条件付き確率に基づく分布推定アルゴリズムによるプログラム進化

柳井孝介† 伊庭斉志†

本論文では，確率モデルに基づくプログラム進化の手法を提案する．提案手法では確率変数が依存関係を持つ確率分布モデルを 2 つ組み合わせ，プログラム集団の確率分布を推定することによりプログラム進化を行う．我々は本手法を XEDP（eXteneded Estimation of Distribution Programming）と呼んでいる．本研究では遺伝的プログラミングおよび先行研究で提案されているモデルと比較実験を行い，提案手法が高い探索性能を持つことを示した．また，実験結果に基づいて，XEDP の確率分布モデル，ロボットプログラムの進化への適応可能性，ブロートについて考察した．

Program Evolution Based on Estimation of Distribution Algorithm Using Conditional Probabilities

Kohsuke Yanai† and Hitoshi Iba†

This paper proposes a novel technique for a program evolution. Our method combines two probabilistic distribution models, in which probabilistic variables have dependency relationships, and a program population is evolved by the repetition of the estimation of distribution and the program generation without any crossover and mutation. We call this framework eXtended Estimation of Distribution Programming (XEDP). This paper shows results of comparative experiments with GP and other related methods. We empirically comfirm that XEDP has higher search performance. Thereafter, we discuss XEDP's distribution model, application to evolution of robot program and bloat.

1. はじめに

本論文では，確率モデルに基づくプログラム進化の手法を提案し，提案手法の特徴について論じる．本研究では木構造で表されたプログラムを進化させることを試みる．

プログラム進化の手法として一般的に知られているものは，Koza によって提案された遺伝的プログラミング（Genetic Programming：以下 GP）である[10)]．GP は進化論的計算手法の代表例であり，現在さかんに研究され，実際的な応用例も数多く発表されている[10),11),27)]．Koza らは，GP が生成した成果のうち人間の創造性に匹敵するものとして 37 の例をあげている[11)]．このうちは 24 の成果は過去に発明され特許となっているもの，あるいは今日，特許取得可能な新しい発明である．現在，欧米の GP 研究者は遺伝的アルゴリズムなどの進化論的計算手法のなかの 3 分の 1 にのぼるといわれている[27)]．

GP においては，様々な交叉と突然変異の方法が提案されている．たとえば，一様交叉や一点交叉[18)]，相同性交叉[12)]，深さ依存型交叉[8),9)]，マクロ突然変異[2)]，自己適応型交叉[1)]，スコアリング交叉[7)] などが提案されている．その一方で，プログラム進化の基本的なアルゴリズムに関しては，GP と同等のパフォーマンスを持つモデルは提案されていない．

本研究では GP とは異なるメカニズムでプログラム進化を行う手法を提案する．提案手法は集団探索ではあるが交叉と突然変異は用いず，プログラムの確率分布を推定することにより進化を行う．我々は確率分布に基づくプログラム進化の手法として，Estimation of Distribution Programming（以下 EDP）を提案している[24),25)]．本論文では EDP を拡張したモデルである Extended Estimation of Distribution Programming（以下 XEDP）を提案する．本論文では GP の標準的な問題に対して XEDP を適用し，GP との比較実験を行い，XEDP と GP の違いについて考察する．

以下では，まず 2 章で研究の背景について述べ，次に 3 章で提案手法について述べる．4 章では XEDP と GP の比較実験の結果を示し，XEDP の特徴について論じる．5 章で XEDP の確率分布モデル，ロボットプログラムの進化への適応可能性について考察し，

† 東京大学大学院新領域創成科学研究科基盤情報学専攻
Department of Frontier Informatics, Graduate School of Frontier Sciences, The University of Tokyo

6 章で結論と今後の課題について述べる．

2. 従来研究

近年，確率モデルに基づく進化アルゴリズムが注目を集めている．これらは EDAs（Estimation of Distribution Algorithms）[14]，あるいは PMBGAs（Probabilistic Moded-Building Genetic Algorithms）[17] と呼ばれ，提案手法も EDAs の一手法といえる．EDAs では集団における個体の分布に着目し，集団内の優秀な個体集団の確率分布を推定し，推定された確率分布を用いて次世代の個体を生成する．EDAs についての詳細は文献 26)，28) を参照されたい．EDAs は多くの問題領域で，従来の GA を凌ぐ性能が得られたとの報告がなされており，エッジヒストグラムを用いて順序表現問題に EDAs を適用する研究[29] や EDAs と強化学習を組み合わせて最適化を行う研究[15]，EDAs をがん細胞分類に応用した研究[16] などがある．

一方，確率モデルに基づく進化アルゴリズムをプログラム進化に適用した研究は例が少ない．Salustrowicz らは，確率モデルを用いたプログラム進化の手法として，PIPE（Probabilistic Incremental Program Evolution）[19] を提案している．PIPE は PBIL（Population-Based Incremental Learning）[3] をプログラム進化に応用したモデルであり，PPT（Probabilistic Prototype Tree）と呼ばれる木構造の確率分布を用いてプログラムの推定と生成を行う．PIPE で用いる確率分布は確率変数がプログラム木の位置に固定されており，さらに互いに独立である．ゆえに，PIPE は親ノードや子ノードなどの周辺のシンボルを無視し，木構造内の位置に依存した分布の推定と記号の生成を行う．PIPE は確率変数がすべて互いに独立であると仮定している点と確率分布を 1 つしか用いていない点で，本研究と大きく異なる．本研究では，確率変数が依存関係（従属関係）を持つ確率分布モデルを 2 つ組み合わせてプログラム進化を行う．

また，PIPE を拡張した手法として eCGP（Extended Compact Genetic Programming）[20] が提案されている．eCGP は PPT の確率変数をいくつかのグループに分割し，グループ内の変数の同時確率分布を推定する．PIPE ではすべての確率変数は互いに独立であったのに対し，eCGP ではいつくかの確率変数を組にして，それぞれの確率変数の組ごとに同時確率を推定する．ゆえに，eCGP は確率変数の依存関係を考慮しているモデルである．eCGP では，互いに独立な変数の集合から進化が始まり，MDL（Minimum Description Length）情報量基準を用いて，世代ごとに確率変数のグループの併合を行う．

我々は確率モデルを用いたプログラム進化の手法として EDP を提案している．EDP では，確率変数が互いに独立な PPT の代わりに，木構造のベイジアンネットワークを用いる．各々の確率変数は親のノードの変数のみと従属関係にある．

文献 22) では，確率文法を用いたプログラム進化の手法として GMPE（Grammar Model-based Program Evolution）が提案されている．GMPE では，文脈自由文法のルールを用意し，それぞれのルールの適用確率を学習することによってプログラム進化を行う．文献 4)，21)，23) でも，確率文法を用いた同様の研究がなされている．

提案手法の基本的なアルゴリズムは，確率モデルを用いたプログラム進化の先行研究と類似している．すなわち，プログラム集団の確率分布のモデルを仮定し，確率分布の推定と推定された確率分布に基づくプログラム生成を繰り返すことによって進化を行う．本研究と先行研究の違いは，使用する確率分布モデルにある．先にあげた先行研究では，我々の提案している EDP も含め，手法の応用範囲が特殊な問題に限定されており，一般にプログラム進化のベンチマークとして知られる問題での有用性が示されていない．特に PIPE 以外の手法では，作為的に作られた GP のだまし問題でしかその有用性が示されていない．本研究では，確率分布のモデルはプログラム進化のパフォーマンスに重大な影響を及ぼすと考え，先行研究とは異なる確率分布モデルを用いることにより，GP と同等のパフォーマンスを持つプログラム進化の手法を構築することを試みる．

