遺伝的プログラミングにおけるプログラム木構造の最適化に関する研究

The study of optimizing the structure of program tree in genetic programming

室伏 康利[*1]
Yasutoshi Murofushi

井上 聡[*1*2]
Satoru Inoue

[*1]埼玉工業大学大学院
Graduate School of Engineering , Saitama Institute of Technology

[*2]埼玉工業大学
Saitama Institute of Technology

Genetic Programming (GP) is well-known solving method for several kinds of procedure. One of the characteristics of this solution is that solved answer in this method is expressed as the tree structure. In the evolutional process in this method, some parts in tree structure which inhibit or slowdown the evolution appear accidentally. To solve this problem, we proposed more effective evolutional algorithm. In our algorithm, system detects the occurrence of useless part of the tree structure in evolutional process and remove them beforehand.

1. はじめに

遺伝的アルゴリズム(Genetic Algorithm:以下 GA)の拡張として遺伝的プログラミング(Genetic Programming:以下 GP)が生まれた.

GA は遺伝子で表現した個体を用いることで組み合わせ最適化問題などのさまざまな問題に対応したメタヒューリスティクスアルゴリズムの 1 つである. GPは GA と違い遺伝子表現に配列であったものを木構造にすることで GA では表現できなかった数式や手続きなどの表現が可能になった. また, GA ではなかった GP 特有の問題点も存在するのも事実である.

その代表的なものとして, GP の木構造が進化していく過程で進化を阻害してしまう部位があらわれデータが無駄に肥大化してしまうことが挙げられる. 解探索途中の木構造が無駄に大きくなってしまうことで解が収束するのを防ぐ結果となってしまう. こういった問題を抱えるため多くの研究がされ改善, 改良されている.

2. 研究内容

本研究では GP の進化の過程において, 世代毎に生成される木構造に対し, その中に含まれる非効率的な部位を抽出し, 最適化していくことで結果として最適解へ効率的に収束する方法について考察する.

そのためにまずは, 非効率的な部位を判別し抽出する必要がある. 非効率的な部位としているが部分的に有益な解を含んでいるため厳密には非効率とは言い切れない場合がある.

主に木構造の生成過程において解への収束を阻害する面が部分解として機能するよりも目立つものを非効率的な部位と呼ぶ場合である.

どのような部位がそのような場合にあたるのかを定義する必要がある. 以下の様なものを例として挙げる.

(1) 部位を 1 つにまとめることができる場合

図 1(左)に示すように, +3 を表すノードが 2 回続くことで表されていたものを+6 と表すことで 1 つのノードに圧縮することができる. 他にも+3 といった部位に-3 が続いた場合には±0 で表すといった形で省略することにより, 効率的な部位にすることが可能である.

(2) 条件分岐が不必要な場合

条件分岐する場合には分岐が不必要な場合が存在する. これは分岐の有無に関わらず分岐後の結果が同一だった場合(図1右)などは分岐自体使用せずに結果だけを使用することで省略し効率的な部位とすることが可能となる.

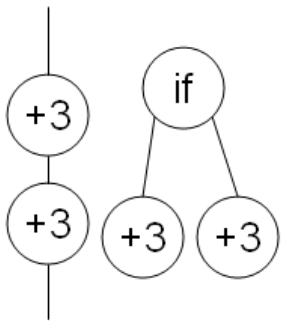

図1 非効率的な1例

以上の様な例が存在するが, これらは GP が解くべきタスクのルールにより様々に変化し, 非効率的であるかどうかはタスク次第である. そのため, タスクに応じた柔軟な対応が必要となる.

3. 最適化シミュレーション

3.1 シミュレーション条件

本研究で提案する最適化法の効果を検証するための実験タスクとして, Santa Fe Trail(人工蟻の探索)問題を使用した. この問題は決められた平面内を歩き回る人工蟻を配置する. その平面内に配置した餌を決められたエネルギー内でどれだけ効率的に集められるかといった問題である. 詳しいルールと終了条件は次のとおりである.

