Vol.2017-HCI-172 No.26
2017/3/7

抽象化とメディア変換に基づく ネットワーク型パズルの提案と可視化

保里 和樹[1,a)]　細部 博史[2,b)]

概要：パズルには多くの種類があり，昔から多くの人に楽しまれている．特に近年はスマートフォンやタブレット上でパズルを楽しむ人が増えている．しかし，既存のパズルは多様なルールがあるにもかかわらず，限られた形状をしていることが多い．パズルの形状を大きく変化させる研究として，杉山らは，既存のパズルは抽象化を経て他のメディアに変換できるとする，抽象化とメディア変換と呼ばれる体系的アプローチを提案した．本研究では，抽象化とメディア変換を杉山らの研究とは異なる種類のパズルに適用することで，より幅広い形状を可能とした新しいパズルを提案する．具体的には，多様なパズルが二次元の格子状であることに着目し，各マスをノード，となりあうマスの関係をエッジとみることで，格子状のパズルをネットワーク，すなわち，グラフとして一般化して表現する．格子状のパズルとして，本研究では特に加算パズルを扱う．さらに，グラフ可視化技術と複数の列の表現方法を用いて，提案するパズルの可視化を行う．グラフ可視化技術として，Kamada-Kawai 法を改良したものを用い，列の表現方法として，列ごとの色分けと，折曲点の生成の 2 つの方法を用いる．

キーワード：抽象化とメディア変換，パズル，グラフ可視化

A New Network-Type Puzzle Based on Divergence via Abstraction and Its Visualization

Kazuki Hori[1,a)]　Hiroshi Hosobe[2,b)]

Abstract:
There are many types of puzzles, and many people have been enjoying puzzles since old days. Especially in recent years, puzzles for smart phones and tablets are increasing, and there are a growing number of people who enjoy such puzzles. However, most of existing puzzles are limited in shape although they adopt various rules. To largely change puzzles in shape, Sugiyama et al. proposed a systematic approach called divergence via abstraction, by which existing puzzles can be converted to other media via abstraction. In this paper, applying this approach to a different puzzle, we propose a new puzzle that allows a more various shape. Specifically, we represent a square grid puzzle as a graph by regarding each square as a node and a side-by-side square relation as an edge, which is based on the observation that many puzzles have a two-dimensional square grid shape. In particular, we treat a square grid puzzle called Kakuro. Also, we visualize such a puzzle by using a graph visualization method and ways for representing multiple lines. We modify the Kamada-Kawai method for graph visualization, and use two ways of representing lines, (1) using different colors for lines and (2) generating bending points.

Keywords: divergence via abstraction, puzzle, graph visualization

[1] 法政大学情報科学研究科
東京都小金井市梶野町 3 丁目 7 番 2 号
[2] 法政大学情報科学部
東京都小金井市梶野町 3 丁目 7 番 2 号
[a)] kazuki.hori.8g@stu.hosei.ac.jp
[b)] hosobe@acm.org

1. はじめに

パズルには多くの種類があり，昔から多くの人に楽しまれている．特に近年はスマートフォンやタブレット上でパズルを楽しむ人が増えている．しかし，既存のパズルは多様なルールがあるにもかかわらず，限られた形状をしていることが多い．

パズルの形状を大きく変化させる研究として，杉山らの研究 [1], [2], [3] がある．杉山らは，既存のパズルは抽象化を経て他のメディアに変換できるとする，抽象化とメディア変換と呼ばれる体系的アプローチを提案した．さらに，このアプローチを用いて二種類の既存のパズルの形状の一般化を行い，それを基に二つの新しいパズルを提案した．また，それらの新しいパズルを視覚的に表現するために，グラフ可視化技術を用いた．

本研究では，抽象化とメディア変換を杉山らの研究とは異なる種類のパズルに適用することで，より幅広い形状を可能とした新しいパズルを提案する．具体的には，多様なパズルが二次元の格子状であることに着目し，各マスをノード，となりあうマスの関係をエッジとみることで，格子状のパズルをネットワーク，すなわち，グラフとして一般化して表現する．格子状のパズルとして，本研究では特に加算パズルを扱う．さらに，グラフ可視化技術と複数の列の表現方法を用いて，提案するパズルの可視化を行う．グラフ可視化技術として，Kamada-Kawai 法 [4] (以下，KK 法) を改良したものを用い，列の表現方法として，列ごとの色分けと，折曲点の生成の 2 つの方法を用いる．

