唯一解を持つ
拡張フィルパズルの自動生成

学生番号　2017381
氏名　吉田茉由
提出年度　2020 年度

目次

1. はじめに

本研究では，**拡張フィルパズル**を取り上げる．フィルパズル[1]では，四角形のマスから成る盤面が与えられ，またスタートとゴールのマスが指定される．そしてこのパズルでは，スタートからゴールまで一筆で全マスをなぞることが問われる．図 1 にパズルの例と解を示す．

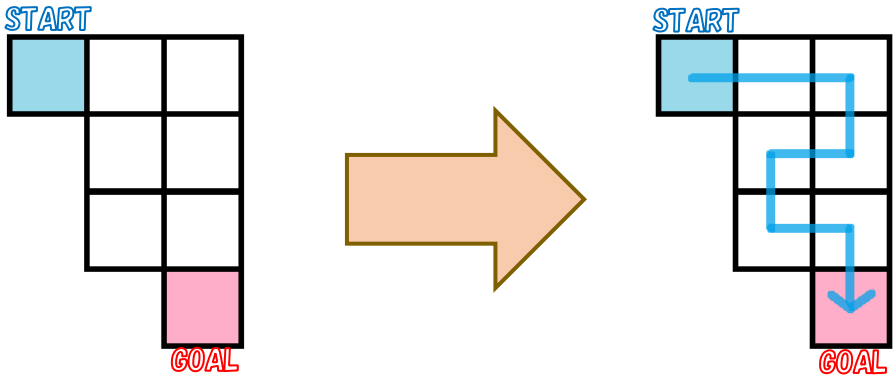


図 1 フィルパズルの問題例とその解

今日，パズルは娯楽のみならず，子供たちの教育に極めて有効なツールであるばかりか，大人の頭のトレーニングにも優れた効果を発揮[2]している．また，近年では認知症の予防にも良い効果をもたらすことが期待される[3]. このことから，幅広い分野で活用できる可能性が期待される．その中でもフィルパズルは，GooglePlay[4]などのアプリストアで多く取り扱われ，一筆書きといったわかりやすいルールで親しみやすい要素を持ったパズルである．しかし，雑誌やネットで手に入る問題には限りがあり，さらに素人が問題を作るのは容易ではない．そのため教育現場の子供たちの興味を引くようなパズルや，地域振興のためのパズルなど，様々な応用に展開するのが困難である．

そこで本研究では，唯一解を持つ拡張フィルパズルの自動生成アルゴリズムを提案する．これは，新しいパズルの問題を自動生成できるようにすることで，娯楽や教育の材料を多く世の中に提供できるようにすることが目的である．ここでパズルが唯一解でなければならない理由は，あてずっぽうでパズルを解くのではなく理詰めで解くために必要な条件だからである．実際，数独など多くのパズルでも解の唯一性が要求されている．また，拡張とはマスの形を任意にとることを許容することである．本来フィルパズルでは四角形のマスしか取り扱われてこなかった．本研究で提案するパズル生成アルゴリズムは，当初パズルを入力とし，このパズルを「修正」することで唯一解パズルを求め，出力する．当初パズルとは，パズルの案に相当するものである．すでに述べたとおり，任意の形をしたマスから成るパズルをアルゴリズムの入力とすることができる．ところがこの当初パズルは解を唯一持つとは限らない．そこでアルゴリズムは，解の唯一性の検証と，解が重複する場合には隣接するマス目を適当に選んで併合することを繰り返すことで，唯一解パズルを探索するのである．そして解の算出と，その唯一性の検証には**数理最適化**の技術を

用いる．

なおパズル生成に関する研究としての本研究の大きな特徴は，経路探索型パズルを取り扱っていることである．これまでクロスワードパズル[5,6]やビルディングパズル[7]など，盤面に値を割り当てるタイプのパズルに関する生成アルゴリズムの研究は知られているが，経路探索型パズルの生成は決して自明ではなく，未解決であった．本研究はそれを解決し，経路探索型パズルの自動生成を試みる最初の研究である．

本研究の実験では，それぞれマスの数・マスの隣接面の数・マスの図形（大きさと形）を変化させたパズルデータを用いて，作成したプログラムが正しく唯一解を導き出せるかを確かめた．その結果，実際に問題として扱えるような 30 マス程度のパズルではどのような図形でも正しく唯一解を求めることができ，計算時間もたかだか 2 秒～12 分程度と実用的なものとなった．また，大きく複雑な北海道の 179 市町村の地図を当初パズルとして用いた実験を行った結果，122 マスの唯一解フィルパズルを正しく求めることに成功し，大人数でパズルを楽しみ学べる新しい利用方法を発見した（図 14）．これらの実験から，提案手法は新たなフィルパズル問題を世の中に多く提供できる可能性を見出すことができた．

本論文の構成は以下のとおりである．まず，2 章で数理最適化について，3 章では問題生成アルゴリズムについて解説する．そして，4 章で実験を行い，5 章でまとめを行う．

2. 準備

2.1. 数理最適化問題

数理最適化問題（最適化問題）とは与えられた制約条件のもと，決められた目的関数の値を最小化，あるいは最大化する問題である[8]．最適化問題の一般形は以下のように表せる．

maximize　　f(x)
subject to　　$x \in F$

この時，関数 f を**目的関数**と呼び，変数 x が集合 F に含まれる条件を**制約条件**，またその集合 F を**実行可能領域**と呼ぶ．目的関数値を最大，あるいは最小にする実行可能解を**最適解**といい，その時の目的関数値を**最適値**という．

最適化問題の例として 0-1 整数最適化問題のナップサック問題を取り上げる．0-1 整数最適化問題とは，解の変数が取る値が 0 と 1 の二種類である問題である．ナップサック問題は，1,…,n の品物が与えられ，ナップサックの容量 W のうち，入れる品物の利得 $p_i(i=1,2,\ldots,n)$の総和を最大化させる品物の組み合わせを求める問題である．品物にはそれぞれ体積 w_i が与えられ，ナップサックの容量を超えて入れることはできない．

入力と出力は以下のようになる．

入力：n 個の品物の利得 $p_1,p_2,\ldots,p_n$ と体積 $w_1,w_2,\ldots,w_n$, およびナップサックの容量 W
出力：品物の体積の総和が W 以下で利得の総和が最大となるような品物の組み合わせ

下の例題を考える．

(ナップサックの容量 W = 7)

