

遺伝的アルゴリズムに基づく紐図形処理

—アヤトリ図形生成方法—

山 田 雅 之[†] 杉 山 貴^{††}
世 木 博 久[†] 伊 藤 英 則[†]

本論文では、空間内の紐を平面図形として表現し、この図形を変形処理する方法について述べる。ここで示される方法は複雑な紐図形を対象としていて、幾何的移動処理と遺伝的アルゴリズムによる処理に基づいており、従来の紐図形処理にくらべ精度の高い図形を生成することができる。本論文ではこの変形処理方法をアヤトリに適用し、いくつかの例をとおしてこの変形処理方法の有用性、および生成される図形の妥当性を示し、さらに、従来の手法との比較を行う。

A String Diagram Processing Based on a Genetic Algorithm

—A Cat's Cradle Diagram Generating Method—

MASASHI YAMADA,[†] TAKASHI SUGIYAMA,^{††} HIROHISA SEKI[†] and HIDENORI ITOH[†]

In this paper, the string in three-dimensional space is represented in a planar string diagram, and a transformation method for this diagram is described. This method is for processing complex string diagrams, and is based on a process using geometric movements and a genetic algorithm, and can generate high quality diagrams. Using this transformation method, a *cat's cradle* shape is generated from a string diagram which represents a history of the finger actions to make the cat's cradle shape. The usefulness of this method is verified by experiments using the cat's cradles. Furthermore, comparisons between this method and the other methods are described.

1. はじめに

紐は衣類に代表されるように生活と密接な関係にある。また紐の諸性質は数学的研究の対象としても注目されており結び目理論という位相幾何学の一分野を成す¹⁾。このような紐を計算機上で表現し、機械的に処理することは興味深い。例えば、位相的に複雑な紐を、その位相構造を変化させることなく変形処理し、さらにその結果を表示する手法の確立とその実現は計算幾何学やグラフィックスなどの立場から見ても意義がある。

本論文では特に、紐の位相構造の複雑さを十分に有する例としてアヤトリ^{2),3)}を取り上げる。これについて著者らはすでに文献 4) において、計算機上での、i)紐状態の表現、ii)アヤトリにおける紐の変形過程

表現、および、iii)これらの表現に基づく紐の変形処理によりアヤトリ出来上がり図形を生成する方法、を述べ、i)とii)については紐状態を表現する平面図形(以降、紐図形と呼ぶ)を用いる方法を提案し、いくつかの例によりこれらの表現方法の有用性を示した。またiii)では、まずアヤトリの動作の履歴を表現する紐図形を与え、この図形に対し、紐図形の幾何的特徴量(交差点数、紐の長さ)の最小化を行うことにより出来上がり図形を生成することを試みた。この最小化はいくつかの幾何学的移動処理により実現され、比較的簡単な図形に対しては最小化が成功し、その結果として良好なアヤトリ出来上がり図形を生成できることを確認した。しかし、図形が複雑になるにつれ、ここで示されている変形手法のみでは最小化が困難となり、良好な出来上がり図形を生成するためには拡張、もしくは異なる手法が必要となることを指摘した。これをうけて、本論文では新たに、遺伝的アルゴリズム(GA)⁵⁾に基づく最小化手法を提案し、これをアヤトリの出来上がり図形の生成のために適用し、その結果に

[†] 名古屋工業大学知能情報システム学科
Department of Intelligence and Computer Science,
Nagoya Institute of Technology

^{††} NEC ソフトウェア中部(株)
NEC Software Chubu Ltd.

ついて述べる。

以下2章では紐状態の表現方法について説明し、3章でGAに基づく紐図形変形処理方法について述べる。さらに、4章ではこの変形処理方法をアヤトリ図形生成のために適用し、幾つかの例によりこの方法の特徴、有用性および、生成される図形の妥当性を示し、最後に文献4)の手法との比較を行う。

2. 紐状態の表現

ここでは紐状態を表現するために図1のような紐図形を用い、これを交差点の位置と連結状態に関するデータより表現する。紐図形において、交差点 c_i (i は交差点の番号) はその近傍にある程度の面積を持ち、四角形で表す。四角形の四つの角 (カド) C_i^n ($n=1, 2, 3, 4$) のうち、互いに対角にある角 C_i^1 と角 C_i^3 、角 C_i^2 と角 C_i^4 を結ぶ対角線はそれぞれ交差点の上方に位置する紐と下方に位置する紐に対応する。また各々の角 C_i^n ($n=1, 2, 3, 4$) には一つの紐が必ず存在し、それぞれ、i) 他の交差点の角 C_j^m ($i \neq j, n=1, 2, 3, 4$) かまたは、ii) 自分自身の交差点の他の角 C_i^m ($n \neq m$)、iii) 端点 e_k (k は端点の番号)、のいずれか一つと連結している (線分によってつながっている)。

3. 紐図形の簡単化処理

ここでは紐図形を簡単化する方法について述べる。ここで簡単化とは紐図形を以下 i), ii) に示す状態に変形することを用いる。

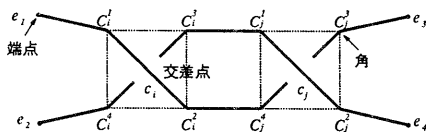


図1 紐図形表現
Fig. 1 A string diagram.

