Vol.2012-MPS-87 No.6
2012/3/1

文法進化における個体の遺伝子定義の改良について

杉浦秀幸[†1]　北　栄輔[†1]

文法進化（Grammatical Evolution：GE）は関数やプログラムの探索を目的とした進化的計算法で，バッカス・ナウア記法（Backus Naur Form: BNF）により定義した文法を用いることで一次元配列の遺伝子型から木構造の表現型を生成することができる．本研究では，GE における遺伝子型を２次元型として収束速度を改善する改良型 GE を提案し，実問題における改良型 GE の適用効果について検討する．

Improvement of Genotype Definition for Grammatical Evolution

Hideyuki Sugiura[†1] and Eisuke Kita[†1]

Grammatical Evolution (GE) is an evolutionary computation in which the phenotype of tree structure can be generated from the genotype of the bit-string according to the grammar defined by Backus-Naur Form (BNF).In this paper, we present Advanced GE, of which genotype definition of bit-string is modified to the two-dimensional chromosome for improving convergence speed.We will discuss the effect of Advanced GE in practical problems.

1. はじめに

近年，様々な分野において，多数の設計変数，複数の目的関数，複雑な制約条件などから定義される大規模組合せ最適化問題をできるだけ短い計算時間で解くことが求められている．大規模組合せ最適化問題は，解空間が広大で複雑であることが多く，厳密な最適解を求めることは，計算量の増加により困難である．しかし，現実の問題においては，最適解である保証がなくとも十分に精度の良い近似解が求まれば満足のいく場合が多い．その為，大規模組合せ最適化問題を効率よく解く手法として進化的計算法が研究されている．進化的計算法とは，生物の遺伝や進化のプロセスを応用し，様々な問題の解析や最適化，学習等に応用する手法である．進化的計算法の代表例として，遺伝的アルゴリズム (Genetic Algorithm: GA) や遺伝的プログラミング (Genetic Programming: GP) などが挙げられる．

GA[1)] では，解の候補を遺伝子とした「個体」の集団を用意し，個体の適合度を計算する．その後，適合度の高い個体を優先的に親個体として選択して，交叉・突然変異を行い，新たな個体を生成する．これらの，個体の生成と遺伝子操作を繰り返す事で，解を探索していく．

GA では遺伝子が一次元配列として定義されるのに対し，GP[2)] は遺伝子を木構造として定義する．そのため，GA では扱うことの難しい多項式やプログラム，条件分岐などを扱うことができる．しかし，遺伝的操作により新たに生成された個体の遺伝子型から，文法的に正しい表現型を生成できる保証はない為，しばしば評価不可能な表現型を生成して，探索性能が劣化する可能性がある．

そこで，1998 年に Ryan らは文法進化（Grammatical Evolution： GE) を提案している[3)]．GE では，あらかじめバッカス・ナウア記法 (Backus Naur Form： BNF) によって 1 次元配列で定義された遺伝型から木構造で定義された表現型への変換を示す文法を定義する．そして，文法で定義された各非終端記号に対する遷移規則数と遺伝子座の値を用いて，遺伝子型から表現型に変換する．その為，生成される個体は常に文法的に正しい表現型を生成することができる．

GE の BNF 文法では，IF 分岐構造を含む文法を定義することができるが, その場合も１つの連続した１次元配列の遺伝子を用いている．その為，遺伝的操作によって，親個体の有用な形質を破壊されやすい場合があり，探索効率が低下してしまう．例として，IF 条件部と IF 実行部が連続して 1 次元遺伝子で定義された場合を考える．遺伝的操作で IF 条件部が変更された場合，それにつづいて定義される IF 実行部の構文も変更されることになる．もし，IF 実行部の構文が正しい場合であっても，それが子個体に正確には引き継がれないので，収束特性が悪化することとなる．そこで本研究では，生成される構文に含まれる各 IF 条件式や IF 実行部をそれぞれ独立した１次元配列を用いて定義し，IF 文全体を二次元配列で定義する遺伝子構造を持つ GE である Grammatical Evolution using 2 Demensional Gene(GE2DG) を提案する．

親個体の形質保存の改善を目的とした方法として，原らが，複数の染色体を持つ GE である Grammatical Evolution using Multiple Chromosomes(GEMC) を提案している[4)]．GE

