2006－CG－122（1）
2006／2／20

折紙の展開図からの折りたたみ形状推定

三　谷　　純†

完成した形が平坦に折りたたまれたものである折紙を対象に，展開図から折りたたみ後の形状を推定する手法を提案する．折り線または紙の輪郭線によって囲まれる閉領域を折紙の構成要素とし，折りたたみ後にそれぞれの要素がどの位置に移動し，重なり順がどのようになるかを計算することで，折りたたみ後の形状を推定する．提案手法を実装したシステムでは，展開図自体が妥当でない場合や，妥当な解が複数ある場合などの考慮も行っている．

Estimation of a folded figure of Origami from the unfolded pattern

Jun Mitani†

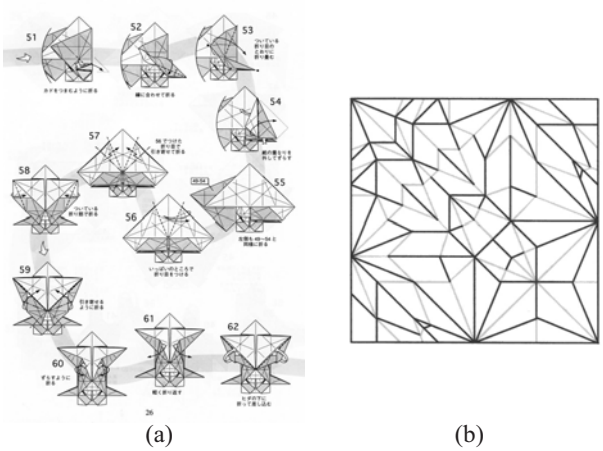
This paper proposes a new method for estimating a folded figure of Origami from the unfolded pattern. The targeted Origami is folded in the flat. The method handles closed areas, that are bounded by crease lines and contour lines, as elements of the Origami structure. The folded figure is calculated by estimating the position and overlapping order of each element. The system that implemented the proposed method can distinct those unfolded patterns that are not be folded in flat correctly.

1. はじめに

紙を折ることで形を表現する「折紙」は日本に古くから伝わる文化の一つであり，教育の場での活用や趣味の一つとして幅広い世代に親しまれている．近年では世界的にも広く認知され，学術的な研究も多くされている．また近年になって作品の創作に「設計」の概念が持ち込まれるようになり，従来の試行錯誤に基づく創作活動から得られるよりも，より複雑で精緻な作品が数多く生み出されるようになった．これらの作品の折り方を他者に伝える方法として「折り図 (図 1(a))」と呼ばれる，折り工程を図で表現したものが用いられるが，複雑な作品の折り図の作成は労力を要する作業であり，このことが様々な作品を伝達することを困難にしている．

そこで，折紙の形状を計算機内に構築し，折り方をアニメーション表示したり，作品を 3DCG で表示することで折り方の伝達を試みる研究がされている．折紙の形状を計算機内に構築する場合には，形状を平面多角形の集合などで表現した幾何情報と，それぞれの多角形の接続関係を表す位相情報を用いてモデル化することが一般的である．しかし，計算機内に折紙の形状データを構築するための手法は未だ確立されていない．従来の CG のインターフェイスで 1 つ 1 つモデリングするのは手間がかかりすぎるため，ユーザーと対話的に折紙を折るような操作でモデルを構築するための研究もされている．また，QR コードを印刷した折紙を写真で撮影して，形状を推定する研究もされている．本稿では，折紙の「展開図 (図 1(b))」から折紙の完成型を計算機で類推する手法を提案する．

なお，本稿で扱う展開図は山折と谷折の区別がされているものとし，さらに折りたたみ後の形が平坦になるもののみを対象とする．



(a)　(b)

図 1　(a) 折り図，(b) 展開図（どちらも馬の作品[1)]）
Fig. 1　(a)Diagram. (b)Crease pattern. (Both are of House[1)]).