3. 提案手法

3.1 XEDP の概要

XEDP は以下の手順に従い探索を行う．

Step 1 初期集団を生成する．
Step 2 個体を評価し，適合度を計算する．
Step 3 終了条件を満たすなら終了．
Step 4 確率分布を学習する．
Step 5 エリート個体を新しい集団にコピーする．
Step 6 確率分布に基づいて個体を生成する．
Step 7 集団を置き換える．

集団数を M とする．Step1 では，初期世代の個体をランダムに M 個生成する．ただし，木の制限サイズに達した場合は強制的に終端記号が生成される．F を非終端記号の集合，T を終端記号の集合とする．

次に，それぞれの個体の適合度を計算し（Step 2），

終了条件を満たせば最適解を報告して終了する（Step 3）．後に述べる実験では，最大の世代数をあらかじめ決めておき，これを終了条件としている．

Step 4 ではトーナメント選択により集団内の r_sM 個の個体を集団内から選択し，選択された個体群の確率分布を推定する．ここで r_s はサンプル比を表す．XEDP で個体を選択するときのトーナメントサイズを以下では T_{xedp} と書くことにする．選択された個体に基づいて確率分布を推定しているため，推定された分布は集団全体の分布よりも望ましい分布であると考えられる．

Step 5 ではエリート戦略を用い，適合度の高い順に r_eM 個の個体を新しい集団にコピーする．ここで r_e はエリート比を表す．Step 6 では残りの $M(1-r_e)$ 個の個体を Step 4 で推定した確率分布を基に生成する．Step 4 で得られた確率分布は，適合度が高い個体に基づいて推定されている．したがって，新しく生成される集団は，前の世代の集団よりも平均して高い適合度を持つことが期待される．

Step 2 から Step 7 は終了条件が満たされるまで繰り返される．

3.2 確率分布モデル

提案手法では，確率木と再帰分布の 2 つの確率分布を組み合わせてプログラム進化を行う．

3.2.1 確 率 木

確率分布モデルとして，木構造のベイジアンネットを用いる．プログラム進化では，プログラム木の最大深さ☆を設定して実験を行うのが一般的である．プログラム木の最大深さを d_{max} とし，非終端記号のうちで最も引数の多い記号の引数の数を a_{max} とする．ベイジアンネットのトポロジは，深さ d_{max} の完全 a_{max} 分木とする．ベイジアンネットの確率変数は，同位置にあるプログラム木のノードに対応する．説明のため，**図 1** のようにプログラム木のノードに番号を割り振る．分かりやすいように図では 2 分木を想定して示してある．実線と点線をあわせた完全木がベイジアンネットのトポロジである．ただし，実際に生成されるプログラムは完全木になるとは限らず，たとえば，図 1 の実線のノードとエッジで示されるように，ベイジアンネットよりも小さい構造になりうる．確率変数 X_i はプログラム木におけるノード i のシンボルを表す．よって確率変数 X_i の変域は $X_i \in F \cup T$ となる．プログラム木内のある位置のノードにおいて，シンボルの生成確率はその親ノードのシンボルにのみ依存する．すなわち，確率木のそれぞれの確率変数はただ 1 つだけの変数に従属し，その従属する変数は親ノードのシンボルを表す．

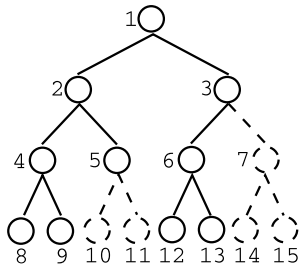

図 1 ノードに番号を割り振ったプログラム木
Fig. 1 Program tree in which nodes are numbered.

たとえば，$F = \{+, -, *, /\}$，$T = \{x_1, x_2\}$ とすると，

$$\mathrm{P}(X_5 = \text{“}+\text{”} \,|\, X_2 = \text{“}/\text{”}) = \frac{2}{7} \qquad (1)$$

は，ノード 2 が / のとき，ノード 5 が + である条件付き確率が 2/7 であることを表す．

確率木により，プログラムの大域的な構造が推定される．我々が過去に提案している EDP は，確率木のみを用いてプログラム進化を行う手法である．また，確率木における確率変数の依存関係をすべてなくしたものが PIPE の PPT である．確率木における変数間の依存関係の必要性は 4 章の実験で示す．

プログラム進化では，有益な部分構造を組み合わせてより大きな部分構造を組み立てることにより，進化が進むと考えられている[10),13),27)]．GP の交叉はこの中心的な役割を果たす．しかしながら，確率木あるいは PPT のみでは，確率変数がプログラム木の中の位置に固定されているため，有益な部分構造を他の位置に移動させることができない[22),24),25)]．この欠点のため，PIPE，eCGP および EDP では，プログラム進化の標準的なベンチマーク問題において，GP と同等のパフォーマンスを得ることができなかったと思われる．そこで本研究では，有益な部分構造を抽出できるようにするため，次項で述べる再帰分布を確率木と組み合わせて用いる．

3.2.2 再 帰 分 布

再帰分布として，次式の条件付き確率を分布モデルとして用いる．

$$\mathrm{P}(Y_1, Y_2, \cdots, Y_{a_{max}} | Y, Y_p) \qquad (2)$$

ここで $Y_1, Y_2, \cdots, Y_{a_{max}}$ は Y が表すノードの子ノードの確率変数であり，Y_p は Y の親ノードの確率変数

☆ 木の根（root）から葉（leaf）までの距離の最大値のことを，「木の高さ」あるいは「木の深さ」という．GP 研究では，「深さ」と呼ぶのが一般的であるので，本論文では「深さ」という表現で統一する．

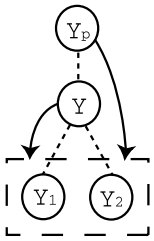

図 2 再帰分布
Fig. 2 Recursive distribution.

である．すなわち，再帰分布は 1 つ上の親と 2 つ上の親のノードシンボルを条件とする子ノードの条件付き確率を表す．**図 2** に再帰分布における確率変数の依存関係を図示する．点線はプログラムの構造を表し，実線の矢印は変数の依存関係を表す．子ノードの同時確率をモデル化しているので，矢印が子ノード全体を指していることに注意されたい．

たとえば，図 1 のプログラム木（実線のノードとエッジ）では，再帰分布の確率分布にマッチングする部分構造は 4 カ所ある．確率変数 Y がノード 2，3，4，6 にそれぞれ対応する．すなわち，ルートノードと末端ノード以外のすべてのノードにマッチングする．子ノードの数が不足しているときは NULL 記号を用いる．たとえば，$a_{max} = 3$ であるのに Y に対応するシンボルの引数の数が 1 の場合は，Y_2 =NULL，Y_3 =NULL となる．

3.3 確率分布の学習

現在の集団からトーナメント選択により $r_s M$ 個の個体を選択する．選択された個体群を S とする．まず，S に基づいて最尤推定を行い，確率木 $\widehat{P}(X_i = x|C_i = c)$ を求める．ここで C_i は X_i の表すノードの親ノードを表す確率変数であるとする．つまり，各ノードごとに親ノードのシンボルに依存した条件付きのシンボル頻度を数え上げ，得られた頻度を正規化して確率を求める．

数式で表現すると次式のようになる．

$$\widehat{P}(X_i = x|C_i = c) = \frac{\#(X_i = x, C_i = c)}{\#(C_i = c)} \quad (3)$$