	1	2	3	4
利得 p_i	28	23	19	16
体積 w_i	5	4	3	2

このナップサック問題は，0-1 整数最適化問題として以下のように定式化される．

目的関数：$28x_1 + 23x_2 + 19x_3 + 16x_4$
制約条件：$5x_1 + 4x_2 + 3x_3 + 2x_4 \leqq 7$　… ①
　　$x_1,x_2,x_3,x_4 \in \{0,1\}$　… ②

制約条件①はナップサックの容量 7 を品物の体積の総和が超えないようにするためのもので，制約条件②は x_i は 0 と 1 の二つの値しかとらない，すなわちナップサックに「入れる」か「入れな

い」かを示している．

解の総数は $2^4 = 16$ である。このうち最適解は$(x_1,x_2,x_3,x_4) = (1,0,0,1)$で，最適値 $28 + 16 = 44$ となる．

一般に品物が n 個あるナップサック問題の解の総数は 2^n で，たとえば $n \geqq 50$ で $2^n \geqq 10^{15}$ となり，最新のスパコンを持ってしても，網羅的探索によって実用的な時間内に最適解を見つけ出すのは困難である．このことはナップサック問題に限らず，多くの複雑な組み合わせ最適化問題についても同様である．整数最適化の技術はこの網羅的探索を可能な限り効率的に行うものである．現在の最先端の**ソルバ**（ソフトウェア）を用いれば，数百～数千変数程度の 0-1 整数最適化問題であれば実用的な時間で最適解を求めることができるとされる．

2.2. グラフ

本研究ではフィルパズルを**グラフ**として取り扱う．

グラフとは，**頂点**の集合と，頂点と頂点を結ぶ**辺**の集合の対から成るものである．グラフ G の頂点集合を V(G)，辺集合を E(G)とする．そしてフィルパズルを，マス目を頂点，隣接する頂点同士を辺で結んだグラフとみなす．グラフ G における**ハミルトンパス**とは，すべての頂点をちょうど 1 度ずつ訪れる路である．

グラフ G における辺 $e = vv' \in E(G)$の**縮約**とは隣接する 2 つの頂点 v,v'を一つの頂点 u に統合し，別のグラフ G'を得ることを言う．すなわち

$$V(G') = (V(G) \cup \{u\}) - \{v,v'\},$$

$$E(G') = (E(G) \cup \{vr \in E(G)\} \cup \{v'r' \in E(G)\}) - \{vv'\}$$

このとき G'=G/e と書く．また，グラフ G における頂点あるいは辺の削除，もしくは辺の縮約によって得られるグラフを G の**マイナー**と言う．

（拡張）フィルパズルとは，グラフが与えられ，かつ 2 つの頂点 s,t が指定されたとき，それらを始点と終点として持つようなハミルトンパスを問う問題である．また，最長パスとは，与えられたグラフの始点 s から終点 t まで最も多く頂点を通る路のことである．明らかにフィルパズルが解をもつとき，かつその時に限り，最長パスはハミルトンパスである．

図 2 にフィルパズルをどのようにグラフとして取り扱うか，また辺の縮約の例を示す．

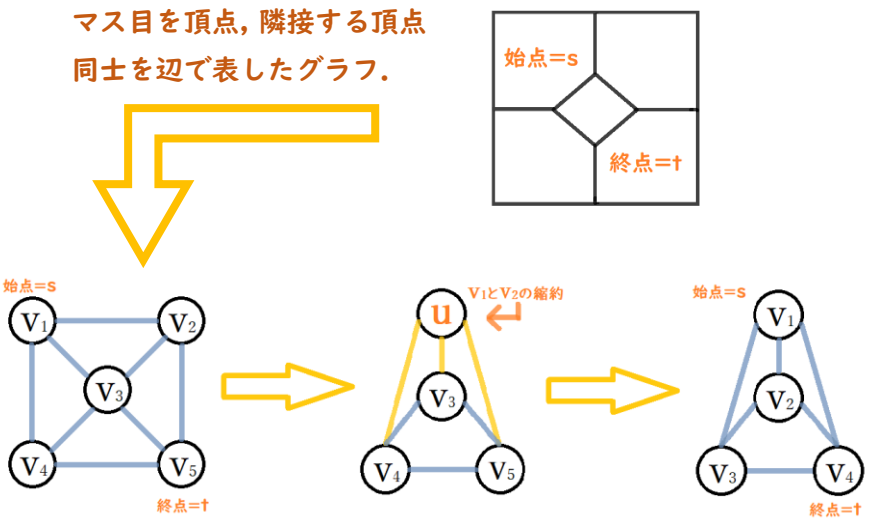


図 2 フィルパズルのグラフへの変換，および縮約の例

3. 問題生成アルゴリズム

3.1. 概要

本章では拡張フィルパズルの問題生成アルゴリズムについて解説する．まず，このアルゴリズムの概略を示す.

入力：グラフ G（当初グラフ），頂点 s,t∈V(G)
出力：グラフ G における最長パスのマイナーを唯一の最長パスとして持つ G のマイナー

1.当初グラフデータを入力.
2.当初グラフに関する**最長パス問題**を解き、当初解を求める.
3.2 で求めた当初解を最長パス**唯一性判定問題**として解き，唯一解であれば解を出力して終了.
4.もし唯一解でなければ当初解と重複解を比較し，当初解が連続して通り，重複解が連続して通らないような 2 つの頂点を併合して縮約する．これによって縮約後のグラフでは重複解（のマイナー）はハミルトンパスではなくなる．この当初解のマイナーが唯一解となるまで縮約を繰り返す.

1.当初グラフデータを入力.

頂点の数とその頂点の隣接関係，すなわち辺を数値データとして入力する．まず，当初グラフの各頂点に番号を振っておく．そして，当初グラフデータを入力する際，各頂点に振られた番号の最小値 1 がスタート地点となり，最大値をゴール地点とする．図 4 の当初グラフに対応する隣接行列は図 3 の通りである．マスの数 10，各マスの隣接関係【マス 1 に隣接するマス:2，4】【マス 2 に隣接するマス:1，3，5】…【マス 10 に隣接するマス:9】を入力する．この隣接行列を下に示す.

