Vol.2012-CE-115 No.5
2012/7/7

プログラム自動生成手法における個体の多様性維持の提案

今福 啓†

本研究では，目的となる機能を持つプログラムをコンピュータ内部で自動的に作成するプログラム自動生成手法において，効率良くプログラムを作成するための手法を提案する．プログラム自動生成手法では，プログラムを表わす個体の多様性が常に維持されることが，目的のプログラムを効率良く作成するために必要とされる．そこで提案手法では，親となる個体群の世代交代の際，親の直系となる子個体群のみを候補とし，そのうち最も高い評価値を持つ個体を選択する．提案手法を用いたシミュレーション結果から，従来手法と比較して高い性能を得られることを示す．

Proposition of A New Method Considering High Diversity for Automatic Program Generation

KEI IMAFUKU†

In this paper, we propose a new method to construct a desired program efficiently for the automatic program generation. To create a program effectively by using automatic program generation, high diversity of individuals which express computer program is required. In the proposed method, only child individuals created from certain parent individual serves as a candidate of replacing with the parent individual. Compared with the existing research, it is shown that the proposed method can obtain the high performance from simulation results.

1. はじめに

教育機関でコンピュータを用いた講義を行う際，個人により習熟度が異なるため，個々の習熟度に合わせたきめ細やかな指導を行うことが必要となる状況がみられる．例えば大学のコンピュータプログラムの講義では，与えられた課題のどの点に関する理解が不足し，何を提示すれば良いのかが学生により違うということが生じる．その際，個人にあわせて指導を行うことが必要となる状況となれば，多くの時間を要することとなり，きめ細やかな教育を行うことが困難になると想定される．

このような状況に，コンピュータが自ら与えられた機能を実現するプログラムを作成する「プログラム自動生成手法」を応用することが期待できる．プログラム自動生成手法では，目的とするプログラムがコンピュータ内部で自動的に作成される．そのため，プログラム教育において，学生が作成したプログラムをもとにして目的となるプログラムを自動的に作成し，自分の作成した内容との比較を行うことで，間違いの箇所や何をどう作成することが完成に近づくのかについて，学生自ら気づくことが可能になると想定される．

プログラム自動生成手法は，遺伝的アルゴリズム（GA）のような進化的計算手法で用いられる操作を応用した手法となっており，遺伝的プログラミング，Grammatical Evolution，遺伝的ネットワークプログラミング，GRAPE（Graph Structured Program Evolution）などさまざまな手法が提案されている．その中で，プログラムを表現する際の制約が低く，かつ作成したコードからコンパイルで実行形式に変換することが不要なため実行しやすい GRAPE がふさわしいと考えられることから，本研究では GRAPE を用いたプログラムの自動生成を効率良く行うことを目的とする．

プログラム自動生成手法では，1 つのプログラムを表現する個体を多数用意する．そして一部の個体を選択し，交叉や突然変異といった操作を適用して，新たな個体を複数作成する．それを元の個体と入れ替えて世代交代を行い，目的となるプログラムが得られるよう個体の集団全体を進化させる．

選択，交叉，突然変異および世代交代の操作において，局所解への収束や進化の停滞を回避するためには，個体の集団が多様な表現を持つことが必要とされる．そのため，GRAPE では MGG（Minimal Generation Gap）とよばれる手法が用いられている．MGG では，親個体群から作成した子個体群を，エリート選択とルーレット選択により選択して親個体と交代する．また，その問題点を修正した手法に JGG（Just Generation Gap）がある．しかし，いずれの手法も実数型 GA における個体の多様性に着目していることと，作成した子個体群のうち，問題にどの程度適した個体であるのかを表す評価値の高いものから順に親個体群と入れ替えを行う．膨大な命令の組み合わせが存在するプログラム自動生成においては，評価値のみに着目した世代交代が多様性の維持に有効に機能するかは不明である．

そこで本研究では，GRAPE での個体の多様性の維持を

† 獨協大学
Dokkyo University.