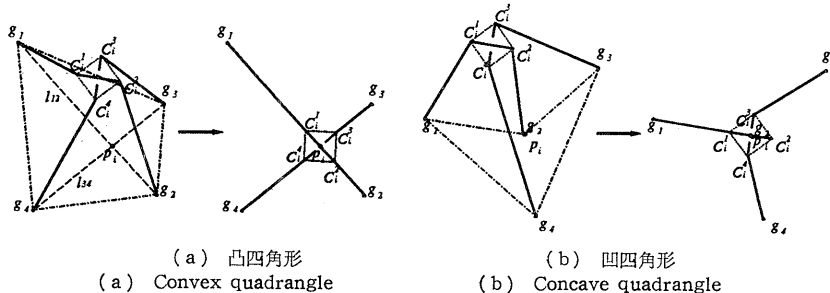


図2 交差点移動
Fig. 2 Crossing movements.

i) 紐の長さ総和が最小である (紐の長さ総和は以下で定義される)。

ii) 交差点数が最小である。

ただし、簡単化の際、紐図形が表現する空間内の紐の位相的性質は保存されるものとする。

3.1 長さ縮小処理

ここではまず紐の長さ総和を定義し、次に紐の長さ総和を最小化するための変形処理を示す。

[紐の長さ総和] 角 C_i^n とこれに連結する他の交差点の角または端点との距離を d_{ci}^n とする。このとき交差点 c_i と連結する紐の長さ d_{ci} を以下とする。

$$d_{ci} = \sum_{n=1}^4 d_{ci}^n. \quad (1)$$

また、端点 e_k とこれに連結する角または他の端点との距離を d_{ek} とし、さらに図形中に存在する交差点と端点の個数をそれぞれ N_c, N_e とする。このとき、紐の長さの総和 L を以下とする。

$$L = \frac{1}{2} \left(\sum_{i=1}^{N_c} d_{ci} + \sum_{k=1}^{N_e} d_{ek} \right). \quad (2)$$

[長さ縮小処理] 以下に示す交差点移動を紐図形中の任意の交差点に繰り返し適用し、紐の長さ総和を減少させる。

[交差点移動] 交差点 c_i の四つの角 C_i^n ($n=1 \sim 4$) が連結している4点を g_n とする (図2)。 d_{ci} を減少させるため以下(1), (2), (3)のように交差点 c_i を移動する。

- (1) 4点 g_1, g_2, g_3, g_4 が凸四角形の頂点となるとき、この凸四角形の二つの対角線の交わる点へ c_i を移動し、さらに(3)を行う (図2(a))。
- (2) 4点 g_1, g_2, g_3, g_4 が凹四角形の頂点となるとき、この凹四角形の凹頂点に c_i を移動し、さらに(3)を行う (図2(b))。
- (3) より d_{ci} が小さくなるように交差点 c_i を回転させる。

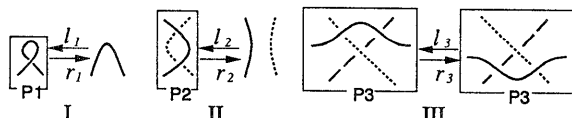
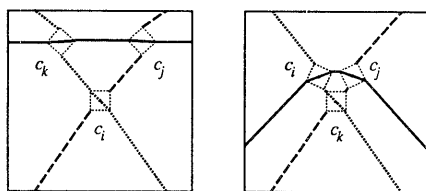


図 3 ライデマイスター移動 I, II, III
Fig. 3 Reidemeister movements I, II, III.



(a) $S > a$ (b) $S \leq a$
図 4 制限つきライデマイスター移動 III
Fig. 4 Restricted Reidemeister movement III.

d_{c_i} が減少するとき、明らかに紐の長さ総和 L も減少する。また交差点移動は紐図形が表現する空間内の紐の位相的性質を変化させない。

3.2 ライデマイスター移動

交差点数を最小化するためにライデマイスター移動 I, II, III¹⁾ (図 3) を適用する。ライデマイスター移動は交差点の連結状態の変形であるが、紐図形が表現する空間内の紐の位相的性質を変化させない。またライデマイスター移動を繰り返し適用することにより交差点数を最小化することができる。

ここではより効率的な紐図形単純化を実現するため、ライデマイスター移動 III に次のような制限を設ける。

【制限つきライデマイスター移動 III】 ライデマイスター移動 III は図 3 に示す図形パターン **P3** に施される。このパターンは三つの交差点 c_i, c_j, c_k からなり、ここではこれらの交差点の接近度 S を次式 (3) により定義する。

$$S = d_{i,j} + d_{j,k} + d_{k,i}, \quad (3)$$

ここで、 $d_{p,q}$ は交差点 c_p と c_q ($p, q = i, j, k$) の距離を表す。

さらにある閾値 TH を設定し、以下のようにライデマイスター移動 III を制限する。

ライデマイスター移動 III は条件: $S < TH$ を満たす **P3** にのみ適用可能である。

すなわち、閾値が $TH = \infty$ であれば、すべての **P3** に対しライデマイスター移動 III が適用可能であり、また、閾値が $TH = 0$ であれば、ライデマイスター移動 III は適用され得ない。ある定数 a を閾値とすることに

```
while  $t \leq T$  do
  while (図形にパターン P1 または P2 が存在する) do
    ライデマイスター移動 I または II を
    これら P1 または P2 に施す;
    長さ縮小処理を施す;
  if (図形にパターン P3 が存在し、
    その接近度  $S$  が  $S < TH$  を満足する) then
    ライデマイスター移動をこの P3 に施す;
   $t := t + 1$ ;
```

図 5 基本単純化処理
Fig. 5 Basic simplifying process.

