2008－MPS－68 (5)
2008／3／4

プログラムサイズを考慮した自動プログラミングのための進化アルゴリズムの提案

白川真一[†1] 長尾智晴[†1]

現在までに遺伝的プログラミングに代表される自動プログラミングの手法が数多く提案されている．通常の自動プログラミング手法では，プログラムサイズを考慮した進化を行うことはないため，多様なプログラムサイズの探索が行えているとは限らない．本報告ではプログラムサイズを考慮した進化アルゴリズムである，Evolutionary Algorithm Considering Program Size (EACP) を提案する．EACP では特有の適応度割り当てと世代交代を行うことで，個体集団内のプログラムサイズの多様性を保持する．EACP をグラフ構造を表現形式とする自動プログラミング手法である Graph Structured Program Evolution (GRAPE) に適用し，階乗，累乗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，多様なプログラムサイズの探索が行えていることを確認する．

Evolutionary Algorithm Considering Program Size for Automatic Programming

Shinichi Shirakawa[†1] and Tomoharu Nagao[†1]

Today, a lot of Automatic Programming techniques have been proposed, such as Genetic Programming. Graph Structured Program Evolution (GRAPE) is one of the recent Automatic Programming techniques. Evolution in usual Automatic Programming techniques is not considered the evolution of program size. Therefore, it would not be search various program sizes. In this paper, a new Evolutionary Algorithm, called Evolutionary Algorithm Considering Program Size (EACP), is proposed. EACP maintains the diversity of program size in the population by using particular fitness assignment and generation alternation. We apply GRAPE with EACP to test problems, factorial, exponentiation and sorting a list. And we show the effectiveness of EACP and confirm evolution of maintaining the diversity of program size.

1. はじめに

現在までに遺伝的プログラミング (GP)[1] に代表される自動プログラミングの手法が数多く提案されている．木構造をプログラムの表現形式とする一般的な GP では，世代交代を繰り返していくうちに木が次第に大きくなってしまう *bloat* という問題が潜在的に存在することが知られている．*bloat* によって，個体集団をプログラムサイズの大きな個体が占めてしまい，探索が停滞する．これに対して，様々な解決策が提案されている．一つのアプローチとして，多目的進化アルゴリズムを用いて，プログラムサイズを目的関数の一つとすることで小さいサイズのプログラムを良い個体と評価して，*bloat* を抑制する Multi Objective Genetic Programming (MOGP) が提案されている[2]．

一方，プログラムの表現形式としてグラフ構造を用いる手法も数多く提案されている[3)-5)]．これらの手法の多くはグラフ構造の表現型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．これによって，*bloat* の問題が回避できる．しかし，個体集団内のプログラムサイズは進化において考慮されていないため，多様なプログラムサイズの個体を探索できているとは限らない．

本報告では，プログラムの進化を行う場合には多様なプログラムサイズを探索する必要があると考え，プログラムサイズの多様性を保持した進化を行う Evolutionary Algorithm Considering Program Size (EACP) を提案する．さらに，グラフ構造を表現形式とする自動プログラミング手法の一つである **GRA**ph structured **P**rogram **E**volution (GRAPE)[5),6)] と EACP を用いてプログラムの自動生成実験を行い，多様なプログラムサイズの探索が行えていることを確認する．

†1 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

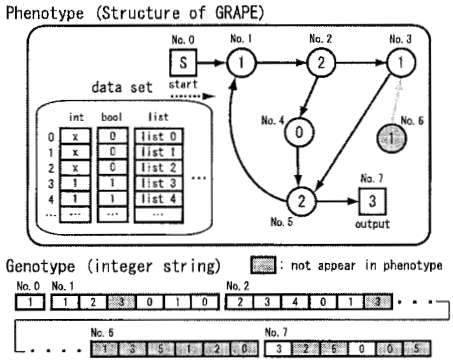

図 1 GRAPE の構造例

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

2. Graph Structured Program Evolution (GRAPE)[5),6)]

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．本報告では GRAPE におけるプログラムサイズを active node の数とする．

通常，GRAPE では世代交代モデルとして Minimal Generation Gap (MGG)[7)] が用いられ，Simple GA を用いた場合に比べて性能が高いことが示されている[5),6)]．しかし，MGG ではプログラムサイズを考慮した進化が行えないため，多様なプログラムサイズについて探索が行えているとはいえない．

3. Evolutionary Algorithm Considering Program Size (EACP)

EACP では特有の適応度割り当てと世代交代を行うことで個体集団内のプログラムサイズの多様性を保つ．各個体は出力値に対する適応度とプログラムサイズを考慮した EACP 独自の適応度が割り当てられる．EACP では MGG 同様に選択圧は生成された家族内だけに限定される．