本稿の第 2 章では関連する研究を紹介し，第 3 章で対象とする展開図とその入力方法について述べる．第 4 章では本手法の詳細を述べ，第 5 章で結果を，第 6 章で考察をま

† 筑波大学大学院 システム情報工学研究科
Department of Computer Science, Univ. of Tsukuba

とめ，第 7 章で今後の展望を述べる．

2. 関 連 研 究

折紙は正方形の紙に対して折り操作を繰り返すことで様々な形を作るものであり，幾何の分野における研究題材として多く取り上げられている[2)3)]．また，国内に留まらず海外でも折紙の設計手法についての研究が行われており，近年では対象とする形状を折り出すための「設計」の概念が導入され，複雑な形状を持つ作品が多く作られるようになった[4)]．折紙を計算機で扱う研究として，内田ら[5)] は紙の物理的な制約条件を元に，折紙の展開図を構成する幾何学的要素から折りたたみ方法を推論するプログラムを提案した．この研究は本稿の扱う研究テーマと非常に近いが，対象とする展開図を構成する折り線が「山折線」と「谷折線」だけでなく,「山折に折って開い線」と「谷折に折って開い線」も含まれる点が，本稿の対象と異なる．つまり内田らの対象とする展開図には，最終的な形を決定するのに用いない折り線も含まれ，本稿が対象とする展開図よりも情報が多い．図 2 は内田らの対象とした鶴の展開図であり，本稿で扱う鶴の展開図（図 5(a)）と比較すると違いが明らかである．

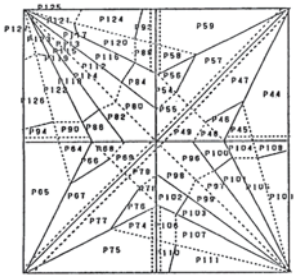


図 2 4 種類の折り線を持つ鶴の展開図[5)]
Fig. 2 Crease pattern of a crane with for types of line[5)].

Miyazaki ら[6)] は，計算機を用いて折紙をディスプレイに表示しながら対話的に操作する手法を提案した．紙を折る操作によって折紙の形状が逐次変化する際の，データ更新の手法を提案している．Kato ら[7)] は,「折り図」の画像を計算機で解析することで，折り操作を推定し，それを元に計算機内の折紙モデルを更新する手法を提案した．この手法は図中に含まれる矢印や折れ線の情報を活用している．三谷ら[8)] は 2 次元バーコードを印刷した紙を折ったものをカメラで撮影し，その写真を元に折りたたみ構造を推定しモデルを構築するする手法を提案した．しかしこの手法では，折り込みを含む形に対応できないため，一般的な作品に応用することは難しい．島貫ら[9),10)] は，折り操作によって得られる妥当な面の重なり順の算出方法を提案しているが，これは特定の切断面に着目して推定しているため，局所的な領域の折りたたみ順を求める方法である．

3. 折紙の展開図

折紙の展開図は「山折線」,「谷折線」および紙の輪郭を表す「輪郭線」の 3 種類の線分の集合から構成される．本稿では展開図から折りたたみ後の形状を推測するため，まず展開図の情報を計算機に入力する必要がある．そのために，展開図を入力するためのエディタ (ORIPA: ORIgami PAttern editor) を開発し，これを用いて展開図データを作成した．

3.1 エディタ機能の実装:ORIPA

展開図編集用のエディタとして，次のような機能を持つアプリケーションを PC 上に Java 言語で開発した．このエディタでは，折紙の展開図を入力し易くするために，以下の方法で線分を入力する機能を実装した．

- 指定した 2 点を端点に持つ線分
- 指定した 2 点を通る線分
- 指定した角を 2 等分する線分
- 指定した 3 点から，3 点の内心を結ぶ 3 本の線分
- 指定した点から指定した線分への垂線
- 指定した線分と，指定した直線に関して対称な線分
- 対称コピー
- 角度と長さの指定

上記の方法で指定する頂点には，線分の交点またはグリッド上の点を指定できる．また，次節で述べる判定で展開図が妥当でない場合には，その箇所をハイライトしてユーザに知らせる機能が実装されている．なお，このエディタは Web 上で公開している[11)]．

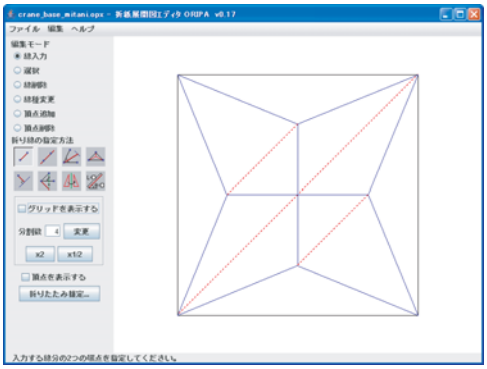


図 3 ORIPA の画面
Fig. 3 Snap shot of ORIPA.

