2004－MPS－52　(12)
2004／12／21

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つ組み合わせて，プログラム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 EDP（Estimation of Distribution Programming）と呼んでいる．本研究では，遺伝的プログラム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，EDP の特徴，探索可能領域について考察した．

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 Estimation of Distribution Programming (EDP). This paper shows results of comparative experiments with GP and other related methods. We empirically comfirm that EDP has higher search performance. Thereafter, we discuss EDP's functions and search space for EDP.

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．本研究ではプログラムを木構造で表現し，このような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．

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．提案手法では，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．我々はこれを Estimation of Distribution Programming（以下 EDP）と呼んでいる [8]．本論文では，GP の標準的な問題に対して EDP を適用し GP との比較実験を行い，EDP と GP の違いについて考察する．また，先に挙げた GP の問題点が，提案手法により克服されるかを否かを調べる．

以下では，まず 2 節で研究の背景について述べ，次に 3 節で提案手法について述べる．4 節では EDP と GP の比較実験の結果を示し，EDP の特徴について論じる．5 節で EDP の機能，探索可能領域について考察し，6 節で結論と今後の課題について述べる．

† 東京大学新領域創成科学研究科
Graduate School of Frontier Sciences, The University of Tokyo

2. 従来研究

近年，確率モデルに基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)[3]，あるいは PMBGAs (Probabilistic Moded-Building Genetic Algorithms)[4] と呼ばれ，提案手法も EDAs の一手法と言える．

しかし，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）[6] を提案している．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．

文献 7) 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．しかし，GP に対する提案手法の優位性が明確に示されていない．

3. 提 案 手 法

3.1 EDP の概要

EDP 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，適合度を割り当てる．
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．F を非終端記号の集合，T を終端記号の集合とする．

Step 4 ではトーナメント選択により集団内の $r_s M$ 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．EDP 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_{edp} と書くことにする．

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元に生成する．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．従って，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．

3.2 確率分布モデル

提案手法では以下の 2 つの確率分布モデルを組み合わせて用いる．

確率木　プログラムの大域的な構造を推定．
再帰分布　有用な部分構造（Building-Block）を推定．

確率木として，木構造のベイジアンネットを用いる．ベイジアンネットのそれぞれの確率変数は，同位置にあるプログラム木のノードに対応する．

再帰分布としては，以下のような条件付き確率を分布モデルとして用いる．

$$\mathrm{P}(Y_1, Y_2, \cdots, Y_{a_{max}} | Y, Y_p) \tag{1}$$

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．

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．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計上し，得られた頻度を正規化して確率を求める．

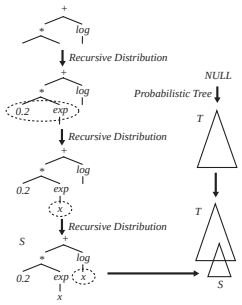

図 1　プログラム生成．
Fig. 1　Program generation.

記述を簡略化するため，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．

$$P' = \eta \widehat{P} + (1-\eta) P_t \tag{2}$$

$$P_{t+1} = (1-\alpha) P' + \alpha \frac{1}{|F| + |T|} \tag{3}$$

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．式（3）では，ラプラス修正 [1] 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（2）で得られた P' と一様分布との重み付け和をとる．

以上は確率木の学習について説明したが，再帰分布についても同様に学習を行う．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，再帰的に頻度を計上して $\widehat{R}$ を推定する．

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．

$$R' = \eta \widehat{R} + (1-\eta) R_t \tag{4}$$

$$R_{t+1} = (1-\alpha) R' + \alpha \frac{1}{|F| + |T|} \tag{5}$$

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 確率木に従ってプログラム T を生成．
Step 2 再帰分布に従って部分木 S を再帰的に生成．
Step 3 T の任意の部分木を S で置き換える．

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 1 参照）．Step 3 は P_r の確率で実行する．

表 1 実験パラメータ.
Table 1 Paremeters for EDP.

r_e	: エリート比	0.005
r_s	: サンプリング比	0.1
T_{edp}	: 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_r	: 置き換え確率	0.9

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 Max 問題，Multiplexer 問題で提案手法（EDP）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.

EDP の特徴を調べるため，いくつかの問題では EDP 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 確率木のみを用いてプログラムを生成（$P_r = 0$）.

Type B 確率木は用いずに，再帰分布だけを用いてプログラムを生成.

Type C 集団から個体をトーナメント選択で選択して，選択された個体に対して再帰分布を用いて部分木を交換（再帰分布による突然変異）.

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

4.1 Max 問題

文献 5)，7) に従い，GP のパラメータは $M = 200$，交叉率と突然変異率はともに 0.995 とした．EDP 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 2 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，EDP 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 8% と低い．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.

Type A（$P_r = 0$）においても探索が成功していることから，Max 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.

文献 7) には，GMPE では探索成功率が 60% を超えるために，平均して 13590 回の評価が必要であったと報告されている．EDP では，13200 回（$= 200 \times 66$）で成功率が 100% となることから，Max 問題においては EDP の方が GMPE よりも探索性能が高いことが分かる.

