社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2007－MPS－66 (6)
2007／9／3

Graph Structured Program Evolutionによる複雑なプログラムの自動生成とその解析

白川真一[†] 長尾智晴[†]

これまでに自動プログラミングを行う進化計算法として，グラフ構造をプログラムの表現形式とする Graph Structured Program Evolution (GRAPE) が提案され，その有効性が示されている．GRAPE の各個体の遺伝子型は一次元の整数列で表現されるため，一般的な GA で用いられるような比較的簡単な遺伝操作を適用することができる．本報告では，GRAPE を用いて階乗，組合せの数を求めるプログラム，リストのソートを行うプログラムの自動生成を行うとともに複数の交叉方法を比較し，その解析を行う．

Automatic Generation of Complex Programs and its Analysis using Graph Structured Program Evolution

Shinichi Shirakawa[†] and Tomoharu Nagao[†]

As an Automatic Programming technique using Evolutionary Computation, Graph Structured Program Evolution (GRAPE) have been proposed and showed the effectiveness so far. The representation of GRAPE is graph structure, and the genotype of GRAPE is a linear string of integers. GRAPE is able to use simple genetic operators used in a usual Genetic Algorithm (GA) because of the genotype of a linear string of integers. In this report, we apply GRAPE to automatic generation of programs (the problems are factorial, combination and sorting a list). We also compare the several crossover techniques and analyze the evolutionary process.

1. はじめに

近年，コンピュータプログラムを自動生成する自動プログラミングに関する研究が非常に活発に行われている．遺伝的アルゴリズム（GA）を用いてコンピュータプログラムを自動的に構築する遺伝的プログラミング（GP）[1] では，一般的にプログラムの表現形式として木構造が用いられ，遺伝操作には部分木の交換による交叉やノードの突然変異などが用いられる．これまでにプログラムの表現形式として木構造ではなく，グラフ構造を用いてプログラムの自動生成を行う手法も研究されている[2)~4)]．グラフ構造を扱う利点としては，木構造に比べて複雑な表現が可能であることやグラフ上での時系列情報の保持が可能であることが挙げられる．

本報告で扱う **GRA**ph structured **P**rogram **E**volution (GRAPE)[5] もグラフ構造を利用した自動プログラミング手法である．GRAPE の特徴としてグラフ構造による表現の拡大，“データセット” を利用した複数データ型の取り扱い，一次元の整数列に対する遺伝操作などが挙げられる．GRAPE の各個体の遺伝子型は一次元の整数列で表現されるため，一般的な GA で用いられるような比較的簡単な遺伝操作を適用することができる．これまで GRAPE では遺伝操作として一様交叉と突然変異が用いられていた．本報告では，GRAPE を用いて階乗 n!，組合せの数 ${}_nC_m$ を求めるプログラム，リストのソートを行うプログラムの自動生成を行うとともに複数の交叉方法を比較し，その解析を行う．

2. Graph Structured Program Evolution (GRAPE)[5]

2.1 GRAPE の構造

GRAPE の構造例を**図 1** に示す．GRAPE のプログラムは有向グラフと “データセット” から構成される．“データセット” は有向グラフ中を流れ，各ノードにおいてそのノードに応じた処理が施される．各ノードでは “データセット” に対する処理や “データセット” を用いた分岐が行われる．GRAPE の実行時

† 横浜国立大学大学院環境情報学府
Graduate School of Environment and Information Sciences, Yokohama National University

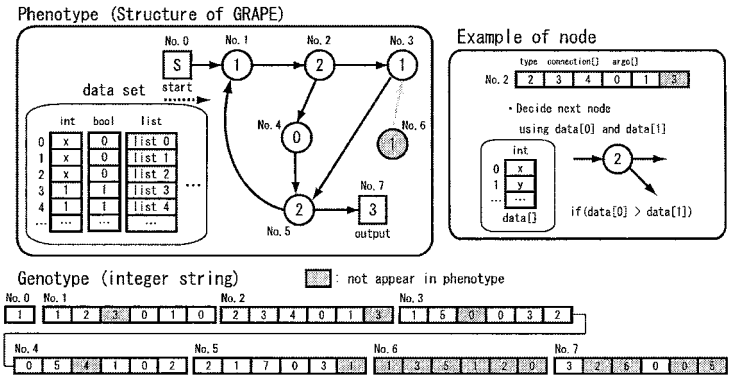

図 1 GRAPE の構造例

Fig. 1 Structure of GRAPE (phenotype) and the genotype which denotes a list of node types, connections and arguments.

