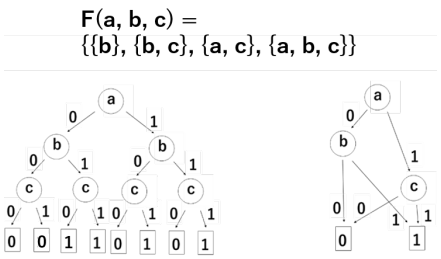

Fig. 2: A conbination set and ZDD

解答アルゴリズムの手順を Algorithm1 に示す. まず問題の大きさを考慮した全体グラフから 1 つのサイクルのみとなる辺の集合を列挙する (Line1). 次に, その中から各マスの辺数制約 (ヒント) に応じた解の候補を,ZDD の演算処理を用いて絞り込む (Line3〜7). この処理をすべてのマスに適用させるスリザーリンクの解を求める. 本研究では ZDD を効率的に操作できる Python パッケージである, Graphillion[5] を使用している. この解答アルゴリズムの特徴として, ヒントを適用したときの集合の数 (解の候補となるサイクルの数) が容易に計算可能な点がある. 構築される ZDD を組み合わせ集合と考えると, 1 つのサイクルのみとなるエッジの集合からその組み合わせの数はヒントを適用させるごとに減少してゆき, 最後のヒントを適用させたときには組み合わせの数はただ一つとなる.

Algorithm 1 Solution Algorithm

1: すべてのサイクルを持つ $ZDD(Zc)$ を構築する
2: **for all** 辺数制約を持つ各マス **do**
3: E' の要素を L 個含むすべての組み合わせを表す $ZDD(Z^i_{E'})$ を構築する
4: Zc の中で $Z^i_{E'}$ の組み合わせ集合を含む集合 Zc' を構築する
5: E' の要素を $L+1$ 個以上含むすべての組み合わせを表す $ZDD(Z^e_{E'})$ を構築する
6: Zc' の中で $Z^e_{E'}$ の組み合わせ集合を含まない集合 Zc'' を構築する
7: $Zc = Zc''$
8: **end for**

4 難易度評価

パズルにおける難易度とは人間が解く際に難しい, 時間がかかると感じる尺度で, 個人差がある. また, 解く人間のパズルに対する知識量, 理解度も難易度に深く関わるだろうと思われる. そこで本論文では個人差のある尺度を用いず, ある程度一般化された尺度として, ヒントの持つ性質に着目した難易度を提案する.

スリザーリンクでは, 一般に“0”や“3”は情報量が大きく, 解答のために確定される辺が決まりやすい. つまり解答を容易にする傾向があるのに対し, “2”が記入されたセルは 4 辺の使用方法が 6 通りもあるため情報量が少ない. こういったヒントの情報は「解 (サイクル) の候補数の削減割合」という観点から考えることができる. またあるヒントで解の候補数が大幅に削減できる場合, その問題は容易な問題であると考えられる. 難易度を数値化するにあたりこの性質を利用する. ヒント適用数と解の候補数を 2 次元グラフ化した場合における面積の大きさは, スリザーリンクの性質上, 問題の難易度と比例関係にあると推測される. よってこの面積が大きいほどに難易度が高くなる傾向にあると推測できる. ただし解候補数の減少量ではなく減少割合を評価するために, 解の候補数は log スケールで計算する. 例として Fig3 にニコリが提供するスリザーリンク [6] の解答過程を示す. ここで, 面積を考える上で, ヒントをど

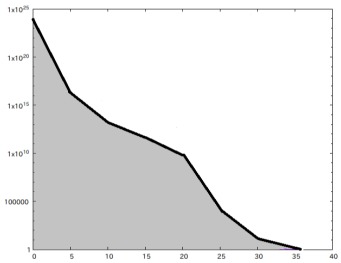

Fig. 3: A sample of Solution Candidate

のように適用させるべきかを考える. ヒントの適用順序によって解の候補数の削減量は変わるため, ヒントの適用順序も重要だと考えられる. 人間は問題がより簡単に解けるように, 解の候補数削減に大きく寄与するだろうと予想されるヒントを先に適用させることで解いていると考えられる. そのためヒントは解の候補数の減少量が大きいヒントから順に適用させていくことが望ましいと考えられる.

