Vol.2017-MPS-112 No.5
2017/2/27

親子の依存を考慮した確率モデルが多段階探索交叉の近傍生成に与える効果

松村 康平[1]　花田 良子[2]　小野 景子[3]

概要：多段階探索交叉 deterministic Multi-Step Crossover Fusion(dMSXF) は，木構造を扱う遺伝的プログラミング (GP) において優れた探索性能を示している．また，単純な確率モデルを近傍生成法に導入することで探索性能を向上させることがわかっている．しかし，従来の確率モデルに基づいた近傍生成法はノードの依存関係を考慮していないため，解に直接影響を与えない形質や壊滅的な影響を及ぼす形質を生成する恐れがある．ここでは，親子間の依存関係を考慮した確率モデルに基づき近傍解を生成することで，効率的に良好な形質の形成を促進させる．本研究では，関数同定問題の複数の例題を対象として，確率モデルが解探索性能に与える効果を検証する．

1. はじめに

遺伝的アルゴリズム (Genetic Algorithm: GA) をはじめとする進化計算で種々の最適化問題を解く際には，主探索オペレータである交叉の設計が重要である．特に組合せ最適化問題のような離散構造を扱う問題においては，両親の形質を受け継ぐよう，問題固有の構造，性質を考慮した交叉の設計が重要である．解が木構造で表現される遺伝的プログラミング (Genetic Programming: GP) においても，交叉の設計が重要な課題である．再帰的な構造を持つ木構造は設計変数であるノードの出現位置 (深さ) や他のノードとの順序関係が解評価に大きな影響を与える場合がある．そのため，両親の良好な形質を効率よく受け継がせる交叉の設計が非常に困難である．また，一般に解表現が固定サイズではなく，設計変数空間を制限することなく自由に探索ができる一方で，探索を阻害する要因としてブロートと呼ばれる現象をしばしば引き起こす．これは，イントロンと呼ばれる冗長なノードの発生や部分木どうしの機能相殺などに起因する．イントロンとは実際には発現しない遺伝子のことであり，取り除かれても解の評価値に影響を与えないものである．これらのような構造の発生によって実行の探索過程でプログラム長が増大し，効率的な探索が阻害され，進化の停滞や実行速度の低下を起こす可能性がある．そのため，GP においては木構造の種々の特徴を考慮した交叉が考案されている．例として，変化を起こすノードや部分木や深さを考慮した交叉，部分木のサイズの制限により交叉での限定的なものにとどめるものなどが挙げられる．その 1 つとして，局所探索のメカニズムを利用した形質遺伝に優れた多段階探索交叉 deterministic Multi-Step Crossover Fusion(dMSXF)[1] がある．これまでに我々は GP に dMSXF を適用するにあたり，両親間に共通するグラフパターンを保存するべき形質ととらえ，それに基づく近傍と距離を導入し，高い探索性能を有することを示した [2]．さらに，探索過程で生成される評価値の高い個体の情報からノードの出現に関する確率モデルを構築し，それに基づいた近傍生成法を提案した [3]．そこでは，ノードの親子間の依存関係を考慮できる Estimation of Distribution Programming(EDP)[4] を dMSXF の近傍生成法に導入することで従来の近傍生成法より高い探索性能を有していることを示した．本研究では，EDP を導入したことによって，解の収束性や最終的に得られる解の質など，dMSXF の探索に与える効果を検証する．検証には関数同定問題の複数の例題を用いる．

2. 木構造における多段階探索交叉

dMSXF[1] は，親 p_1 から親 p_2 に向けて局所探索を行うことで，両親の形質の受け継ぎ方が多様な子個体群を生成する．dMSXF のアルゴリズムを以下に示す．親 p_1, p_2 から生成される子個体群を $C(p_1, p_2)$ と表す．

[1] 関西大学大学院理工学研究科
Graduate School of Science and Engineering, Kansai University
[2] 関西大学システム理工学部
Faculty of Engineering Science, Kansai University, Japan
[3] 龍谷大学理工学部
Department of Electronics and Informatics, Ryukoku University, Japan

