条件付き確率 PIPE

鈴木彰　柴原一友　乾伸雄　小谷善行
東京農工大学

概要

遺伝的アルゴリズムの研究の一分野に分布推定アルゴリズムがある。これを遺伝的プログラミングに応用したアルゴリズムとして PIPE がある。遺伝的プログラミングで生成される木構造の多くではノードの親子間の依存関係が存在する場合が多い。本論文では PIPE でのプログラム進化時に利用される確率ベクトルを親ノードとの条件付き確率により表記することにより、適応度の収束性能の向上をはかった。

PIPE with the Conditional Probability

Akira SUZUKI　Kazutomo SHIBAHARA　Nobuo INUI　Yoshiyuki KOTANI
Tokyo University of Agriculture and Technology
akira-s@fairy.ei.tuat.ac.jp　k-shiba@fairy.ei.tuat.ac.jp　nobu@cc.tuat.ac.jp　kotani@cc.tuat.ac.jp

Abstract

The PIPE proposed by Salustowicz is an estimation of distribution algorithm developed for the genetic programming. We improve the PIPE algorithm by introducing the conditional probability in this paper. A fundamental idea of our method is to describe the occurrence probability of a tree as an individual of the genetic programming using the product of the conditional probability of each node. The conditional probability is obtained by the occurrence probability of a symbol of a child node after a symbol of the parent node is given. Experimental results showed that our method was effective for some problems against the PIPE.

1. はじめに

現在、多くの分野において遺伝的アルゴリズム(以下 GA と呼ぶ)が適用されている。GA とはダーウィンの進化論を起点とし、ある問題の解候補を遺伝子により表現し、遺伝子を１つの解と捉え、遺伝子に交叉や突然変異などの遺伝的操作を行うことにより解空間を効果的に探索する多点型探索アルゴリズムである。

近年、GA の研究の一分野として分布推定アルゴリズム (EDAs:Estimation of Distribution Algorithms)がある[1]。従来の GA の研究では遺伝子の良い部分列(以下、ビルディングブロックと呼ぶ)を保存できるような交叉手法を考案することに注目が置かれたが、遺伝子中のビルディングブロックが遺伝子中に散在している場合、単純な交叉手法ではビルディングブロックが破壊されてしまう。この問題に対し、問題をビルディングブロックの確率モデルとして考え、それにより解空間を探索する手法が EDA である。

EDA では各遺伝子は確率ベクトルにより求められる。確率ベクトルにより個体群を生成し、その個体群から良い個体群を抽出し、良い個体群から確率ベクトルを学習する。この手続きを繰り返すことにより、解空間を探索する。GA での EDA として PBIL[2]、cGA[3]、UMDA[4]等がある。

一方、GA を拡張した手法として遺伝的プログラミング(以下、GP と呼ぶ)[5]がある。GA では遺伝子の長さは固定であるが、GP では遺伝子の長さは可変である。また、GP では問題構造に依存した表現で解を示すことができるという特徴がある。

本論文では、GP に EDA を応用したアルゴリズムである PIPE[6]をノード間の依存関係を利用する

ことにより改良し、適応度の収束速度の向上を目的とする。

本論文は次のように構成されている。第2章ではGPの説明を行う。第3章では従来のPIPE手法を示す。第4章では提案する条件付き確率PIPEを示す。第5章では提案手法の有効性を実験により示す。第6章では考察を行い、第7章で条件付き確率PIPEのまとめを行う。

2. 遺伝的プログラミング

GPではプログラムを木構造(遺伝子)で表現する。遺伝子は1つの解候補を表し、個体と呼ばれる。

GPの基本的なアルゴリズムは次のようになる。

Step1. 初期個体群を生成する
Step2. 各個体の適応度(評価値)を計算する
Step3. 交叉、突然変異、選択などの遺伝的操作を行うことにより次世代を生成
Step4. 終了条件を満たすまで、Step2～Step3を繰り返す

GPではGAに比べ、次の2つの特徴がある。

(1) 遺伝子の長さが可変である
(2) 問題構造に依存した表現で解を示せる

遺伝子の長さが可変であるため、固定長の遺伝子をもつGAに対し、より高い表現力を持たせることができる。また、GAでは多くの場合、0,1の2値により遺伝子を表現するか、実数値により遺伝子を表現するのだが、GPでの解は問題に依存した記号で示される。例えば、関数近似の問題であれば四則演算やlogやexp等の数学関数などにより解を表現する。GPの個体例を図1に示す。これは、解(logX+(1*2))を表している。