†1 名古屋大学大学院情報科学研究科
Nagoya University, Graduate School of Information Science

では非終端記号ごとに終端記号へ遷移するまでに使用する遺伝子の数が異なる．その為，遺伝子操作によってある遺伝子座の値が変化し，選択される遷移規則も変化すると，遷移する元の非終端記号を終端記号に遷移するのに使用される遺伝子座の範囲が，親個体よりもずれてしまう．するとそれ以降の遺伝子座に対して選択される遷移規則が連鎖的に変化してしまい，親個体の有用な形質が失われてしまう問題がある．この問題に対し，GEMC では定義した文法の非終端記号と同じ数だけの染色体を個体の遺伝子型に用意する．遺伝子型から表現型に変換する際には，選択された非終端記号と対応する染色体を用いる事で，親個体の形質を保存しやすくなる．

また本稿では GE2DG と GEMC を組み合わせた GE(GEMC+GE2DG) についても提案する．そして，GE2DG,GEMC,GE2DG+GEMC と既存 GE を実問題に適用した際の探索性能について比較検討を行う．

本論文は次のような構成になっている．まず第 2 章では，既存 GE と改良型 GE の解説を行う．第 3 章では既存 GE と改良型 GE を関数推定問題に応用した際の探索性能の違いついて実験と考察を行う．第 4 章では本稿全体のまとめと今後の課題をまとめる．

2. 文法進化

オリジナルの GE のアルゴリズムは以下の様になる．

(1) BNF 文法の定義
問題に合わせて，遺伝子型を木構造の表現型に変換する文法の定義をする．

(2) 初期個体群の生成
初期集団として N 個の個体をランダムに生成する．

(3) 遺伝子型の表現型への変換
(1) で定義した文法と遺伝子座の値に基づいて，遺伝子型から表現型を生成する．

(4) 適合度評価
個体の適合度を計算する．

(5) 終了条件
設定した終了条件を満たせば最も良い適合度を持つ個体を出力して終了する．

(6) 遺伝子操作と子個体の生成
エリート数だけ適合度の高い個体を順に，次世代の個体として保存する．その後，エリート保存戦略にて保存された個体と合わせて，N 個の子個体が生成されるまで，次の 3 ステップを繰り返す．

(a) 選択
母集団から 2 個の親個体を選択する．

(b) 交叉
6a にて選択された 2 個体の交叉点をランダムに選び，2 個体の交叉点をもとに染色体を交換する．

(c) 突然変異
染色体の一部をランダムに変化させる．

(7) 世代交代
現在の集団を (6) にて生成された子個体群に置き換え，3 へ戻る．

2.1 複数の染色体を持つ GE(GEMC)

以下に原らの提案した複数の染色体を用いた GEMC のアルゴリズムについて説明する．GEMC では，定義した文法の非終端記号と同じ数だけの染色体を個体の遺伝子型に用意する．例として表 1 の文法を用いる場合を考える．

表 1 文法の例 (1)
Table 1 Syntax example (1)

(A)	<expr> ::= <expr><op><expr>	(A0)
	| <var>	(A1)
(B)	<op> ::= +	(B0)
	| -	(B1)
	| *	(B2)
	| /	(B3)
(C)	<var> ::= X	(C0)
	| Y	(C1)
	| Z	(C2)

表 1 の文法では，非終端記号は <expr>，<op>，<var> の 3 つなので，個体は 非終端記号の数と同様に 3 つの異なった染色体を持つ．以下に遺伝子型の例を示す．個体の染色体は上から順に，<expr>，<op>，<var> を展開するために使用される．

GEMC の遺伝子構造：

2	1	5	10	.....
9	12	8	3	.....
7	6	16	11	.....