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}
v_1	×	1	0	1	0	0	0	0	0	0
v_2	1	×	1	0	1	0	0	0	0	0
v_3	0	1	×	0	0	1	0	0	0	0
v_4	1	0	0	×	1	0	1	0	0	0
v_5	0	1	0	1	×	1	0	1	0	0
v_6	0	0	1	0	1	×	0	0	1	0
v_7	0	0	0	1	0	0	×	1	0	0
v_8	0	0	0	0	1	0	1	×	1	0
v_9	0	0	0	0	0	1	0	1	×	1
v_{10}	0	0	0	0	0	0	0	0	1	×

図 3 例題の隣接行列

2.当初グラフに関する最長パス問題を解き，当初解を求める．

読み込んだ当初グラフに関する最長パス問題として，始点 v_1=s から終点 v_{10}=t まで最も多くの頂点を通過する路を解き，当初解を求める．図 4 の例で求めた場合，当初解は図 5 のように求められた．なお最長パス問題をどのように解くかは 3.2 で解説する．

3.当初解の他に重複解を求める．

求めた当初解がすべての頂点を通っていなければ，通らなかった頂点とその頂点に接続する辺を削除，また，当初解以外に重複解があるかを唯一性判定問題として解くことで求め，なければ現在のパズルを出力して終了する．例題からは図 6 のような重複解が求められた．なお唯一性判定問題は 3.3 で説明する．

4.当初解と重複解を比較し，当初解が連続して通り重複解が連続して通らない 2 つの頂点を縮約する．

当初解と重複解を比較し，当初解が連続して通り重複解が連続して通らない 2 つの頂点が存在した場合，その頂点同士を併合し，縮約する．縮約後は再度唯一性判定問題として解き，当初解のマイナーが唯一解となるまで縮約を繰り返す．

例題の当初解（図 5）と異なるルートのマスを縮約する．図 5 では 1→4→7→8→5→2→3→6→9→10，図 6 では 1→2→3→6→5→4→7→8→9→10 というパスであることから，当初解が連続して通り，重複解が連続して通らない 2 つのマスの 1, 4 を併合する（つまりグラフ上では対応する辺を縮約する）．唯一解となるまで縮約を繰り返した結果，最終的に求められた唯一解フィルパズルは図 7 である．

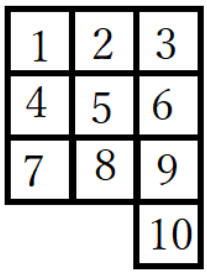


図 4 当初盤面例

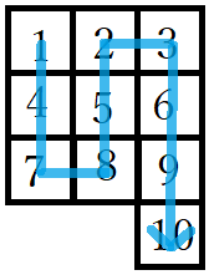


図 5 例題の当初解

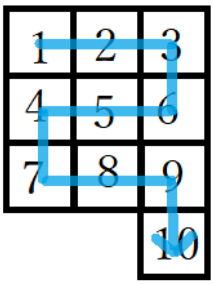


図 6 重複解の例

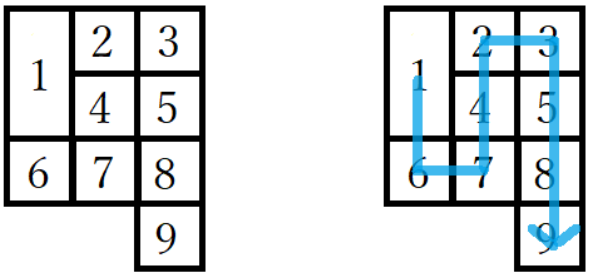


図 7 （左）削除後のパズル/（右）求められた唯一解パズル

3.2. 最長パス問題

最長パス問題とは，与えられた当初パズルのスタートとゴールまでのパスで最も長いパスを問う問題である．

最長パス決定に 2 章で述べた数理最適化を用いて解を求める．

以下，0-1 変数を任意の辺$(v_i,v_j)\in E(G)$について，

$$x_{ij} = \begin{cases} 1 \text{ 辺}(v_i, v_j)\text{は解に含まれる.} \\ 0 \text{ 辺}(v_i, v_j)\text{は解に含まれない.} \end{cases}$$

任意の頂点 $v_i\in V(G)$について，

$$y_i = \begin{cases} 1 \text{ 辺}v_i\text{は解に含まれる.} \\ 0 \text{ 辺}v_i\text{は解に含まれない.} \end{cases}$$

と定め，任意の頂点 $v_i\in V(G)$ について，ポテンシャル変数 z_i を導入する．この z_i は部分巡回路除去制約(後述)に用いるもので，実数変数である．

入力はファイルから頂点の数 n，頂点の隣接リスト A を読み込む．パズルデータは各頂点に 1,2,3,…,n と番号が振られており，始点は 1，終点は n（最大値）と定める．

目的関数:解が通過する頂点の個数 Y

制約条件:

（1）始点から出る辺は 1 本

（2）頂点 v_i に入る辺の本数は y_i に等しい

（3）頂点 v_i から出る辺の本数は y_i に等しい

（4）終点に入る辺は 1 本

（5）部分巡回路除去

目的関数は解が通過する頂点の個数で，制約条件は以下にそれぞれ解説していく．例として，概要の図 4 の例題を用いる．

まず，(1) は始点から出る辺のうちちょうど 1 本を選ばなければならないことを要請するものである．これは必ず始点から始めるためにある条件であり，図 4 の例題では，

$$x_{12} + x_{14} = 1$$

と表される．

また，制約条件 (2)，(3) は，始点 v_1 および終点 v_n を除くすべての頂点 $v_i(i \in \{1,...,n-1\})$において，

-頂点 v_i を通過する場合 ($y_i=1$)：v_i に入る辺の内ちょうど 1 本，v_i から出る辺のちょうど 1 本を選ばなければならないことを要請するものである．

-頂点 v_i を通過しない場合 ($y_i=0$)：v_i に入る辺，v_i から出る辺のいずれも選んではならないことを要請するものである．

図 4 の例題のマス番号 2 で示すと，

$$x_{12} + x_{32} + x_{52} = y_2$$

$$x_{21} + x_{23} + x_{25} = y_2$$

と表せられる．

(4) は終点 v_n に入る辺のうちちょうど 1 本を選ばなければならないことを要請するもので，例題の終点 v_{10} より示すと，

$$x_{9,10} = 1$$

となる．

(5) は部分巡回路除去のための制約で，巡回セールスマン問題において部分巡回路を生成しないためのポテンシャル制約 (MTZ 制約) [9]の応用である．これは $z_i - z_j + nx_{ij} \leqq n-1$ と表され，もし解が辺(v_i,v_j)を通過するのであれば$(x_{ij}=1)$，$z_j \geqq z_i+1$ にならなければならないことを要請するものである．また，もしすべての頂点が解に含まれるのであれば，$z_1 = 1, z_2 = 2,...,z_n = n$ となる．