目的とした新しい手法を提案する．提案手法では，親個体群と子個体群を世代交代する際，元の親子体から作成された，直系となる子個体群のみを世代交代の候補とすることで，置き換え対象となる親個体群の持つ多様性の維持を目的とする．

以下では，2 章で本論文に関連する研究について述べ，3 章で個体の多様性維持のための手法の提案を行い，4 章で提案手法を用いたコンピュータシミュレーションの結果と考察を述べる．

2. プログラム自動生成と多様性維持の研究

最初に，本研究で扱うプログラム自動生成手法である GRAPE[4]について述べる．GRAPE は，例えば最大公約数を求めるなど，目的となる機能を実現するプログラムを人が作成する代わりにコンピュータ内部で自動的に作成する手法である．GRAPE の 1 つの個体は，複数の「ノード」と「データセット」から構成され，それを複数連結して 1 つの個体を構成する．そして個体を複数集めて個体群を構成し，相互にノードを参照した進化を通じて，正しいプログラムに近づくよう内容を更新する．

GRAPE で 1 つの命令を表すノードの内容を図 1 に示す．ノードは整数を一次元状に並べたものからなり，それを命令の種類や使用する変数であるデータセットに応じて，プログラムの 1 命令に変換し実行する．ノードには番号が付けられており，1 つのノードでの命令を実行後，次の遷移先に移動して同様の処理を繰り返す．最後に，外部に出力するデータセットの番号が記述されたノードの命令を実行し，プログラムの外部にデータセットの内容を出力して実行を終了する.個体から変換されるプログラム構造の例を，図 2 に示す.

ノードの総数と整数値の数は，あらかじめ決められた数だけ用意する．ノードの整数値は，命令の種類，移動先のノード番号（2 つ），使用するデータセットの番号（3 つ）の順に並んでいる．これらの要素はすべて使用されるわけではなく，命令の種類に応じて使用する個数が決められているため，命令によっては使用されない場合がある．

GRAPE では，プログラム実行前にデータセットに初期値を格納し，目的となる出力値が得られた際に，正しいプログラムが作成できたものとする．例えばx^2を計算するプログラムを作成する場合，初期値としてデータセットに 1, 2, 3, 4, 5 を順に入力し，それぞれの値から 1, 4, 9, 16, 25 の出力値が得られたならば，正しいプログラムが構築できたとする．

GRAPE のノードは，最初ランダムな整数値とする．そして，各個体が求めるプログラムにどの程度近い内容となっているのかに応じて，評価値を与える．個体群は，以下で述べる「世代交代モデル」にもとづきさらに高い評価値を持つ内容へと進化させることで，徐々に正しいプログラムへと進化していく．

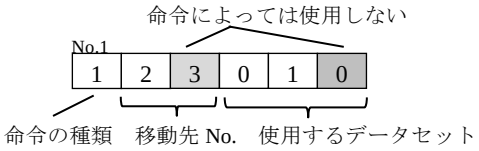

図 1　GRAPE のノードの例
Figure 1　Example of the node in GRAPE

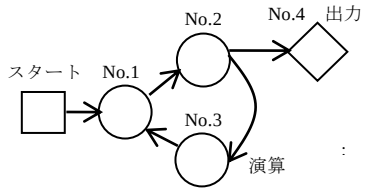

図 2　プログラム構造の例
Figure 2　Example of the structure of the program

個体群の進化の際，ノード全体の持つ値の多様性が維持されることが，目的のプログラムを効率良く得る際に必要となる．そのための手法として提案されている MGG と JGG について述べる．これらの手法は「世代交代モデル」とよばれている．いずれの手法も，個体群とよばれる複数（N 個）の個体の中から一定数 n_p を親個体群として重複しないようランダムに選択し，その個体に対して交叉，突然変異を行い，子個体群となる新たな個体を複数個 n_c だけ作製する点は共通している．異なるのは，親個体群から子個体群への世代交代の方法である．

MGG では，親個体群のうち，n_p-2 個をそのまま元の個体集団に戻し,残りの 2 個を子集団群に加え,家族とする．そして家族の各個体に対して目的となるプログラムへの近さを表す評価値を計算し，その値が最も高い個体と（エリート選択とよばれる),評価値の大きさに応じたルーレット選択により選んだ個体を元の集団に戻す．JGG では，作成した n_c 個の子個体群のそれぞれの評価値を計算した後，評価値の良い方から上位 n_p 個を元の集団に戻す．