【dMSXF のアルゴリズム】

1. p_1，p_2 を両親，その子個体群 $C(p_1,p_2)=\phi$ とする．
2. 探索初期点 $x_1=p_1$，k=1 とし，x_1 を $C(p_1,p_2)$ の要素として加える．
3. ステップ k における探索点 x_k の近傍解を μ 個生成し，その集合を $N(x_k)$ とする．ただし，$N(x_k)$ のすべての近傍解 $y_i(0<i<\mu)$ はかならず $d(y_i,p_2)<d(x_k,p_2)$ を満たさなければならない．
4. $N(x_k)$ の中で最も良い解 y を選択する．$x_{k+1}=y$ とし，x_{k+1} を $C(p_1,p_2)$ の要素として加える．
5. $k=k+1$ とし，$k=k_{max}$ あるいは x_k が p_2 に等しくなれば終了．そうでなければ，3 にもどる．

3. 木構造における距離と近傍

木などのグラフ構造において，ノード間の接続といった形状として現れる情報は個々のグラフの特徴や頻出パターンなどを理解する上で，重要な情報の一つである．また，部分的な構造が持つ意味や作用を理解・対比するにあたっては，その構成要素であるノードの一致性もしばしば考慮される．GP が扱う多くの問題においても，木の部分構造および個々のノードの記号が個体の評価値に大きく寄与する．ここでは，両親間で共通する部分構造およびノードを，子に遺伝させるべき形質ととらえ，それらを破壊しないような近傍生成法，およびそれに対応する距離を定義し，dMSXF に適用している [2].

3.1 距離の定義

木の距離を定義するにあたり，対応する遺伝子座を固定するために 2 つの木の最大共通部分木（Largest common subtree: LCST）を求める．最大共通部分木は，複数のグラフにおける共通の連結部分グラフのうち，ノード数が最大のものをいう．LCST の抽出において得られた，対応するそれぞれの木のノードを，同一の遺伝子座のノードと定義する．これは幹となる構造をもとに 2 つの木の位置合わせを行うことに相当する．図 1 に LCST の例を示す．図中，u_* と v_* は木 T_a，T_b それぞれのノードであり，黒の実線で示された部分が T_a，T_b における LCST である．また，破線で示される部分は，それぞれの木の特有にみられるノードであり，対応する遺伝子座は定義されない．

木の距離は次のように定義される．T_a，T_b で同一の遺伝子座で記号が互いに異なるノードの集合をそれぞれ U_a，U_b とする．T_a で LCST に含まれないノードの集合を D_a，T_b で LCST に含まれないノードの集合を D_b とし，2 つの木の距離 $d(T_a,T_b)$ を式 (1) で定義する．$|\cdot|$ は要素数（ノードの個数）を示す．ここで $|U_a|$=$|U_b|$ である．

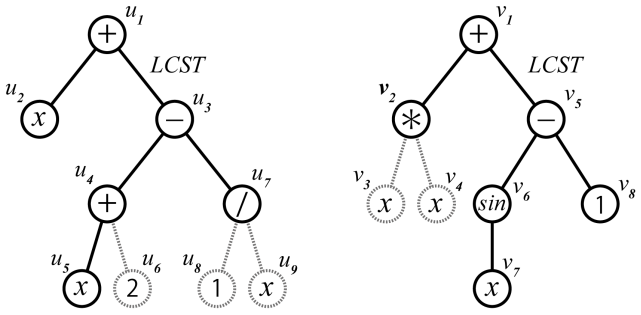

図 1　T_a，T_b の最大共通部分木

$$d(T_a,T_b)=|U_a|+|D_a|+|U_b| \tag{1}$$

これは同一遺伝子座で異なるノード，および LCST に含まれない各木特有の部分木に属するノードの総数を示している．図 1 の例では，U_a と D_a はそれぞれ $\{u_2, u_4, u_5, u_7\}$，$\{u_6, u_8, u_9\}$ となる．また，U_b と D_b はそれぞれ $\{v_2, v_6, v_7, v_8\}$，$\{v_3, v_4\}$ であり．距離 $d(T_a,T_b)$ は $4+3+2=9$ となる．