ここで $\#(X_i = x, C_i = c)$ はノード i が x でかつその親ノードが c である個体の数を表し，$\#(C_i = c)$ はノード i の親ノードが c である個体の数を表す．

式が煩雑になるので，以下では確率分布の引数は書かないことにする．推定された分布 $\widehat{P}$ に基づいて，以下の式を用いて確率木を漸近学習する．

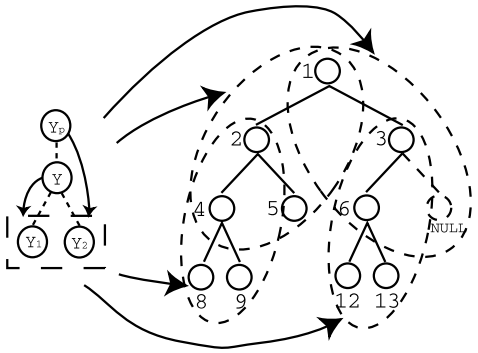

図 3 $\widehat{R}$ の推定
Fig. 3 Estimation of $\widehat{R}$.

$$P' = \eta\widehat{P} + (1-\eta)P_t \quad (4)$$

$$P_{t+1} = (1-\alpha)P' + \alpha\frac{1}{|F|+|T|} \quad (5)$$

ここで P_t は t 世代目の確率木である．η は学習率であり，過去の分布への依存度を表す．0 に近付くほど確率分布の変動は少なくなり，特に $\eta = 1$ の場合は，過去の分布をまったく参照せずに t 世代目の集団のみに基づいて確率分布を更新する．式 (5) では，ラプラス修正[5] として知られる手続きを行っている．α はラプラス修正率を表す定数であり，式 (4) で得られた P' と一様分布との重み付け和をとる．α が 1 に近いほど，P_{t+1} は一様分布に近くなる．ラプラス修正は，確率分布にノイズを入れ，すべての確率を非零にする効果がある．

再帰分布についても，確率木と同様に学習を行う．選択された個体群 S に基づいて，再帰分布 $\widehat{R}$ を最尤推定する．再帰分布の形にマッチングするすべてのノード，すなわちルートノードと末端ノード以外のすべてのノードに対して，再帰的に頻度を数え上げ，$\widehat{R}$ を推定する．**図 3** に $\widehat{R}$ の推定の例を示す．図 3 では，再帰分布の構造にマッチングする部分構造が全部で 4 つあり，この 4 つの構造すべてに対して記号の頻度を数える．これを選択された S 個の個体すべてに対して行う．

R_t を t 世代目の再帰分布とする．確率木の学習と同様に以下の式で再帰分布を学習する．

$$R' = \eta\widehat{R} + (1-\eta)R_t \quad (6)$$

$$R_{t+1} = (1-\alpha)R' + \alpha\frac{1}{|F|+|T|} \quad (7)$$

3.4 プログラムの生成

プログラムの生成は以下の手順に従う．

Step 1 確率木を用いてプログラム T を生成．

Step 2 再帰分布を再帰的に用いて部分木 S を生成．

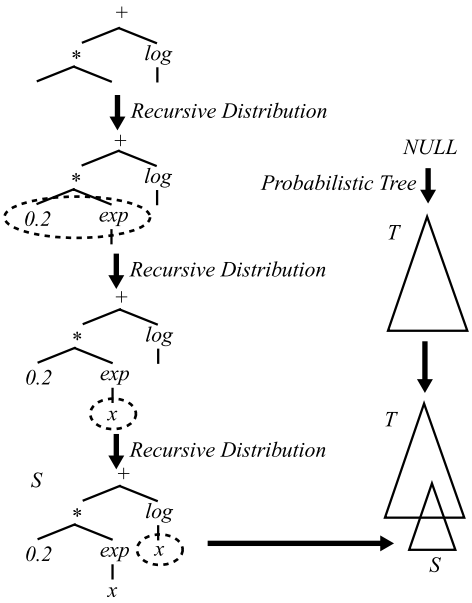

図 4 プログラム生成
Fig. 4 Program generation.

Step 3 T の任意の部分木を S で置換.

まず，確率木 P_t に従ってルートノードから順にノードシンボルを決定し，プログラム T を生成する．次に，深さ 2 の木をランダムに生成し，この木を基に再帰分布 R_t を再帰的に用いて部分木 S を生成する（**図 4** 参照）．ただし，S を T に挿入したときに木の深さが d_{max} を超えないように S を生成する．d_{max} を超える場合は強制的に終端ノードを生成する．Step 3 は P_{rep} の確率で実行する．すなわち，$1 - P_{rep}$ の確率で部分木の置き換えを行わず，T をそのまま新しい個体とする．

4. 比較実験

プログラム探索のベンチマーク問題として知られる最大値問題，6 ビットマルチプレクサ問題，関数同定問題において，提案手法（XEDP）と GP の比較を行った．実験で用いたパラメータを**表 1** に示す．集団数 M や最大世代数，その他のパラメータは，先行研究に従ってそれぞれのベンチマーク問題ごとに変更した．

以下では $prog_i$ で集団の i 番目の個体のプログラム木を表す．プログラム木が引数を持つ場合，$prog_i(X)$ でそのプログラム木に X を代入した値を表す．プログラム木が引数を持たない場合，単に $prog_i$ でそのプログラム木が返す値を表す．また特に明記していない場合，終端記号と非終端記号の定義は文献 10) に従う．GP ではトーナメント選択を用い，トーナメントサイズを T_{gp} で表すことにする．

表 1 XEDP のパラメータ
Table 1 Paremeters for XEDP.

r_e	: エリート比	0.005
r_s	: サンプリング比	0.1
T_{xedp}	: トーナメントサイズ	25
η	: 学習率	0.5
α	: ラプラス修正率	0.1
P_{rep}	: 置き換え確率	0.9

XEDP の特徴を調べるため，XEDP のオペレータを部分的に用いる手法に対しても実験を行い比較を行った．

Type A 再帰分布は用いない．確率木のみを用いてプログラムを生成する．

Type B 確率木は用いない．再帰分布だけを用いてプログラムを生成する．

Type C 再帰分布は用いない．確率木を用いてプログラムを生成した後，プログラムの任意の部分木を突然変異する．

Type D 確率木と再帰分布の両方を用いる．ただし，確率木として確率変数が独立であるモデルを用いる．

Type A は $P_{rep} = 0$ としたモデルであり，EDP と完全に等価である．Type B では図 4 の部分木 S をそのまま個体とする．Type C では再帰分布で生成した部分木で置き換えるのではなく，ランダムに生成した部分木で置き換える．XEDP と Type C を比較することにより，再帰分布による生成が真に有効か否かが分かる．Type D はベイジアンネットの代わりに，完全に独立な確率変数の集合を確率木として用いるモデルである．Type D との比較により，確率木における変数の依存関係の有効性を調べる．最大値問題と関数同定問題ではそれぞれ GMPE と PIPE による探索結果が報告されているので，これらの手法とも比較を行った．また GP は以下の 3 種類のものを用意し，比較を行った．

- 交叉のみの GP
- 突然変異のみの GP
- 交叉と突然変異両方用いる GP

以下では，新しい集団を生成する際に交叉オペレータを用いる割合を P_c，突然変異オペレータを用いる割合を P_m，コピーを用いる割合を P_r とする．

4.1 最大値問題

まず，GP での探索が困難であることが知られている最大値問題において，提案手法の探索性能を調べた．

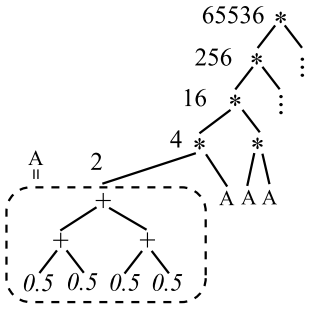

図 5 最大値問題の最適解
Fig. 5 Optimum solution for Max problem.