また, ヒントの適用順序を難易度の要素とする場合, 問題を解く上で必ずしも適用させる必要のないヒントが存在することに注意したい. これを冗長なヒントと呼ぶ. 人間にとって冗長なヒントは解く手がかりとして重要であるが, 解答アルゴリズムにとっては問題解答に必要のない制約のため, 無視して処理されてしまう. よって冗長なヒントが多いほど難易度を反比例的に下げる, つまり難易度評価を行う上で, 冗長なヒントを考慮する必要がある.

冗長なヒントの数を補正項として評価に加える上で, すべてのマスにヒントが配置されている問題 (Fig4 左) を考える. この問題は冗長なヒントが 6 マス分存在しており, 問題を容易にしている. 次に, これに対しマス目が約 2 倍の数の問題 (Fig4 右) を考える. この問題は冗長なヒントが 13 マス分存在しており, こちらも同じく問題を容易にしている. ただヒントによって解の形状が芋づる式に解けるわけではなく, 仮定を置いて解く必要のある箇所も存在する. つまり, すべてのマスにヒントが配置されているからといって, 難易度を固定化するような評価は相応しくないと考えられる.

また, 冗長なヒントの数のみを単純に評価とする場合, 問題の大きさ, ヒントの配置が可能なマスの数によって冗長なヒントを配置し得る数が多くなり, 結果として

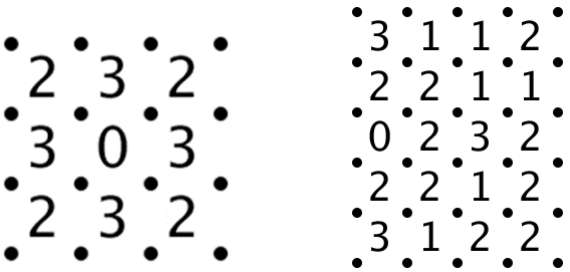

Fig. 4: Perfect-Hint Slitherlink

問題の大きさが大きいほど難易度が簡単になるような補正項になる可能性が高い. よって本研究では, ヒントの数と冗長なヒントの数の比率を補正項として加える評価を提案する. あくまで冗長なヒントは人間にとって問題を容易にする傾向にあるヒントなので, 冗長なヒントの数によって難易度が面積によって算出される値の半分以下になることは避けたい. よって補正項は範囲が 2 分の 1 から 1 の間となるように定める. ヒント数を H, 冗長なヒント数を E とすると補正項 P は式 1 で表される.

$$P = \frac{H}{H+E} \tag{1}$$

本研究では面積, 冗長なヒントの二点を評価基準とする難易度を提案する. ヒント数を H, ヒント適用数を x, ヒントを適用した際の解候補数を C（x）とする (ヒント適用順序は解候補数減少量の大きいヒントから順に適用する). また, 冗長なヒント数を E とすると難易度 D は式 2 で表される.

$$D = \sum_{x=0}^{H-1} \log \frac{C(x)+C(x+1)}{2} \times P \tag{2}$$

5 問題自動生成アルゴリズム

5.1 問題生成

問題生成を行うにあたり, 本研究ではメタ戦略を用いた手法を提案する. メタ戦略に含まれるアルゴリズムは, (1) 過去の探索履歴を利用して新たな解を生成する, (2) 生成した解を評価し次の解の探索に必要な情報を取り出す, という操作の反復より成る. すなわち, 生成された解のどのような情報を探索履歴として記憶するか, 探索履歴をどのように利用して新たな解を生成するか, に対するアイデアの集合がメタ戦略である [7]. これは局所探索法の一般化とも言える. 本論文においてはパズルの生成を最適化問題と考え, 難易度を評価とすることで, 一つの問題に対して難易度のより低い, もしくはより高い問題を自動生成することを目的としている.