[1]では，実数型 GA を用いて問題の解を求める際，MGG で採用されているルーレット選択が最適な解への収束を遅らせ，エリート選択が局所解からの脱却を遅らせる原因になると述べている．[2]では，[1]と同じく実数型 GA の問題において，MGG と JGG で解を求める性能に選択方法が与える影響を分析し，一部の問題において JGG が MGG 以上に有効であることを示している．

MGG と JGG は，評価値の良いものを残すという点は共通しており，[2]でも世代とともに元の個体群のうち，子個体群で置き換えられる数が増加することが示されている．そのため，多様性の維持よりは，評価値を向上させる点を重視した世代交代モデルといえる．

GRAPE は個体がプログラムを表わすため，構築できるプログラムの自由度が極めて高い．そして目的となるプログラムを構築するためには，多様な要素が必要となる．そのことから，特に集団の多様性を維持することを重視した世代交代モデルが必要となる．しかし，単にランダムに個体を構築して多様性のみを維持するだけでは，個体群を正しいプログラムに近づけることはできない．世代交代につれて評価値を上げつつ,かつ高い多様性を維持することが，GRAPE での正しいプログラム作成を効率化すると考えられる．

以上の点から，本研究では初期個体の持つ要素の多様性を残すため，親個体群の直系を子個体群として残すことで集団全体の多様性を維持する，新しい世代交代モデルを提案する．

3. 提案手法

提案する世代交代モデルでは，最初に N 個の個体群から重複しないよう n_p 個の親個体群をランダムに選択する．親個体群のうち,i 番目の親個体から子個体群を作成する際，交叉は i 番目の親個体と，n_p 個の親個体群からランダムに選択した 1 つを用いて一様交叉により行う．さらに，低い確率で作成個体の一部を他の値に変更する突然変異により，各親個体から n_c 個の子となる個体群（総数 $n_p \times n_c$）を作成する．

次に，子個体群の評価値を計算する．そして，親個体の直系となる子個体群の中から，最も評価値の高いものを親個体と入れ替え世代交代を行う.提案手法の手順を以下に，子個体群の作成過程を図 3 に示す．

【提案手法の手順】

1. 時刻$t = 0$とする．
2. 初期の個体集団$P(0)$をランダムに生成し，各個体の評価値を計算する．
3. $t := t + 1$とする．
4. $P(t-1)$より個体$P'(t-1) = \{P'_1(t-1), \cdots, P'_{n_p}(\mathrm{t}-1)\}$をランダムに重複しないよう選択し，$P(t-1) := P(t-1) - P'(t-1)$とする．
5. $P'_i(t-1)\ (i = 1, \cdots n_p)$のそれぞれから，子個体群$C_i(t) = \{C_{i1}(t), \cdots, C_{in_c}(t)\}$を作成する．
6. $C_i(t)\ (i = 1, \cdots n_p)$のそれぞれから評価値の最も高いものを選択し，$P(t-1)$に加える．
7. 決められたシミュレーション期間に至るまで，3.に戻る．

提案手法では，選択した親個体群のそれぞれから，直系となる n_c 個の子個体群が作成され，子個体群の 1 つがその親個体と入れ替わる．また交叉においても，子個体にはその親となる個体の要素が必ず使用される．そのため，世代交代の際に個体群が高い親個体から構成された子個体ばかりで占められることがなくなり，親個体群の多様性が世代交代後にも受け継がれる．これらの手順により，初期にランダムに作成された個体群が持つ集団の多様性を，世代交代を繰り返した後にも維持することが可能になると考えられる．

なお，提案手法は[3]で述べられている子個体の生成に近いものとなっている．本論文の手法とは各親個体の直系を子個体として作成する点は共通しているが，世代交代の際に，親の直系の子個体群のすべてから評価値が上位のものを選択して親個体群と入れ替えており，親の直系のみを残していない点が異なっている．