3.2 展開図の妥当性

展開図はエディタで入力されるが，入力の時点では何ら制約が設けられていないため，任意の折れ線を入力可能であり，入力された展開図が妥当な展開図でない可能性がある．妥当でない展開図とは，その折り線の通りに折り操作を行っても，実際に平坦に折りたたむことが不可能な展開図のことを言う．

与えられた展開図が平坦に折りたためるために必要十分

な，展開図の内点に接続している折れ線の条件（局所平坦条件）は文献 12)~14) より，次の通りである．

(1) 折れ線の数は偶数である
(2) 山折線の数と谷折線の数の差の絶対値は 2 である
(3) 折れ線の成す角のひとつおきの和は 180 °である
(4) 折れ線の成す角が鈍角の時，その角をはさむ 2 つの線の折れ線属性（山折/谷折）は等しい

これらの条件を全ての内点が満たす展開図は平坦に折りたたむことができる．入力された展開図に対して，上記の条件を満たすかを調べることで，妥当でない展開図を識別できる．

4. 展開図の折りたたみ

本章では展開図に基づいて計算機内で紙の折りたたみを行い，折りたたみ後の形状を構築する手法を述べる．

4.1 多角形面素の取得

折り線または輪郭線で囲まれた閉領域を，本稿では「多角形面素」と呼ぶこととする．また文脈から誤解のない場合には単に「面」と呼ぶこともある．

この多角形面素は，折り線または輪郭線を図 4 中の F_0〜F_3 のように反時計回りに巡回して得られる．多角形面素には次のような性質がある．

- 多角形面素は互いに重なり合わない
- 多角形面素の和は折紙の形に等しい
- 折紙の形が凸多角形であれば，すべての多角形面素は凸多角形である

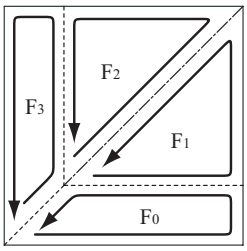


図 4 展開図からの多角形面素の取得
Fig. 4 Extraction of polygon-elements from a crease pattern.

4.2 折りたたみ

本手法ではまず，多角形面素どうしの重なり順を考慮せずに，折りたたみ操作によって，それぞれの多角形面素が折りたたみ平面上でどの位置に移動するかを推定する．はじめに起点となる任意の多角形面素を決定し，それに隣接する多角形面素を再帰的に巡回する．隣接面に移動する毎に，2 面間を介する稜線によって（山折，谷折の別に依らずに）多角形面素を折り返す（つまり平面上で折り線を介して反転させる）ことを行う．この反転操作は，各多角形面素に対して高々1 回行うものとする．これにより，基点となる面から奇数回の移動で到達する場合は面の向きが裏に，偶数回の移動で到達する場合は面の向きが表になる．全ての多角形面素を巡回した時点で折りたたみ操作が完了する．

展開図に誤りが無い場合，展開図上で隣接している多角形面素の組は，折りたたみ後でも同じ位置の辺を共有するはずである．従って，折りたたみ後に稜線を共有しなくなるものが存在したら，局所平坦条件の (1)〜(3) を満たさない展開図であると判断できる．

4.3 折りたたみの結果

上記の手法で，展開図から多角形面素を取得し，その折りたたみ後の位置を算出することで図 5 の結果が得られる．図 (a) は鶴の展開図であり，(b) は折りたたみ後に表面が上を向く多角形面素を網掛けで表している．(c) は折りたたみ後の多角形面素の輪郭を表示したもの，(d) は透過色で塗りつぶしたものである．これにより，面の向きが隣接する多角形面素で互いに異なること，及び前述の方法で妥当な折りたたみ後の形状を得られることがわかる．なお，この時点ではまだ多角形面素の重なり順は不定である．