ここからはある問題から難易度を上昇させた問題を生成するアルゴリズム (Algorithm2) について説明する. まず問題の大きさを設定したのち, 初期解としてランダムなサイクル S を生成する.(Line1) その後 S に従ってすべてのマスにヒントを加えた問題を Sp とする (Line2). 次に Sp から任意の数ヒントを抜き出した問題を Sh とする ただし解が一意に定まる条件は満たし続けるようにヒントを抜き出す.(Line3). 次に Sh の難易度を評価する. (Line4) 次に S の一部に変更を加えたサイクルを近傍解 S ' とし (Line6), その S ' に対し, 難易度の評価までの処理を同様に行う (Line7). 難易度が S より高ければ, その近傍解を新たな解として受理する (Line8). この近傍生成と解の評価を終了条件まで繰り返すことで難易度のより高い問題を自動で生成することを可能にする. 難易度の低い問題を生成する場合は Line8 の不等号を逆にすればよい. 次に S に対する一部の変更, 近傍解生成の処理について説明する.

5.2 近傍解生成

Algorithm2 における近傍解について述べる. スリザーリンクではヒントの数字が同種, 同数あっても配置が異なる場合, 解は全く異なるものが出力される. 同じくヒントの配置が同じであっても数字が異なる場合解は異なる. また, ヒントの数字と配置が異なる場合であっても同じ解が出力される場合も存在する. 上記のような, ヒントの配置もしくは数字, 解の形状のいずれかを固定した場合, 解の探索範囲を狭めることになり解の大きな遷移が見込めない. そこで本研究では解の形状に着目し, 解の形状を一部変更したときのヒントの一部の変化を近傍解と定める.

Algorithm 2 問題自動生成アルゴリズム

```
1: (初期解生成) 初期解としてランダムにサイクル S を生成する.
2: S に従ってすべてのマスにヒントを加えた問題 Sp を生成する.
3: Sp から任意の数ヒントを抜き出した問題 Sh を生成する.
4: Sh の難易度を評価し D(Sh) とする.
5: for all (反復) 終了条件 do
6:    (近傍解生成) S の近傍解 S' を生成する.
7:    S' の難易度 S'h を評価し D(S'h) とする.
8:    if (探索法) D(S'h) > D(Sh) then
9:       S = S'
10:   end if
11: end for
```

Algorithm 3 近傍解生成アルゴリズム

```
1: S における辺の X% を使用する, 別のサイクル S' をランダムに一つ選択する
2: S' において, S のヒントの位置と数字が一致している箇所にヒントを加える
3: while true do
4:    空きマスに対し任意の数ヒントを追加する. ∗但しヒントの数字は解の形に準ずる
5:    if 解が一意に定まる then
6:       break
7:    else
8:       追加分のヒントを抜き出す
9:    end if
10: end while
```

解となるサイクル S に対する近傍解生成のアルゴリズムを Algorithm3 に示す. まず S における辺の X %を使用する, 別のサイクルをランダムに一つ生成し S ' とする (Line1). 辺の使用率が高いほど近傍解のサイクルが S に近くなる. 次にヒントの位置と数字が S と一致している箇所にヒントを加える (Line2). これにより S と同じサイクルの部分グラフを表すヒントの位置が S と同様になる. 最後に空きマスに対し任意の数だけヒントを追加する (Line3). ヒントの追加処理はランダムにマスを選択し, ヒントの数字の種類を, 解の形に準ずるようにしてそのマスに配置する. その際, 解が一意に定まるようにヒントを追加する必要があるが, もし定まらなかった場合は追加した分のヒントを抜き出し, 一意に定まるまでランダムに加える処理を繰り返す.X=80 の場合の一例を Fig5〜8 に示す.