3.2 近傍の生成

近傍解を生成するにあたり，LCST に含まれるノードに関する 3 種の操作 Replace，Delete および Insert を導入する．これらの 3 つの操作はノードの引数の違いによって選択され，適用後も非終端記号となる関数の引数の個数に関する制約を満たすようになっている．各ステップにおける暫定解 x_k について，x_k と p_2 に関する LCST を抽出する．それに基づき，以下を x_k に確率的に複数回適用し近傍解 $y_i(0<i<\mu)$ を生成する．ここでは，$[1, 2\times d(p_1,p_2)/k_{max}-1]$ の範囲のランダムな整数 s とすると，$d(x_k,y_i)\geq s$ となるまで適用する．なお，手法の説明にあたり，次の表記を用いる．

- **child(n)**：ノード n の子ノードの集合
- **parent(n)**：ノード n に対する親ノード
- **st(n)**：ノード n を根とする部分木
- **arg(n)**：ノード n が持つべき子ノードの数
- **symbol(n)**：ノード n の記号

以下に各操作について説明する．

- **Replace**(u，v)
  Replace は $arg(u)=arg(v)$ を満たす場合に起こる操作で，ノード u の記号を v の記号に置き換える．
  Step1. $arg(u)=arg(v)$ を満たす同一遺伝子座のノード u,v を U_a,U_b から一つ選択する．
  Step2. $symbol(u)$ を $symbol(v)$ に置き換える．
- **Delete**(u，v)
  Delete は $arg(u)>arg(v)$ の場合に起こる操作で，u 以下にある深さ 1 の部分木を削除する．

Step1. $arg(u) > arg(v)$ を満たす同一遺伝子座のノード u,v を U_a,U_b から 1 つ選択する．

Step2. $child(u)$ の中で LCST に含まれないノードを一つランダムに選び，それを u^* とする．

Step3. 部分木 $st(u^*)$ に含まれる終端記号で最も深い位置に存在するノードを一つなんらかの基準で選び，それを (u^{**}) とする．

Step4. $parent(u^{**})$ を (u^{**}) に置き換える．

- **Insert(u，v)**
  Insert は $arg(u) < arg(v)$ の場合に起こる操作で，u 以下にノードを付け足す．

  Step1. $arg(u) < arg(v)$ を満たす同一遺伝子座のノード u,v を U_a,U_b から 1 組選択する．

  Step2. $arg(u) - arg(v)$ 個の終端記号をなんらかの基準で生成し，u の子ノードとして末尾に挿入する．

  Step3. $symbol(u)$ を $symbol(v)$ に置き換える．

4. 確率モデルに基づく近傍生成法

従来の dMSXF は局所探索において近傍解を生成する際に，問題特有の制約条件を満たすために，Insert の Step2 や Delete の Step3 においてノードをランダムに置換，挿入している．そのため，効率的に良好な近傍解を生成することが困難である．そこで，探索過程で生成される個体群の情報より得られるノードの出現に関する確率モデルを構築し，それに基づき終端ノードを確率的に生成することを考えた．本研究では，Sałustowicz らによって提案された木構造の推定を行うためのデータ構造である確率木 (Probabilistic Prototype Tree: PPT)[5] を採用している．PPT は，あらゆる問題の木構造を表現するために完全 n 分木が用いられている．ただし，n は出現しうる非終端ノードの中で最大の引数の数である．また，親子ノードの依存を考慮するために柳井らによって提案された Estimation of Distribution Programming(EDP)[4] を近傍生成法に導入する．

4.1 EDP

多くの木構造は親子関係ノードに強い依存があり，親ノードが子ノードへ与える影響が大きい．EDP は親子関係ノードを扱うことのできるモデルの 1 つであり，木構造内の位置と親ノードのシンボルに依存した分布の推定により子ノードの生成を行う．EDP における確率木の例を図 2 に示す．EDP のそれぞれの確率表には探索過程で得られた個体のその位置でのシンボルの出現頻度が格納されている．確率表の行が親ノードのシンボル，列が子ノードのシンボルを示している．なお，親ノードを持たないルートノードには確率表が存在しない．

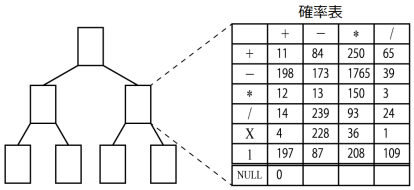

図 2 EDP における PPT の例

4.2 近傍生成法における確率モデルの利用

EDP を dMSXF の近傍生成法に導入する．dMSXF における近傍操作に EDP を導入するにあたり，Delete と Insert を改良する．確率木において，探索の過程でノードが出現する頻度も保持し，Delete の操作で削除対象になるノードを出現頻度が低いものほど高い確率で選択する．また，Insert における操作では，適用ノードにおける確率表の確率分布に従い，シンボルを選択する．確率モデルに基づいた Delete，Insert のアルゴリズムを以下に示す．