次に開始記号を <expr> として，GEMC において遺伝子型から表現型に変換する手順を説明する．開始記号は <expr> なので個体の一番上の染色体から値を読み込む．一番上の染色体における最初の遺伝子座の値は 2，遷移規則数は 2 となり剰余は 0 なので <expr><op><expr> が選択される．表 2 にの変換の手順を示す．

表 2 変換の手順
Table 2 Process of translation

置換前	遺伝子値	遷移数	剰余	遷移規則	置換後
<expr>	2	2	0	(A0)	<expr><op><expr>
<expr><op><expr>	1	2	1	(A1)	<var><op><expr>
<var><op><expr>	7	3	1	(C1)	Y <op><expr>
Y <op><expr>	9	4	1	(B1)	Y - <expr>
Y - <expr>	5	2	1	(A1)	Y - <var>
Y - <var>	6	3	0	(C0)	Y - X

以上のアルゴリズムにおいて，例えば遺伝子操作によって個体の一番上の染色体における最初の遺伝子座の値が 2 から 3 に変わった場合を考える．開始記号である <expr> は <var> に置き換えられるが，<var> の展開に使われる値は遺伝子操作前と同じ一番下の染色体における最初の遺伝子座の値である．その為，<var> は遺伝子操作前と同様，Y に置き換えられる．以上のように GEMC では遺伝子操作に因って遺伝子座の値が変わっても親個体の有用な形質を保持する事が期待できる．

2.2 2 次元の遺伝子型を用いた GE(GE2DG)

以下に本研究で提案する GE2DG における遺伝子型の構造について説明する．GE2DG では表現型に含まれる関数や多項式をそれぞれ独立した 1 次元配列を用いて表現する．例として表 3 の文法を用いる場合を考える．GE2DG における遺伝子型の構造は図 1 の様になっている．1 行目は IF 分岐制御部，最下行は ELSE 用 IF 計算式生成部とする．N(N=1,3,5…) 行目は IF 条件生成部，N+1 行目は IF 計算式生成部とし，N 行目と N+1 行目によって生成される IF 条件式と IF 計算式を分岐 M(M=(N+1)/2) と呼称する．遺伝子型から表現型を生成する際は，IF 分岐制御部の n 番目の値を読み込み，遷移規則に従って分岐 n を生成するかどうかを決定する．分岐 n を生成しない場合は，ELSE 用 IF 計算式を生成し，分岐 n 以降の分岐を生成しない．以上の様にする事で 1 行目の IF 分岐制御部を用いて表現型の IF 分岐の数を制御できる．また IF 条件式の開始記号を<expr><op><expr>,IF 計算式の開始記号を<expr>とする．

表 3 文法の例 (2)
Table 3 Syntax example (2)

(A)	<element> ::= if(<expr> <cond_op> <expr>){ <expr> }else{ <element> } | <expr>	(A0) (A1)
(B)	<cond_op> ::= < | >	(B0) (B1)
(C)	<expr> ::= <expr><op><expr> | <var>	(C0) (C1)
(D)	<var> ::= X | Y	(D0) (D1)
(E)	<op> ::= + | - | * | /	(E0) (E1) (E2) (E3)

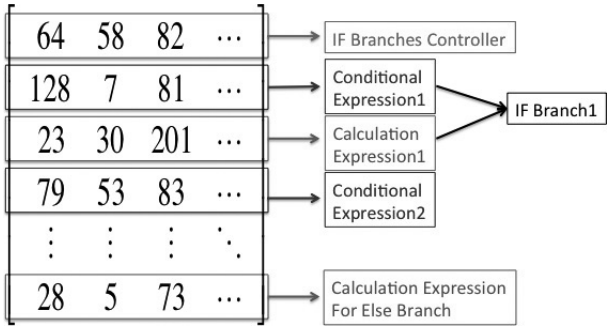

図 1 GE2DG における遺伝子型の構造

既存の GE のおいて，IF 分岐を含む場合の交叉では，IF 条件式部と IF 計算式部といった様に，異なる対象を発現する為に利用される遺伝子座同士でも交叉をする事があり得る．その為，交叉による適合度の改善が効率的に行えないと考えられる．そこで GE2DG における交叉は 2 個体間で，IF 条件式部と IF 条件式部といった様に同じ対象を発現する為に利用される 1 次元配列同士で行う様にする事で，交叉による適合度の改善が期待できるとともに，親個体の有用な形質を次世代に残しやすくなると考えられる．

2.3 GEMC+GE2DG

本節では 2.1 と 2.2 で述べた 2 つの手法を合わせたアルゴリズム（以下 GEMC+GE2DG）

を提案する．GEMC+GE2DG は GE2DG のように構文に含まれる IF 条件式や IF 計算式毎に遺伝子配列を保持するとともに，GEMC のように非終端記号の数だけ遺伝子配列内に 1 次元配列を保持する遺伝子構造を持つ．以下に表 3 に定義した文法を用いる際の遺伝子構造を図 matrix に示す．また GE2DG と同様に IF 条件式の開始記号を<expr><op><expr>,IF 計算式の開始記号を<expr>に変更する．