3.3. 解の唯一性判定問題

解の唯一性判定問題には最長パスを問う問題の定式化に下に示した制約条件を付加して解を求めている．これは，当初解を実行可能領域から取り除く制約である．

$$\sum_{\{v_i,v_j\} \in H} x_{ij} \leq |H| - 1$$

H は当初解のパスである．この問題の最適値が |H| と等しい場合は，G_t (入力した当初グラ

フデータ G から H に含まれない頂点およびそれに接続する辺を削除して得られるグラフ）は複数の解を持つと結論づけることができる．つまりは，G_t において当初解 H に属している頂点を結ぶ辺をすべて足し合わせたときに，当初解の辺の総数$|H|-1$ よりも多ければ重複解があると判断され，それ以下であれば当初解が唯一の最長パスであると判断できるのである．この時 $y_i=1$ となるような頂点 v_i の集合，および $x_{ij}=1$ となるような辺(v_i,v_j)の集合が G_t における解 $H'\neq H$ を構成する．

3.4. 問題例の更新

概要でも述べたように，唯一解判定をするにあたり，当初解の解と異なる重複解が存在するのかどうかを求めた後，重複解があれば比較し，その解が唯一解になるまで縮約を繰り返し問題を更新する．ここでは例題（図 8）を用いて解説していく．

図 8 は 10 マスの正六角形で構成された当初パズルである．そして図 9（1）はこのパズルの当初解である．

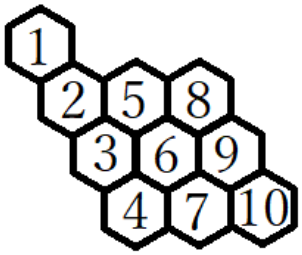
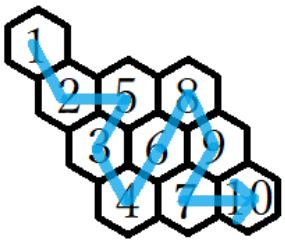


図 8 唯一性決定問題例題　当初パズル例

まず，このパズルに関する唯一性判定問題を解き，当初解のマイナーの他に重複解が存在するのかを調べる．図 8 の例題で求めた当初解のマイナー（図 9（1））は 1→2→5→3→4→6→8→9→7→10 となったが重複解（図 9（2））を求めると 1→2→3→4→6→5→8→9→7→10 となった．この 2 つを比較すると，3 番目に訪れる頂点でパスが別れた．そこで次に縮約を行う．当初解が連続して通り重複解が連続して通らない 2 つの頂点は 2 と 5 であるので，その 2 つの頂点を併合して一つの頂点とし，辺の繋がりも併合する（図 9（3））．これを行うことで重複解を一つ減らすことができ，唯一解に近づく．この当初解のマイナーが唯一解となるまで再度マス番号を振り最適化と縮約を繰り返していく．

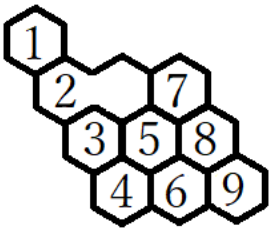
再び最適化を行った当初解のマイナーは 1→2→3→4→5→7→8→6→9 となった（図 9（4））．重複解を求めると 1→2→3→4→6→5→7→8→9 となり 5 番目に訪れる頂点で再びルートが別れたので，当初解が連続して通り重複解が連続して通らない 4 と 5 を縮約する．再度最適化を行う．

次に求めた当初解のマイナーは 1→2→3→4→6→7→5→8 で，重複解はなくなった．これで解の唯一性を導き出せたこととなる．最終的に得られる唯一解パズルは図 9（7）に示したものとなる．

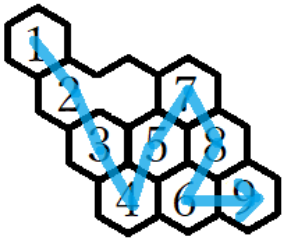


（1）当初解

（2）重複解 1

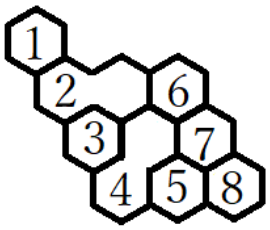


（3）更新したパズル 1



（4）当初解のマイナー

（5）重複解 2



（6）更新したパズル 2

（7）求められた唯一解パズル

図 9 問題例の更新例

4. 計算実験

フィルパズルのグラフデータを入力することで唯一解を持つフィルパズルの解を生成し，問題生成にかかった時間と隣接関係を表示するプログラムを Python[10]で作成し，実験を行った．また，数理最適化問題を解くにあたって，pulp ライブラリ[11]を用いている．作成したプログラムの全行数は約 340 行である．4.1 では計算時間に関する実験，4.2 では応用可能性に関する実験を行う．

以下，使用したコンピュータはクロック周波数:2.50GHz,メインメモリサイズ:8.00GB である．また，各計算時間は 3 回の計算の平均を取っている．

4.1. 計算時間に関する実験

この実験ではマスとマスの数をそれぞれ変えて，計算時間を求めた．それぞれ正方形/正五角形/正六角形で 10・20・30 個のマス目を用いた当初パズルで問題を生成した．各当初パズルはランダムに作る．これを図 10 に示す．