- 改良 **Delete(u，v)**

  Step1. $arg(u) > arg(v)$ を満たす同一遺伝子座のノード u,v を U_a,U_b から一つ選択する．

  Step2. $child(u)$ の中で LCST に含まれないノードを 1 つランダムに選び，それを u^* とする．

  Step3. 部分木 $st(u^*)$ に含まれる終端記号で最も深い位置に存在するノードの集合を T とする

  Step4. T に含まれるノードに対応する PPT の確率表を参照し，確率表のノードの出現頻度をもとにルーレット選択を行い，選ばれたノードを (u^{**}) とする．

  Step5. $parent(u^{**})$ を (u^{**}) に置き換える．

- 改良 **Insert(u，v)**

  Step1. $arg(u) < arg(v)$ を満たす同一遺伝子座のノード u,v を U_a,U_b から 1 組選択する．

  Step2. $arg(u) - arg(v)$ 個の子ノードを T として，u の末尾に挿入する．

  Step3. $symbol(u)$ を $symbol(v)$ に置き換える．

  Step4. T に対応する PPT の確率表を参照し，確率表の確率分布に従って T のシンボルを決定する

5. 数値実験

関数同定問題を用いて解の収束性，および解の質を検証する．解の収束については評価計算回数，解の質については個体に含まれるイントロンの数に注目する．比較として確率モデル Probabilistic Incremental Program Evolution(PIPE)[5] を dMSXF の近傍生成に導入した手法 [6](dMSXF+PIPE) と確率モデルを用いない dMSXF を用いる．また，EDP を用いた手法を dMSXF+EDP と呼ぶ．

5.1 関数同定問題

連続関数同定問題とは，複数の入力値と出力値のセットに対して，それらを最小の誤差で近似する連続関数を求める問題である．検証に用いる例題を以下に示す．

$$f_{opt1} = x^4 - x^3 + x^2 - x \quad (2)$$

$$f_{opt2} = xsinx(cosx - 0.5) \quad (3)$$

推定するにあたり，f_{opt1} については非終端記号の集合 V^{NT}={+, -, *, /}，f_{opt2} については V^{NT}={+, -, *, /, sin, cos} を用いる．また終端記号の集合は，両問題とも V^{T}={x, 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9} を用いる．非終端記号については，算術演算は 2 項演算子，他の sin，cos は単項演算子で，終端記号における 0〜1 の数字は実数，x は変数とする．f_{opt1} の変数の定義域は [-1,1]，f_{opt2} の変数の定義域は [0,10] である．いずれも，サンプル点を 21 点，等間隔に与える．目的関数はサンプル点における真値と絶対誤差の総和とする．最小化問題であり，0 による除算などの演算例外を含む個体は評価値は∞とする．それぞれの式と等価な式 (最適解) が得られたときに同定成功とする．

5.2 イントロンと致死形質の定義

EDP が探索性能に与える効果を検証するにあたり，探索中に出現したイントロンと致死形質の数を比較する．致死形質とは，0 による除算のような解に致命的な影響を及ぼす形質とする．解が問題に対して有効である形質を有している場合においても，致死形質が存在すると解全体の評価値は非常に悪くなる．イントロンとは，解に影響を与えない部分木であり，例として 0 との加算が挙げられる．このような特徴を有する最小の単位を直接イントロンと呼ぶ．また，致命的な影響を及ぼす致死形質の最小単位を直接致死形質と呼ぶ．実験に用いる例題における直接イントロンと直接致死形質の例を図 3 に示す．図中 r は，実数とする．解を評価する過程で，例に挙げた直接イントロンに帰着されるものを間接イントロンと呼ぶ．同様に，致死形質においても間接致死形質と呼ぶ．

EDP は，イントロンや致死形質のような解に悪影響を与える形質の生成を陽に禁止していないが，良好な解の形質を保存し参照時にその発生を促進することで，イントロンや致死形質を抑制することが期待できる．