より、図 4(a) の **P3** にはライデマイスター移動 III は適用できないが、図 4(b) の **P3** には適用できることになる。

この制限の単純化処理における影響は 4 章で示される。

3.3 基本単純化処理

図 5 に示す単純化処理を基本単純化処理と呼ぶ。この処理は長さ縮小処理とライデマイスター移動に基づく。

ここでは図 6 に示す例を用いて基本単純化処理を説明する。

1. まず図形中にパターン **P1** または **P2** である箇所が存在する間、その箇所に対しライデマイスター移動 I または II を施し、次に長さ縮小処理を施す。図 6(a) にはパターン **P1** または **P2** である箇所は存在しない。この図形は長さ縮小処理により図(b)に推移する。
2. 次に、制限付きのライデマイスター移動 III を図形中のパターン **P3** である箇所施す。このとき接近度が閾値より小さい箇所のみライデマイスター移動 III が適用される。図(b)においてはパターン **P3** である箇所が二つあり、ある閾値 a に対しこの二つのいずれの接近度 S も $S < a$ を満足し、それぞれに対しライデマイスター移動 III が適用される。
3. ライデマイスター移動 III の後、パターン **P1** または **P2** である箇所が生成されるならば、この **P1** または **P2** である箇所に対しライデマイスター移動 I または II を適用し、交差点数を減少させる。図(c)ではパターン **P2** が生成されており、これに対しライデマイスター移動 II を適用することにより図(d)が得られる。
4. 再び長さ縮小処理を施すことにより、交差点数と紐の長さ総和が共に最小である図(e)が得られる(これら 1~4 は計算機シミュレーションの過程である)。

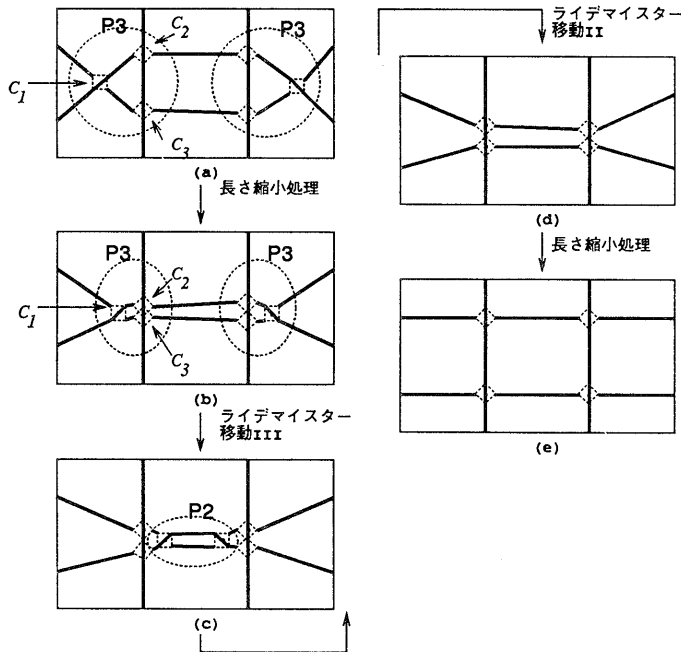


図 6 基本簡単化処理による実行例

Fig. 6 An execution example of the basic simplifying process.

3.4 GA による拡張

上述の長さ縮小処理は紐の長さ総和 L を小さくするには有効な手段であるが、局所的な変形を用いた逐次処理であり、山登り的に L が最小である図形を探索することから一般に L を十分最小化することができない。そこで本論文では、長さ縮小処理に加え遺伝的アルゴリズム (GA) を適用し L を最小化することを試みる。

端点の位置が固定され、交差点の連結状態が一定であるとき、紐の長さ総和 L は図形中の各交差点の位置の組合せにより定まることから、 L の最小化は組合せ最適化問題と見なせる。交差点数が増すにつれ交差点位置の組合せの数も増大するが、GA はこのような組合せの数の大きい最適化問題の近似解法として有用であり、さらに、確率的な処理候補探索アルゴリズムであることから局所解から抜け出すことができるという特徴を持つ。

ここでは GA が不得手とする局所探索のために長さ縮小処理を用い、これにより効率的に紐の長さ総和の最小化を行い、また GA の確率的な処理候補探索により局所解から抜け出し、最適解をみつめることのできる簡単化処理を、長さ縮小処理と GA を組み合わせることにより実現し紐図形の簡単化のために適用