連絡先:〒369-0293 埼玉県深谷市普済寺 1690 埼玉工業大学大学院, m9007wqk@sit.sc.jp

(1) ルール設定

- 蟻は動作（向きの変更，前進する）を行うたびにエネルギーを1失う．
- 蟻は 1 つ前のセル内に餌があるかどうか判別が可能である．
- 餌がある場所に進んだとき餌の所持数のパラメータに1を加える．マス上の餌を消去する．

(2) 終了条件

- エネルギーがなくなってしまったとき
- 空間内の範囲外に出てしまったとき
- 予め指定したゴールに辿り着いたとき

また，蟻の動作をプログラム木で生成するために以下の非終端記号（表 1）と終端記号（表 2）を用いた．

表 1 非終端記号の定義

非終端記号	
If_Food_Ahead(x,y)	前方に餌がある場合 x を実行し，ない場合には y を実行する．
Prog2(x,y)	x,y を順次実行する．
Prog3(x,y,z)	x,y,z を順次実行する．

表 2 終端記号の定義

終端記号	
RIGHT	右に 90 ° 向きを変える．
LEFT	左に 90 ° 向きを変える．
MOVE	前方に 1 歩前進する

非効率的な部位の抽出は全ての世代で行う場合と，10 世代ごとに行う方法の 2 パターンで行う．効率化は通常の GP の1世代の最後の操作として効率化の手法を導入する．

適応度は以下のように設定した．

$$(\text{適応度}) = \frac{\text{餌の取得数}}{\text{餌の全数}}$$

比較に用いる値は以上の 2 つを比較対象として用いた．

- 適応度
- 収束までの世代数

この解を求める進化の過程において，非効率的な部位を抽出し効率化を図ったプロセスを採用した今回の 2 つの手法と，採用しない従来手法とを比較する．

3.2 実行結果

本研究で提案する効率化の手法において，その対象となる木構造の部位を図 2 に示す．このような部位に対して 2 章で示したような効率化を施すことによって，その進化の過程にいかなる変化があったのかを図 3 に示す．従来手法と新手法を比較すると，新手法を用いた場合，その平均適応度が早い段階で収束域レベルまで達しており，またその平均適応度は従来手法より高いことが確認できた．

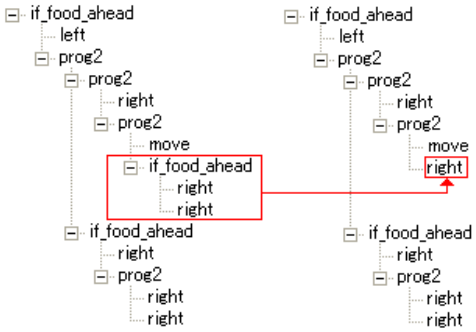

図 2 木構造を効率化できる例

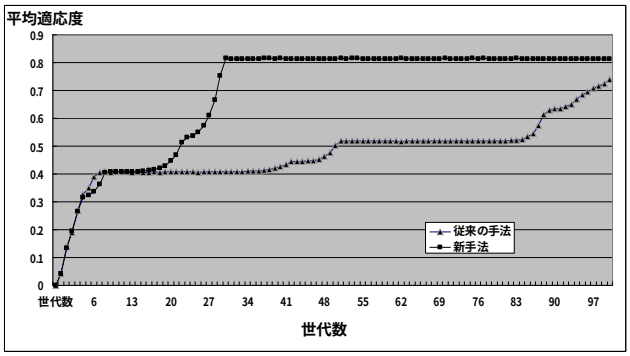

図 3 平均適応度の時間変化

4. 考察と課題

今回のシミュレーションにおいて，非効率な部分の圧縮，除去プロセスが，GP での進化プロセスの効率化に一定の効果をもたらすことが確認できた．しかしながら非効率的な部位の判別には予め発見した基本的なものを発見，効率化したに過ぎない．また，今回検証に用いた Santa Fe Trail では簡単なルートを用いたため，本研究で提案しているアルゴリズムの有用性を主張するには，より多様なルートでシミュレーションを行う必要があると考えられる．その際には世代数などの値も増減し，違いをより明確にしたい．これからの課題として，他の実験タスクでの検証，非効率的な部位の抽出，効率化のアルゴリズムを提案し，更なる効率的手法を展開していくことである．

参考文献

[平野 00] 平野 廣美，遺伝的アルゴリズムと遺伝的プログラミング，パーソナルメディア社，2000.

[伊庭 94] 伊庭 斉志，遺伝的アルゴリズムの基礎―GA の謎を解く，オーム社，1994.

[伊庭 01] 伊庭 斉志，遺伝的プログラミング入門，東京大学出版会，2001.