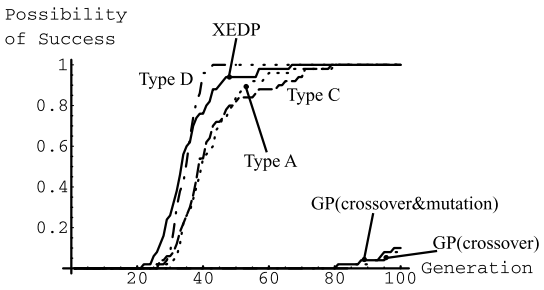

図 6 最大値問題における探索成功率
Fig. 6 Cumulative frequency of successful runs for Max problem.

最大値問題では，$T = \{0.5\}$ と $F = \{+, \times\}$ を用いて最大の値を返すような木を探索する．ただし，木の最大の深さは 7 と制限されている．**図 5** に示すように，0.5 と + で 2 を返す部分木 A を構成し，A を × でかけあわせていく木が最大値を返す．ゆえに，最適解の木は 65536 を返す．0.5 を掛けると，値が小さくなることに注意されたい．この性質のため，最大値問題では早熟な収束を起こし，局所解に陥りやすいことが報告されている[13]．個体 i の適合度 fit_i は

$$fit_i = prog_i \tag{8}$$

とした．

文献 13)，22) に従い，GP の集団数は $M = 200$ とした．交叉のみの GP は $P_c = 0.995$，$P_c = 0$，$P_r = 0.005$，突然変異のみの GP は $P_c = 0$，$P_c = 0.995$，$P_r = 0.005$，交叉と突然変異両方を用いる GP は $P_c = 0.795$，$P_c = 0.2$，$P_r = 0.005$ とした．文献 13) には木の深さが 7 の場合，$T_{gp} = 3$ が最も探索成功率が高いことが報告されているため，本実験でも $T_{gp} = 3$ とした．XEDP の集団数は GP と同様に $M = 200$ とした．以上の設定で，実験を 50 回繰り返した．

図 6 は，最適解が得られた試行の割合を各世代ごとに示している．グラフから分かるように，XEDP では 66 世代目までに，毎回必ず探索が成功しているのに対し，交叉のみの GP では，100 世代目でも成功率は 8% と低い．交叉と突然変異の両方を用いる GP でも同様に探索成功率は低い．突然変異のみの GP では，100 世代までに最適解が得られたことは一度もなかった．また，GP の失敗した試行では適合度がほとんど変化しておらず，進化が行われていなかった．

Type B では，100 世代までに最適解が得られたことは一度もなかったが，その他の確率モデルを用いる手法，すなわち Type A，Type C，Type D および XEDP では，いずれも GP に比べはるかに高い探索性能を示している．ゆえに，最大値問題においては，確率木によるプログラム生成が有効であることが分かる．また，再帰分布を用いないモデルである Type A および Type C に比べ，XEDP と Type D が高い探索成功率を持つことから，再帰分布で生成された木による部分的な置換が有効に働いていることが分かる．

文献 22) には，探索成功率が 60% を超えるためには，GMPE では平均して 13590 回の評価が必要であったと報告されている．XEDP では 13200 回（$= 200 \times 66$）で成功率が 100% となることから，最大値問題においては XEDP の方が GMPE よりも高い探索性能を持つことが分かる．

4.2 6 ビットマルチプレクサ問題

6 ビットマルチプレクサ $F_{6mp} : \{0,1\}^6 \to \{0,1\}$ に等価な木を探索することを考える．実験結果から分かるように，6 ビットマルチプレクサ問題は，交叉が有効な問題である．本節の実験の目的は，交叉が有効な問題における XEDP の探索性能を調べることである．形式的には，F_{6mp} は

$$F_{6mp}(a_0, a_1, d_0, d_1, d_2, d_3) = d_{2^1 a_1 + a_0} \tag{9}$$

と書ける．a_i をアドレスビット，d_i をデータビットといい，6 ビットマルチプレクサはアドレスビットが指定したデータビットの値をそのまま返す．非終端記号と終端記号は，

$$F = \{and, or, not, if\}$$
$$T = \{a_0, a_1, d_0, d_1, d_2, d_3\}$$

とする．6 ビットマルチプレクサ問題では，これらの終端記号と非終端記号を用いて，6 ビットマルチプレクサ回路とまったく同じ出力をする木を探索する．文献 10) に従い，適合度は誤った出力をした回数とした．入力例は 64 パターンすべてを用いた．すなわち，x_{jk} を整数 j を二進数表示したときの k 桁目とすると，個体 i の適合度 fit_i は

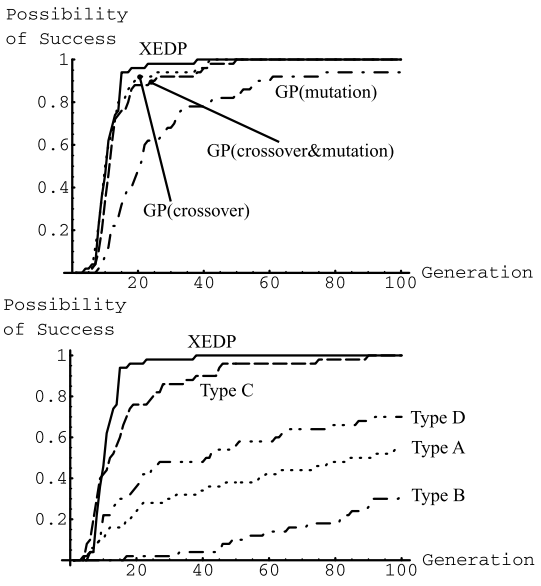

図 7 6 ビットマルチプレクサ問題における探索成功率.
Fig. 7 Cumulative frequency of successful runs for a 6-bit multiplexer problem.

$$fit_i = 64 - \sum_{j=0}^{63} \mid prog_i(x_{j1}, \cdots, x_{j6}) - F_{6mp}(x_{j1}, \cdots, x_{j6})| \quad (10)$$

とした.

文献 10) に従い, GP の集団数は $M = 1000$ とした. 交叉のみの GP は $P_c = 0.9$, $P_c = 0$, $P_r = 0.1$, 突然変異のみの GP は $P_c = 0$, $P_c = 0.9$, $P_r = 0.1$, 交叉と突然変異両方を用いる GP は $P_c = 0.7$, $P_c = 0.2$, $P_r = 0.1$ とした. トーナメントサイズは, $T_{gp} = 2$ がしばしば推奨されるが, 実験の結果, $T_{gp} = 7$ の方が探索性能が高かったため, $T_{gp} = 7$ とした☆. 木の最大深さは $d_{max} = 12$ とした. XEDP の集団数は GP と同様に $M = 1000$ とした. 以上の設定で, 実験を 50 回繰り返した.

図 7 に各世代ごとの探索成功率を示す. 上の図は XEDP と GP の探索成功率の違いを示しており, 下の図は Type A〜D と XEDP の探索成功率の違いを示している. 図 7 より, 6 ビットマルチプレクサ問題において, XEDP は交叉のみの GP と同程度の探索性能を持つことが分かる. また, 突然変異のみの GP に比べ, 明らかに交叉のみの GP の方が探索性能が高い. ゆえに, 交叉が有効な問題においても, XEDP は高い探索性能を持つことが分かる.