5.3 実験パラメータ

dMSXF，dMSXF+PIPE，dMSXF+EDP の 3 手法を比較する．いずれも母集団サイズは 50，計算終了世代を 200 とする．また，探索の途中で最適解を得た時点で計算終了とする．世代交代モデルは dMSXF[1] の原論文のものを用いる．初期解はあらかじめ定めたサイズの範囲内でランダムに生成する．ここでは [25,35] で検証する．3 手法における局所探索のステップ数，各ステップにおける近傍生成数は (k_{max}, μ)=(3,4) とした．

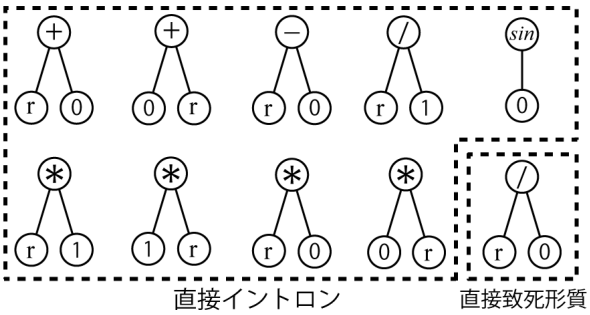

図 3 イントロンと致死形質

表 1 3 手法の最適解到達率と評価計算回数の比較

		dMSXF		dMSXF+PIPE		dMSXF+EDP	
f_{opt}	initial size	%opt	#eval	%opt	#eval	%opt	#eval
f_{opt1}	[25, 35]	0.72	9.3x10^5	0.96	6.7x10^4	0.96	6.2x10^4
f_{opt2}	[25, 35]	1.00	4.7x10^4	1.00	2.9x10^4	0.98	2.8x10^4

($k_{max} = 4, \mu = 5$)

表 2 3 手法の直接イントロンと間接イントロンの比較

		dMSXF		dMSXF+PIPE		dMSXF+EDP	
f_{opt}	initial size	direct	indirect	direct	indirect	direct	indirect
f_{opt1}	[25, 35]	3.1	5.2	2.3	5.4	2.7	4.9
f_{opt2}	[25, 35]	3.5	6.7	2.8	6.0	2.5	5.3

direct:直接イントロン，indirect:間接イントロン
(x10^{-1})

表 3 3 手法の直接致死形質と間接致死形質の比較

		dMSXF		dMSXF+PIPE		dMSXF+EDP	
f_{opt}	initial size	direct	indirect	direct	indirect	direct	indirect
f_{opt1}	[25, 35]	4.5	6.4	2.5	5.4	2.3	6.8
f_{opt2}	[25, 35]	7.3	16	4.5	14	2.3	12

direct:直接致死形質，indirect:間接致死形質
(x10^{-3})

PIPE，EDP については確率木はすべての葉の深さが 8 の完全 2 分木とし，いずれのノードも初期の記号の度数分布を頻度 1 の一様分布とする．確率モデルの確率分布学習には，dMSXF の探索時に生成される近傍解を使用する．dMSXF の探索におけるステップ k で生成された近傍解の中で，最も評価値の高い解を x_k とする．確率分布学習には，$x_k(1 \leq k \leq k_{max})$ で評価値の高い上位半数を使用する．そのため，1 回の交叉において確率分布に登録される個体数はステップ数 k_{max} の半分である $k_{max}/2$ となる．確率木の参照時，確率分布学習時には解と確率木でマッチングを行い，ノードの位置を特定する．生成された解において深さが 8 よりも深い位置にあるノードは確率木のノードとマッチングするものがない．更新の場合にはそのノードを無視，参照の場合にはすべての記号の出現確率は一様分布に従うものとする．確率木は 1 回目の試行の開始と同時に初期化し，2 回目以降の試行は前の試行で更新された確率木を引き継ぐ．