には，まず“データセット”に初期値として入力値などをセットする．その後，スタートノードからプログラムを開始して各ノードを遷移していくことで処理が行われる．GRAPE ではグラフ構造をプログラムの表現形式としているため，分岐やループを含む複雑なプログラムの表現が可能である．さらに，グラフ中を流れる“データセット”に様々なデータ型を用意することで，複数のデータ型を 1 つのプログラム中で扱うことができる．各ノードではあらかじめ定められたデータ型を使って処理を行う．

GRAPE では表現型のグラフ構造を遺伝子型にマッピングし，遺伝子型に対して遺伝操作を行う．GRAPE の遺伝子型は各ノードの種類，接続，引数を定義した一次元の整数列で表現される．遺伝子型から表現型に変換する際，ノードの種類によっては接続先や引数を指定する遺伝子が表現型に発現しない場合もある．GRAPE の総ノード数はあらかじめ指定するため，染色体の遺伝子長は固定長になる．総ノード数は固定であるが，接続の状態によって使用されるノード（active node）と使用されないノード（inactive node）があるため，表現上はノード数は可変となる．

3. GRAPE における交叉方法の比較

GRAPE の各個体の遺伝子型は一次元の整数列で表現されるため，一般的な GA で用いられるような比較的簡単な遺伝操作を適用することができる．本報告では GRAPE における交叉方法の違いによる性能を比較するために，次に示す交叉方法を用いた．また，いずれの交叉方法も突然変異と併用される．

- 一様交叉（Uniform Crossover; UC）
  確率 P_c によって整数列にマスクパターンを生成し交叉点を決定する．本報告では P_c の値として 0.1 と 0.5 を用いる．
- 一様ノード交叉（Uniform Node Crossover; UNC）
  確率 P_c によってノードごとにマスクパターンを生成し交叉点を決定する．一様交叉との違いは整数列に対する交叉ではなく，ノード単位での交叉を行う点である．本報告では P_c の値として 0.1 と 0.5 を用いる．
- 二点ノード交叉（Two Point Node Crossover; TPNC）
  一様乱数によってノード単位で交叉点を 2 点決定する．
- 交叉を行わない（Mutation Only; MO）
  遺伝操作として交叉を使用せず，突然変異だけを用いる．突然変異は突然変異率 P_m によって遺伝子単位で発生するものとする．選択された遺伝子はランダムに変更される．本報告では P_m の値として 0.02 と 0.1 を用いる．木構造を扱う GP においては交叉がほとんど無意味であるという報告もされている[6]．

4. プログラムの自動生成実験

本節では GRAPE を階乗 n!（factorial），組合せの数 ${}_nC_m$（combination）を求めるプログラム，リストのソート（sorting a list）を行うプログラムの自動生成に適用し各種交叉方法の比較，解析を行う．

4.1 実験の設定

実験で使用した各パラメータ値を**表 1** に示す．生成されたプログラムを実行する際には，ノードの遷移回数に制限を設けることで停止しないプログラムに対応した．遷移回数が制限回数に達したプログラムは適応度として 0.0 が与えられる．本実験ではこの最大ノード遷移回数（Execution step limits）を 500（factorial），1000（combination），3000（sorting a list）と

表 1 実験に用いた各パラメータ値
Table 1 The parameters of GRAPE.

Generation alternation model	MGG*
The number of generations	100000
Population size	500
Children size (for MGG)	50
Crossover rate P_c	0.1, 0.5
Mutation rate P_m	0.02, 0.1
The number of nodes	50

* Minimal Generation Gap[7)]

した．これは，目的のプログラムを実行するために十分な数であると考えられる．本論文で用いるGRAPEのノード関数は整数型のデータに対する四則演算やリストの交換，比較などの基本的なものであり，問題ごとに設計は行わない．

個体の評価に用いるトレーニングデータとして，factorialの実験では0から5の入力値を，combinationの実験では，入力値（n, m）の組30例を，sorting a listの実験では，ランダムに発生させた長さ10から20のリスト30例を用いた．