☆ 集団数が 1000 を超える場合には, Over-Selection が有効であることが報告されている[10]. しかしながら, Over-Selection はルーレット選択に基づく方式である. 我々の実験では, トーナメント選択で Over-Selection と同様の選択効果を実現できるようにするため, 理論計算を行い, $M = 1000$ では $T_{gp} = 7$, $M = 2000$ では $T_{gp} = 15$ とした.

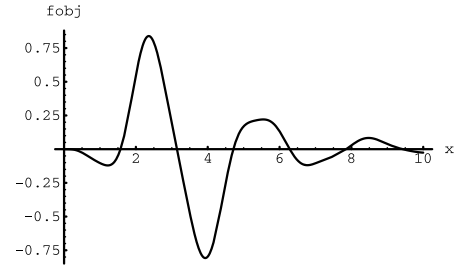

図 8 目的関数の概形
Fig. 8 Target function to be approximated.

一方, 確率モデルを用いる手法どうしの比較からは, Type A, B および D では進化がうまく行われなかったことが分かる. また, XEDP が Type C よりも高い探索成功率を持つことから, 6 ビットマルチプレクサ問題においても, 再帰分布で生成された木による部分的な置換が有効であることが分かる.

4.3 関数同定問題

本実験では PIPE との比較を主な目的とする. 関数同定問題は, 与えられた入力出力例から例を満たすような関数を同定する問題である. PIPE との比較を行うため, 同定する目的関数として

$$f_{obj}(x) = x^3 \cos(x) \sin(x) e^{-x} (\sin^2(x) \cos(x) - 1) \quad (11)$$

を用いた. $f_{obj}(x)$ の概形を**図 8** に示す. この関数は, 文献 19) でベンチマークとして用いられたものである. 非終端記号と終端記号は

$$F = \{+, \times, -, \%, sin, cos, exp, rlog\}$$
$$T = \{x, 0.1, 0.2, \cdots, 1.0\}$$

とした. 文献 19) に従い, 個体 i の適合度 fit_i は

$$fit_i = 100 - \sum_{j=0}^{100} |prog_i(X_j) - f_{obj}(X_j)| \quad (12)$$
$$X_j = 0.1j \quad (13)$$

により計算した.

文献 19) に従い, GP の集団数は $M = 2000$ とした. 交叉のみの GP は $P_c = 0.9$, $P_c = 0$, $P_r = 0.1$, 突然変異のみの GP は $P_c = 0$, $P_c = 0.9$, $P_r = 0.1$, 交叉と突然変異両方を用いる GP は $P_c = 0.7$, $P_c = 0.2$, $P_r = 0.1$ とした. トーナメントサイズは, $T_{gp} = 2$ がしばしば推奨されるが, 実験の結果, $T_{gp} = 15$ の方が探索性能が高かったので, $T_{gp} = 15$ とした. 木の最大深さは $d_{max} = 17$ とした. XEDP の集団数は

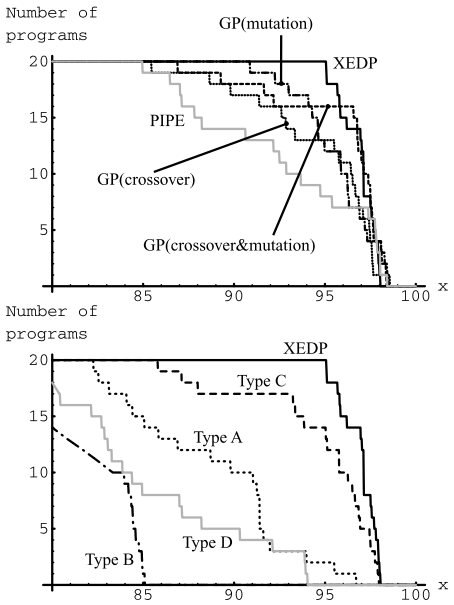

図 9 関数同定問題において 50 世代目で適合度が x を超えた試行の回数

Fig. 9 Number of runs for a regression problem, in which maximum fitness at the 50th generation is greater than x.

GP と同様に $M = 2000$ とした．以上の設定で，実験を 20 回繰り返した．

式 (11) を目的関数とする関数同定問題では，探索が成功する（適合度が 100 になる）可能性がほぼ 0 であるため，探索成功率で比較することができない．したがって，文献 19) と同様に 50 世代目の最大適合度で探索性能を比較した．**図 9** は，50 世代目で適合度が x より大きかった試行の回数を示している．上の図は XEDP と GP および PIPE の探索性能の違いを示しており，下の図は Type A〜D と XEDP の探索性能の違いを示している．なお，文献 19) にも GP による探索結果が報告されているが，再実験したところ文献 19) よりも良い結果が得られたため，我々の実験で得られた結果を示している．図 9 からは，20 回の試行で得られた最大適合度は GP の方が良いが，GP は試行ごとのばらつきが大きいことが分かる．たとえば，50 世代目で適合度（誤差）が 95 より大きくなる確率は，XEDP では 100% であるが，交叉のみの GP では 65% と小さい．関数同定問題では，XEDP は GP や PIPE よりも安定して探索することができることが分かる．また，交叉のみの GP に比べ，明らかに突然変異のみの GP の方が探索性能が高い．ゆえに，突然変異が有効な問題においても，XEDP は高い探索性能を持つことが分かる．

表 2 関数同定問題における 50 世代目の最大適合度の平均と標準偏差

Table 2 Mean and standard deviation of maximum fitness values at the 50th generation for a regression problem.

手法	平均	標準偏差
XEDP	96.92	0.97
GP（交叉のみ）	94.76	3.58
GP（突然変異のみ）	95.69	2.11
GP（交叉＋突然変異）	95.96	3.25
PIPE	92.94	4.65
Type A	88.94	4.39
Type B	82.49	2.33
Type C	94.73	3.65
Type D	85.71	4.85

一方，関数同定問題においても，XEDP が Type C よりも高いパフォーマンスを持つことが分かる．よって，関数同定問題においても，再帰分布で生成した木による部分的な置換が有効であることが分かる．

表 2 に 50 世代目の最大適合度の平均と標準偏差を示す．集団内の平均ではなく，20 回の試行での最大適合度の平均であることに注意されたい．表 2 から，XEDP が平均して最も高い適合度を持ち，また標準偏差が最も小さく，試行ごとのばらつきが小さいことが分かる．

5. 考　　察

5.1 XEDP の探索性能

最大値問題，6 ビットマルチプレクサ問題，関数同定問題において，XEDP は GP 以上の探索性能を示した．ゆえに，汎用的なプログラムの進化の手法として，少なくとも GP と同程度には有効な手法であると考えられる．

PIPE は実数値関数の探索問題にしか適用できない．また，eCGP および GMPE は作為的に作られた GP のだまし問題での探索性能しか報告されておらず，標準的な GP のベンチマーク問題での有効性が示されていない．一方，関数同定問題，最大値問題において，XEDP はそれぞれ PIPE，GMPE よりも高い探索性能を示した．XEDP は確率モデルを用いたプログラム進化の手法である PIPE や eCGP，GMPE よりも，汎用性が高く，探索効率の良い手法であると考えられる．総じて，XEDP はプログラム探索の手法としてきわめて有効な手法であると思われる．

PIPE は関数同定問題とパリティ問題[10] において有効であることが報告されている[19]．一方 4.3 節では，関数同定問題において交叉のみの GP と突然変異のみの GP では性能差がないことが分かった．また文

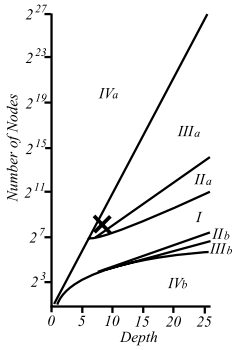

図 10 GP での探索が困難と予想される空間.
Fig. 10 Predicted regions in which search by GP is difficult.