2. 関連研究

2.1 抽象化とメディア変換に基づくパズル

杉山らは抽象化とメディア変換の概念を提案し，本概念に基づき置換パズル [1], [3] と巡回パズル [1], [2] の二つのパズルを提案した．置換パズルとは，パズルの操作を要素間の置換として特徴づけられるパズルであり，ルービックキューブに代表される．一方，巡回パズルとは，パズルに同じ操作を繰り返すとパズルの状態が周期的に変化し一定回数後に元に戻るという特徴をもつパズルであり，ライツアウトに代表される．またこれら二つのパズルについてパズルジェネレータを作成し，それぞれの可視化を行った．置換パズルにおいてはパズルの対称性を重視した新しいグラフ可視化アルゴリズムを提案した．

2.2 Kamada-Kawai 法

KK 法 [4] とは力指向アルゴリズムを基にしたグラフ描画アルゴリズムであり，シンプルな式で合理的な配置を計算することができる．この手法の特徴としてつながった要素間の距離が一定になるため，要素が密集する，つながった要素が離れすぎるといった状況をある程度防ぐことができる．

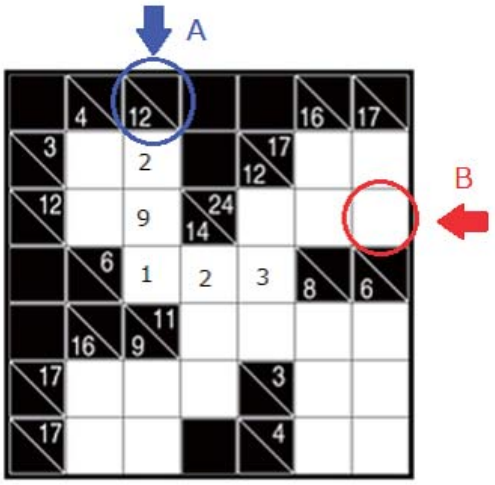

図 1　加算パズルの例
Fig. 1　A Kakuro puzzle

3. 加算パズル

本研究では抽象化とメディア変換を加算パズルに対して適用する．加算パズルとは，図 1 B のようなブロックに数字を入れて，図 1 A のような各三角マスから始まる縦または横それぞれのブロックに入る数字の合計が，三角マスの中の数字と等しくなるようにするパズルである．三角マスの中の数字は，斜線の左下なら縦，右上なら横の列の値の合計を示す． またブロックに入れる値は 1〜9 の整数のみで，縦横それぞれで同じ三角マスから連なるブロックに同じ値を入れてはならない．

例として，図 1 A の三角マスは斜線の左下に 12 が入っているため，このマスの下に連なる三つのブロックに入れる値の合計を 12 にしなくてはならない．この時，2, 5, 5 や 4, 4, 4 といった同じ値を用いた組み合わせを使うことはできない．

4. 提案手法

4.1 パズルの抽象化

加算パズルは，各列に記入した数字の合計が決められた値になるようにするパズルである．よって加算パズルを定義するためには，各要素に入る値，各要素の列への所属とその順序，各列の要素の値の合計が定まっていればよい．このことから加算パズルを抽象化したもの M を次のように定式化できる．

$M = (X, C, \varphi, L)$
$X = \{x_1, x_2, ..., x_n\}$：要素の集合
$C = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$：要素に入る値の集合
$\varphi : X \to C$：各要素に入る値
$L = \{(t_1, r_1), (t_2, r_2), ..., (t_m, r_m)\}$：列の集合

$r_i = \{x_{j1}, x_{j2}, ..., x_{jki}\}$
($p \neq q$ のとき$\varphi(x_{jp}) \neq \varphi(x_{jq})$)
$t_i = \varphi(x_{j1}) + \varphi(x_{j2}) + ... + \varphi(x_{jki})$