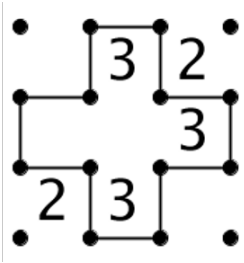

Fig. 5: Cycle [S]

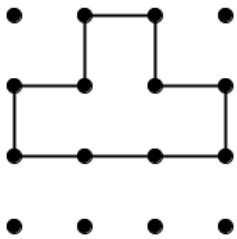

Fig. 6: [S'] Line1

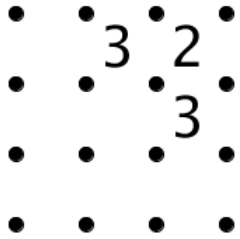

Fig. 7: Line2

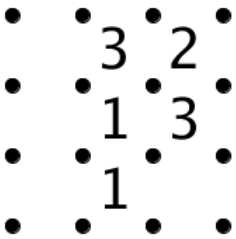

Fig. 8: Line3

6 実験

6.1 実験設定

3 章及び 5 章で述べた問題解答アルゴリズム, 問題生成アルゴリズムを用いて問題の自動生成を行う. 本研究の目的はメタ戦略による難易度を考慮した問題自動生成である. そのため問題の自動生成に加え難易度が正しく評価できているか検証する必要がある. 今回は問題の自動生成に加え, 難易度の妥当性に関する検証実験としてアンケートを行った. 実験設定は以下の通りである.

- 問題の大きさ : 頂点 6 × 6 (マス目 5 × 5)
- 使用するヒント数 : 全マス目の 50% (13 個)
- 探索法 : Simulated Annealing[7)]
- 終了条件 : 問題生成アルゴリズム 2000 ステップ
- 使用率 X(Algorithm7,Line1) : 80% , 60% の 2 種類

様々な問題を生成することを目的としてるため, SA の温度は高い状態をある程度保ったままゆるやかに下降するよう設定している. アンケートの概要 : 被験者にはランダムに生成された問題, その問題から難易度が簡単になるように生成された問題, 難しくなるように生成された問題の 3 種類を 1 セットとして 2 セット合計 6 問 (a〜f) を解いてもらう.1 セットごとに難易度の高い順に並び替えてもらい, 提案する難易度との比較を行う. また, 最も難易度の高いと感じた問題を選択してもらう.Fig19〜24 にアンケートに使用した問題を示す.Table1 にアンケートに使用した問題の難易度を示す.

6.2 実験結果

実験はランダムな初期解から易化させた場合と難化させた場合の 2 種類を 1 試行として, 10 試行分行った. 自動生成された問題 (Fig9〜14) については, 10 試行分の実験から典型的な例を 1 試行分ずつ抽出している.Fig9〜11 は近傍解生成における元の解の辺の使用率 80%, Fig12〜14 は近傍解生成における元の解の辺の使用率 60%の結果を示している. 問題の分析として, 難易度上昇問題生成について 1 試行で生成されたスリザーリンクの難易度の推移を X=80%, X=60%の場合でそれぞれ Fig15,Fig16 に示している. また, 本実験で生成されたスリザーリンクの中で最小及び最大難易度の問題を Fig17 に示している. アンケートの結果は Table2, Table3 に示している.