献 2) には，パリティ問題に関しては突然変異の方が成績が良いか，少なくとも同程度であることが報告されている．以上から，PIPE は突然変異が有効な問題において良いパフォーマンスが得られる手法であると考えられる．提案手法では，交叉が有効な問題においても，突然変異が有効な問題においても，比較的良いパフォーマンスが得られている．

最大値問題における比較実験より，GP での探索が困難な問題において，XEDP での探索が有効である場合があることが示された．文献 6) では，**図 10** に示すように探索空間を分割し，それぞれの部分空間の探索可能性について論じている．図 10 の III_a と III_b は，標準的な GP で探索したときに，個体が生成される確率が 1% 未満である部分空間を表している．最大値問題の最適解は図 10 の×印の位置に存在する．図 10 からも，GP での探索が困難な部分空間において，XEDP の探索が成功していることが分かる．前述したように，最大値問題は初期収束に陥りやすい問題である．XEDP は初期収束に陥ることなく，探索することができている．

5.2 XEDP の確率分布モデル

最大値問題，6 ビットマルチプレクサ問題，関数同定問題のすべてにおいて，Type B の探索性能は最も悪かった．ゆえに，大域的な構造の推定がプログラム探索において重要と考えられる．また，いずれの問題においても，XEDP は Type C よりも高い探索性能を示した．これは，再帰分布によって生成された木がランダムに生成された木とは異なることを意味している．ゆえに，再帰分布を用いることにより，プログラムの有益な部分構造を抽出することができたと考えられる．以上から，プログラムの大域的な構造の推定に加えて，再帰分布による部分構造の推定を行うことで，プログラム進化を効率良く行うことができると結論付けられる．

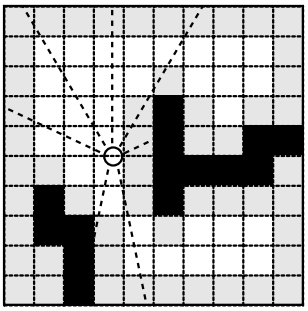

図 11 不規則な壁に囲まれた部屋に置かれているロボット
Fig. 11 A robot located in an irregular room.

最大値問題では確率木による探索が有効であることが分かった．よって，確率木を用いることで，GP では探索できない空間でも探索が可能になると考えられる．この機能により，6 ビットマルチプレクサ問題および関数同定問題において，XEDP は GP よりも高い探索性能を示したと考えられる．

Type D はプログラムの大域的な構造の推定と部分構造の推定の両方を行うことができるが，XEDP とは異なり，プログラムの大域的な構造を推定する際にノード間の依存関係を扱うことができない．Type D は XEDP に比べはるかに劣る探索性能を持つことから，プログラムの大域的な構造を推定する際は，ノード間の依存関係を無視できないことが分かる．

PIPE はプログラムの大域的な構造のみを推定し，さらにノード間の依存関係を無視する．eCGP はノード間の依存関係を扱うことが可能であるが，プログラムの部分的な構造を推定することができない．PIPE も eCGP もともに確率変数が木構造の位置に固定されているため，有益な部分構造を抽出し，それを他の位置に移動させることができない．一方で，GMPE は確率文法を用いるため，プログラムの部分的なつながりのみを推定し，プログラムの大域的な構造を推定できない．以上の考察により，提案手法で用いた確率分布モデルは，先行研究で用いられたモデルに比べ，プログラム進化により適した優れたモデルであると思われる．

5.3 ロボットプログラムの進化への適応可能性

ロボットプログラムの進化への XEDP の適用可能性を調べるため，Wall-following 問題における XEDP の探索性能を調べた．Wall-following 問題では，**図 11** に示すような不規則な壁に囲まれた部屋で，壁に沿って移動するロボットのプログラムを探索する[10]．ロボットは 7 つのセンサを持ち，部屋の中を自由に動き回ることができる．非終端記号は

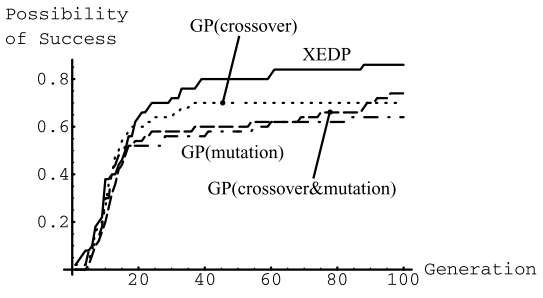

図 12 Wall-following 問題における探索成功率
Fig. 12 Cumulative frequency of successful runs for Wall-following problem.

$$F = \{if, prog2, +, \times, -, sensor\}$$

とした．ここで，if は 3 つ引数をとり，第 1 引数が正なら第 2 引数を評価して返し，そうでないときは第 3 引数を評価して返す．また $sensor$ は引数を 1 つとり，引数の値に従って対応するセンサからの出力値を返す．各センサは，センサの向いている方向の壁までの距離を出力する（図 11 参照）．終端記号は

$$T = \{MF, MB, ML, MR, 1, 2, \cdots, 10\}$$

とした．ここで MF，MB，ML，MR はロボットの動作を実行する命令であり，それぞれ前に進む，後ろに進む，左に曲がる，右に曲がるを実行する．適合度は通過した壁際のタイルの枚数から通過した壁際でないタイルの枚数を引いた数とした．図 11 の灰色のタイルが壁際のタイルを表し，白色のタイルが壁際でないタイルを表す．

GP の集団数は $M = 1000$ とした．交叉のみの GP は $P_c = 0.9$，$P_c = 0$，$P_r = 0.1$，突然変異のみの GP は $P_c = 0$，$P_c = 0.9$，$P_r = 0.1$，交叉と突然変異両方を用いる GP は $P_c = 0.7$，$P_c = 0.2$，$P_r = 0.1$ とした．$T_{gp} = 7$ とし，木の最大深さは $d_{max} = 10$ とした．XEDP の集団数は GP と同様に $M = 1000$ とした．以上の設定で，実験を 50 回繰り返した．

図 12 に各世代ごとの探索成功率を示す．XEDP は進化の後半に探索成功率が上昇し，50 世代目における探索成功率は，XEDP の方が GP よりも高いことが分かる．一方，交叉のみの GP は，40 世代目以降で探索成功率の変動がないことが分かる．**図 13** に，XEDP の探索により得られたロボットプログラムで動作したロボットの軌跡を図示する．ロボットプログラムの進化に対しても，XEDP が有効であることが分かる．

5.4 確率変数の依存関係

確率木のトポロジは木構造以外にも考えられる．たとえば，兄弟ノードどうしで依存関係を持つようなモデルがある．同様に，再帰分布に用いるモデルおいても，より依存関係が多いモデルが考えられる．

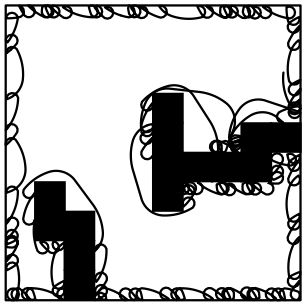

図 13 Wall-following 問題において XEDP で得られたプログラムによる軌跡．
Fig. 13 Acquired trajectory of a robot program obtained with XEDP for Wall-following problem.