3.1 EACP のアルゴリズム

図 2 は EACP の世代交代の概略を示している．EACP のアルゴリズムを次に示す．

Step 1. Initialization:
世代数を t=0 にセット．N 個の個体を初期個体集団 $P(t)$ としてランダムに生成する．

Step 2. Selection of Parents:
2 つの親個体 M を個体集団 $P(t)$ からランダムに選択する．

Step 3. Recombination:
親個体 M に交叉，突然変異を適用し，m 個の子個体 C を生成する．

Step 4. EACP Fitness Assignment:
個体集団 $P(t)+C$ を用いて親個体 M と C に EACP fitness を割り当てる（*cf.* Section 3.2）．

Step 5. Selection:
家族 $M+C$ の中で EACP fitness が最も高い個体と 2 番目に高い個体を選択し，個体集団 $P(t)$ 内の 2 つの親個体 M と入れ替え，$P(t+1)$ を得る（ここで，個体 A と B が同じ EACP fitness を割り当てられていた場合，プログラムの出力値に対する適応度 F が高いほうをより良い個体とする．i.e. $(F_{eacp}(A) = F_{eacp}(B)) \wedge (F_A > F_B) \Rightarrow$ A is evaluated better individual.）

Step 6. Stop Criteria:
終了条件を満たしていれば終了，そうでなければ $t = t+1$ として step 2 へ．

3.2 EACP の適応度割り当て

EACP の適応度割り当てでは，個体集団内のプログラムサイズを考慮して適応度が割り当てられる．EACP では，個体集団内で同じプログラムサイズの個体が多数存在する場合，これらの個体の適応度は低くなる．

EACP の適応度割り当ての手順を次に示す．

(1) 式 (1) によって各個体 $i \in (M+C)$（親と子個体）について $n(i)$ を計算する．$n(i)$ は個体 i とプログラムサイズが同じ，かつプログラムの出

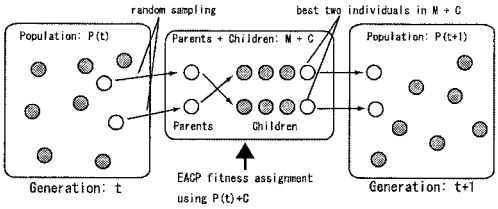

図 2 EACP の世代交代の概要
Fig. 2 Outline of generation alternation in EACP.

力値に対する適応度が同じ，または大きい個体の数である．

$$n(i) = \sum_{j \in P(t)+C, j \neq i} x_i(j), \qquad (1)$$

$$x_i(j) = \begin{cases} 1 & \text{if } (S_i = S_j) \wedge (F_i \leq F_j) \\ 0 & \text{otherwise} \end{cases}$$

ここで，S_i は個体 i のプログラムサイズ，F_i は個体 i のプログラムの出力値に対する適応度である．この場合，F の値が大きいほど良い個体であることを示す．GRAPE では，プログラムサイズ S は表現型における active node の数である．

(2) 各個体 $i \in (M + C)$ は式 (2) によって EACP fitness value ($F_{eacp}(i)$) を割り当てられる．

$$F_{eacp}(i) = \frac{1}{n(i)+1} \qquad (2)$$

図 3 は EACP の適応度割り当ての例である．図 3 では，個体 A と同じプログラムサイズで出力値に対する適応度が大きい個体が 2 つ存在する (i.e. $n(A) = 2$)．そのため，個体 A の $F_{eacp}(A)$ は 0.33 となる．個体 B と同じプログラムサイズで出力値に対する適応度が大きい個体は存在しない (i.e. $n(B) = 0$) ため，$F_{eacp}(B)$ は 1.0 となる．個体 C と同じプログラムサイズで出力値に対する適応度が同じ個体が 1 つ存在する (i.e. $n(C) = 1$) ため，$F_{eacp}(C)$ は 0.5 となる．

4. プログラムの自動生成実験

本節では GRAPE を用いて階乗 n! (factorial)，累乗 a^b (exponentiation) 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を行い，進化アルゴリズムとして SGA，MGG，SPEA2，EACP を用いた場合の比較を行う．SPEA2 は代表的な多目的進化アルゴリズムの一つである．

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．SGA ではトーナメント選択 (トーナメントサイズ:2) とエリート保存戦略 (エリート数:1) を採用した．SPEA2

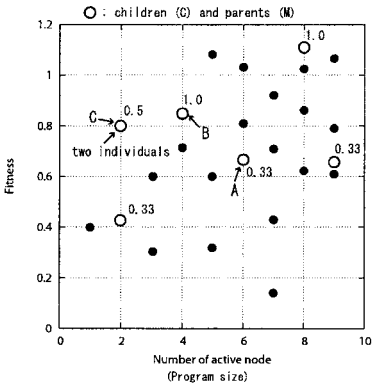

図 3 EACP の適応度割り当ての例
Fig. 3 Example of fitness assignment in EACP.