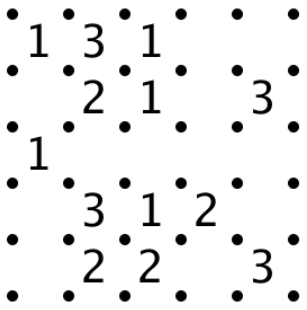

Fig. 9: The initial solution(D=19.97)

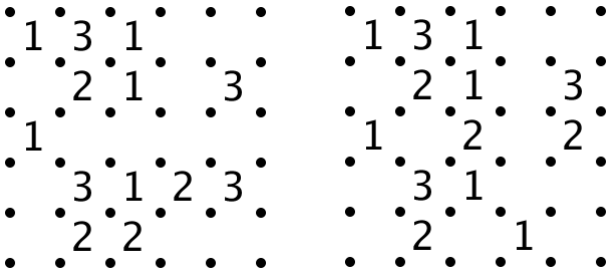

Fig. 10: Generated easy(D=17.83)

Fig. 11: Generated hard(D=21.58)

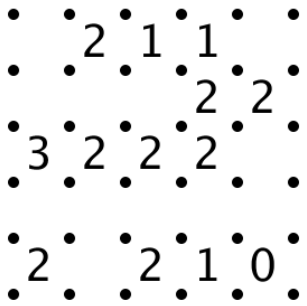

Fig. 12: The initial solution(D=22.07)

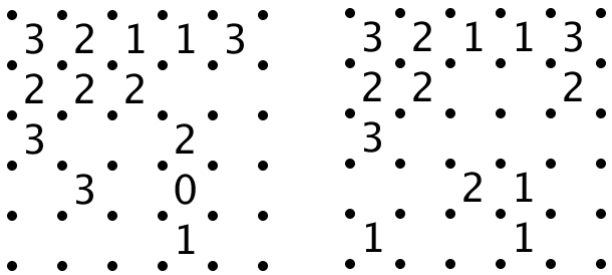

Fig. 13: Generated easy(D=15.51)

Fig. 14: Generated hard(D=28.06)

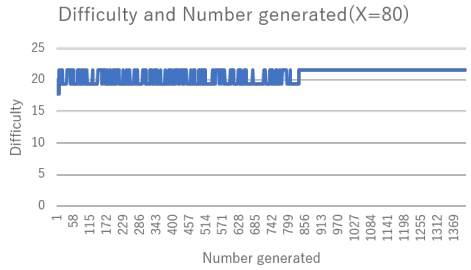

Fig. 15: Difficulty and Nunber generated (X=80)

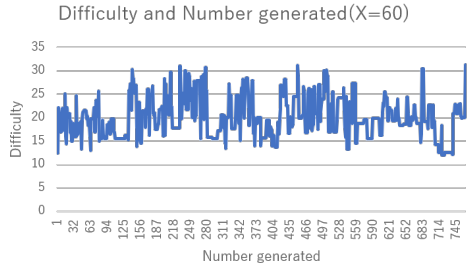

Fig. 16: Difficulty and Nunber generated (X=60)

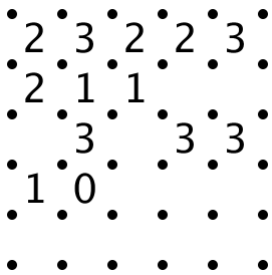

Fig. 17: Min difficulty puzzle(D=10.56)

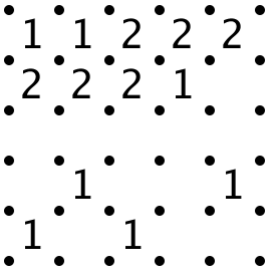

Fig. 18: Max difficulty puzzle(D=31.32)

6.3 考察

生成されたスリザーリンクの問題については. 難易度が上昇しても下降しても解の形が変化していない箇所, ヒントの数字と配置が変化していない箇所があることから近傍解の探索が正しく処理されていることがわかる. また, 解の形状が変化していないにもかかわらずその位置付近のヒントが増加している場合が存在しているが, これはヒント数を固定しているために起こった現象だと言える. 次に生成されたスリザーリンクの問題の分析だが. 近傍解の辺に対する元の解の辺の使用率が 80%の場合, ほとんど同じ難易度の解が生成されている. これは問題の大きさの関係上, 辺の使用率が高い場合に近傍解の類似性が高くなりすぎたことで, 局所解に陥る可能性が極めて高くなっていることが原因と考えられる. 一方, 近傍解の辺に対する元の解の辺の使用率が 60%の場合は, 解の形状がある程度変更されることで, 近傍解の類似性を保ちながらも局所解からの脱出が容易になっており, 様々な難易度の問題を生成できていると考えられる. 以上の結果から, 問題規模を変更することで, 近傍解の辺に対する元の解の辺の使用率の最適値も変更されることが予想される.