表 3 確率分布で表すことができるパターン数
Table 3 Number of patterns which can be represented by each of the probabilistic distribution models.

確率分布	パターン数	サンプル数	R
確率木	n^2	M	10
再帰分布	n^4	$L/2 \times M$	5

実験で用いた確率木と再帰分布で表せるパターン数とサンプル数の概算を**表 3** に示す．ここで L は平均のプログラム長，n は終端記号と非終端記号の種類の数を表す．また，R は $n = 10$，$M = 1000$，$L = 100$ としたときの 1 世代あたりのサンプル数とパターン数の比（サンプル数/パターン数）を表す．ここでは 6 ビットマルチプレクサ問題を想定したため，上の値を用いて概算した．平均プログラム長 L は実験で得られたデータを基に，優良個体の平均の長さのおおよその値から決めた．依存関係を増やすと，確率分布で表すことのできるパターン数は指数関数的に増えていくので，サンプル数が不足し十分な推定ができない．ゆえに，本実験で用いた確率分布モデル以上に確率変数の依存関係を増やしても，探索性能が向上しないと予想される．

5.5 ブロート

一見したところ無駄なコードが集団内に蓄積され，プログラムが長くなっていく現象のことをブロートという．ブロートは実行時間の遅延や過学習を引き起こすと考えられている[27]．

6 ビットマルチプレクサ問題におけるプログラム木のサイズの遷移を**図 14** に示す．図 14 は各世代での集団内の平均プログラム長を表している．ただし 20 回実験を行い，その平均を図示している．6 ビットマルチプレクサ問題では XEDP と GP の差がそれほど

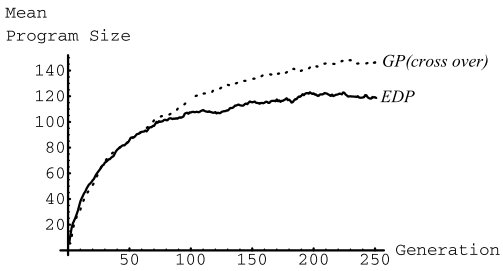

図 14 6 ビットマルチプレクサ問題におけるプログラム長の平均.
Fig. 14 Mean of program size for a 6-bit multiplexer problem.

顕著でなかったため，6 ビットマルチプレクサ問題でのプログラム長の成長を調べた．

図 14 より，XEDP においてもブロートが生じていることが分かる．しかし，GP と比較してみると，プログラム長の増加の割合が小さいことが分かる．GP では前の世代の個体を基に新しい個体を生成しているため，プログラム全体には無害な冗長部分が増加していく．GP においてブロートの制御は困難であることが知られている．一方で，XEDP では 0 から新しい個体を生成しているため，冗長な部分は蓄積されにくい．ゆえに，XEDP ではブロートの影響が GP に比べて小さいと考えられる．

6. おわりに

本論文では確率モデルに基づくプログラム進化の手法を提案し，提案手法がプログラム進化の手法として高いパフォーマンスを持つことを示した．また，確率木と再帰分布の双方を用いること，確率変数の依存関係がある確率分布モデルを用いることが有効であることが確認された．

今後の課題としては，実用的な問題への適用，XEDP のパラメータの最適化，パラメータの値に対する探索の安定性の調査などがあげられる．また，インタラクティブ進化計算（Interactive Evolutionary Computation：以下 IEC）への応用も検討している．IEC では，適合度関数により自動的に個体を評価するのではなく，人間が直接，個体を評価して適合度を与える．IEC では，コンピュータと人間とのインタラクションに意味がある．人間はコンピュータに感性を教え，コンピュータは人間にひらめきや発想を与えるという，この人間とコンピュータの関係が重要視される．XEDP では，確率分布を用いて進化を行うため，人間の感性をシステム内に蓄積しやすいのではないかと考えている．

これまで，GP と同等な性能を出すようなプログラム進化のモデルを構築することは困難であると思われていた．本研究は，プログラム進化の新しい方向性を切り開くものと思われる．

参考文献

1) Angeline, P.: Two selfadaptive crossover operations for genetic programming, *Advances in Genetic Programming II*, MIT Press (1995).
2) Angeline, P.J.: Subtree Crossover Causes Bloat, *Genetic Programming 1998: Proc. 3rd Annual Conference*, pp.745–752, Morgan Kaufmann (1998).
3) Baluja, S.: Population-Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning, Technical Report CMU-CS-94-163, Carnegie Mellon University, Pittsburgh, PA (1994).
4) Bosman, P.A.N. and de Jong, E.D.: Learning Probabilistic Tree Grammars for Genetic Programming, *Proc. 8th International Conference on Parallel Problem Solving from Nature: PPSN-2004* (2004).
5) Cestnik, B.: Estimating probabilities: A Crucial Task in Machine Learning, *Proc. 9th European Conference on Artificial Intelligence*, pp.147–149 (1990).
6) Daida, J.M. and Hilss, A.M.: Identifying Structural Mechanisms in Standard Genetic Programming, *Proc. Genetic and Evolutionary Computation Conference GECCO-2004*, pp.1634–1649, Springer-Verlag (2003).
7) Iba, H. and de Garis, H.: Extending Genetic Programming with Recombinative Guidance, *Advances in Genetic Programming 2*, pp.69–88, MIT Press (1995).
8) Igel, C. and Chellapilla, K.: Investigating the Influence of Depth and Degree of Genotypic Change on Fitness in Genetic Programming, *Proc. Genetic and Evolutionary Computation Conference GECCO-1999*, Vol.2, pp.1061–1068, Morgan Kaufmann (1999).
9) Ito, T., Iba, H. and Sato, S.: Depth-Dependent Crossover for Genetic Programming, *Proc. 1998 IEEE World Congress on Computational Intelligence*, pp.775–780, IEEE Press (1998).
10) Koza, J.R.: *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press (1992).
11) Koza, J.R.: Introduction to Genetic Programming, *2004 Genetic and Evolutionary Computation Conference Tutorial Program* (2004).
12) Langdon, W.B.: Size Fair and Homologous