する。以降これを拡張簡単化処理と呼ぶ。

3.4.1 個体の表現

ここでの処理対象は紐図形からなる集合であり、個々の紐図形を個体と呼び、 m 個の個体からなる集合を集団と呼ぶことにする。個体 k ($1 \leq k \leq m$) は紐図形であることから ($Position^k$, $Topology^k$) と表現できる。ここで、 $Position^k$ と $Topology^k$ はそれぞれ個体 k に存在する全交差点の位置に関するデータと連結状態に関するデータを表す。さらに個体 k の交差点数を N_k とし、個体 k における交差点 i の位置 (座標) を p_i^k とすると、交差点の位置に関するデータ $Position^k$ は以下に示すような列により詳細表現できる。

$$p_1^k, p_2^k, \dots, p_{N_k}^k.$$

3.4.2 CRP

拡張簡単化処理では、集団に対し交差点移動とライデマイスター移動からなる処理 CRP (Crossing and Reidemeister movements Process) が施される (図 7)。以下に説明されるように CRP ではライデマイスター移動が実行されるつど、全個体の連結状態 $Topology$ が統一されるがこれは次節に示す遺伝的操作を効果的に行うためである。

CRP はまず、集団から紐の長さ総和が最も小さい個体 k (以後、最良個体と呼ぶ) を選び、個体 k のパターン $P1$ またはパターン $P2$ である箇所に対しライデマイスター移動 I または II を施し、交差点を削除

```

while  $t \leq T$  do
  集団から最良個体  $k$  を選ぶ;
  while ( $k$  にパターン  $P1$  または  $P2$  が存在する) do
    ライデマイスター移動 I または II を
    これら  $P1$  または  $P2$  に施す;
     $Topology^k$  を  $Topology^h$  ( $1 \leq h \leq m$ ) にコピーする;
    全個体に対し長さ縮小処理を施す;
    集団から最良個体  $k$  を選ぶ;
  if ( $k$  にパターン  $P3$  が存在し、
    その接近度  $S$  が  $S < TH$  を満足する) then
    ライデマイスター移動 III をこの  $P3$  に施す;
     $Topology^k$  を  $Topology^h$  ( $1 \leq h \leq m$ ) にコピーする;
   $t := t + 1$ ;
  
```

図 7 CRP

Fig. 7 CRP.

する。このとき、ライデマイスター移動により個体 k の連結状態 $Topology^k$ は変化するが、ここではすべての個体の連結状態を統一するために、変化した個体 k の連結状態 $Topology^k$ を他の個体の $Topology^h$ ($1 \leq h \leq m$) にコピーする。次に集団中の各個体に長さ縮小処理を施し、各々の紐の長さ総和を減少させる。そして再び最良個体を選び、今度は制限つきライデマイスター移動Ⅲを施し、その後、前述と同様に集団中の個体の連結状態を統一する。CRP は期間 T の間上記の処理を繰り返す。

CRP において、各個体の連結状態 $Topology$ は最良個体の連結状態のコピーであり、よって集団中の個体間の相違は交差点位置 $Potition$ のみである。

3.4.3 遺伝的操作

拡張簡単化処理では遺伝的操作を用いて、CRP により得られた各個体の交差点位置 $Position$ を合成し、より紐の長さ総和の小さい個体を生成する。ここでは以下に示す三つの操作、選択、交叉、突然変異、が用いられる。

選択: 式(4)により各個体の適応度 f を求める。

$$f(L) = A/L^2, \quad (4)$$

ここで L は個体の紐の長さ総和とし、 A は定数とする。さらに、集団から一対の個体を選ぶ。このとき適

選ばれた2個体 k, h :
 $k: p_1^k, [p_2^k, p_3^k, p_4^k], p_5^k, p_6^k$
 $h: p_1^h, [p_2^h, p_3^h, p_4^h], p_5^h, p_6^h$
 $\Rightarrow$ 生成される2個体 k', h' :
 $k': p_1^k, [p_2^h, p_3^h, p_4^h], p_5^k, p_6^k$
 $h': p_1^h, [p_2^k, p_3^k, p_4^k], p_5^h, p_6^h$

図 8 交叉: p_2^k と p_2^h を入れ換える

Fig. 8 Crossover: p_2^k and p_2^h are exchanged.

応度の高い個体ほど高い確率で選ばれるものとする。

交叉: 一対の個体の交差点位置を組み換え、新しい交差点位置を生成する。さらに生成された交差点位置を持つ一対の個体を生成する。なおここでは、交差点位置を組み換えるために二点交叉を用いる。この交叉は、まず、 $Position$ 上の二点をランダムに選び、これら二点間にある交差点位置を入れ換える (図 8)。

突然変異: 交叉により生成された個体のある交差点の位置の値を一定の確率で乱数値に置き換える。

これらの遺伝的操作により生成された個体と生成に用いた個体との相違は交差点位置のみであり連結状態は等しい。よって遺伝的操作は個体すなわち紐図形が表現する空間内の紐の位相的性質を変化させない。