次に生成されたスリザーリンクの難易度についてだが, 難易度は最小で 10 程度の問題, 最大で 30 程度の問題を生成することができた. 数値上では約 3 倍の違いがみられる 2 問だが, これを筆者がそれぞれ解いてみたところ, 最低難易度の問題は仮定を置いて解くようなヒントの配置ではなく, また複数のヒントによってはじめて確定するような辺もないように思えた. つまり一つのヒントで確定する辺が存在し, そこから芋づる式に解ける問題であるように思えた. 一方の最大難易度の問題は仮定を置いて解くような箇所が中盤に存在し, スリザーリンクの定石を利用しても解けない箇所が存在した. つまり一つのヒントによって確定する辺が少なく, 複数のヒントを同時に考慮することによってはじめて確定するような辺が多い問題であるように思えた.

アンケートはランダムに生成された問題, その問題から難易度が簡単になるように生成された問題, 難しくなるように生成された問題の 3 種類を 1 セットとして 2 セット解いてもらった結果であるが, 1 セット目は 21 人中 15 人, 約 70%が提案している難易度評価と同様の難易度であると判断している. また 2 セット目に関しても同じく 21 人中 15 人, 約 70%が提案している難易度評価と同様の難易度であると判断している. またアンケートの中で最も難易度の高い (b) に関しては 21 人中 21 人が a, b, c の中で最も難しいと判断し, 全体の中で最も難しいと判断した人も 21 人中 13 人と半分以上の人が最も難しいと判断している.

これらの結果から本研究の提案している難易度評価はある程度有用であると考えられる. 人間が難易度を判定する上で判断材料となるのは, ヒントの適用によって確定する辺がない状況, つまり芋づる式に解けなくなり, 仮定を置く必要がある状況が何度訪れるのか, その回数とその状況に陥るまでにどれだけヒントの適用が進んでいるか, その進行状況であると予想していた. アンケート結果からそれらの判断材料と解の候補数という評価に相関がある可能性が示唆されていると言える.

7 まとめ

本研究ではペンシルパズルの一種であるスリザーリンクに対して, ZDD によって列挙可能な解の候補数の減少量と冗長なヒントの数を評価とした難易度を設定し, その難易度評価に基づいて, メタ戦略を用いてより難しい, もしくはより簡単な難易度の問題を自動で生成するアルゴリズムを提案し, そのアルゴリズムが問題生成において有用であることを示した. このアルゴリズムは難易度評価を工夫すれば他のパズルにも応用できると考えられる. ただ, ZDD を用いた解の候補数の減少量を評価とする手法は, 高速化しているとはいえ計算時間が大きく, 問題の規模が大きい場合には問題生成に非常に時間がかかってしまう.ZDD ではなくパズルごとの専門的な知識による難易度指標を与えれば, より高速に問題の自動生成ができるだろう.

参考文献

1) 土出 智也 , 真貝 寿明 : 数独パズルの難易度判定 - 解法ロジックを用いた数値化の提案- , 大阪工業大学紀要. 理工篇 56(1), 1/18, (2011)

2) 那須 律政 : モンテカルロ木探索による数独少数ヒント盤面の生成 , 第 54 回プログラミング・シンポジウム 173/180 , (2013)

3) 湊 真一： 超高速グラフ列挙アルゴリズム, 1/177, 森北出版, ,(2015)