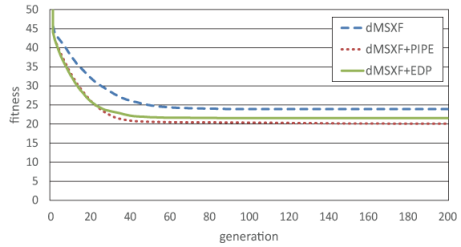

図 4　母集団全体の平均評価値の推移

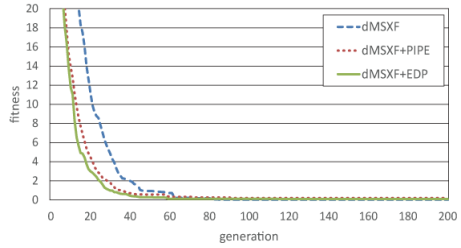

図 5　最良解の平均評価値の推移

5.4　実験結果

表 1，表 2，表 3 に 3 手法の比較結果をそれぞれ示す．これらは 50 試行の結果であり，表 1 に最適解を得た確率，すなわち同定成功した試行の割合 (%opt) と試行中に解を評価した回数 (#eval) の比較結果，表 2 に評価した解 1 個体あたりに含まれるイントロンの比較結果示す．また，表 3 には致死形質の比較結果を示している．これらの結果より，EDP を近傍生成法に導入した dMSXF は従来の dMSXF よりも少ないイントロン，致死形質，評価計算回数でありながら高い探索成功率を有していることがわかる．dMSXF+EDP と dMSXF+PIPE の違いは，探索成功率と評価計算回数で見られるが，イントロンや致死形質については，f_{opt2} における直接致死形質の一部分でしか確認できなかった．

f_{opt2} における最良解と母集団全体の平均評価値の遷移を図 4，図 5 に示す．図 4 より，最終的に得られる母集団の平均評価値は dMSXF+PIPE のほうが優れていることがわかる．これは，各試行で最適解を得た時点で探索を終了していることに起因する．表 1 より，dMSXF+EDP は最も高い探索成功率を有しながら少ない評価計算回数で最適解を得ていることがわかる．また，図 5 より dMSXF+EDP は最良解の更新が早いことが確認できる．このことから，EDP を導入することで探索の早い段階で発見的に最適解を得ていると考えられる．

6.　おわりに

木構造最適化のための多段階探索交叉 dMSXF において，親子間の依存関係を考慮した確率モデルに基づく近傍生成法が探索に与える効果を検証した．数値実験より，関数同定問題において EDP の導入により，少ない評価計算回数，かつ高い探索成功率を得られることがわかった．しかし，イントロンや致死形質においての違いは一部にしか確認できなかった．これは，本実験における探索で最適解とみなす解を発見した時点で，計算を打ち切っていることが原因だと考えられる．dMSXF+EDP は発見的に最適解を得ているため，母集団全体が収束する前に計算が終了している．一方，dMSXF+PIPE は最適解の探索に時間をかけ，長い間確率表を更新している．そのため，dMSXF+PIPE では確率木の確率表に偏りが生じる．今後は，この偏りを考慮した確率分布学習の改良を目標とする．

謝辞 本研究は科研費 26330290 の助成を受けたものです．

参考文献

[1] Ikeda, K., and Kobayashi, S.: deterministic Multi-step Crossover Fusion: A Handy Crossover for GAs, Proc. Parallel Problem Solving fron Nature VII, pp.162–171 (2002).
[2] Hanada, Y., Hosokawa, N., Ono, K. and Muneyasu, M.: Effectiveness of Multi-step Crossover Fusions in Genetic Programming, Proc. IEEE Congress on Evolutionary Computation (CEC 2012), pp. 1743–1750 (2012).
[3] 松村康平, 花田良子, 小野景子,:木構造最適化におけるノードの依存関係を考慮した近傍探索に基づく多段階探索交叉，計測自動制御学会・第 26 回インテリジェント・システム・シンポジウム，2016-10，pp. 178-181
[4] Yanai,K. and Iba,H.:Estimation of distribution programming based on Bayesian network, Proceedings of the Congress on Evolutionary Computation 2003, pp. 1618–1625(2003).
[5] Sałustowicz, R. P. and Jurgen Schmidhuber, J: Probabilistic Incremental Program Evolution, Evolutionary Computation, Vol. 5, No. 2, pp. 123-141, 1997
[6] Matsumura, K., Hanada, Y., and Ono, K.: Probabilistic Model-Based MultiStep Crossover genetic programming, Kohei Matsumura･Yoshiko Hanada･Keiko Ono, Proc. 2016 Joint 8th International Conference on Soft Computing and Intelligent Systems and 2016 17th International Symposium on Advanced Intelligent Systems，2016-8, pp.154-159