以下に簡単な例を示す．

$X = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8\}$
$C = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$
$\varphi : x_1 \to 1, x_2 \to 2, x_3 \to 3, x_4 \to 4,$
$x_5 \to 5, x_6 \to 6, x_7 \to 7, x_8 \to 8$
$L = \{(t_1, r_1), (t_2, r_2), ..., (t_7, r_7)\}$
$r_1 = (x_1, x_2, x_3), r_2 = (x_6, x_7, x_8)$
$r_3 = (x_1, x_4, x_7), r_4 = (x_2, x_5, x_7)$
$r_5 = (x_2, x_4, x_6), r_6 = (x_3, x_5, x_6), r_7 = (x_3, x_8)$
$t_1 = \varphi(x_1) + \varphi(x_2) + \varphi(x_3) = 1 + 2 + 3 = 6$
$t_2 = 21, t_3 = 12, t_4 = 14, t_5 = 12, t_6 = 14, t_7 = 11$

4.2 パズルの可視化

定義したパズルを可視化するとき，要素の位置をランダムに決定してしまうと，配置によっては列の把握が困難になってしまう可能性がある．そのため要素がある程度わかりやすい配置になるよう，本研究では各要素の配置に KK 法を用いる．ただしそのままではいくつかの問題点があるため，それらについての対策を行う必要がある．

4.2.1 Kamada-Kawai 法の改良

KK 法を用いて要素の位置決定を行った場合に，図 2 の四角形 19 のようにほかの要素と一つしかつながりを持たない要素がよい位置に配置されず，その要素につながった要素が引っ張られてしまうことで最終的なパズルの形状の一部が歪んでしまう場合がある．パズルの形状が歪むことで要素間の距離が狭くなってしまうと，要素がエッジの上に重なってしまう可能性が増え，パズルの可視化に問題が起きてしまう．本研究ではこの問題の対策として，ほかの一つの要素としかつながりを持たない要素の位置を決める際に，一定の確率で，その要素につながる要素にさらにつながる要素のいずれかと位置を交換してから位置決定を行う．図 3 の例では四角形 19 と図中の A か B の要素の位置を入れ替えた後に計算を進めることで，問題を解決することができる．

4.2.2 列の表現方法

位置を決定した要素をただ繋いだだけでは列の識別ができなくなってしまうため，以下の通り，列ごとに色分けする等の対策を行う．

(1) 列ごとに色分けをする．単純に列ごとに色分けをして表示する事で識別を容易にする．ただし，この方法には，列の数が多くなるにつれ，似た色の列同士が交差する箇所がでてくる可能性が増え，識別が困難になるという問題がある．
(2) 曲げるための点を別に用意する．要素同士を単純につないだ場合に列の識別が困難になる原因として，列同士が交差した点で列が曲がってしまうと，その列が次にどの要素に向かっていくかわからなくなってしまうことが挙げられる．そこで要素と要素を結んだ直線上に，曲がるための点を新たに作成し，そこでのみ曲がるようにすることで，列同士が交差した点では曲がる事がなくなり，列の識別が容易になると期待できる．