4) Ryo Yoshinaka , Toshiki Saitoh , Jun Kawahara , Koji Tsuruma , Hiroaki Iwashita , Shin-ichi Minato : Finding All Solutions and Instances of Numberlink and Slitherlink by ZDDs , Algorithms (2012)

5) akeru Inoue, Hiroaki Iwashita, Jun Kawahara, and Shin-ichi Minato: "Graphillion: Software Library Designed for Very Large Sets of Labeled Graphs," International Journal on Software Tools for Technology Transfer, Springer, vol.18, issue 1, 57/66, (2016)

6) 株式会社ニコリ　スリザーリンクのおためし問題 http://www.nikoli.com/ja/puzzles/slitherlink/

7) 今堀 慎治 , 柳浦 睦憲 , 概説メタ戦略 , オペレーションズ・リサーチ 58(12), 695/702, (2013)

8) 白井 裕己 , 五十嵐 力 , 但馬 康宏 , 小谷 善行 : スリザーリンク解答システムと問題作成システム , ゲームプログラミングワークショップ 2006 論文集 , 32/39, (2006)

9) S. B. Akers, Binary decision diagrams : IEEE Transactions onComputers, vol. C-27, No. 6, p509-516, (1978)

10) 湊 真一：BDD/ZDD を基盤とする 離散構造と演算処理系の最近の展開 , 電子情報通信学会 基礎・境界ソサイエティ Fundamentals Review 2011 年 4 巻 3 号 ,224/230, (2011)

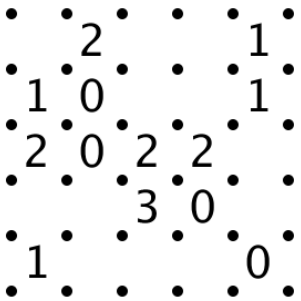

Fig. 19: puzzle (a) of questionnaire

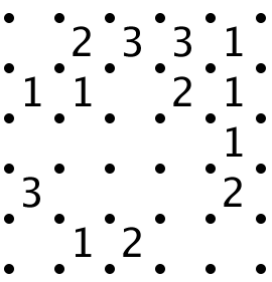

Fig. 20: puzzle (b) of questionnaire

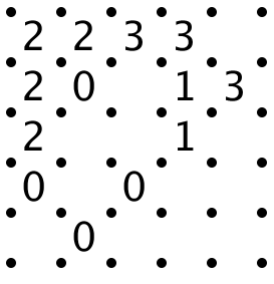

Fig. 21: puzzle (c) of questionnaire

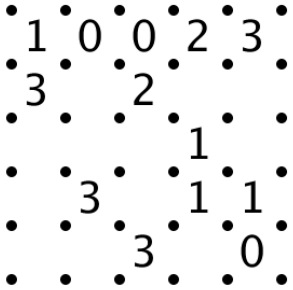

Fig. 22: puzzle (d) of questionnaire

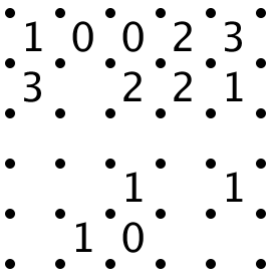

Fig. 23: puzzle (e) of questionnaire

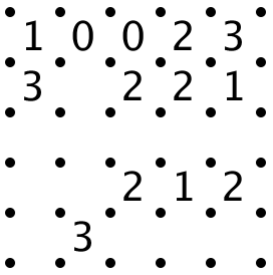

Fig. 24: puzzle (f) of questionnaire

Table 1: Puzzle of questionnaire

puzzle	difficulty	rank
a	18.23	4
b	28.07	1
c	15.84	5
d	20.67	3
e	12.38	6
f	25.77	2

Table 2: Results of questionnaire

puzzle rank	number of people (21)
b → a → c	15
b → c → a	6
f → d → e	15
f → e → d	4
d → f → e	2

Table 3: Results of questionnaire 2

most difficulty puzzle	number of people (21)
b	13
f	8