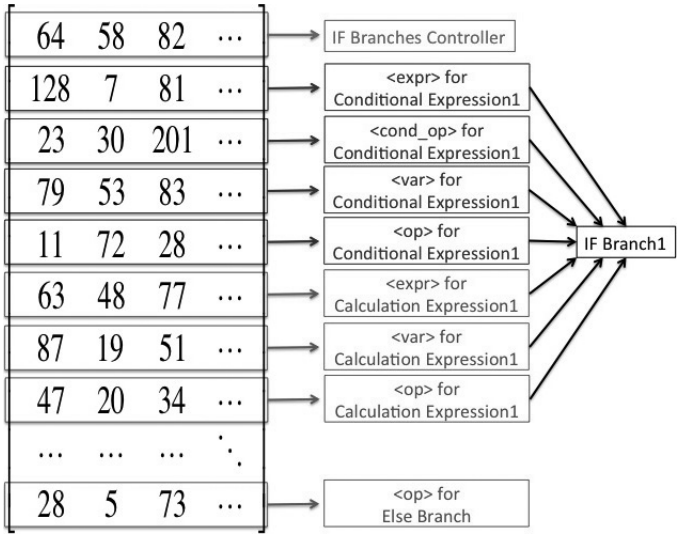

図 2 GEMC+GE2DG における遺伝子構造

1 行目は IF 分岐制御部とし，以降は IF 条件式生成部，IF 計算式生成部の繰り返しとする．また遺伝子構造の最後は ELSE 用 IF 計算式生成部とする．IF 条件式生成部，IF 計算生成部ともに文法に定義された非終端記号の数だけの 1 次元配列を持つ，上から N 個目の IF 条件式 IF 計算式とによって生成される分岐を分岐 N と呼称する．遺伝子型から表現型を生成する際は，GE2DG と同様に IF 分岐制御部の n 番目の値を読み込み，遷移規則に従って分岐 n を生成するかどうかを決定する．分岐 n を生成しない場合は，ELSE 用 IF 計算式を生成し，分岐 n 以降の分岐を生成しない．以上の様にする事で 1 行目の IF 分岐制御部を用いて表現型の IF 分岐の数を制御できる．また GEMC+GE2DG における交叉は 2 個体間で，IF 条件式部と IF 条件式部といった様に，おなじ対象を発現する為に利用される遺伝子配列同士で行う．その際，<expr> 用の染色体は <expr> 用の染色体同士で行うといった様に，同じ非終端記号を発現する為に利用する染色体間で遺伝子座の値の交換を行う．以上のようなアルゴリズムを用いる事で，MCGE が持つ単独の多項式や関数を発現する場合の親個体の有用な形質の保持の効果と，GE2DG の複数の多項式や関数を発現する場合における，親子や個体の有用な形質の保持の効果を併せ持つことができると考えられる．

3. 解 析 例

3.1 関数同定問題

本節では関数同定問題を対象に GEMC，GE2DG，GEMC+GE2DG と既存 GE の探索性能について比較実験を行う．本実験では，真の関数 f としてステップ関数である式 (1) を用いる．

$$f(x) = \begin{cases} 0 & (x < -3) \\ -1 & (-3 \leq x \leq 3) \\ 1 & (3 < x) \end{cases} \tag{1}$$

式 (1) において，x の値を-10 から 10 まで 0.1 刻みで変化させた計 201 個のサンプル点を実験に用いる．サンプルデータから $\bar{f}$ を GEMC，GE2DG，GEMC+GE2DG と既存 GE を用い，それぞれ予測させる．数式を GE で生成するため，表 4 のような文法を定義する．開始記号は既存 GE,GEMC においては <element>，GE2DG と GEMC+GE2DG においては IF 条件式部が <expr> <cond_op> <expr>，IF 計算式部が <expr> である．