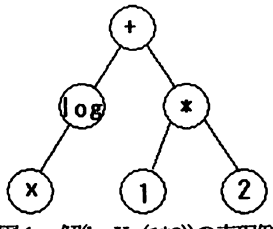

図1 解(logX+(1*2))の表現例

3. PIPEアルゴリズム

PIPE(Probabilistic Incremental Program Evoluation)[6]とは、PBILとGPを組み合わせた手法である。PIPEでは各ノードに確率ベクトルを用意する。確率ベクトルはそのノードにどの記号がどれくらいの確率で出現するかを決めたもので、その確率に基づいて個体群を生成する。その個体群の内、もっとも適応度の高い個体方向へ確率ベクトルを学習させる手法である。PIPEでの木構造の例を図2に示す。

ノード2の
確率ベクトル

+	0.02
-	0.03
*	0.05
/	0.05
log	0.05
1	0.5
2	0.3

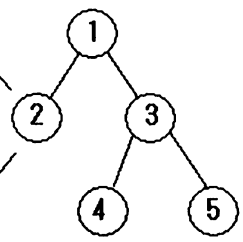

図2 PIPEでの木構造例

PIPEの具体的なアルゴリズムは次のようになる。

PIPEアルゴリズム

Step1. 確率ベクトルを初期化する
Step2. 終了条件を繰り返すまで Step3～Step4 を繰り返す
Step3. 一定確率でGBLを行う。残りはELを行う
Step4. 一定確率で確率ベクトルを突然変異する

GBL(Generation Based Learning)は実際に集団を生成してから最良個体に関して、確率ベクトルを更新する手法であり、EL(Elite Learning)は進化中のエリート解から確率ベクトルを更新する手法である。そのアルゴリズムは次の様になる。

GBLアルゴリズム

Step1. 確率ベクトルにより個体群を生成
Step2. 個体群を評価する
Step3. 最良個体から確率ベクトルを学習

ELアルゴリズム

Step1. 進化中のエリート解から確率ベクトルを学習

確率ベクトルの学習の更新式は次のようになる。まず、N個の個体群の中で最良個体をProg_b、進化中で見つかった最良解をProg_{el}とすると、Prog_bが生成される確率は次の式(1)のようになる。

$$P(\mathrm{Prog}_b) = \prod_{j \in \mathrm{Prog}_b} p(I_j) \qquad (1)$$

$P(\mathrm{Prog}_b)$:プログラムbが生成される確率
$P(I_j)$:記号I_jの出現確率

ここで、学習ではこのプログラムbが生成される確率を増やせばよいので、次の式(2)のようになる。

$$P_{target} = P(Prog_b) + (1 - P(Prog_b)) \cdot lr \cdot \frac{e + FIT(Prog_{el})}{e + FIT(Prog_b)}$$

REPEATUNTIL $P(Prob_b) \geq P_{target}$

$$P(I_j) = P(I_j) + lr \cdot (1 - P(I_j)) \quad (2)$$

ただし、$j \in Prog_b$

P_{target}:学習の目標確率

$P(Prog_b)$:プログラムbが生成される確率

lr:学習率

e:適応度定数

FIT(Prog):プログラムの適応度

$P(I_j)$:プログラムbの

ノードjの記号I_jの出現する確率

この式では適応度により学習の目標確率が変わるので、エリート解より適応度が明らかに悪いとその方向へは学習されにくい。

確率ベクトルの突然変異は次の式(3)、式(4)のようになる。式(3)により突然変異の起こる確率を求め、突然変異による確率ベクトルの更新式は式(4)のようになる。

確率ベクトルの更新が行われた後、正規化を行い確率の和が1になるようにしている。

$$P_{M_P} = \frac{P_M}{z \cdot \sqrt{|Prog_b|}} \quad (3)$$

P_{M_p}:突然変異確率

P_M:突然変異パラメータ

z:命令数(終端記号の種類数+

非終端記号の種類数)

$|Prog_b|$:プログラムbの木の大きさ

$$P(I) = P(I) + mr \cdot (1 - P(I)) \quad (4)$$

P(I):突然変異対象ノードの

記号の確率

mr:突然変異の学習率

4. 条件付き確率PIPE

第3章でPIPEを説明したが、PIPEでは各ノードは独立として扱っている。一方、EDAでは多変数間の関係を考慮して設計する手法が多数提案されている[7][8]。PIPEにおいても、2変数間の関係を取り入れることにより、より高い表現力を実現することができる。