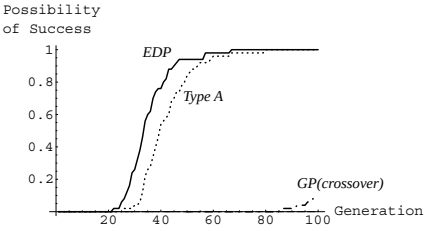

図 2 Max 問題における探索成功率
Fig. 2 Culmative frequency of successful runs for Max problem.

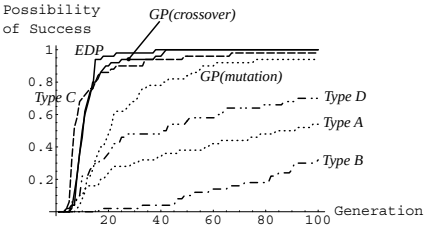

図 3 Multiplexer 問題における探索成功率.
Fig. 3 Culmative frequency of successful runs for a 6-bit multiplexer problem.

4.2 Multiplexer 問題

文献 2) に従い，GP のパラメータは $M = 1000$，交叉率と突然変異率はともに 0.9 とした．EDP 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 3 に探索成功率を示す．図 4 は，世代 i までに 99% の確率で最適解を得るために必要な個体の評価回数 $I(M, i, 0.99)$ を各世代ごとに示している．$I(M, i, 0.99)$ の定義について文献 2) を参照されたい．図 3，4 より，Multiplexer 問題において，EDP は GP と同程度の探索性能を持つことがわかる．プログラム探索においては，しばしば

$$\min_i I(M, i, 0.99) \tag{6}$$

によって探索アルゴリズムの性能が評価される [2]．表 2 に各手法の $I(1000, i, 0.99)$ の最小値を示す．表 2 より，EDP が最も少ない評価回数で 99% 以上の探索成功率を達成できることが分かる．また，Type C が EDP に次ぐ探索性能を持つことから，Multiplexer 問題では再帰分布によるプログラム生成が有効であることが分かる.

5. 考　　察

Max 問題，Multiplexer 問題において，EDP は GP と同等かそれ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

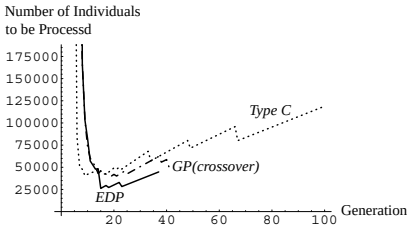

図 4 Multiplexer 問題における $I(1000, i, 0.99)$.
Fig. 4 Plot of $I(1000, i, 0.99)$ values for a 6-bit multiplexer problem.

表 2 Multiplexer 問題における $I(1000, i, 0.99)$ の最小値
Table 2 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
EDP	15	26189
GP（交叉のみ）	19	40000
GP（突然変異のみ）	35	109493
Type A	22	322428
Type B	91	1187850
Type C	9	40416
Type D	25	194315

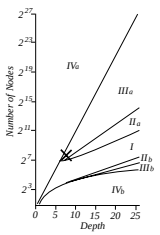

図 5 予想される探索困難な領域.
Fig. 5 Predicted regions of search difficulty.

と同程度には有効な手法であると考えられる．

Max 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，EDP での探索が有効である場合があることが示された．図 5 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% 未満である領域を表している．Max 問題の最適解は図 5 の×印の位置に存在する．図 5 からも，GP での探索が困難な領域において，EDP の探索が成功していることが分かる．

Multiplexer 問題においては，突然変異のみの GP よりも Type C の探索性能の方が良いことから，再帰分布による生成はランダムな生成ではないことが分かる．また，交叉のみの GP は Type C とほぼ同程度の性能を示しているため，EDP の再帰分布による生成は，GP の交叉と同様の働きをしていることが予想される．

一方，Type B による探索が失敗していることから，確率木と再帰分布の組み合わせが有効であることが分かる．また Type D の探索も失敗しているため，確率木における確率変数の依存関係がプログラム探索に重要な役割を果たしていると考えられる．

Max 問題では Type A による探索が成功している．よって確率木による生成は探索可能領域を広げる役割があると考えられる．この機能により，Multiplexer 問題において，EDP が Type C や GP よりも高い探索性能を示したと予想される．

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 GP と同等かそれ以上の探索性能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．

参考文献

1) Cestnik, B.: Estimating probabilities: A Crucial Task in Machine Learning, *Proceedings of the 9th European Conference on Artificial Intelligence*, pp. 147–149 (1990).
2) Koza, J. R.: *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press (1992).
3) Larranaga, P. and Lozano, J. A.: *Estimation of Distribution Algorithms*, Kluwer Academic Publishers (2002).
4) Pelikan, M., Goldberg, D. and Lobo, F.: A Survey of Optimization by Building and Using Probabilistic Model, Technical Report 99018, IlliGAL (1999).
5) Poli, R. and Langdon, W. B.: *Foundations of Genetic Programming*, Springer-Verlag (2002).
6) 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, Springer-Verlag, pp. 213–220 (1997).
7) Shan, Y., McKay, R., Baxer, R., Abbass, H., Essam, D. and Nguyen, H.: Grammar Model-based Program Evolution, *Proceedings of the Congress on Evolutionary Computation: CEC-2004*, pp. 478–485 (2004).
8) Yanai, K. and Iba, H.: Program Evolution by Integrating EDP and GP, *Proceedings of Genetic and Evolutionary Computation Conference GECCO-2004* (2004).