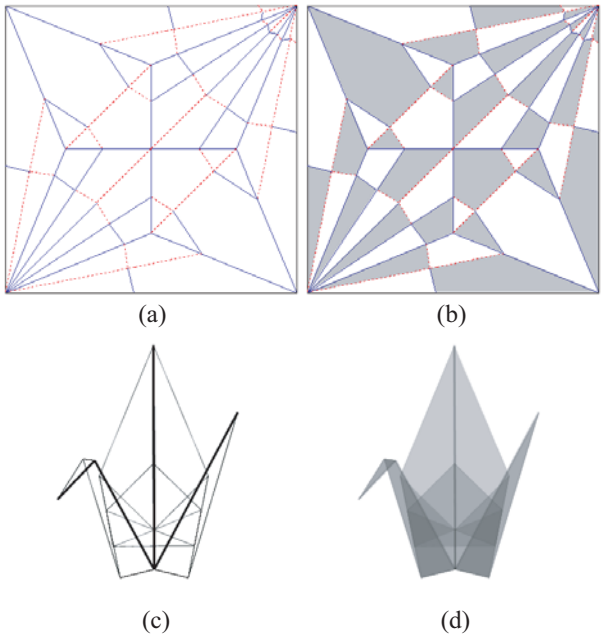


図 5 (a) 展開図，(b) 面の向きによって色分けされた展開図，(c) 多角形面素の輪郭，(d) 透過色で塗りつぶした多角形面素
Fig. 5 (a)Crease pattern. (b)Polygon-elements colored by the direction of normal. (c)Contours of polygon-elements. (d)Polygon-elements filled with transmitting color.

4.4 多角形面素の重なり順の決定

先ほどの折りたたみ操作で，平面上での各多角形面素の位置を得ることができるが，それぞれの重なり順は求まらない．以降では，各多角形面素の重なり順を決定する方法を述べる．なお，図 6 に示す「ねじり折り」のような，重なり順の関係が閉ループを成すような作品は本稿では対象外とする．

多角形面素の重なり順には制約があり，折り線に従って折り操作を行う場合，多角形面素どうしが衝突して実現で

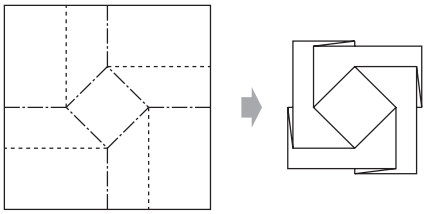


図 6 ねじり折り
Fig. 6 Twisting fold.

きない重なり順が存在する．例えば，図 7 を例に挙げると，F_1 を最下層に配置した場合 (a) の展開図では F_0 と F_2 のどちらが上でも問題ないが，(b) は F_2 が F_1 の下になることができない．また，この例から，多角形面素の重なり順を決定するには，展開図の位相情報からだけでは判断できず，多角形面素の形状に基づく幾何的な判定が必要があることがわかる．

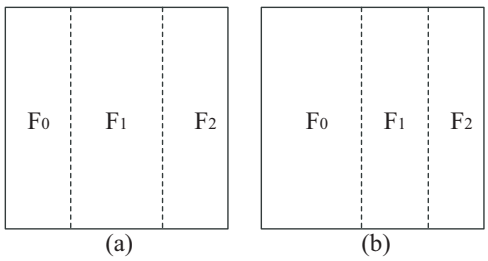


図 7 多角形面素の重なり順の制約（波線は谷折）
Fig. 7 Constraint for overlapping order.