3.4.4 拡張簡単化処理

拡張簡単化処理を以下 step 1 から step 5 および図 9 に示す。

step 1. m 個の個体からなる初期集団を生成する。

初期集団の各個体の連結状態 $Topology$ は入力図形 (簡単化したい図形) のそれと同じであるものとし、交差点位置は乱数により与えられるものとする。

step 2. 期間 T の間、集団に CRP を施す。

step 3. 次世代の集団を形成するため遺伝的操作を用いて m 個の個体を生成する。

step 4. 世代数が N になるまで step 2 と step 3 を繰り返す。

step 5. N 世代目の集団から最良個体を選びこれを出力とする。

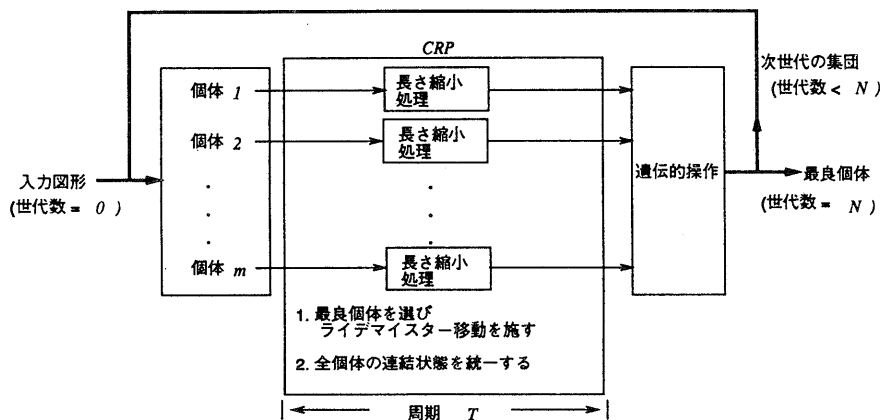


図 9 拡張簡単化処理

Fig. 9 Expanded simplifying process.

4. 応用例アヤトリ

ここでは、文献 4) においてアヤトリ過程を表現するものとして提案された重ね合わせ図形からアヤトリの出来上がり形の生成するため、上述の簡単化処理を用いる。以下ではまず重ね合わせ図形を説明する。

4.1 重ね合わせ図形

〔ネット部とハンドル部〕 アヤトリを図 10(a) のような平面図形を使って表現し、さらに、紐を絡める指を図中では点 (・) で表す。また、図 10(a) のような点を含む紐図形を図 10(b) に示すように二つの領域に分ける。一方の領域は互いに交差する紐のみが存在し、点は存在しない領域である。もう一方の領域は点と紐が双方存在し、かつ、交差点が存在しない領域である。以降、前者の領域をネット(部)、後者の領域をハンドル(部)と呼び、ネットとハンドルを区別するときはハンドルを斜線で図示する。また、ネットとハンドルの境界を紐が通過する点を端点とする。

〔基本動作〕 ここでは、三つの基本動作、i) 取る、ii) 外す、iii) 入れ換える、を定義する。

- i) 取る：ネット部の紐を指で取る。
- ii) 外す：ハンドル部の紐を外す。
- iii) 入れ換える：同一の指に掛かっているハンドル部の二本の紐の内側と外側を入れ換える。

これら i) ~ iii) の基本動作に対応するネットとハンドルの変形を図 10(c) に示す。

〔重ね合わせ図形〕 基本動作に対応する図形のネット部を重ね合わせた図形をネット部の重ね合わせ図形と呼ぶ。

ネット部の重ね合わせ図形によるアヤトリの変形過程表現について述べる。ここではアヤトリの代表的な

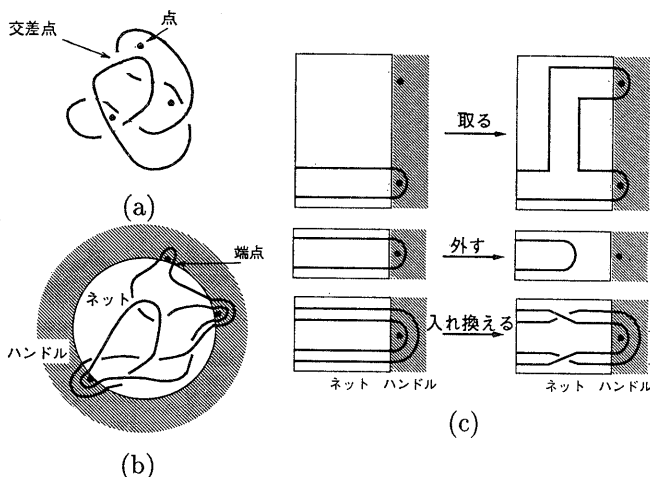


図 10 (a) 紐図形, (b) ネットとハンドル, (c) 基本動作
Fig. 10 (a) A string diagram, (b) Net and Handle, (c) Basic operations.