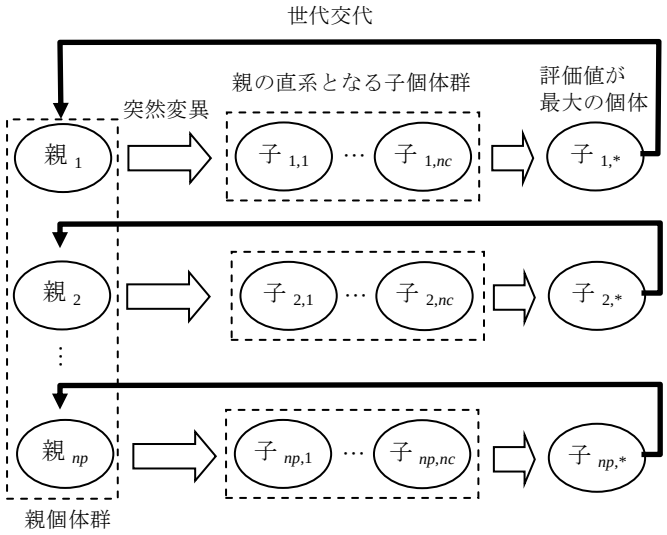

図 3　提案する世代交代モデル

FIgure 3　A New Generation Alternation Model

4. シミュレーション

提案手法を用いたシミュレーションを行った．従来の GRAPE と提案手法の結果比較を行うため，シミュレーションは[4]で実行されたものと同じ問題および個体の評価方法により行い，正しい入出力が得られた回数についてまとめる．なお，以下では先行研究の[4][5]では扱われていない問題として，最大公約数を求めるプログラムの獲得についても行った．

[4]では，以下に述べる GRAPE の入出力の値には整数を用いている．しかし，さまざまな問題への応用を検討する際，すべての問題が整数の枠組みで解けるわけではないため，本論文では入出力の組および GRAPE 内部での変数として，すべて実数を用いた．

シミュレーションでは，GRAPE における個体群の総数 $N = 500$，ノードの長さを 50，進化の際に選択する親の数 $n_\mathrm{p} = 50$，各親個体から作成する子個体の数 $n_\mathrm{c} = 6$とした．また，すべての問題においてシミュレーション期間は

10000 ステップとし，その期間内に正しい入出力が得られない場合，プログラム作成に失敗したと判断した．

GRAPE のノードで使用した命令の種類を表 1 に示す．No.は命令の番号，ノードは命令実行後に遷移するノードの数，引数は使用するデータセットの種類，機能は命令の動作の説明である．例えば No.1 の命令は，データセット x と y を加算して z に代入した後，指定するノード（1 つ）へ遷移することを表わす．

一般的なコンピュータプログラムを作成する際には，命令として条件分岐，繰り返しを多用する．そのため，プログラム自動生成手法で効率的なプログラム作成には，それらを含むことが必要になると考えられる．くり返し命令の 1 つとして，実行内容を指定した回数くり返す for 命令があるが，GRAPE ではグラフ構造型なっており，そのまま実装することは難しい． [4][5]にはくり返し命令が含まれていないが，指定箇所を実行するごとにデータセットの値を加算し，それが一定値を超えたときに別のノードに条件分岐するようノードが並べば実現することは可能である．しかし，そのような内容が高い確率で発生することは想定できない．そこで本論文では，for に相当する命令を記号に含むことで，目的となるプログラムの自動生成を行う上での効率化を検討する．

表 1　GRAPE で使用するノード関数
Table 1　Node functions using in GRAPE

No.	ノード	引数	機能
1	1	x, y, z	$z \coloneqq x + y$
2	1	x, y, z	$z \coloneqq x - y$
3	1	x, y, z	$z \coloneqq x * y$
4	1	x, y, z	$z \coloneqq x/y$
5	1	x, y, z	$z \coloneqq x \bmod y$
6	1	x, y	swap x and y
7	2	x, y	if $x > y$ goto node1 else goto node2
8	2	x, y	if $x < y$ goto node1 else goto node2
9	2	x, y	if $x = y$ goto node1 else goto node2
10	2	x, y, z	if $x \leq y$ goto node1 else goto node2 & $x \coloneqq x - z$
11	2	x, y, z	if $x \geq y$ goto node1 else goto node2 & $x \coloneqq x + z$
12	0	x	output x

表 1 において，命令 No.10 は if 命令と同様に条件分岐を行う際，同時に条件部で使用するデータセット x から指定したデータセット z を引き，それが指定した値 y を下回るならばノード 1 に，そうでなければノード 2 へ遷移する．命令 No.11 も同様であり，for 命令に近い動作を実行する内容となっている．