解が木構造で表される問題では近接ノード間に依存関係が存在すると予想できる。この考えをPIPEに取り入れる。各ノードは親ノードの非終端記号(子ノードを必要とする記号)の数だけ確率ベクトルを用意し、親ノードの記号により各ノードの記号の出現確率を求める。この方法を本論文では条件付き確率PIPEと呼ぶ。その表現例を図3に示す。

ノード1での記号

ノード2での出現記号	+	-	*	/
+	0.1	0.1	0.2	0.3
-	0.2	0.1	0.05	0.1
*	0.1	0.4	0.05	0.15
/	0.2	0.1	0.2	0.2
1	0.3	0.2	0.1	0.15
2	0.1	0.1	0.4	0.1

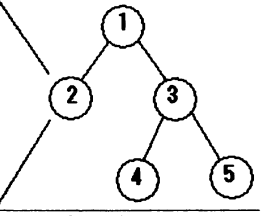

図3 条件付き確率PIPEでの表現例

条件付き確率PIPEのアルゴリズムは通常のPIPEアルゴリズムと同じものを用いるが、学習の更新式を変更する。まず、プログラムbの生成確率は次の式(5)のように定義する。

$$P(Prog_b) = \prod_{j \in Prog_b} P(I_j \mid I_k) \quad (5)$$

ただし、kはjの親ノード

$P(Prog_b)$:プログラムbが生成される確率

$P(I_j \mid I_k)$:

プログラムbのノードjの親ノードkの記号I_kが出現したときの子ノードjの記号I_jの出現確率

確率ベクトルの学習式は次の式(6)のようになる。

$$P_{target} = P(Prog_b) + (1 - P(Prog_b)) \cdot lr \cdot \frac{e + FIT(Prog_{el})}{e + FIT(Prog_b)}$$

REPEART UNTIL $P(Prob_b) \geq P_{target}$

$$P(I_j \mid I_k) = P(I_j \mid I_k) + lr \cdot (1 - P(I_j \mid I_k)) \quad (6)$$

P_{target} :学習の目標確率

$P(Prog_b)$:プログラムbが生成される確率

lr:学習率

e:適応度定数

FIT(Prog):プログラムの適応度

$P(I_j|I_k)$:

プログラムbのノードjの親ノードkの記号I_kが出現したときの子ノードjの記号I_jの出現確率

突然変異に関しては、突然変異が起きるかの判定式である式(3)は従来のPIPEと変わらないが、式(4)の更新式に関しては、次の式(7)のようになる。

$$P(I|I_k) = P(I|I_k) + mr \cdot (1 - P_j(I|I_k)) \quad (7)$$

$P(I|I_k)$:

突然変異対象ノードの親ノードkの記号が出現したときの突然変異対象ノードの記号の確率

mr:突然変異の学習率

5. 実験

5.1 実験概要

適応度の収束速度を調べるために、従来のPIPEと条件付き確率PIPEの比較実験を行う。関数近似問題、11マルチプレクサー問題、遭難者を探せ問題の3つの問題で、従来手法と提案手法を比較する。

5.1.1 関数近似問題

関数近似問題とはGPにおいてもっとも基本的な問題であり、ある関数 f(x)の n 個の観測値 $\{f(x_1), f(x_2) \cdots f(x_n)\}$ から元の関数を特定する問題である。

この問題をまとめると表1のようになる。

表1 関数近似問題

目的	Sin関数の近似関数を求める。
適応度事例	100個の観測値(x,sin(x))
非終端記号	+,−,*,／
終端記号	1,2,3,4,5,6,7,8,9,x