面の重なり順を決定する問題は NP 問題であることが証明されている[15] ため，本手法では，取り得る全ての重なり順を対象に探索を行い，折紙として妥当な重なり順であるものを抽出することとする．ところで，多角形面素の数を N とすると，その重ね順の場合の数は $N!$ 通りあり，N が増大するとすぐに場合の数は爆発してしまう．例えば図 5 の鶴の例では多角形面要素の数が 52 であり，$52! = 10^{67}$ 通りの重なり順を全て調べるのは現実的でない．

そこで，できるだけ判定の数を減らすために図 8 に示すアルゴリズムを用いる．このアルゴリズムでは，始めに空のスタック FaceStack を用意し，そこに多角形面素を重なり順に格納することを行う．addFace 関数では自身を再帰的に呼び出すことで，順番に面をスタックに追加するが，面の重なり順として妥当でない面が追加された場合はスタックからポップし，次の面を格納することを行う．これにより，全ての重なり順を調べるよりは遙かに効率よく妥当でないものを判定の対象からはずすことができる．全ての多角形面素を問題なくスタックに格納し終わったら，妥当な重なり順が求まったものと判断できる．

解が 1 つも見つからなかった場合は，展開図が局所平坦条件の (4) を満たしていないことがわかる．

図 8 の addFace 関数では，スタックに面を追加してよいか否かを判定するための関数 canAddFace を呼び出しているが，この関数はスタックの最上位に，引数の多角形面素 F を追加できる場合に true を，そうでない場合に false を返すものとする．ここで，スタックに F を追加できる場合とは，F を追加しても次の 2 つの条件が満たされる場合である．

(1) 折線での折り曲げ方が展開図の山折，谷折と矛盾しない
(2) 任意の断面で面の交差が生じない

上記の条件の判定方法を説明する前に，面の稜線を CE:Contour Edge（輪郭線），UE:Upper Connect Edge（上に接続する稜線）および LE:Lower Connect Edge（下に接続する稜線）に分類する．面の表側が上を向いている場合，谷折の稜線は LE，山折の稜線は UE となる．面が下を向いている場合はこの逆となる（図 9）．面の表側が上下のどちらを向いているかは前章の折りたたみ操作で確定している．

```
addFace() {
  foreach(FaceStack に含まれない Face f) {
    // f を FaceStack の末尾に追加してよいかチェック
    if(FaceStack.canAddFace(f)) {
      FaceStack.push(f);
      if(全ての面の順番が決まったら) {
        処理を終了; // 妥当な解が見つかった
      }
      addFace(); // 再帰的に次の層へ
    }
  }

  if(FaceStack.empty()) {
    処理終了; // 妥当な解が見つからなかった
  } else {
    FaceStack.pop();
  }
}
```

図 8 多角形面素の重なり順の決定
Fig. 8 Algorithm for finding valid overlapping orders.

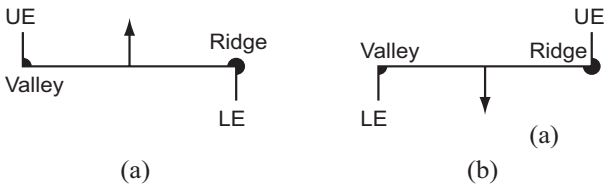


図 9 面の向きと接続（矢印は面の向きを表す）
Fig. 9 Direction of normal and face-connection.

折紙の多角形面素の重なり順に，スタックに下から順番に面を追加していくことを考えると，追加する面 F に LE を介して隣接する面 F' が，必ずスタック内に存在する必要がある（スタック内に F' が存在しない場合，F よりも F' が上方に配置されることになり，LE の谷折，山折が展開図と矛盾することとなる）．つまり，このことが上記 (1) の条件である．

(2) の条件の判定では，新しく配置する面 F が既にスタックに格納されている面 F' の UE の接続を妨害しなければよい．各面 F' に含まれる未接続の UE(e) を，面 F が覆い隠す位置にある場合 (図 10(a)) は false となる．(図 10 では，最上位の面を太線で表している．また灰色の領域は理解しやすくするために幅を持たせてあるが幾何的には幅ゼロの領域である．) e が面 F の UE と重なり合う位置にある場合 (図 10(b)) は true である．e が面 F の LE と重なり合う位置にある場合，面 F と隣接する面 $F_{neighbor}$ との位置関係によって条件が異なる．