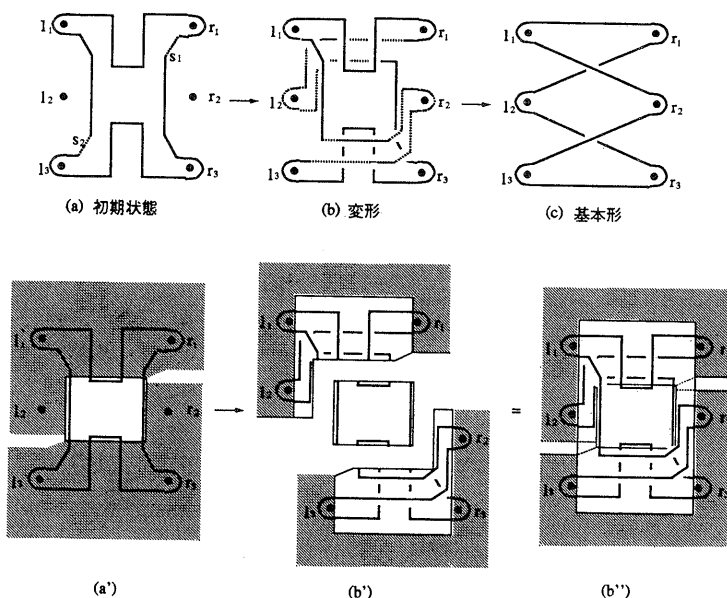


図 11 初期状態から基本形への変形過程の表現
Fig. 11 A representation of transformation process from Initial state to Basic pattern.

例³⁾を用いて基本動作の重ね合わせにより変形過程を表現できることを示す⁴⁾。

〔例 1〕 アヤトリの初期状態を図 11(a) とする。また、ここでは図 11(c) を基本形とする。初期状態から基本形への変形過程は(a)に示す線分 s_1 を指 l_2 で取り、線分 s_2 を指 r_2 で取るにより表現することができる (図 11(b))。この変形過程は図 11(a'),

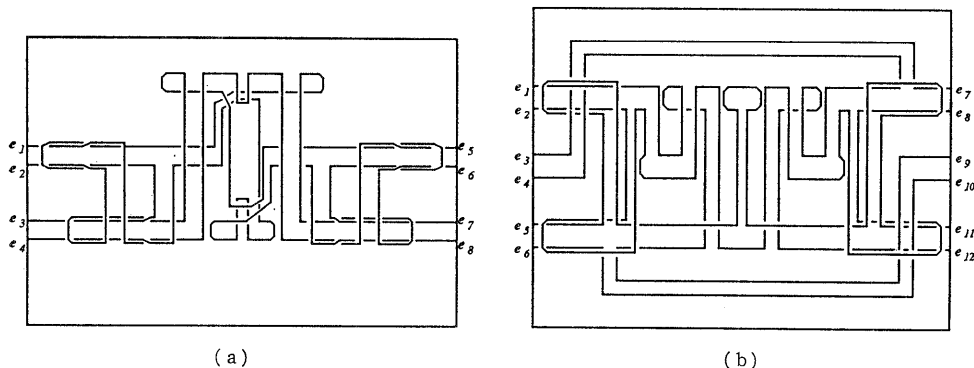


図 12 重ね合わせ図形, (a) 二段ばしど, (b) 星
Fig. 12 Overlapping diagrams, (a) *Bridge*, (b) *Star*.

(b'), (b'')に示すように, 図 11(a')のハンドルを基本動作の「取る」に対応する図形に置き換えることにより表現できる.

【例 2】同様に, 基本動作に対応する図形の重ね合わせにより「二段ばしど」と「星」の変形過程を表現できる. それぞれの重ね合わせ図形を図 12(a)と(b)に示す(図中, ハンドル部は省略する).

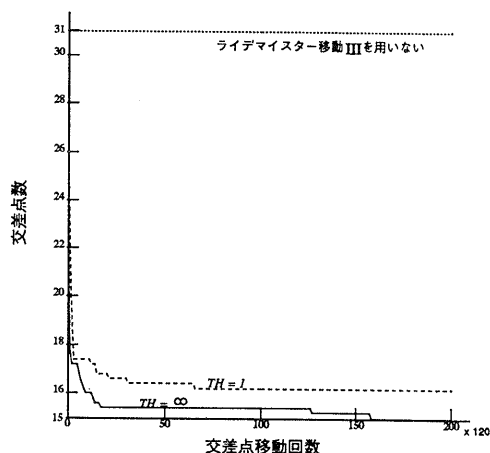
4.2 実行例

二段ばしどの重ね合わせ図形(図 12(a))に本簡単化処理を適用した結果を示す.

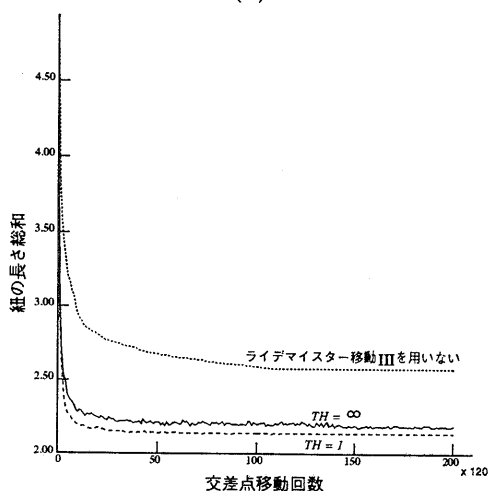
3.2 節で制限つきライデマイスター移動Ⅲについて述べたが, ここではまず, この制限の効果について述べる. 図 13 は基本簡単化処理を用いたときの実行結果であり, 交差点移動を 120 回行うごとに制限つきライデマイスター移動Ⅲを行うものとした.