適応度関数はfactorialとcombinationの実験では，次の式(1)を用いた．

$$fitness = 1.0 - \frac{\sum_{i=1}^{n} \frac{|T_i - O_i|}{|T_i| + |T_i - O_i|}}{n} \quad (1)$$

ここで，T_i はトレーニングデータ i の正しい出力値，O_i は生成されたプログラムのトレーニングデータ i に対する出力値である．n はトレーニングデータの総数である．Sorting a listの実験ではプログラムの出力がリストとなるため，適応度関数として次の式(2)を用いた．

$$fitness = 1.0 - \frac{\sum_{i=1}^{n} \frac{\sum_{j=0}^{l} (1 - \frac{1}{2^{d_j^i}})}{l_i}}{n} \quad (2)$$

ここで，d_j^i はトレーニングデータ i の j 番目の要素の位置と，プログラムが返したリストの要素の距離である．l_i はトレーニングデータ i のリストの長さ，n はトレーニングデータの総数である．これらの適応度関数では適応度は [0.0, 1.0] の範囲で与えられ，大きい数値ほど良い個体であるといえる．さらに，式(1)，式(2)で求めた適応度が1.0であった場合，適応度関数は次の式(3)を使用する．

$$fitness = 1.0 + \frac{1}{N_{active}} \quad (3)$$

ここで，N_{active} はプログラム中で使用されているノード数（the number of active nodes）である．この適応度関数では，active nodeが少ないプログラムほど良い個体としている．以上から各個体の評価としては，まず誤差の小さい個体ほど良い評価を与え，理想の出力が得られる個体にはactive nodeが少ない個体ほど良い評価を与える．つまり，評価値として1.0を超えた個体はトレーニングデータに対しては100% 正しい出力を返すプログラムであるといえる．

表 2 各交叉方法による成功率の比較
Table 2 Comparison of success rate using several crossover methods.

Cross method	factorial	combination	sorting
UC (P_c=0.1)	86	**3**	**76**
UC (P_c=0.5)	16	0	0
UNC (P_c=0.1)	91	1	75
UNC (P_c=0.5)	34	0	51
TPNC	80	2	72
MO (P_m=0.02)	**96**	2	67
MO (P_m=0.1)	90	0	2

Factorialの実験では，“データセット”にサイズ10の整数型のデータを用意し，プログラム実行時に，このデータ型のdata[0]-[4]に入力値n，data[5]-[9]に定数1をセットし初期値とする．Combinationの実験では，“データセット”にサイズ15のdouble型のデータを用意し，プログラム実行時に，このデータ型のdata[0]-[4]に入力値n，data[5]-[9]に入力値m，data[10]-data[14]に定数1.0をセットし初期値とする．Sorting a listの実験では，“データセット”に入力リストとサイズ15の整数型のデータを用意し，整数データ型のdata[0]-[4]に入力リストの長さ（List Length），data[5]-data[9]に0，data[10]-[14]に定数1をセットし初期値とする．

それぞれの実験について100回の独立な試行を行い評価する．

4.2 実験の結果と考察

100回の試行のうち，トレーニングデータに対して正しい出力が得られるプログラムが生成された試行回数をカウントしたもの（成功率）を表2に示す．いずれの問題においても，“UC 0.1”，“UNC 0.1”，“MO 0.02”は他の交叉方法と比べて良好な結果を示した．Factorialの実験では交叉を行わない“MO 0.02”が最も成功率が高かったが，combination，sorting a listの実験では交叉の有効性が観測できる．“UC 0.5”，“UNC 0.5”，“MO 0.1”では他の交叉方法と比べて，表現上で親と大きく異なる子個体が生成されやすいと考えられる．このことから，親から子へと大きな変更が少ない交叉方法を用いるほうが効率の良い進化が行えると考えられる．

図 2 はsorting a listの実験における各世代ごとの成功率を示したものである．“UNC 0.1”と“MO 0.02”が早い世代から正しい出力を得る個体を発見できていることに対し，“UC 0.1”では後半にかけての成功率の上昇が著しい．

図 3 は各世代における最良個体のプログラム中で

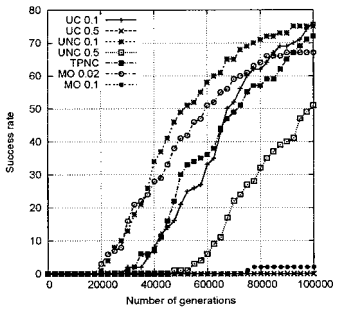

図 2 成功率の推移 (sorting a list)
Fig. 2 Transitions of success rate (sorting a list).