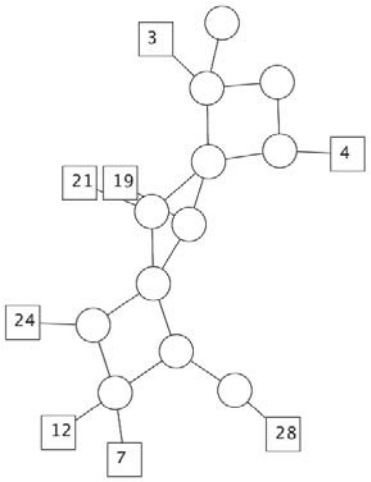

図 2 歪んだ配置
Fig. 2 A distorted layout

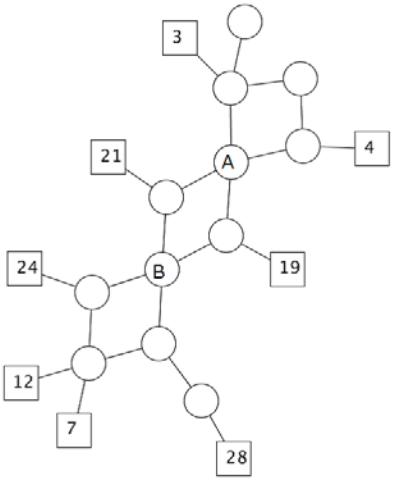

図 3 改善された配置
Fig. 3 An improved layout

5. 実装

本研究では実装に Processing を用いた．またパズルの内容を入力するためのもの，パズルを可視化し実際に遊ぶためのものの二つにわけて実装した．

5.1 入力内容・入力画面

コンピュータでパズルを定義する場合，人の手で要素の数，各要素に入る値，列の数，各列にどの要素が所属するか

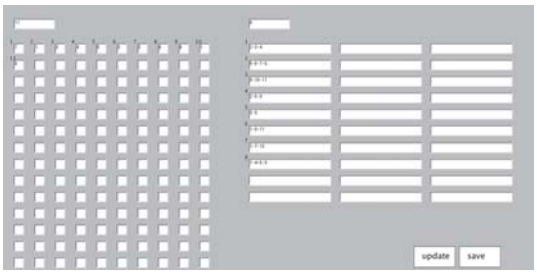

図 4　入力画面
Fig. 4　An input screen

及びその順番が入力されれば残りの項目はプログラム上で定義できるので，これらの項目を入力する形でパズルジェネレータを作成した (図 4)．パズルの内容は txt ファイルとして保存される．

5.2　表示方法・表示画面

まずパズルの情報を txt ファイルから読み込む．このとき txt ファイルには各列に所属する要素の値の合計と実際に表示する列の情報がないためここで設定する．また KK 法で利用するため各要素にどの要素がつながっているかの情報もここで設定する．次に KK 法を用いて各要素の位置を決定する．このときパズルの中心が画面の中心からずれてしまい，この後画面下部に設置するボタン (図 5 の下部) に重なってしまうことがある．そこで KK 法を実行した後最も画面の右側と左側の要素の x 座標からパズルの中心の x 座標を調べ，パズルの中心の x 座標が画面の中心の x 座標に来るようにすべての要素をスライドする．また最も画面の上側の要素の y 座標を調べ，画面下側のボタンに要素が重なってしまわないよう最も上の要素が画面の上部の端に近くなるようすべての要素をスライドする．その後ボタンを設置しパズルを表示する．ボタンは入力したい数字を選択するためのもの，入力した回答が正しいかチェックするためのもの，答えがわからなかったとき回答を表示するものを実装した．

6.　実験

本研究で提案した KK 法の改良法がどの程度の効果があるか実験を行う．また本ジェネレータがどの程度の規模のパズルが作成できるか，要素のつなぎ方を変えて実験を行う．

6.1　Kamada-Kawai 法の改良

6.1.1　実験方法

KK 法と改良した KK 法を 20 個のパズルに対して適用し性能を比較する．評価方法としてはほかの要素と一つしかつながりを持たない要素に関係するエッジと他のエッジの交差数と実行時間を用いる．実行時間の計測区間として，KK 法で利用する各種パラメータを設定し終え，要素の位置を決定し始めたところからすべての要素のエネルギーが安定してループを抜けたところまでとする．改良した KK 法では，4.2.1 節に述べた確率を 10 %とする．また改良した KK 方法については，同じパズルに対して 10 回この手法を適用し，交差数と実行時間についてこの平均をとって評価する．