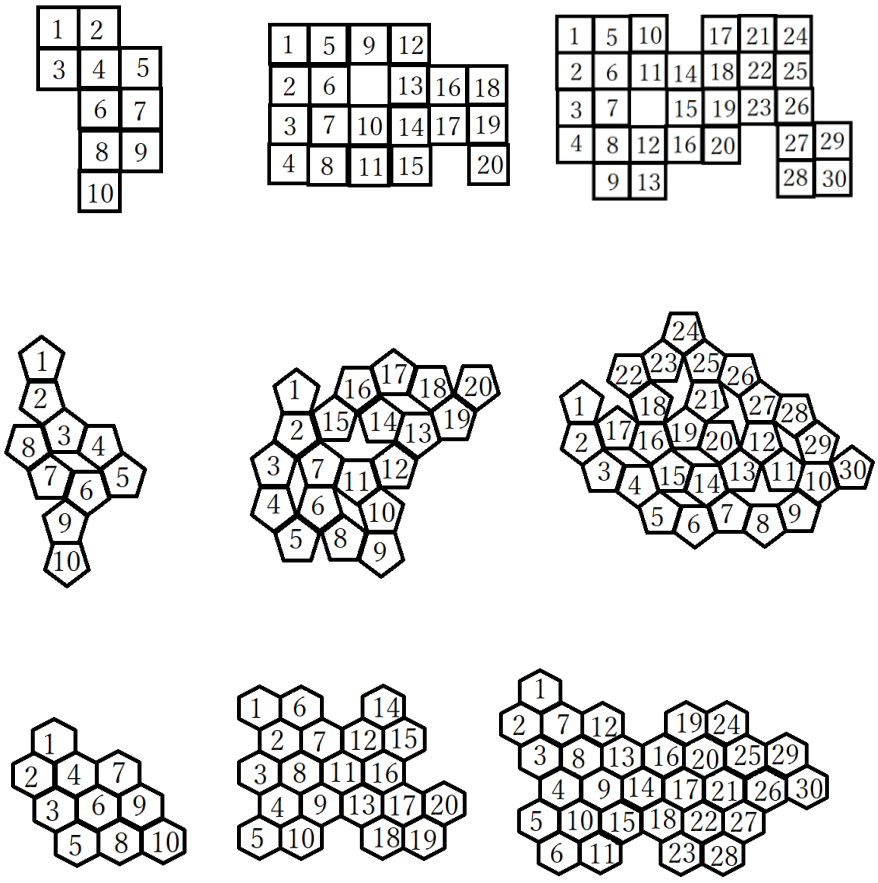


図 10 実験に用いた当初パズル
（上から）正方形/正五角形/正六角形/（左から）10 マス/20 マス/30 マス

各盤面から求められた唯一解パズルを図 11 に，唯一解パズルを求めるのにかかった計算時間を表 1 に，出力した唯一解パズルのマス数を表 2 に示した．

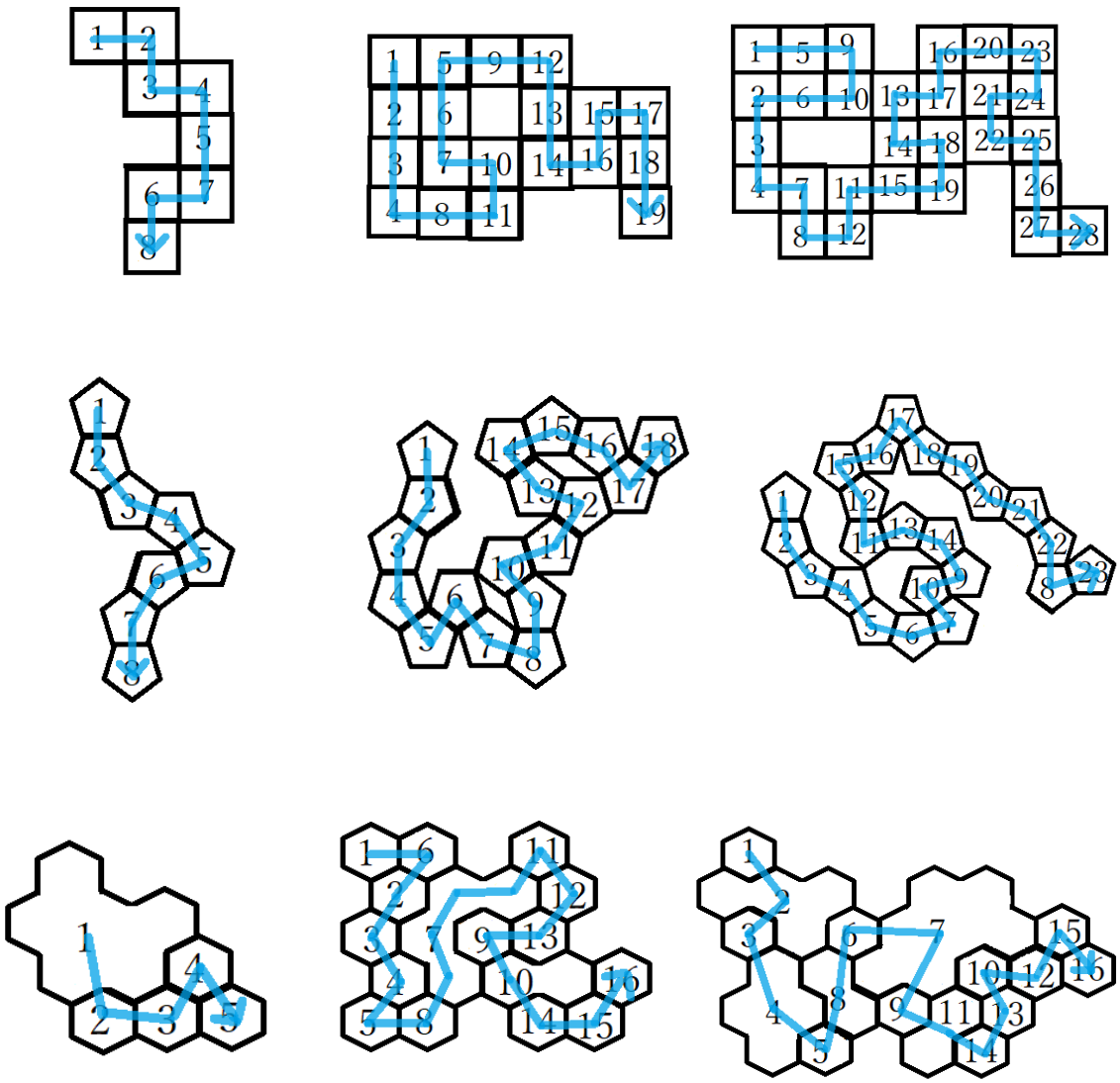


図 11 各唯一解パズル
（上から）正方形/正五角形/正六角形
（左から）10 マス/20 マス/30 マス

表 1 各フィルパズルの生成時間

生成時間(秒)		*入力したマスの数*		
		10 マス	**20 マス**	**30 マス**
図形	**正方形**	0．63	1．798	2．631
	正五角形	0．686	0．821	1．85
	正六角形	2．065	4．511	40．817

表 2 出力された唯一解パズルにおけるマスの数

出力された唯一解パズルにおけるマスの数		*入力したマスの数*		
		10 マス	**20 マス**	**30 マス**
図形	**正方形**	**8**	**19**	**28**
	正五角形	**8**	**18**	**28**
	正六角形	**5**	**16**	**16**

