Information Processing Society of Japan

5L－9

汎用Linear GPシステムの実装と評価 *

徳井 直生† 伊庭 斉志† 石塚 満†

東京大学工学部電子情報工学科‡

1 はじめに

遺伝的プログラミング（GP）とは、遺伝的アルゴリズム（GA）の拡張であり、進化論的計算手法のプログラム自動合成への応用である[1, 2]。C/C++言語によるGPシステムの多くは、ポインタで表現された木構造を遺伝子として扱う。その一方で機械語のようなリニアな遺伝子を扱うLinear GPも提案されている[5]。しかし、現在ポインタ方式のシステムが広く使われているのに対して、汎用のLinear GPシステムは作られていない。そこで本研究では、C言語を用いた汎用LinearGPシステムを構築する。GP個体の木構造を配列で表現することによってポインタ参照のオーバーヘッドを減らし、高速かつメモリ消費の小さいシステムを目指す。

2 システムの概要

本システム"LGPC:Linear GP System in C"の概要を示す。システム全体を適合度計算などの問題固有部分と遺伝オペレータなどの共通部分とに分け、問題記述ファイルの書き換えによって様々な問題に対応できるようにした。また、高速な探索を可能にする手法として注目されている自動関数定義（ADF:Automatic Difined Function）を実装した[3]。

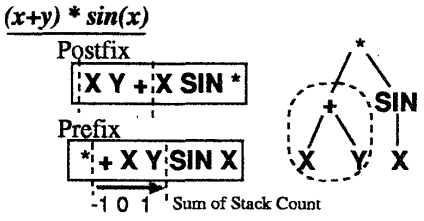

図 1: GP個体の表現形式とスタックカウント

* LGPC:Linear GP System in C, Its Implementation and Evaluation

† Nao Tokui, Hitoshi Iba, Mitsuru Ishizuka

‡ University of Tokyo,7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan

e-mail: tokui@miv.t.u-tokyo.ac.jp

2.1 GP個体の表現

GP個体の表現形式としては、引数の後に関数を書くPostfix形式と、その逆のPrefix形式が考えられる（図1）。Postfix形式ではIf-then-else等の制御文の実装が困難なので、本研究ではPrefix形式を採した[4]。実際には、ノードの種類別に振ったid番号を配列に格納することで各GP個体を表現する。

2.2 プログラムの構造解析

配列表現された木構造を解析するために、スタックカウント（スタックに対するプッシュとポップの回数の差、以下SC）の合計が部分木内で1になることを利用する方法[4]を用いた（図1）。

2.3 遺伝オペレータ

初期化は配列の先頭から順に、関数や定数、変数をSCが1になるまでランダムに格納する。本システムでは、最大配列長だけではなく木構造の深さによっても個体の大きさを制限できるようにした。交叉は、SCの計算によって求めた部分木の交換による。また先頭からのSCが等しい点を交叉点として入れ替えるというGA的な一点交叉も採り入れた。さらに、初期化と交叉の手法を組み合わせることで突然変異を実現した。

2.4 GP個体の評価

Prefix形式の遺伝子は、配列を先頭から順に読みだして再帰的に処理することで実行される。具体的には、配列内の位置を示すグローバル変数をインクリメントし、そのノードの評価関数を呼び出す動作を繰り返す。

3 評価実験

本システムを評価するために、C言語を用いたポインター方式のGPシステムとして、最も一般的なSGPC（Simple GP System in C）との比較実験を行なった。

NII-Electronic Library Service

Information Processing Society of Japan

3.1 記号当てはめ問題

$(x,y) = (0,0),(0.1,0.05),...,(0.9,0.405)$ の 10 組の入出力値を訓練データとして、未知関数 $y = x^2/2$ の近似式を求めさせた。個体数 1000 の集団に対して 100 世代にわたって、最良個体の適合度（2 乗誤差）、世代交代にかかる時間を計測した結果、LGPC は SGPC の 5 分の 1 程度の時間で SGPC 以上の適合を示した（図 2,3）。また、100 世代目の評価時のメモリ消費量を計測したところ、SGPC が 8236（ページ単位）かかるのに対して、LGPC は 890（SGPC の 10.7%）であった。