表 1 実験に用いた各パラメータ値
Table 1 The parameters used in the experiments.

The number of evaluations	5000000
Population size	500
Child size (for EACP, MGG)	50
Uniform crossover rate P_c	0.1
Crossover rate	0.7
Mutation rate P_m	0.02
The maximum number of nodes	30

ではアーカイブサイズを 500 とし，目的関数としてプログラムの出力値に対する適応度とプログラムサイズを用いた．つまり SPEA2 を用いた場合，小さなプログラムサイズの個体が生存しやすい．

本実験で用いる 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．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．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，exponentiation の実験では，入力値 (a, b) の組 9 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．また，本実験では正規化された誤差をプログラムの出力値に対する適応度 F として使用した．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．さらに，生成したプログラムの出力値が全てのトレーニングデータと一致した場合には，ノード遷移回数が少ないプログラムのほうが良い個体であるという評価関数を用いる．

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

表 2 各アルゴリズムの 100 回試行中の成功回数の比較
Table 2 The number of successful runs after 5000000 fitness evaluations for each algorithm over 100 runs.

	SGA	MGG	SPEA2	EACP
Factorial	19	80	7	**95**
Exponentiation	2	**39**	0	37
Sorting a list	1	72	0	**80**

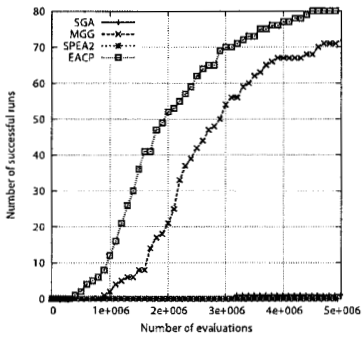

図 4 Sorting a list の実験における成功回数の推移
Fig. 4 Transitions of the number of successful runs for each algorithm over 100 runs (Sorting a list).

をカウントしたものを表 2 に示す．いずれの問題においても MGG，EACP が高い性能を示していることが確認できる．Factorial，Sorting a list の実験では EACP が最も高い成功回数を達成しており，EACP の有効性を示すことができた．**図 4** は Sorting a list の実験における各世代ごとの成功回数の推移を示したものである．EACP が早い段階で正解のプログラムを生成することができており，さらにどの世代においても最も高い成功回数を示していることが確認できる．**図 5** は個体集団内のプログラムサイズ（active node 数）の分散の推移である．EACP が一定の値を保ち，個体集団内に多様なプログラムサイズの個体を保持しているのに対して，他の進化アルゴリズムでは世代が進むに連れてあるプログラムサイズに個体が偏っていることが分かる．このことから，目的である個体集団内で多様なプログラムサイズを保持する進化が実現できたといえる．SPEA2 では個体が小さなプログラムサイズに偏ってしまい，大きなプログラムサイズの個体を探索できないため，成功率が低くなってしまったと考えられる．

5. ま と め

本報告では，プログラムサイズを考慮した進化アルゴリズムである EACP の提案を行った．グラフ構造をプログラムの表現形式とする GRAPE を用いて階乗，累乗，ソーティングのプログラムの自動生成を行い，EACP の性能を評価した．他の進化アルゴリズムと比較したところ，EACP は高い成功回数を達成した．また，多様なプログラムサイズの個体を探索することができることを確認した．

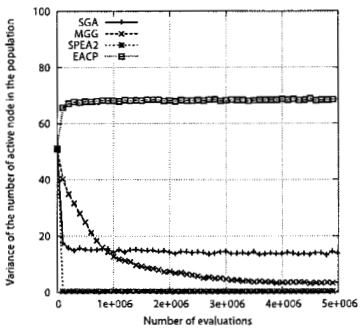

図 5 Sorting a list の実験における個体集団内のプログラムサイズの分散の推移
Fig. 5 Transitions of the variance of the number of active nodes (program size) in the population for each algorithm over 100 runs (Sorting a list).

今後は様々な問題に EACP を適用し性能評価を行うとともに，GRAPE 以外の自動プログラミング手法に適用し検証を行う予定である．

参 考 文 献

1) Koza, J. R.: *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press (1992).
2) Bleuler, S., Brack, M., Thiele, L. and Zitzler, E.: Multiobjective Genetic Programming: Reducing Bloat Using SPEA2, *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*, IEEE Press, pp.536–543 (2001).
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) 白川真一，長尾智晴：Graph Structured Program Evolution によるプログラムの自動生成，電気学会論文誌 C， Vol.129, No.3 (2008).
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).