4.1 問題

以下に，シミュレーションで正しいプログラムを構築する際に使用した問題の詳細を示す．

4.1.1 階乗

与えられた値 a から，$b = a!$を計算する．GRAPE への正しい入出力の組(a,b)として，(0,1) (1,1) (2,2) (3,6) (4,24) (5,120) を選択した．また GRAPE における各個体の評価値は，その出力をもとに次式から計算した．

$$1.0 - \frac{\sum_{i=1}^{n} \frac{|c_i - e_i|}{|c_i| + |c_i - e_i|}}{n} \quad (1)$$

n は入出力の組の数，c_i, e_i はそれぞれ i 番目の入力に対する正しい解と GRAPE からの出力を表す．$i = 1, \cdots n$ のすべての c_iとe_i が一致すると，評価値は最大値 1 となる．データセットには，初期値として 0～4 に a，5～9 に定数 1 を代入した．

4.1.2 累乗

与えられた値 a, b から，$c = a^b$を計算する．GRAPE への正しい入出力の組(a,b,c)として，(2,0,1) (2,1,2) (2,2,4) (3,3,9) (3,4,27) (3,5,81) (4,6,4096) (4,7,16384) (4,8,65536) を選択した．各個体の評価値は，式(1)から計算した．データセットには，初期値として 0～2 に a，3～5 に b，6～8 に定数 1 を代入した．

4.1.3 フィボナッチ数列

与えられた値 $a_0 = 0, a_1 = 1$ から， $a_{i+2} = a_{i+1} + a_i$ $(i \geq 0)$ を計算する．GRAPE への正しい入出力の組(i, a_i)として，(1,1) (2,1) (3,2) (4,3) (5,5) (6,8) (7,13) (8,21) (9,34) (10,55) (11,89) (12,144) を選択した．また各個体の評価値の計算には，式(1)を用いた．データセットには，初期値として 0～4 に i，5～9 に定数 1 を代入した．

4.1.4 最大公約数

与えられた値 a, b の最大公約数 c を計算する．GRAPE への入力の組 (a, b) として，$a = 1, \cdots, 60$, $b = 60$ の 60 通りを用意し，それぞれに対する最大公約数 c を正しい出力とした．各個体の評価値の計算には式(1)を使用し，データセットには初期値として 0～4 に a，5～9 に b，10～14 に定数 1 を代入した．

4.2 シミュレーション結果

4.1 節で示した各問題に対し，提案手法を用いて得られた結果を表 2 に示す．また，表 3 に[4]および[5]における結果を示す．[4]では GRAPE のノード数をさまざまに変更してシミュレーションを行っているため，表 3 ではその中で最も良い結果を示した．表 2 と表 3 を比較すると，本論文の提案手法により性能が大幅に向上していることがわか

る．

[4][5]では最大公約数を求める問題を行っていないため，結果の分析のため，図 4 にシミュレーション結果で得られた最大公約数を求めるプログラムを示す．プログラムは実行順に並べられており，各行において先頭の数値は行番号，$に続く値は使用するデータセットで，goto に続いて遷移先の行番号が記述されている．なお，33 行目のみ次に 6 行目に遷移するため，行末に遷移先を記した．構築されたプログラムは，ユークリッドの互除法とよばれるアルゴリズムで解を求める内容となっているが，考察は次節で行う．

表 2 提案手法での成功回数

Table 2 The number of times of a success using the proposed method

問題	成功回数
階乗	100
累乗	100
最大公約数	97
フィボナッチ数列	71

表 3 参考文献での成功回数

Table 3 The number of times of a success using the method in bibliography

問題	[4]の成功回数	[5]の成功回数
階乗	69	68
累乗	45	99
最大公約数	（結果なし）	（結果なし）
フィボナッチ数列	8	12

```
 0:if $11>=$11 goto 6 else goto 44
 6:$7=$9 mod $2
10:if $1>$1 goto 9 else goto 28
28:$4=$3*$12
45:$12=$11/$4
 5:$8=$7+$7
42:if $7>=$12 goto 40 else goto 26 & $7:=$7+$12
40:swap $2 $9
29:$13=$6-$2
20:if $11==$9 goto 17 else goto 46
46:if $6==$6 goto 18 else goto 35
18:$0=$3+$11
33:$2=$7 mod $1, goto 6
26:if $7<=$3 goto 30 else goto 14 & $7:=$7+-$13
30:output $2
```