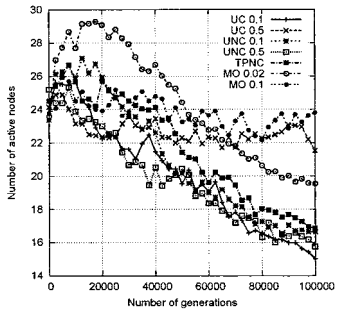

図 3 プログラム中で使用されているノード数の推移 (sorting a list)
Fig. 3 Transitions of the number of active nodes (sorting a list).

使用されているノード (active node) 数の推移である．式 (3) により，active node の数が少ない個体のほうが良いという評価をしているため，解が発見できた後の世代後半では active node 数が減少する傾向にある．世代前半を見てみると，"MO 0.02" の active node 数が大きいという特徴が分かる．これは低い確率の突然変異だけで次の世代の個体を生成するため，ノード間の接続が大きく変わることが少なく，比較的多くのノードが接続されやすいという特徴をもっていると考えられる．

最後に，combination の実験で自動生成された GRAPE のプログラム例を図 4 に示す．このプログラムは任意の入力 n，m に対して，${}_nC_m$ を返す一般的なプログラムとなっている．

5. ま と め

本報告では GRAPE を用いて，階乗 n!，組合せの数 ${}_nC_m$ を求めるプログラム，リストのソートを行うプログラムの自動生成を行い，交叉方法を変えて性能の評価を行った．active node 数の推移をみると，"MO 0.02" においてノード数が多くなる傾向が観測された．

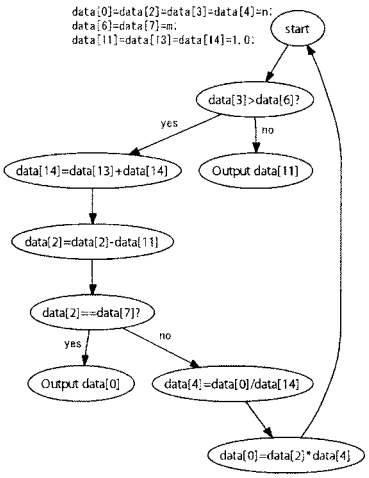

図 4 獲得したプログラムの例 (combination)
Fig. 4 Example of GRAPE program (combination).

今後は複雑なプログラムを生成するために有効な遺伝操作についてさらに検討を行う予定である．

参 考 文 献

1) Koza, J. R.: *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press (1992).
2) Teller, A. and Veloso, M.: PADO: A New Learning Architecture for Object Recognition, *Symbolic Visual Learning*, Oxford University Press, pp.81–116 (1996).
3) Miller, J. F. and Thomson, P.: Cartesian Genetic Programming, *Proceedings of EuroGP'2000*, LNCS, Vol.1802, Springer-Verlag, pp.121–132 (2000).
4) 平澤宏太郎，大久保雅文，片桐広伸，古月敬之，村田純一：蟻の行動進化における Genetic Network Programming と Genetic Programming の性能比較，電気学会論文誌 C，Vol.121, No.6, pp. 1001–1009 (2001).
5) Shirakawa, S., Ogino, S. and Nagao, T.: Graph Structured Program Evolution, *Proceedings of Genetic and Evolutionary Computation Conference (GECCO'07)*, Vol. 2, pp. 1686–1693 (2007).
6) Luke, S. and Spector, L.: A Comparison of Crossover and Mutation in Genetic Programming, *Genetic Programming 1997: Proceedings of the Second Annual Conference (GP97)*, Morgan Kaufmann, pp.240–248 (1997).
7) 佐藤 浩，小野 功，小林重信：遺伝的アルゴリズムにおける世代交代モデルの提案と評価，人工知能学会誌， Vol.12, No.5, pp.734–744 (1997).