実験の結果，どのパズルも正確に唯一解フィルパズルを生成することに成功し，ほとんどのパズルが数秒ほどで生成され，どのパズルもかなりの早さで唯一解フィルパズルを生成できるということがわかった．また，出力した唯一解パズルのマスの数は約 0.5～5 割程度の減少が見られたが，平均して約 2 割程度の減少に収まった．マスの各図形で比較すると，まず，正方形ではマスの数が 10 増えるごとに約 1 秒程度の問題生成時間の増加が見られた．さらに正五角形・正六角形でも同じようにマスの数が増えるごとに問題生成時間が少しだけ増加した．このことから，マス目の増加によって計算時間が増加するが，非実用的となるまでに時間は増加していないことが確認できた．そして，特に著しい結果となったのは 30 マスの正六角形で，他の多角形に比べて時間がかかり，問題生成時間は約 40 秒となった．これは当初パズルが 30 マスであるにも関わらず，出力された唯一解パズルが 16 マスしか持たないことから，唯一解判定を 14 回，他の事例よりも多く行ったことに起因すると考えられる．しかしこのようなケースにおいても 1 分もかからずに唯一解フィルパズル問題を生成することに成功している点から，実用の可能性は大いにあると考えられる．

次にマスの図形と大きさをランダムに混ぜてパズルデータを入力し，正しく唯一解フィルパズルが求められるかを実験する．

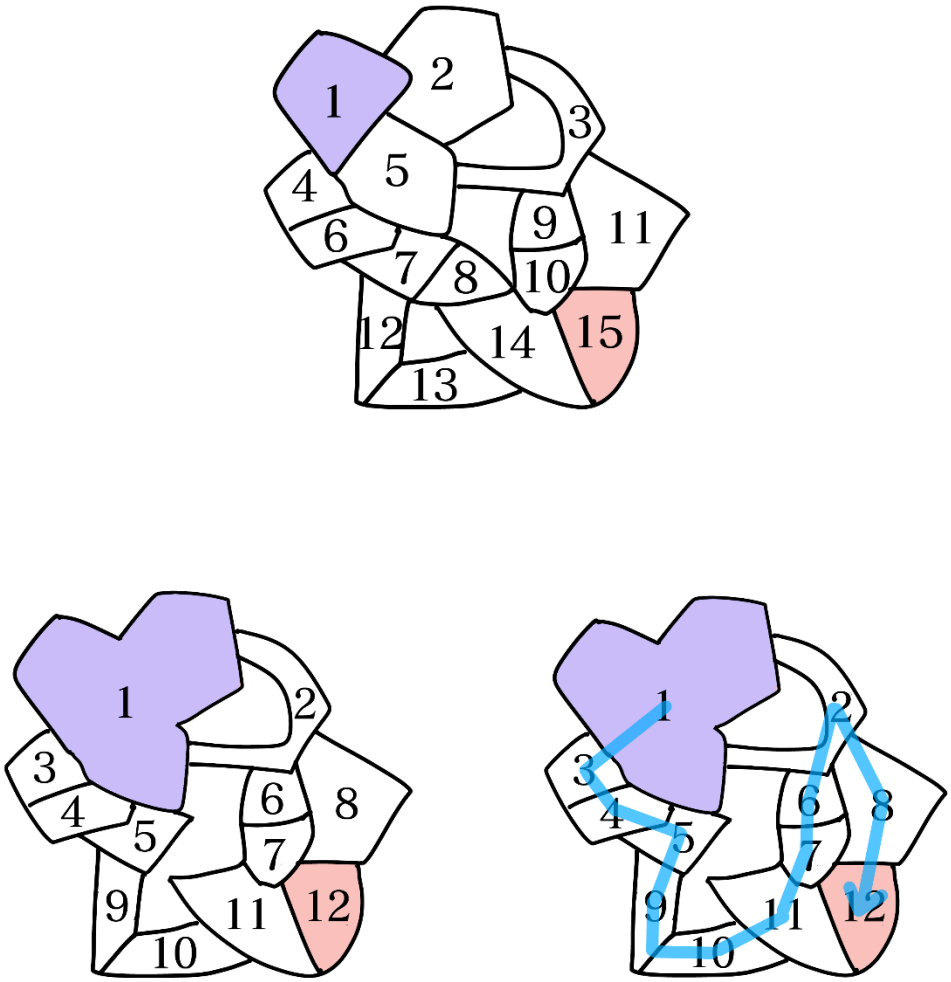


図 12 ランダムな図形と大きさのマスで構成したパズル
（上）当初盤面/
（左下）更新された当初解のマイナー/（右下）求められた唯一解フィルパズル

図 12 の上図はランダムな形と大きさの図形で作った当初盤面，左下図は更新された当初解のマイナー，右下図は生成された唯一解フィルパズルである．計算の結果，当初解に含まれなかったマス 8 は削除され，マス 2 とマス 5 がマス 1 に併合された．この結果から，このプログラムは図形や大きさに囚われずに，どのような形でも正しく唯一解フィルパズルを求められるということが確かめられた．また，生成時間は約 3.07 秒とかなりの速さで求めることができた．このような一般的なサイズ（フィルパズルの場合，マス目の数は 5～40 程度）の問題であれば，どんな形のパズルでも比較的短時間で問題が生成できることから，人の手間を省くだけでなく多くの種類の問題を提供できると期待される．