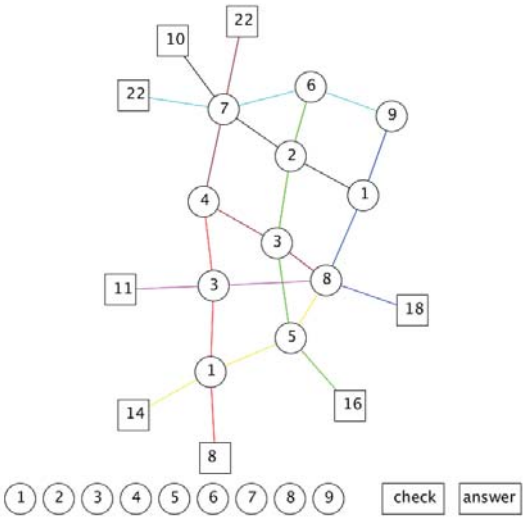

図 5　表示画面
Fig. 5　An output screen

6.1.2　結果

実験対象とした 20 個のパズルのうち代表的な 5 個についての結果を表 1 に示す．エッジの交差数については，全体的に改良した KK 法の結果が元の KK 法の結果と比べて交差数が少なくなっているか，悪い結果でも同じくらいの交差数となり，元の KK 法の結果と比べて悪くなる場合は少なかった．また元の KK 法の結果と比べて悪くなってしまった場合でも，パズルの形状が大きく悪化してしまう場合はほとんど見られなかった．実行時間については改良した KK 法の方が元の KK 法の結果と比べて長くかかった．平均実行時間では 1.5 倍から 3 倍程度の時間がかかる場合が多いが，個々の実行時間を見ると元の KK 法の結果とあまり変わらない場合から 5 倍以上かかっている場合もあり，かなり幅が大きかった．またループを抜ける条件にかからない微妙な状態には陥る場合があり，そうなってしまった場合ループの上限回数に達してしまい，表 1 のパズル 14 のように実行時間が大幅に増加してしまう，といった問題がまれに発生した．逆に表 1 のパズル 1 のように元の KK 法で同じような状態になってしまっている場合に改良した KK 法ではその状態を抜け出して実行時間が短くなった場合もあった．

6.2　列の表現の実験方法

規模を変えたいくつかのパズルに対して，3 種類のつなぎ方をためす．

表 1　実験結果
Table 1　Experimental results

パズル	手法	交差数	実行時間 (ミリ秒)
1	KK	1	3856
	改良 KK	0.3	166.2
2	KK	1	179
	改良 KK	0.4	260.1
5	KK	3	349
	改良 KK	0.9	457.2
10	KK	2	193
	改良 KK	1.4	277.1
14	KK	2	252
	改良 KK	1.2	1634.2

6.2.1　色分け
列ごとに色分けをする．色の配分のしかたは事前に設定していた 13 色を順番に配分する方法と，各列にランダムで色を設定する方法をためす．

6.2.2　折曲点の生成
曲げるための折曲点を別に用意する．各列の合計を表示している場所を始点とし，そこからその列の最初の要素を結んだ直線上の最初の要素から一定距離離れた場所に一つ目の折曲点を設定する．次に一つ目の折曲点と 2 番目の要素を結んだ直線上の 2 番目の要素から一定距離離れた場所に 2 番目の折曲点を設定，というように最後の要素の一つ前まで折曲点を設定していく．その後合計を表示している場所，一つ目の折曲点，二つ目の折曲点，…，最後の折曲点，最後の要素，とつないでいく．この方法と，最後の要素でも折曲点を設定しそこを終点とする方法の二つをためす．

6.2.3　併用
上記の二つの方法を同時に使用する．色分けについては事前に色を設定しておく方法のみ使用し，折曲点を用いる方法は両方の方法を使用する．両方同時に使用することでどちらか片方では判別が難しい場面でも判別が容易になると期待できる．

6.3　列の表現の結果
6.3.1　色分け
事前に使用する色を決めておく方法は図 6 のように列数が 13 より少ないうちは問題がなかったが，13 より多くなると同じ色の列が交差してしまい列の判別ができなくなってしまうことがあり，列の数が多くなるほど同じ色の列が交差する数が多くなった．色をランダムに設定する方法では，列数が少ないときでも似た色の列が交差してしまい列の判別が困難になってしまうことがあったが，事前に色を決定していた方法と比べて列数が多くなっても列を判別できることが多かった．