Tree Genetic Programming Crossovers, *Proc. Genetic and Evolutionary Computation Conference GECCO-1999*, Vol.2, pp.1092–1097, Morgan Kaufmann (1999).
13) Langdon, W.B. and Poli, R.: *Foundations of Genetic Programming*, Springer-Verlag (2002).
14) Larranaga, P. and Lozano, J.A.: *Estimation of Distribution Algorithms*, Kluwer Academic Publishers (2002).
15) Paul, T.K. and Iba, H.: Reinforcement Learning Estimation of Distribution Algorithm, *Proc. Genetic and Evolutionary Computation Conference GECCO-2003*, Springer-Verlag (2003).
16) Paul, T.K. and Iba, H.: Selection of the Most Useful Subset of Genes for Gene Expression-Based Classification, *Proc. Congress on Evolutionary Computation: CEC-2004* (2004).
17) Pelikan, M., Goldberg, D. and Lobo, F.: A Survey of Optimization by Building and Using Probabilistic Model, Technical Report 99018, IlliGAL (1999).
18) Poli, R. and Langdon, W.B.: On the Search Properties of Different Crossover Operators in Genetic Programming, *Genetic Programming 1998: Proc. 3rd Annual Conference*, pp.293–301, Morgan Kaufmann (1998).
19) Salustowicz, R.P. and Schmidhuber, J.: Probabilistic Incremental Program Evolution: Stochastic Search Through Program Space, *Machine Learning: ECML-97*, van Someren, M. and Widmer, G.(Eds.), Vol.1224, pp.213–220, Springer-Verlag (1997).
20) Sastry, K. and Goldberg, D.E.: Probabilistic model building and competent genetic programming, *Genetic Programming Theory and Practise*, pp.205–220, Kluwer (2003).
21) Shan, Y., McKay, R., Abbass, H. and Essam, D.: Program Evolution with Explicit Learning: A new frame work for program automatic synthesis, *Proc. Congress on Evolutionary Computation: CEC-2003*, pp.1639–1646 (2003).
22) Shan, Y., McKay, R., Baxer, R., Abbass, H., Essam, D. and Nguyen, H.: Grammar Model-based Program Evolution, *Proc. Congress on Evolutionary Computation: CEC-2004*, pp.478–485 (2004).
23) Tanev, I.: Implicatons of incorporating learning probabilistic context-sensitive grammar in genetic programming on evolvability of adaptive locamotion gaits of snakebot, *CD=ROM of Workshop Proceedings: Workshop on Evolvability in Evolutionary Computations, Genetic and Evolutionary Computation Conference: GECCO-2004* (2004).
24) Yanai, K. and Iba, H.: Estimation of Distribution Programming based on Bayesian Network, *Proc. Congress on Evolutionary Computation: CEC-2003*, pp.1618–1625 (2003).
25) Yanai, K. and Iba, H.: Program Evolution by Integrating EDP and GP, *Proc. Genetic and Evolutionary Computation Conference GECCO-2004* (2004).
26) 佐久間淳，小林重信：確率分布推定に基づく実数値 GA の新展開，人工知能学会誌，Vol.18, No.5, pp.479–485 (2003).
27) 伊庭斉志：遺伝的プロミング入門，東京大学出版会 (2001).
28) 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，Vol.18, No.5, pp.487–494 (2003).
29) 筒井茂義：エッジヒストグラムを用いる順序表現向き確率モデル GA の提案，人工知能学会論文誌，Vol.18, No.4, pp.173–182 (2003).

付　　録

文献 10) に従って，世代 i までに確率 z で最適解を得るために必要な個体の評価回数 $I(M,i,z)$ の定義を以下に述べる．ここで M は集団数を表す．世代 i での探索成功確率を $P(M,i)$ とし，確率 z で探索を成功させるために必要なおおよその繰返し数を r とすると，

$$(1-P(M,i))^r = 1-z \tag{14}$$

$$\Leftrightarrow r = \frac{\log(1-z)}{\log(1-P(M,i))} \tag{15}$$

ここで，

$$I(M,i,z) = M(i+1)r \tag{16}$$

$$= M(i+1)\frac{\log(1-z)}{\log(1-P(M,i))} \tag{17}$$

と定義する．プログラム探索においては，しばしば

$$\min_i I(M,i,0.99) \tag{18}$$

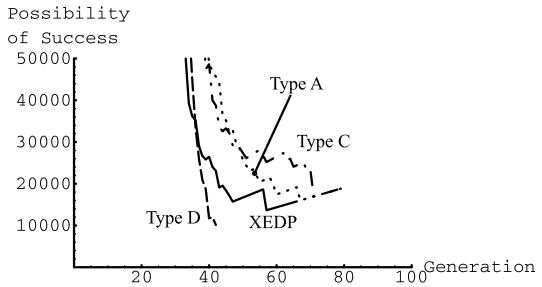

図 15 最大値問題における $I(200,i,0.99)$.
Fig. 15 Plot of $I(1000,i,0.99)$ values for Max problem.

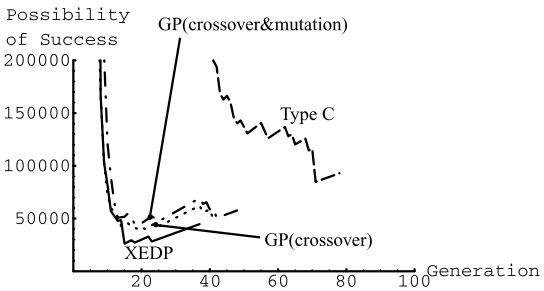

図 16 6 ビットマルチプレクサ問題における $I(1000, i, 0.99)$
Fig. 16 Plot of $I(1000, i, 0.99)$ values for a 6-bit multiplexer problem.

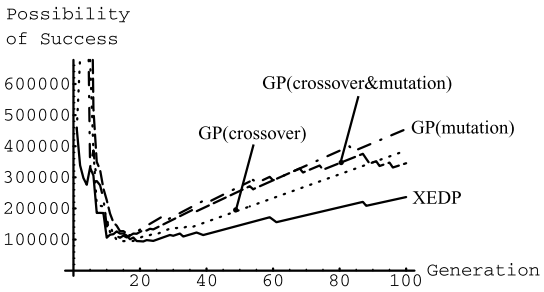

図 17 Wall-following 問題における $I(1000, i, 0.99)$
Fig. 17 Plot of $I(1000, i, 0.99)$ values for Wall-following problem.

表 4 最大値問題における $I(200, i, 0.99)$ の最小値
Table 4 Minimum value of $I(200, i, 0.99)$ for Max problem.

手法	最小となる世代数	最小値
XEDP	57	13655
GP（交叉のみ）	98	1093550
GP（交叉＋突然変異）	98	865432
Type A	67	16009
Type C	71	16951
Type D	42	10123

によって探索アルゴリズムの性能が評価される[10]．

最大値問題，6 ビットマルチプレクサ問題，Wall-following 問題における $I(M, i, 0.99)$ を**図 15**，**図 16**，**図 17** に，各手法の $I(M, i, 0.99)$ の最小値を**表 4**，**表 5**，**表 6** にそれぞれ示す．$I(M, i, 0.99)$ のグラフが途中で途切れているのは，$P(M, i) = 1$ では $I(M, i, 0.99)$ が定義できないためである．

(平成 16 年 11 月 21 日受付)
(平成 17 年 1 月 11 日再受付)
(平成 17 年 1 月 24 日採録)

表 5 6 ビットマルチプレクサ問題における $I(1000, i, 0.99)$ の最小値
Table 5 Minimum value of $I(1000, i, 0.99)$ for a 6-bit multiplexer problem.

手法	最小となる世代数	最小値
XEDP	15	26189
GP（交叉のみ）	19	40000
GP（突然変異のみ）	35	109493
GP（交叉＋突然変異）	19	43439
Type A	22	322428
Type B	91	1187850
Type C	19	64538
Type D	25	194315

表 6 Wall-following 問題における $I(1000, i, 0.99)$ の最小値
Table 6 Minimum value of $I(1000, i, 0.99)$ for Wall-following problem.

手法	最小となる世代数	最小値
XEDP	21	93912
GP（交叉のみ）	15	94887
GP（突然変異のみ）	14	105635
GP（交叉＋突然変異）	17	112938

柳井 孝介（正会員）
2001 年東京大学工学部電子情報工学科卒業．現在，同大学院新領域創成科学研究科基盤情報学専攻博士課程に在学中．進化計算，複雑系，人工生命に興味を持つ．

伊庭 斉志（正会員）
1985 年東京大学理学部情報科学科卒業．1990 年同大学院工学系研究科情報工学専攻博士課程修了．工学博士．同年電子技術総合研究所入所．1998 年東京大学大学院工学系研究科電子情報工学専攻助教授．1999 年同大学院新領域創成科学研究科基盤情報学専攻助教授．2004 年同大学院新領域創成科学研究科基盤情報学専攻教授．進化システムおよび人工知能基礎研究に従事．特に遺伝的プログラミング，知能ロボット，バイオインフォマティクスに興味を持つ．