4.2. 応用可能性に関する実験

では実際に娯楽や教育においてパズル問題として扱えるのか，イラストを用いてフィルパズルを作ってみる．

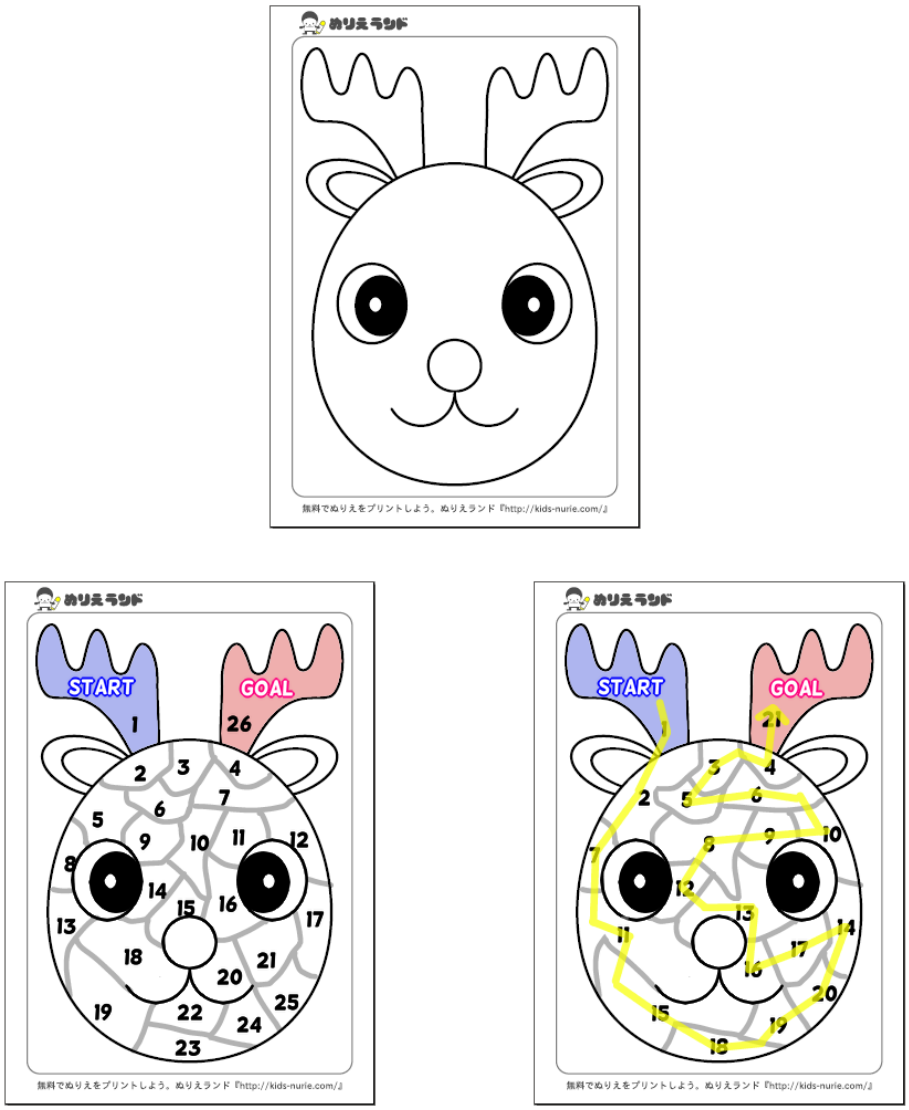


図 13 イラストを用いたフィルパズル実験例
（上）当初パズル/（左下）任意にマス目を配置/（右下）求められた唯一解パズル

図 14（上）に示したトナカイのイラスト[12]を用いて，26 マスのパズル盤面を作成した．なお，目・鼻・口などの元のイラストの境界線は跨げないこととする（縮約によって元のイラストの線が消えてしまう可能性があるため）．プログラムを実行したところ，21 マスの唯一解フィルパズルを求めることができ，計算時間は約 15.85 秒となった．この結果から，イラストを用いてマスを分割することでも唯一解フィルパズルを求められるとわかり，イラストや図形を用いた娯楽や教育向けの多様な種類のパズルを生成できるという利点を発見できた．

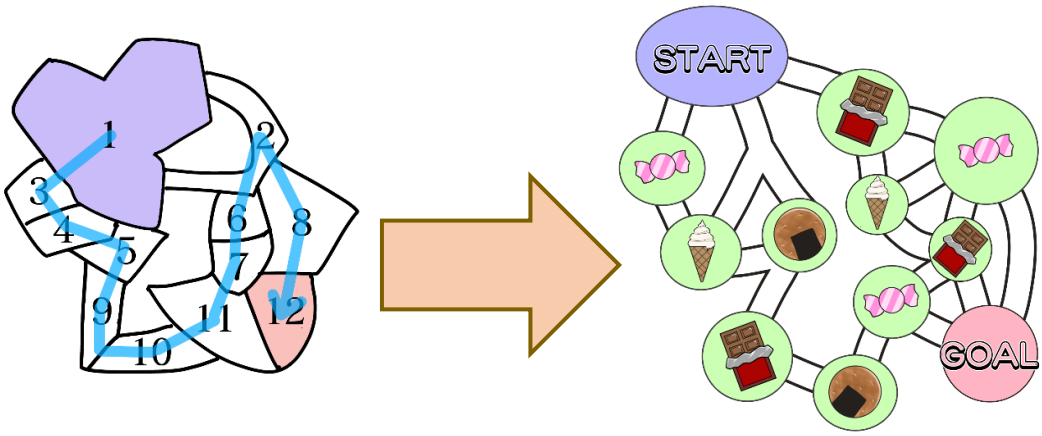


図 14 唯一解フィルパズルのパズル問題作成例

さらに，図 12 で求めた唯一解フィルパズルを応用し，頂点と辺の関係を用いて唯一解フィルパズル迷路を作成した（図 14）．このパズルは，スタートからすべてのおやつを一度ずつ通ってゴールにたどり着くパズルである．このように，マスの盤面だけでなく迷路といったフィルパズルも作成可能であることから，種類の豊富なフィルパズル問題を提供できると期待でき，実際に娯楽や教育の現場での使用が望める結果となった．

以上の実験から，様々な組み合わせの図形でも必ず唯一解を求められることがわかったため，さらにマスを多くし，より複雑にしたパズルでも正しく唯一解を求められるか，また，計算時間は実用的であるかを確かめるべく，北海道の地図[13]（離島を除く）を用いて北海道最南端の松前町から最北端の稚内市までのパスを求めた．