制限つきライデマイスター移動Ⅲにおける閾値を十分小さく設定すると, ライデマイスター移動Ⅲは実行されず, 交差点数は最小化されない. 図 12(a)はライデマイスターⅠおよびⅡにより交差点数は 31 まで減少するが, それ以下にはならない(図 13(a)). 結果的に図 13(b)に示されるように紐の長さ総和も最小化されない.

これに対し, 閾値を十分大きく設定する($TH=\infty$)ことによりすべてのパターン **P3** に対しライデマイス



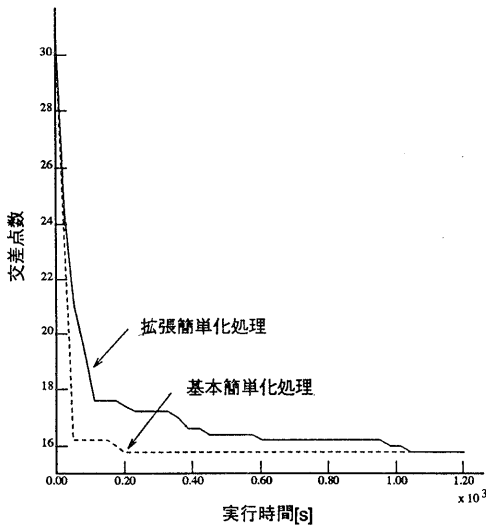
(a)



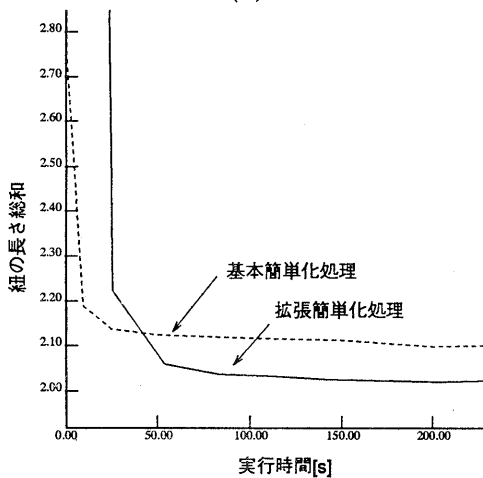
(b)

図 13 閾値 TH の影響, (a) 交差点移動の実行回数と交差点数, (b) 交差点移動の実行回数と紐の長さ総和

Fig. 13 Effects of threshold TH , the number of execution times of the crossing movements vs. (a) the number of crossings and (b) the total string length.



(a)



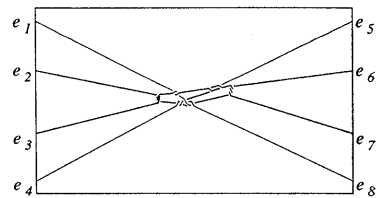
(b)

図 14 基本簡単化処理と拡張簡単化処理との比較,
(a) 実行時間と交差点数, (b) 実行時間と紐
の長さ総和

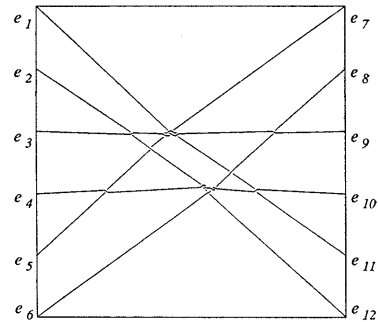
Fig. 14 Comparisons between the basic and the
expand simplifying process, the execu-
tion time vs. (a) the number of cross-
ings and (b) the total string length.

ター移動Ⅲが実行可能となり、それゆえ交差点数は早期に最小化される。しかしライデマイスター移動Ⅲが頻繁に実行され連結状態が安定しないため、図 13 (b)に示されるように、紐の長さ総和もまた安定しない。

以上より閾値の最適化の必要性が示された。閾値を適当な値に設定する ($TH=1$) ことにより交差点数と



(a)



(b)

図 15 生成される図形, (a) 二段ばしど, (b) 星

Fig. 15 Generated diagrams, (a) *Bridge*,
(b) *Star*.

紐の長さ総和を比較的効率的に減少させることができる。

次に基本簡単化処理と GA を用いた拡張簡単化処理との比較を行う。図 14 はそれぞれの処理を 10 回試行し、平均をとったものであり、横軸は実行時間であり縦軸は (a) 交差点数と (b) 紐の長さ総和である。また、拡張簡単化処理における個体数は 20 とし、各個体に 900 回交差点移動を実行することによって遺伝的操作を施した。すなわち、CRP の周期 T は 900×20 回の交差点移動とその間に行われるライデマイスター移動に要する時間の和に相当する。

拡張簡単化処理では、個体数に比例して交差点移動の実行回数が増加するため、個体数が大きいほど簡単化の速度は減少する (図 14 (a) と (b))。しかしながら図 14 (b) に示されるように、基本簡単化処理に比べ紐の長さ総和の小さい図形を発見することができる。