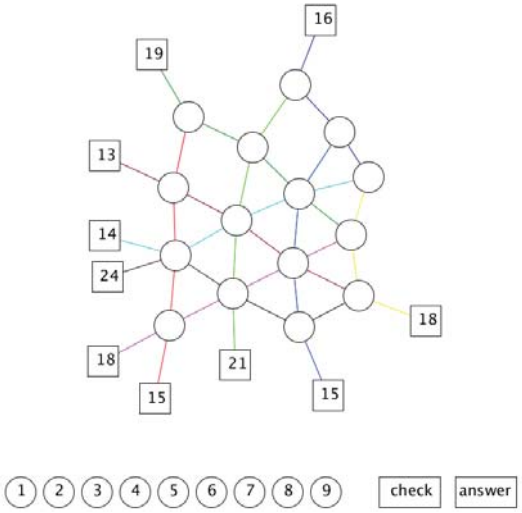

図 6　色分け
Fig. 6　Using different colors for lines

6.3.2　折曲点の生成
最後の要素を終点とする方法は，すべての要素がばらばらに配置されているパズルでは規模によらずしっかりと折曲点がすべて表示され列の判別が可能だった．しかし一部でも要素同士が密集した場所ができてしまうと，折曲点が近くの要素と重なって見えなくなってしまい列の判別ができなくなってしまうことがあった．またまれに二つの線が同じ方向から同じ要素につながってしまい，折曲点が同じ場所に設定されて判別できなくなる問題が発生した．またどこで列が終了しているかがわかりにくく，すぐに列を判別することは難しかった．最後の要素でも折曲点を設定する方法でも同じ問題が発生したが，図 7 のようにどこで列が終わるかはっきりしたことで列の判別はもう一つの方法と比べて容易になった．折曲点の増加により要素と折曲点が重なってしまう可能性が少し増えてしまうが，全体的なわかりやすさはこちらのほうがよいと考えられる．

6.3.3　併用
両方同時に使用することで同じ色の列が交差してしまっても判別が可能となった．しかし折曲点が別の要素と重なってしまった場合は折曲点を生成する方法と同様に列の判別が困難になってしまった．また折曲点を用いる二つの方法を比較すると，色分けしたことにより最後の要素を終点とする方法でも列の判別は容易になった (図 8) が，最後の要素でも折曲点を生成する方法がより判別しやすかった．

7.　議論
7.1　Kamada-Kawai 法の改良
エッジの交差数については元の KK 法の結果と比べて同じくらいか少なくなっている場合が多く，改良した KK 法を用いる有用性はあると考えられる．交差数が同じくら

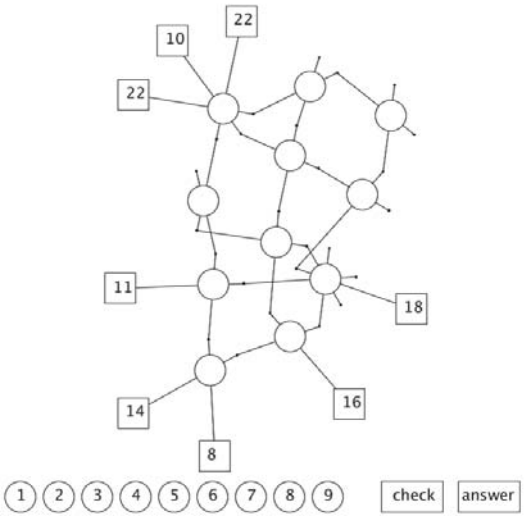

図 7　折曲点の生成
Fig. 7　Generating bending points

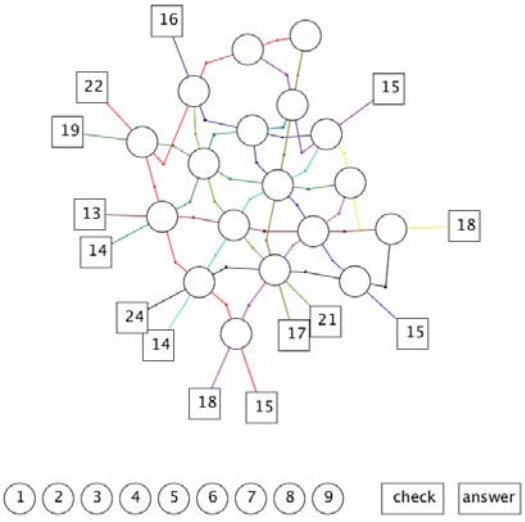

図 8　併用
Fig. 8　Combination

い，または増えてしまう原因としては，以下の二つが考えられる．

- 確率で要素の位置の入れ替えを行うか決めているため，交差してしまっているエッジに関係する要素がそのままになってしまう場合がある．
- 位置を入れ替えた結果，エッジが交差してしまう場所に陥ってしまう場合がある．

一つ目の問題の解決案として，要素の位置を入れ替える確率を上げるといったことが考えられる．ただし要素の位置を入れ替える確率を上げると実行時間が伸びてしまう，また位置を入れ替え続けて設定したループ回数内に終わらなくなってしまう，といった問題が発生することが予想されるため，適切な確率を調査する必要がある．二つ目の問題の解決案として，よくない位置の入れ替えが行われた際にその入れ替えをなかったことにするといったことが考えられる．問題として，よくない入れ替えだったのかどうか判断する評価基準をどうするか，また早い段階で入れ替えが行われた場合の評価をどうするか考える必要がある．