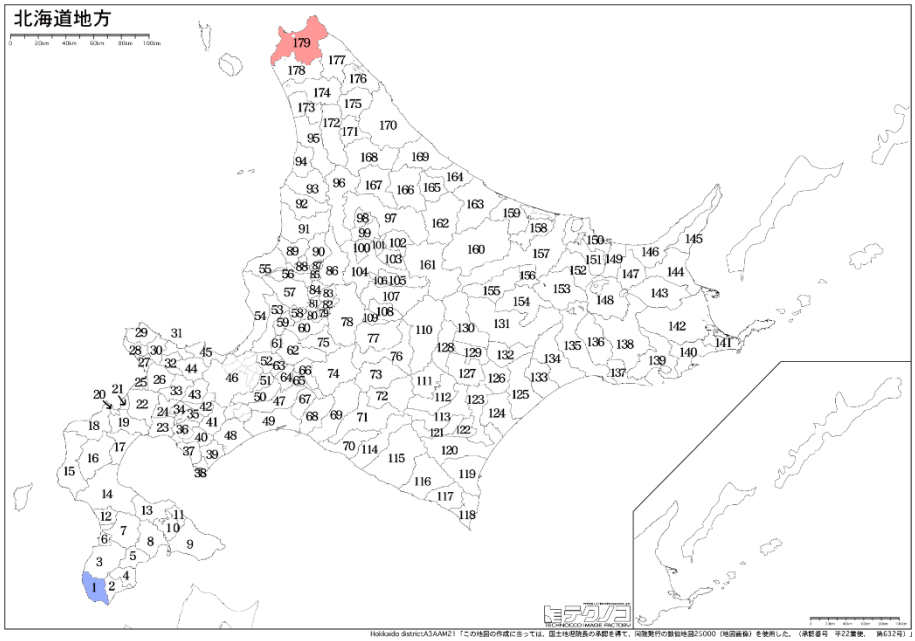


図 15 北海道の市町村の当初盤面

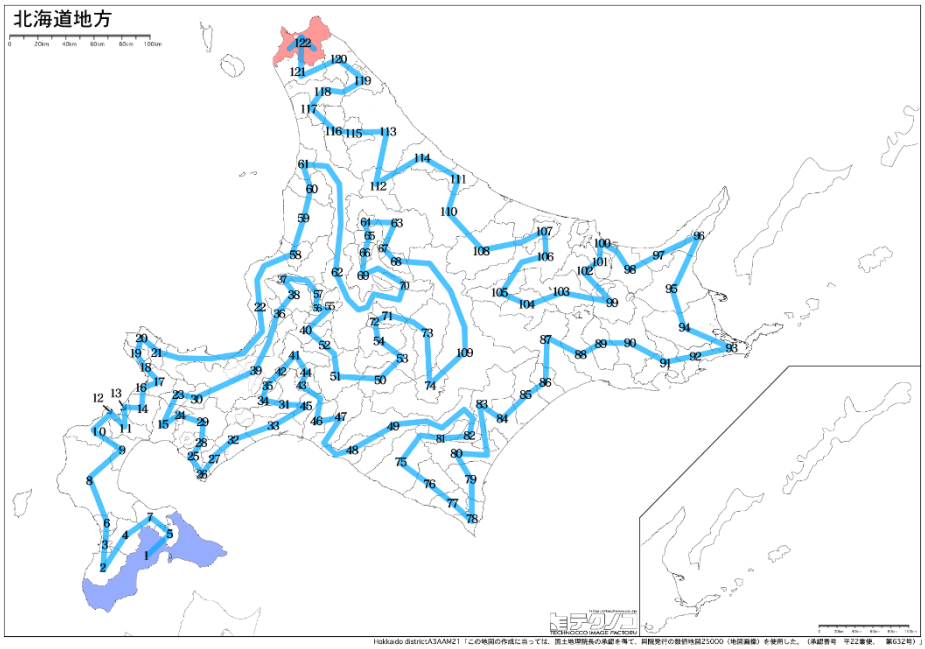


図 16 北海道の市町村パズルから求められた唯一解フィルパズル

この実験ではマスの数が多くかなりの時間がかかると予想されたことから，使用機器を変更し，MacBook Pro （macOS Catalina 10.15.7，プロセッサ：2.8GHz QuadCore Intel Core i7，メモリ：16GB 2133MHz LPDDR3）で IBM ILOG CPLEX version 12.8 を用いて行った．

実験の結果，図 16 のようなパズルを求めることができた．計算時間は約 319.822 秒，マスの数は 179 から 122 になった．この結果から，このプログラムは 100 以上の複雑なマス目でも，5 分程度の短い計算時間で正しく唯一解フィルパズルを求められると確かめられた．このような大きく複雑なパズルでは問題として一人で解くことは困難であるが，なぞったマスの数での対戦やチーム戦，世界大会などに応用が可能で，今までにない新たなパズルの楽しみが増えることが期待される．

5. まとめ

本論文では唯一解を持つ拡張フィルパズルを自動生成するアルゴリズムを開発し，プログラムを行った．そして，Python プログラムを用いての自動生成実験を行った結果，どんな図形で構成されたパズルでも正確に唯一解フィルパズルを求めることができるとわかり，娯楽や教育においてより多くのフィルパズル問題を世の中に提供できる可能性を見出せた．また，今までにない複雑で大きな唯一解フィルパズルを生成することに成功し，チーム戦や世界大会などフィルパズルの新しい使い方が望まれる結果となった．もしこのプログラムが実際に用いられたとするならば，ゲームや雑誌などの娯楽では，イラストや地図または迷路などを用いた豊富な種類のフィルパズルで新しい面白さを提供でき，教育や脳トレにおいては，より複雑になったフィルパズルで頭を鍛えることができると期待される．さらに，その唯一解フィルパズルは自動で生成されることから，人の手間を減らしながらもより多くの種類の問題を手にすることができるという利点がある．この結果から，この唯一解フィルパズルの自動生成プログラムはかなりの実用性があるとわかった．一方，難易度の調整ができないことや，手入力によるミス，手入力と使用機器によって時間がかかってしまうことなどの問題がある．よって，今後の課題として，難易度調整，データの画像入力とパズル画像の出力，計算時間の短縮などが挙げられる．

謝辞

本論文は，多くの方々からのご協力を受け執筆することができました．ご指導いただいた先生，学生論文賞の審査・評価をしてくださった先生方に心より感謝の意を表します．

参考文献

[1] Fill one-line puzzle game on the App Store : https://apps.apple.com/gb/app/fill-one-line-puzzle-game/id1091456207 (2020 年 11 月 8 日閲覧)
[2]東田大志/東大・京大パズル研究会：〈東大・京大式〉頭がはっきりするパズル, 文藝春秋, 2013.
[3]佐古田三郎《国立病院機構刀根山名誉院長》: 認知症を予防する！「ひと筆書き」ドリル 100, 2018.
[4] ゲーム -Google Play の Android アプリ : https://play.google.com/store/apps/category/GAME?hl=ja&gl=US(2020 年 12 月 11 日)
[5]佐藤潤一, クロスワードパズル生成問題の新しい定式化, 小樽商科大学 商学部 卒業論文, 2017.
[6]西島善治, 反復局所探索法に基づいたスケルトンパズルの生成アルゴリズム, 小樽商科大学 商学部 卒業論文，2018.
[7]田中良弥，ビルディングパズルの自動生成，小樽商科大学 商学部 卒業論文，2017.
[8]大堀隆文・加地太一・穴沢務：例題で学ぶ OR 入門，コロナ社，2017.
[9]ポテンシャル制約（MTZ 制約）：C.E. Miller, A.W. Tucker and R.A. Zemlin: Integer programming formulations and travelling salesman problems: Journal of the ACM, Vol.7, pp. 326-329 (1960)
[10] Python：https://www.python.org/(2020 年 11 月 6 日閲覧)
[11] Pulp ライブラリ：https://pypi.org/search/?q=pulp(2020 年 11 月 6 日閲覧)
[12]ぬりえランド・無料ぬりえ 600 枚以上：kids-nurie.com(2020 年 12 月 8 日閲覧)
[13]北海道の白地図：https://technocco.jp/n_map/hokkaidoarea.html(2020 年 11 月 8 日閲覧)