- F' が $F_{neighbor}$ より下の場合 (図 10(c)) は true である．
- F' が $F_{neighbor}$ より上の場合で，F' と $F_{neighbor}$ が重なり合う場合 (図 10(d)) は false である．
- F' が $F_{neighbor}$ より上の場合で，F' と $F_{neighbor}$ が重なり合わない場合 (図 10(e)) は true である．

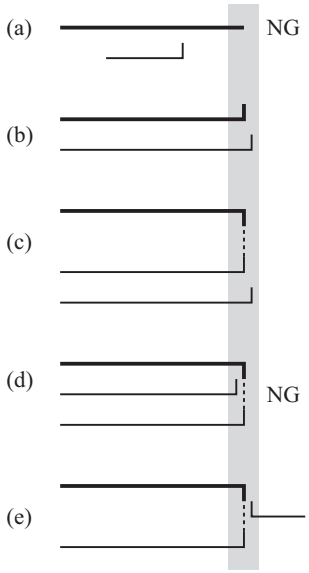


図 10 多角形面素の重なり順と稜線の関係
Fig. 10 Relations between polygon-element and edges.

5. 結　　果

エディタで展開図を作成し，本手法を CPU:Pentium Mobile Processor2.0GHz，RAM1.0GB の PC に実装したシステムで実験を行った．対象とした展開図は図 11 に示す 4 つで，それぞれ (a) は図 4 に示した単純な例，(b) は中割折りの例，(c) は兜，(d) は鶴の半分の展開図である．鶴に関しては展開図が対称な形状であることを考慮して，その半分の展開図に対して評価を行った．実験結果は表 1 に示す通りである．表中の N は多角形面素の数，N' は面の重なりが妥当であるかの判定を行った回数，#Ans は得られた解の数，time は計算に要した時間（ミリ秒単位）である．図 11 中の多角形面素に振られている数字は，最初に見つかった解の面の重なり順を表している．

表 1 結果
Table 1 Results.

	N	N!	N'	#Ans	time(ms)
(a)	4	24	10	1	0
(b)	8	4.0×10^4	79	3	10
(c)	18	6.4×10^{15}	8.3×10^4	2778	2.2×10^3
(d)	26	4.0×10^{26}	3.4×10^6	5.0×10^5	1.2×10^5

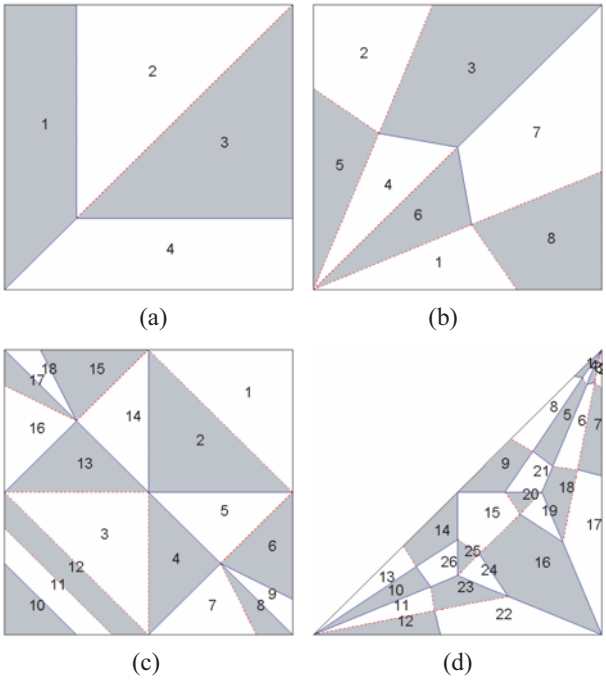


図 11 結果
Fig. 11 Results.