図 4 最大公約数のプログラム

FIgure 4 Program for greatest common divisor

4.3 考察

表 2 と表 3 の結果を比較すると，提案手法で取り入れた個体群の多様性を維持する手法が，GRAPE により与えられた問題の正しいプログラムを構築する際に有効に機能しているといえる．MGG や JGG では，作成した子個体群の評価値の良いものを優先的に世代交代に用いているが，多様性の維持を考慮することが，プログラム自動生成手法では重要な役割を担っていることがシミュレーション結果から見て取れる．

提案手法で獲得できたプログラムの詳細を分析する．図 4 に示す最大公約数のプログラムから，全く使用されない変数に対する計算や，条件が必ず成り立つ条件分岐を除いたものを図 5 に示す．

```
1(6):$7=$9 mod $2
2(28):$4=$3*$12
3(45):$12=$11/$4
4(42):if $7>=$12 goto 5 else goto 7 & $7:=$7+$12
5(40):swap $2 $9
6(33):$2=$7 mod $1, goto 1
7(30):output $2
```

図 5 最大公約数のプログラム

Figure 5 Program for greatest common divisor (unnecessary line is omitted)

図 5 では，各行の先頭にある行番号を新たに付け直している．それにあわせて，4，6 行目の遷移先の番号を変更している．また，カッコ内は元の行番号を表わす．

(a, b) の組の最大公約数を求める際，$9 と$2 の初期値はそれぞれ b と a であり，その余りが$7 に計算される．もし$7 が 0 となり$2=$a$ が解ならば，7 行目に遷移し，$2=$a$ が解として出力される．そうでなければ，5 行目で$2 が$9 に入り，6 行目で$7=$b$ mod a が$2 に代入されて 1 行目に戻る．これにより，$a$ と b を a で割った余りとの最大公約数を求めることになり，ユークリッドの互除法として機能していることが分かる．

表 1 の中で，くり返し命令に相当する No.10,11 の有無がどの程度の性能の違いを生むのかについてもシミュレーションを行った．その結果を表 4 に示す． 表 2 と比較すると，若干ではあるが成功回数が減少していることから，くり返し命令がプログラム自動生成の際の効率化に影響しているといえる．

なお，本研究では作成されたプログラムの長さを評価していない．そのため，図 4 で示すプログラムには無駄な行が多く含まれたと考えられる．プログラムの進化の際，行数も考慮に入れた評価を行うことで，無駄な行数の増加を軽減できると考えられる．

表 4　提案手法での成功回数（No.10,11 を使用しない場合）

Table 4　The number of times of success using the proposed method (without functions No.10 and 11)

問題	成功回数
階乗	98
累乗	94
最大公約数	87
フィボナッチ数列	67

5. おわりに

本研究では，目的となる機能を人の代わりにコンピュータ内部で自動的に作成するプログラム自動生成手法の性能向上のため，個体群の多様性を維持するための手法を提案した．提案手法では，進化させるために選択した親個体から子個体群を作成する際に，その直系のみを残すことで，個体群としての多様性を維持している．そして，提案手法を用いたシミュレーション結果から，従来手法と比較して大幅に性能を向上させられることを示した．

参考文献

1 小林重信：実数値 GA のフロンティア、人工知能学会論文誌、Vol.24、No.1、(2009).
2 秋本洋平、永田祐一、佐久間淳、小野功、小林重信：実数値 GA における生存選択モデルとしての MGG と JGG の解析、人工知能学会論文誌、Vol.25、No.2、(2010).
3 宮前惇、佐久間淳、小野功、小林重信：インスタンスベース政策最適化のための実数値 GA と非ホロノミック制御系への適用、人工知能学会論文誌、Vol.24、No.1、(2009).
4 Shinichi Shirakawa, Shintaro Ogino and Tomoharu Nagao:Graph Structured Program Evolution, Proceedings of the Genetic and Evolutionary Computation Conference 2007 (GECCO '07), Vol.2, pp.1686--1693, London, England, 7-11 July (2007)
5 石堂眞大、白川真一、長尾智晴：グラフ構造のプログラム自動生成手法への ADF の導入、電子情報通信学会総合大会、D-8-8、(2009).