適応度の設計は次の式(8)のようにした。

$$\sum_{n=1}^{100}\{(\text{GPによって生まれた関数}) - (\text{目的関数})\}^2 \quad (8)$$

この式(8)の評価関数を設計することにより、関数近似問題を式(8)の評価値を 0 にする最小化問題と考えることができる。

5.1.2 11マルチプレクサー問題

11 マルチプレクサー問題とは回路設計問題の 1 つで、11マルチプレクサーとは次の図4のようなものを言う。

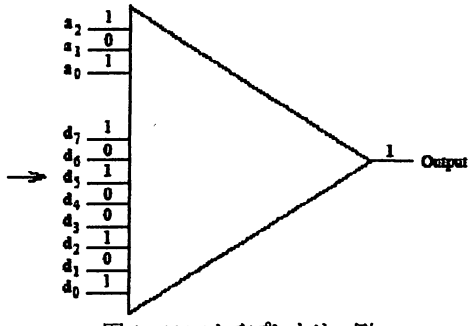

図4 11マルチプレクサー例

11 マルチプレクサーとは 3 つのアドレスビットと8つのデータビットを入力とし、アドレスビットが指すデータビットを出力する素子である。図4の例ではアドレスビットが$(101)_2 = 5$を指しているので、データビットd_5の値を出力する。

11マルチプレクサー問題とはこの11マルチプレクサーと同等の振る舞いをする素子をプリミティブな関数により表現する問題である。

この問題をまとめると表2のようになる。

表2 11マルチプレクサー問題

目的	11 マルチプレクサーをプリミティブな関数により製作する。
適応度事例	11 マルチプレクサーの全入力パターン($2^{11} = 2048$通り)
非終端記号	IF,AND,OR,NOT
終端記号	3つのアドレスビット a1,a2,a3 8つのデータビット d1,d2,d3,d4,d5,d6,d7,d8,d9

評価関数は次の式(9)のように設計した。

$$(\text{適応度}) = 2048 - (\text{正しく分類できた数}) \quad (9)$$

この式(9)では先の関数近似問題と同様に適応度

が0に近ければ近いほど、良い解と言うことができ、適応度が0になれば、11マルチプレクサーと同じ機能を持つ解を生成できたと言える。

5.1.3 遭難者を探せ問題

遭難者を探せ問題とは、日経サイエンスのホームページ[9]に掲載されているパズルであり、次の様な問題である。

『遭難者が1辺10kmの正方形状の原野のどこかで大怪我をしてしまった。歩けないので救難信号を発信しているが，この信号の到達距離は2kmだ。救助隊がその電波の到達範囲に入れば，方向探知機を使って直ちに遭難者を助けに行くことができる。あなたの任務は，救難信号の届く範囲に，できるだけ早く，確実に入ることだ。ただし救助隊の四輪駆動車は，この正方形の辺上であれば，どの地点から進入してもよいものとし，連続した線に沿ってのみ動けるものとする。また，その速度は10分に1kmとする。300分以内に確実に遭難者を見つける方法を考えてもらいたい』

この問題は次の図5のようになる。

図5 遭難者を捜せ問題

ただし、このままでは、最適化する要素として、自動車の進入座標と自動車の軌跡があり、単純にGPを適用することはできない。そこで、単純化遭難者を捜せ問題として次の2点の条件を加える。

- 進入座標は(0,5)とする
- 遭難者を500分以内に見つける

進入条件を(0,5)としたことで、自動車の軌跡を求めるプログラムをGPで生成する。また、進入座標を指定したことで、最適解が300分以内で見つからない恐れがあるので、500分に延長する。

この問題は次の表3のようになる。

表3 遭難者を捜せ問題

目的	遭難者をできるだけ短い時間で捜す経路を見つける
適応度事例	10×10 マス内の 1000 個の点(遭難者)
非終端記号	ATG,IFLTZ,+,−,*,／
終端記号	X,Y,R

遭難者がどの位置にいても探せる経路ということは次のように考えた。

「10km×10km の正方形内に遭難者を無数にばらまき、すべての遭難者を制限時間内に探せる経路」

遭難者の数を増やせば増やすほど、その経路が問題の解である精度が高まるのは明かであるが、ここでは、10km×10km の正方形に 1000 人のランダムな位置の遭難者を配置する。