6. 考　　察

単純な例題である図 11(b) については，一見すると中折りであると判断できるが，他に 2 通りの折り方があることが判明し，興味深い結果を得ることができた．図 11 の解は，中折りではない折り方の例である．

兜の例では，実質的に異なる解は著者自身が実際に紙で確認した折り方の場合の数は 4 通りであるが，4 万近くの解が抽出された．これは，互いに関与しない面の重なりが存在する場合，解となる重なり順が増大するためである．例えば，図 12 の例では，左右の多角形面素をそれぞれグルー

プ G_0，G_1 に分け，それぞれの面の数を N_0，N_1 とした場合，${}_{(N_0+N_1)}C_{N_0}$ 通りの，外見的に同一のものが生成されることになる．これは，異なるグループに属する面との重なり順の前後は得られる形状に影響を与えないためである．

鶴の例では，多角形面素の数が 26 であり，全ての場合は実時間で計算しきれないが，効率的に不要な判定を除くことができた結果，判定の数は 3.4×10^6 に収まっている．しかし，この例でも兜の例と同じ理由で解の数が膨大になっている．これらの解の中から，折りたたんだ結果の形状が同一のものを除去し，本質的に異なるものを抽出する後処理を実装する必要があると考えられる．

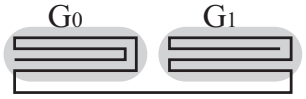


図 12 互いに関与しない面の重なり
Fig. 12 Polygon-elements with no effect to result.

7. 展　　望

本研究で実装したアルゴリズムで，展開図から折りたたみ後の形状を構築できることを示したが，さらに探索の数を減らし，以下に高速化するかが課題である．例えば，図 12 に示した例のように，互いに影響を与えないグループを効率的にまとめることで高速化が図れるかもしれない．また，解が無数に得られた場合に，本質的に異なるものだけを抽出する方法を確立する必要がある．

参 考 文 献

1) 小松英夫: 馬, 折紙探偵団, Vol. 10, No. 6, pp. 22–32 (2000).
2) 川崎敏和: バラと折り紙と数学と, 森北出版株式会社 (1998).
3) 深川英俊: 折紙の数学, 森北出版株式会社 (2002).
4) Lang, R. J.: *Origami Design Secrets: Mathematical Methods for an Ancient Art*, AK Peters, Ltd. (2003).
5) 内田忠, 伊藤英則: 折り紙過程の知識表現とその処理プログラムの作成, 情報処理学会論文誌, Vol. 32, No. 12, pp. 1566–1573 (1991).
6) Miyazaki, S., Yasuda, T., Yokoi, S. and Toriwaki, J.: An Origami Playing Simulator in the Virtual Space, *The Journal of Visualization and Computer Animation*, Vol. 7, No. 1, pp. 25–42 (1996).
7) Kato, J., Watanabe, T., Hase, H. and Nakayama, T.: Understanding Illustrations of Origami Drill Books, 情報処理学会論文誌, Vol. 41, No. 6, pp. 1857–1873 (2000).
8) 三谷純: 2 次元バーコードを用いた紙の折りたたみ構造の認識とモデル化, 情報処理学会研究報告 2005-CVIM-150, pp. 115–122 (2005).
9) 島貫博, 加藤ジェーン, 渡邉豊英: 展開図を用いた折り紙操作過程における手順毎の折り方の構成, 電子情報通信学会技術研究報告, Vol. 102, No. 55, pp. 71–78 (2002).
10) Shimanuki, H., Kato, J. and Watanabe, T.: Construction of 3-D Paper-made Objects from Crease Patterns, *in Proc. of IAPR Conference on Machine Vision Applications (MVA2005)*, pp. 35–38 (2005).
11) 三谷純: 折紙展開図エディタ ORIPA. http://mitani.cs.tsukuba.ac.jp/pukiwiki-oripa/.
12) Kawasaki, T.: On the Relation Between Mountain-creases and Valley-creases of a Flat Origami, *Proceedings of the First International Meeting of Origami Science and Technology*, pp. 229–237 (1989).
13) Justin, J.: Towards a Mathematical Theory of Origami, *Proceedings of the Second International Meeting of Origami Science and Scientific Origami*, pp. 15–29 (1997).
14) Hull, T.: On the Mathematics of Flat Origamis, *Congressus Numerantium*, Vol. 100, pp. 215–224 (1994).
15) Bern, M. and Hayes, B.: The complexity of flat origami, *Proceedings of the seventh annual ACM-SIAM symposium on Discrete algorithms*, pp. 175–183 (1996).