適合度は，サンプル点における真の関数 $f(x)$ と各手法よって生成した関数 $\overline{f}(x)$ との平均二乗誤差を用いる．平均二乗誤差は式 (2) で与えられる．

$$E = \sqrt{\frac{1}{201}\sum_{i=1}^{201}(f_{(x_i)} - \overline{f}_{(x_i)})^2} \tag{2}$$

また，実験に用いる 4 つの手法の共通パラメータ設定と各手法専用のパラメータを以下に示す．実験は異なる乱数列を用いて 100 回の試行を行い，適合度の平均値によって比較する．なお，GE のパラメータにおける交叉率と突然変異率は，事前実験において最も良い結果を示した値を設定している．

表 4 定義した文法 1
Table 4 Syntax definition of exmaple 1

(A)	<element> ::= if(<expr> <cond_op> <expr>){ <expr> }else{ <element> }	(A0)
	| <expr>	(A1)
(B)	<cond_op> ::= <	(B0)
	| >	(B1)
(C)	<expr> ::= <expr> <expr> <op>	(C0)
	| <var>	(A1)
(D)	<var> ::= <X>	(D0)
	| <num>	(D1)
(E)	<op> ::= +	(E0)
	| -	(E1)
	| *	(E2)
	| /	(E3)
(F)	<X> ::= x	(F0)
(G)	<num> ::= 0	(G0)
	| 1	(G1)
	| 2	(G2)
	| 3	(G3)
	| 4	(G4)
	| 5	(G5)
	| 6	(G6)
	| 7	(G7)
	| 8	(G8)
	| 9	(G9)

表 5 シミュレーションパラメータ 1
Table 5 Parameters (Example 1)

世代数	1000
個体数	100
選択方法	トーナメント選択
トーナメントサイズ	5
エリート保存選択で残す個体数	5 体
交叉方法	一様交叉
交叉率	0.8
突然変異率	0.3

表 6 既存 GE のパラメータ 1
Table 6 Original GE parameter (Example 1)

個体長	10000

表 7 GEMC パラメータ 1
Table 7 GEMC parameters (Example 1)

染色体数	7
染色体の長さ	1500

表 8 GE2DG のパラメータ 1
Table 8 GE2DG parameters (Example 1)

遺伝子型の行数	10
遺伝子型の 1 行の長さ	1000

表 9 GEMC+GE2DG のパラメータ 1
Table 9 GEMC+GE2DG parameters (Example 1)

染色体数（IF 条件式）	6
染色体数（IF 計算式）	5
遺伝子型の行数	50
染色体の長さ	200

3.2 結果と考察

本実験における，最良個体の適合度の平均値のグラフを図 3 に示す．図の横軸は世代数，縦軸は平均二乗誤差を表す．

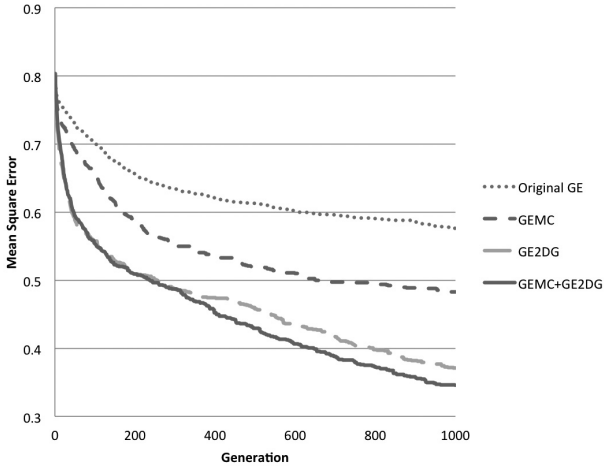

図 3 関数同定問題の実験結果

結果を見るとGE2DGとGEMC+GE2DGが同程度に良い収束特性を示している．これはGE2DGとGEMC+GE2DGに採用されている，各IF分岐を独立した遺伝子配列で生成する遺伝子構造がステップ関数の推定に有用であったといえる．GE2DGとGEMC+GE2DGにて生成された個体をみると，IF分岐が破壊されずに，IF条件式やIF計算式の改善がなされている事が多い．またGE2DGとGEMC+GE2DGの結果にあまり違いが見られないのは，ステップ関数内の関数がf(x)=1やf(x)=0といった様に，簡単な多項式であることが原因だと考えられる．一方で既存GEとGEMCにて生成された最良個体をみると，ほとんどの個体でIF分岐が出現していない．また最良個体にIF分岐が出現する事があっても次の世代ではIF分岐が無い個体が最良個体になる事が多い．これは同次元の1次元配列を用いて複数の生成対象を発現させる事が，親個体の有用な形質を破壊してしまう事を示している．また，既存GEとGEMCを比較するとGEMCの方が良い適合度を示している事を考えると，単独の多項式を発現する個体を進化させる場合，GEMCの方が効率的に進化させることができる事が分かる．