遭難者を探せ問題でのGPへの入力は現在の自分のXY座標値を入力する。GPで生成されたプログラムは入力されたXY座標値での車の進行角度を返す。

評価関数は次の式(10)のように設計した。

$$(\text{適応度}) = \sum_{i \in GN} \text{TIME}_i + 50500 * \text{NN} \qquad (10)$$

GN : 1000個の遭難者でゴール(救出)できた数

TIME_i : 救出した遭難者iを救出するまでにかかった時間

NN : 1000個の遭難者で救出できなかった数

5.2 実験結果

5.2.1 関数近似問題

関数近似問題の実験結果を図6、図7に示す。

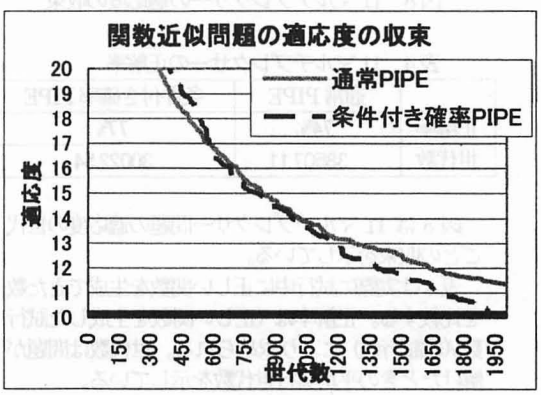

図6 関数近似問題の適応度の収束

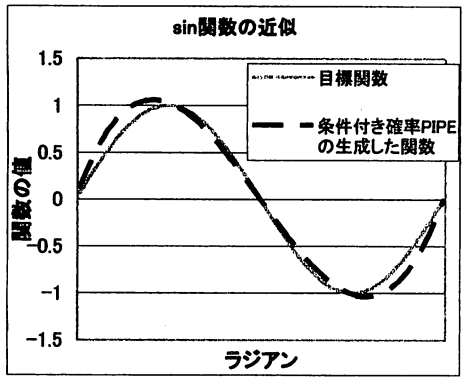

図7　条件付き確率の生成した関数

図6は世代数ごとの適応度の収束を示している。

図7は条件付き確率PIPEの生成した関数が実際のsin関数に近似できているかを示した物である。

5.2.2　11マルチプレクサー問題

11マルチプレクサー問題の実験結果を図8、表4に示す。

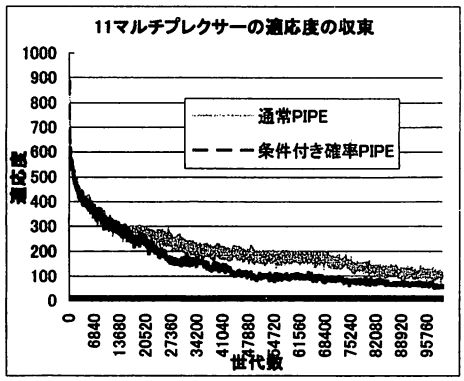

図8　11マルチプレクサーの適応度の収束

表4　11マルチプレクサーの正解率

	通常PIPE	条件付き確率PIPE
正解率	74%	77%
世代数	38507.11	30022.54

図8は11マルチプレクサー問題の適応度の世代ごとの収束を示している。

表4は実際に試行中に正しい関数を生成できた数を比較する。正解率は（正しい関数を生成した試行数)/(総試行数）により求められる。世代数は問題が解けたときの平均実行世代数を示している。

5.2.3　遭難者を探せ問題

遭難者を探せ問題の実験結果を図9、図10に示す。

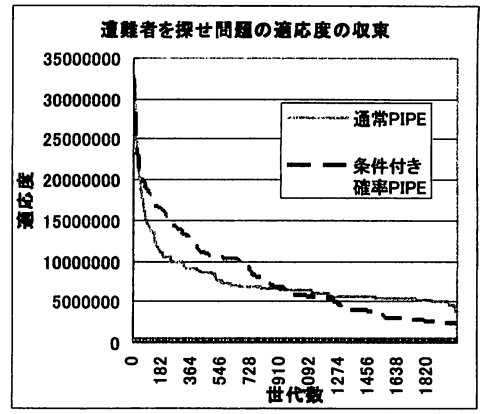

図9　遭難者を探せ問題の適応度の収束

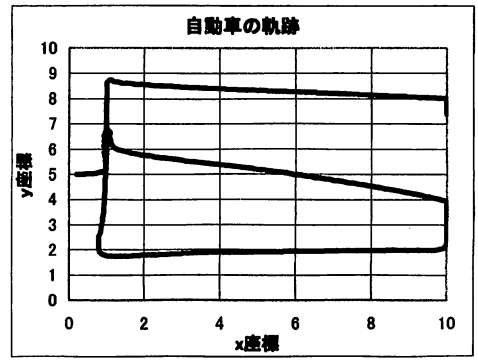

図10　最良解の自動車の軌跡

図9は遭難者を探せ問題での適応度の収束を示している。遭難者を探せ問題では世代数の上限を2000に設定した。

図10は条件付き確率PIPEが生成した最良解の移動軌跡である。この軌跡では、ランダムにばらまかれた1000人すべてを救出する軌跡できる。また、1000人すべてを救出するまでにかかった時間は約400分であった。