最後に、図 12 (a) と (b) に対し拡張簡単化処理を施すことにより得られる図形をそれぞれ図 15 (a) と (b) に示す。

5. おわりに

文献 4) で述べた簡単化手法は、前処理として交差

点数が極小である図形をライデマイスター移動を用いて生成し、次に、生成された図形に対し長さ縮小処理を施す。この手法の問題点の一つは前処理により得られる図形、すなわち交差点数が極小である図形が一般に複数あることである。例えば図 12(a) の重ね合わせ図形から交差点数を極小にして得られる図形は 5 個ある。図 12(b) においては、85 個の図形が得られる（これらの図形は互いに交差点数は等しいが、連結状態が異なる）。このことから従来までは、各交差点数極小図形に長さ縮小処理を施して得られた図形の中から妥当なものを選び、それをアヤトリの出来上りの図形としていた。この問題を回避するため、本論文では交差点数と紐の長さ総和を同時に最小化する基本単純化処理を提案した。さらに、より良好な図形を得るため、基本単純化処理を GA により拡張することを提案し、その拡張方法を示した。その結果、妥当な図形（図 15）を一意的に生成することができた。しかしながら、この処理方法ではライデマイスター移動の適用のタイミングや適用箇所の最適化が必要であり、これに関しては、制限つきライデマイスター移動Ⅲを提案し実験的考察を行ったに留まっており、より詳細な解析が今後の課題となる。また GA を適用することは比較的精度の高い図形が得られる利点があるが、その反面単純化の速度が低下するため、並列処理などによりこれを改善することが課題となる。

最後に、本手法の欠点と利点について述べ結論とする。

実際のアヤトリには摩擦など幾つかの物理的要素が影響しているため、単純化処理のみではうまくその形を生成できないものがある。理想的な図形を生成するために物理的制約を考慮する必要があるが、ここで示した表現や処理にこれを厳密に反映させるのは困難であり、むしろこのような場合には文献 6)、7) で示されている紐の忠実（物理的）な表現*のほうが適している。

一方、紐のような空間内の複雑な対象を計算機上で表現し変形処理する際、処理の効率化のためには簡単な表現と処理が必要であり、このことから本手法のように幾何的特徴に着目して表現し、幾何的量に関する処理により変形処理を実現することは有効であり、さらにまた GA などの最適化手法の適用も容易となる。

* 3次元デジタル空間を考え、ひもを空間内で接するセルの集合により表現し、ひもの長さを短くする処理を、セルを減少させるアルゴリズムにより実現している。

参考文献

- 1) Kauffman, L. H.: *On Knots, Annals of Mathematics Studies*, Vol. 155, Princeton Univ. Press, Princeton (1987).
- 2) Haddon, K.: *Cat's Cradles from Many Lands*, New York (1911).
- 3) 野口 広: 母と子のやさしいあやとり遊び, 梧桐書院 (1987).
- 4) 山田雅之, Rahmat Budiart, 伊藤英則, 世木博久: アヤトリにおけるひも図形変形過程の表現とその処理, 情報処理学会論文誌, Vol. 35, No. 3, pp. 497-504 (1994).
- 5) Goldberg, D.E.: *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley (1989).
- 6) 齊藤豊文, 横井茂樹, 鳥脇純一郎: 3次元デジタル線図形のトポロジー結び目の解析, 電子情報通信学会技術研究報告, PRU 90-83 (1990).
- 7) Toriwaki, J., Yokoi, S. and Saito, T.: Understanding Forms by Man and Computer Using Computer Graphics and Image Processing, Ishizaka, S. ed., *Proc. 2nd Int. Symposium for Science on Form*, pp. 219-231 (1990).

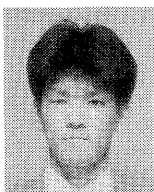
(平成 6 年 2 月 4 日受付)

(平成 6 年 6 月 20 日採録)



山田 雅之 (正会員)

1992 年名古屋工業大学電気情報工学科卒業。1994 年同大学院博士前期課程修了。同年同大学知能情報システム学科助手。並列処理、遺伝的アルゴリズム等に興味を持つ。人工知能学会会員。



杉山 貴 (正会員)

1971 年生。1994 年名古屋工業大学工学部電気情報工学科卒業。遺伝的アルゴリズムによる図形処理に関する研究、人工生命による生態系のシミュレートに関する研究に従事。

現在、NEC ソフトウェア中部(株)第三応用システム事業部勤務。



世木 博久 (正会員)

1979年東京大学工学部計数工学科卒業。1981年同大学院工学系研究科修士課程修了。同年4月より三菱電機(株)中央研究所に勤務。1985年～1989年(財)新世代コンピュータ技術開発機構に出向。1992年4月より名古屋工業大学工学部知能情報システム学科助教授。工学博士。論理プログラミング、演繹データベース等に興味を持つ。電子情報通信学会、人工知能学会、ACM、IEEE Computer Society 各会員。



伊藤 英則 (正会員)

1974年名古屋大学大学院工学研究科博士課程電気・電子専攻満了。工学博士号取得。同年日本電信電話公社入社、横須賀研究所勤務。1985年(財)新世代コンピュータ技術開発機構出向。1989年より名古屋工業大学教授、現在知能情報システム学科所属。これまでに、数理言語理論とオートマトン、計算機ネットワーク通信OS、知識ベースシステムなどの研究と開発に従事。電子情報通信学会、人工知能学会、ファジー学会各会員。