4. 結　　論

本研究では，文法進化における個体の遺伝子定義の改良手法について述べた．既存GEと改良型GEを2つの実問題に適用し，探索性能に関して比較実験を行った．関数同定問題では，GE2DGとGEMC+GE2DGが良い結果を示した．実験ではステップ関数を用いたため，IF分岐を有する関数の探索を効率的に行う事のできるGE2DGやGEMC+GE2DGが良い結果を示したと考えられる．今後の課題と展望として，他の実問題に対してGEMC+GE2DGを適用することが挙げられる．GEは文法の定義を改良する事で，エージェントの行動決定や様々な最適化問題に適用することができる．本研究では関数同定問題に対して実験を行い，考察を行ったが，さらに他の実問題に対してもGEMC+GE2DGを適用することで，GEMC+GE2DGの適用可能性について検討することができる．

参 考 文 献

1) J.H.Holland. **Adaptation in Natural and Artificial Systems**. The University of Michigan Press, 1975.
2) J.R.Koza. **Genetic Programming: On the Programming of Computers by Means of Natural Selection**. MIT Press, 1992.
3) C.Ryan, J.J.Collins, and M.O'Neill. Grammatical Evolution: Evolving Programs for an Arbiturary Language. **Proc. of 1st European Workshop on Genetic Programming**, pp.83-95, 1998.
4) 原章，山口智久，市村匠，高濱徹行，複数の染色体を用いたGrammatical Evolution,2008.
5) 久保幹雄．**組合せ最適化とアルゴリズム**．共立出版，2000.
6) 伊庭 斉志. **進化論的計算の方法**. 東京大学出版会. 1999.
7) W.Banzhaf. Genotype-Phenotype-Mapping and Neutral Variation – A case study in Genetic Programming. **Proceeding of Parallel Problem Solving from Nature III**, pp.322-332,1994.
8) P.Whigham. Search Bias, Language Bias and Genetic Programming. In Genetic Programming 1996: **Proceedings of the First Annual Conference**, pp.230-237, 1996.
9) C.Ryan and M.O'Neill. **Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language**. 2003.
10) A.Brabazon, M.O'Neill, R.matthews, and C.Ryan. Grammatical Evolution and Corporate Failure Prediction. **Proceedings of the Genetic and Evolutionary Computation Conference**, pp.1011-1018, 2002.
11) M.O'Neill, A.Brabazon, C.Ryan, and J.J.Collins. Evolving Market Index Trading Rules using Grammatical Evolution. **Lecture Notes In Computer Science**, Vol. 2037, pp.343-352, 2001.
12) W.Banzhaf, P.Nordin, R.E.Keller, and F.D.Francone. **Genetic Programming: An Introduction: On the Automatic Evolution of Computer Programs and Its Applications**, 1998.
13) 黒田卓也, 改良型GrammaticalEvolutionの実問題への応用, 2009.
14) 伊庭 斉志. **遺伝的アルゴリズムの基礎**. オーム社. 1994.
15) G.Syswerda, **Uniform crossover in genetic algorithms**, Morgan Kaufmann Publishers,1989
16) O 'Neill1, A.Brabazon, **Recent Adventures in Grammatical Evolution**, Proceedings of CMS, 2005
17) O 'Neill, M.Ryan, Under the hood of grammatical evolution. **Proceedings of the First Genetic and Evolutionary Computation Conference**, pp.1143-1148,1999.
18) 岩澤博人，北栄輔．Grammatical Evolutionの文法選択方法の変更による性能改善．**情報処理学会研究報告**，Vol.2007，No.128，pp.101-104，2007.
19) Hans-Paul Schwefel. **Evolution and Optimum Seeking**. John Wiley & Sons,Inc, 1995.