6.　考察

実験結果から通常のPIPEに比べ、適応度の収束性能という点で提案手法が有効であることが言える。これは、例えば、関数近似問題では従来のPIPEでは、イレギュラーな木、例えば、（3／0）などの生成を防ぐ事ができなく、また、親ノードの学習結果が変化したときに従来のPIPEでは、子ノードにおいても同様に今までの学習情報が有効に利用できなくないことが多くなるが、条件付き確率PIPEでは

親ノードの学習結果に関わらず、常に学習情報が有効に利用できるためだと考える。

条件付き確率PIPEの有効性は、関数近似問題や11マルチプレクサー問題、遭難者を探せ問題など、多種の分野の問題においても一定の有効性が見られることから、問題に依存することは無いと考えられる。しかし、図6や図8の適応度収束と比べ、図9での収束は進化の初期では通常のPIPEの方が、収束が速く、進化が進むにつれ条件付き確率PIPEの方が、収束性能が高くなるということが分かる。これは、条件付き確率PIPEでは、各ノードは親ノードの非終端記号の数だけ確率ベクトルを用意する事になり、学習対象の確率ベクトルの数が増える。このため、条件付き確率PIPEではより多くの学習回数が必要になるからだと考えられる。故に、進化初期では学習が進んでいないため、適応度の収束が遅くなるが、学習が充分進むにつれ、適応度は通常のPIPEより速く収束したものと考えられる。

条件付き確率PIPEの生成した結果を見てみると、関数近似問題では充分近似できている関数が生成されていると思われるが、遭難者を捜せ問題では、10km×10kmの正方形の端に接した経路が生まれる事など、良い解が生成されているとは思われない。これは、いくつかの原因が考えられる。第一に終端記号、非終端記号の設計であり、設計により、解の表現力が違ってくる。また、先ほど説明した、条件付き確率PIPEでの必要な学習回数は設計した非終端記号の数に比例すると考えられる。よって、進化中に確率の極端に低い非終端記号を動的に減らしていくなどの工夫が考えられる。第二にPIPEの探索上の性質の問題がある。PIPEでは通常のGAに比べ、交叉や突然変異などの遺伝的操作にランダム性が少ないため、大域探索性能が低い。つまり、図10の解はすでに局所解に落ち込んでしまっている可能性がある。

7. おわりに

本論文では、多くのGPのプログラムに見られる近接ノード間の強い依存関係を利用することにより、従来のPIPEより収束性能の高い条件付き確率PIPEを提案した。

今後の課題として、親ノードとの関係のみではなく、異なるノード間の依存関係を発見し利用できるような枠組みを作ること、非終端記号の数と適応度収束性能の検証、遭難者を探せ問題での最適なGPパラメータ設計などがあげられる。

参考文献

[1]倉橋 節也, 勝又 勇治, 寺野 隆雄：ベイジアン最適化手法と分布推定アルゴリズムの動向, 人工知能学会誌18巻5号, pp.487-493, 2003.

[2]Baluja : Population-Based Incremental Learning : A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning , Technical Report CMU-CS-94-163, Pittsburgh, 1994

[3] Georges Harik : The Compact Genetic Algorithm, IEEE-EC, 1998.

[4] H. Mühlenbein, G. Paaß : From recombination of genes to the estimation of distributions I. Binary parameters, 1996.

[5]John R.Koza : Genetic Programming, The MIT Press, 1992.

[6] Rafal Salustowicz, Jürgen Schmidhuber : Probabilistic Incremental Program Evolution, Machine Learning: ECML-97, 1997.

[7] Martin Pelikan, Heinz Mühlenbein : The Bivariate Marginal Distribution Algorithm , Advances in Soft Computing - Engineering Design and Manufacturing, 1999.

[8] Jeremy S. De Bonet, Charles L. Isbell, Jr., Paul Viola : MIMIC: Finding Optima by Estimating Probability Densities, Advances in Neural Information Processing Systems, 1996.

[9] 日経サイエンスホームページ : http://www.nikkei-science.com/