実行時間については全体的に元の KK 法の結果と比べて長くかかってしまうが，今回扱った規模のものに対して扱うならば大きな問題にはならないと考えられる．ループを抜ける条件にかからない微妙な状態に陥ってしまい大きく実行時間が伸びてしまう場合があるという問題については，あまり多く発生する問題ではないため，想定より多く時間がかかっている場合はやり直すといった方法で改善することが可能であると考えられる．

7.2　列の表現方法

3 種類の方法を比べると，列の数が少ない場合は色分けが優秀だったが，列の数が多くなるにつれ折曲点を生成する方法のほうが列の判別が可能なパズルが多くなった．また併用したものは全体的に折曲点のみの場合より判別が容易だったため，列数が少ない場合は色分けのみ，列数が多い場合は併用したものを使用するとよいと考えられる． 折曲点を利用する方法は，実験結果から始点と終点がはっきりしていると列の判別が容易になると考えられる．今回提案したパズルでは各列に所属する要素の値の合計を表示する場所が始点としての役割をしていたため始点として特別に点を用意することはしなかったが，同じような格子状のパズルにこの方法を用いる場合は始点を用意したほうがよいと考えられる．

問題点として，要素が密集した場所ができてしまうと，特に折曲点を利用する方法で適切に列の表現ができなくなってしまう場合がある．この問題の原因として，作成するパズルに対して画面が小さすぎること，KK 法の位置決定に問題があることの二つが考えられる．一つ目の問題はパズルの全体が画面内に収まらなくてもよいとして，画面をドラッグするなどして全体を見ることができるようにすることで解決できると考えられる．二つ目の問題を今回の手法を変えずに解決するためにはパズルの形状を制限しなくてはならないため，他のアルゴリズムを併用する 等の対策を考える必要がある．

7.3　他のパズルやゲームへの適用

本研究で提案するパズルの可視化手法は，列を扱う他のパズルやゲームに対しても適用可能だと考えられる．特に扱う対象が数字から文字に変わっただけのクロスワードパズルは本研究の手法をそのまま利用することで実装可能と考えられる．その他のパズルやゲームではオセロやペイントロジックに対して適用可能だと考えられる．特にオセロにはニップなど盤の形状を変化させたものが存在するため，適用は容易であると考えられる．ただしオセロでは最初に置く石の位置をどうするか，またペイントロジックで

はどのようにして最終的な絵を見せるか，について考える必要があると考えられる．

8. おわりに

本研究では抽象化とメディア変換を用いてパズルの形状を一般化し，それを基により幅広い形状で作成可能なパズルの提案とそのパズルの可視化を行った．小規模，中規模のパズルの可視化は考えていた通りにできたが，大規模なパズルや一部の小・中規模のパズルはうまく可視化できなかった．本研究ではパズル全体を画面内に収めたが，パズル全体が画面内に収まらなくてもよいとするならばより大規模なパズルの可視化も可能だと考えられる．今回適切に表示できなかったパズルの可視化を可能とすること，他の格子状のパズルに対しても今回の手法を適用し，適切に表現可能か調べること，また本研究ではパズルの難しさ等を考慮していないため，パズルの難易度がどのように変わっているか調べることが今後の課題となる．

謝辞 本研究は JSPS 科研費 JP25540029 の助成を受けたものである．

参考文献

[1] Sugiyama, K., Osawa, R. and Hong, S.-H.: Puzzle Generators and Symmetric Puzzle Layout, *Proc. APVIS*, CRPIT, Vol. 45, ACS, pp. 97–105 (2005).
[2] 前田篤彦，杉山公造，間瀬健二：巡回パズルのメディア変換とパズル・ジェネレータの試作，情報処理学会研究報告：ヒューマンインターフェース (HI)，Vol. 101, pp. 41–48 (2002).
[3] 前田篤彦，杉山公造，間瀬健二：置換パズルのメディア変換とパズル・ジェネレータの試作，情報処理学会研究報告：ヒューマンインターフェース (HI)，Vol. 101, pp. 33–40 (2002).
[4] Kamada, T. and Kawai, S.: An Algorithm for Drawing General Undirected Graphs, *Information Processing Letters*, Vol. 31, No. 1, pp. 7–15 (1989).