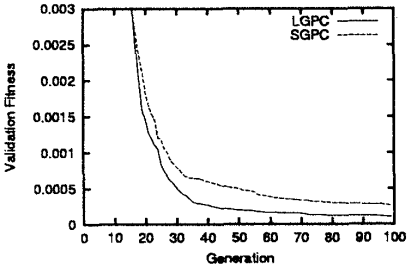

図 2: 最良個体の適合度

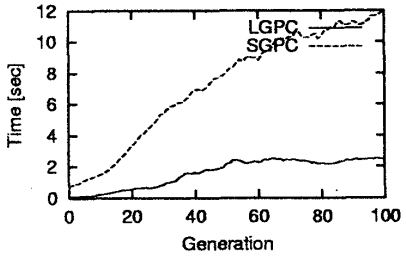

図 3: 世代交代に要した時間

3.2 時系列予測問題

非線形な振舞いをする Macky-Grass 微分方程式

$$\frac{dx(t)}{dt} = \frac{0.2x(t-17)}{1+x^{10}(t-17)} - 0.1x(t)$$

から生成される時系列を過去のデータ $x(t-1), x(t-2)$ 等をもとに予測する [2]。図 4、5 の結果では、LGPC が SGPC の 2 倍程度の速度で収束している。

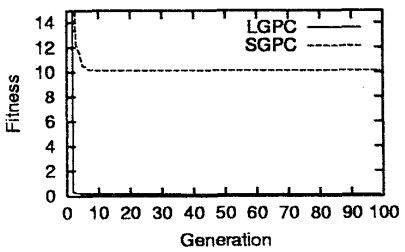

図 4: 最良個体の適合度

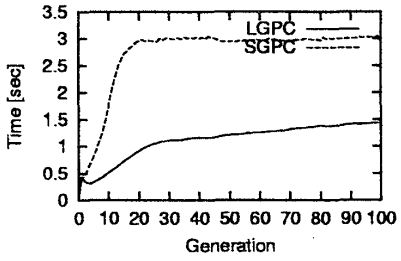

図 5: 世代交代に要した時間

4 考察

時系列予測では、三角関数や平方根などの比較的複雑な関数を使用したため、関数計算にかかる時間が長い。記号当てはめに比べて実行時間の短縮率が小さいのは、評価にかかる時間に占める関数計算の割合が大きいためであると考えられる。

また、SGPC を使った場合に x(t) の予測として x(t-1) という局所解に陥る現象が見られた。LGPC でも若い世代で同様の現象が観察されたが、その後、より好成績な予測式へと移っている。したがって、LGPC は一点交叉と ADF 等によって SGPC とは異なるより広範囲な探索を行っているものと推測される。

5 おわりに

従来のポインタ方式の GP との比較実験から、本システムが高速かつ低メモリ消費であることが確かめられた。現在、条件文による制御構造を含む ANT 問題についても同様の実験を行っている [2]。また、ADF や一点交叉が Linear GP の収束率にあたえる影響等の研究は今後の重要な課題である。

参考文献

[1] John Koza:"Genetic Programming, On the Programming of Computers by means of Natural Selection",MIT Press,1992

[2] 伊庭斉志:"遺伝的プログラミング", 東京電機大学出版局,1996

[3] John Koza:"Genetic Programming II: Automatic Discovery of Reusable Subprograms",MIT Press,1996

[4] Mike J.Keith and Martin C.Martin:Genetic Programming in C++ : Implementation Issues. In Kinnear Jr.,K.E.,editor,*Advances in Genetic Programming*,chapter 13,pages 285-310,MIT Press,Cambridge,MA,1994

[5] Wolfgang Banzhaf, Peter Nordin, Robert E.Keller, Frank D.Francone:"Genetic Programming An Introduction",dpunkt.verlag and Morgan Kaufman Publishers, Inc.,1998

NII-Electronic Library Service