2D4-4

遺伝的プログラミングにおけるイントロン発現の検出に関する研究

The study of detecting the appearance of introns 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

procedure evolutionary whose solution is expressed by tree structure. During the evolutional process, unnecessary increase of the tree structure called bloat can be seen. This excess increase of tree structure is induced by introns which appear in the evolutional process. In this study, we proposed the method detecting and omitting the intros which do not influence on fitness value in early period of evolutional processing.

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, これらは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.

そのような部位は大きく分けて2つある.

一つ目にブロートがある. これは簡潔にまとまっていないものである.

2つ目が, イントロンである. これは取り除かれても結果が変わらないものである. これも2つに分けることが出来る.

(1) 意味的イントロン

実行されるが適合度に影響を与えないもの（図1左）.

(2) 構文的イントロン

意味的イントロンと違い実行自体が行われないもの. 図1右では上の構造により b が実行されない場合が当てはまる.

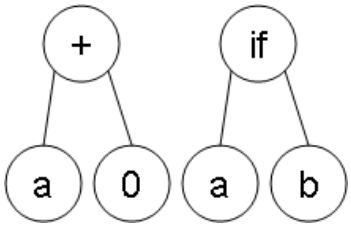

図1. イントロンの例

ブロートは可変長のデータに現れる現象であり, 実行の遅延や, 過学習などにつながるとされている.

イントロンは存在することで交叉の破壊から守ると言った面がある. また, 新しいコードの生成に一役買うとされている.

本研究では, イントロンの発現を検出し, 状況に応じてそれらを除去, 残留させることで効率的な木構造の生成を行う.

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, 多数決判定問題を使用した. この問題は 3 ビットの入力状態のうち 2 ビット以上が ON になっていれば結果として ON を, それ以外だと OFF を返す関数を作成する問題である.

今回用いた終端記号は 3 ビットの状態を表す3つ D1, D2, D3 を用いた. 非終端記号として, AND,OR, NOT 演算を行うものを用いた. また, AND 演算ものは引数が3, 2のものの2つがある. それらを組み合わせ, 多数決を表す関数を作成していく. また, 教師データとして表1を使用した. この教師データと目標値としシミュレーションを行う.

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

表 1. 教師データ

D1	D2	D3	出力
1	1	1	ON
1	1	0	ON
1	0	1	ON
1	0	0	OFF
0	1	1	ON
0	1	0	OFF
0	0	1	OFF
0	0	0	OFF

適合度は以下のように設定した.

$$\text{適合度} = \frac{\text{適合している出力数}}{\text{教師データの出力数}}$$

以下の値を従来の手法と今回の新手法の比較対象として用いた.

- 適合度
- 収束までの世代数
- 1 世代あたりのノード数の平均

この解を求める進化の過程において, イントロンをある一定の世代数において抽出を行い, それらをある一定の確率で除去すると言った手法を行った.

3.2 実行結果

実行結果を図2, 3, 4 に示す.

図 2 は今回抽出したイントロンの一例を示す. 図2の例では、not と not の論理演算は適合度に影響を与えない意味的イントロンに分類されるため, 今回イントロンとして抽出した.

図 3 は x 軸を世代数, y 軸を平均適合度とし, 平均適合度を比較したものである. 適合度ではイントロンを除去したことにより新手法では適合度にばらつきがあるが, 従来の手法よりも早く解に収束している.

図 4 は x 軸を世代数, y 軸を平均ノード数とし平均ノード数を比較したものである. 平均ノード数においては, 新手法が従来の手法よりも少ないことが目に見えてわかる.

これにより, イントロンはの発現を抽出し除去することで適合度の変化は少なかったが, ノード数を減らすことは確認出来, また, 従来の手法よりも効果を上げることが出来た.

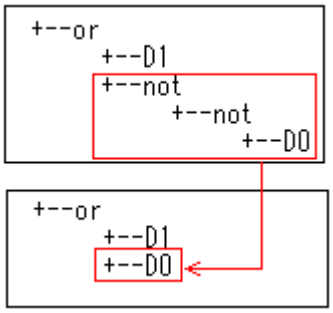

図 2 抽出したイントロン

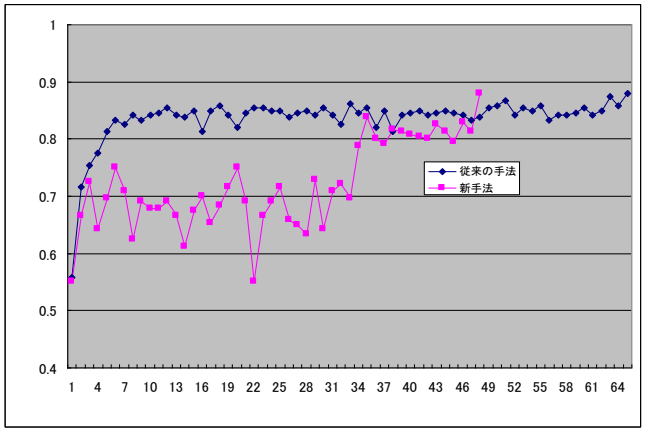

図 3 世代毎の平均適合度

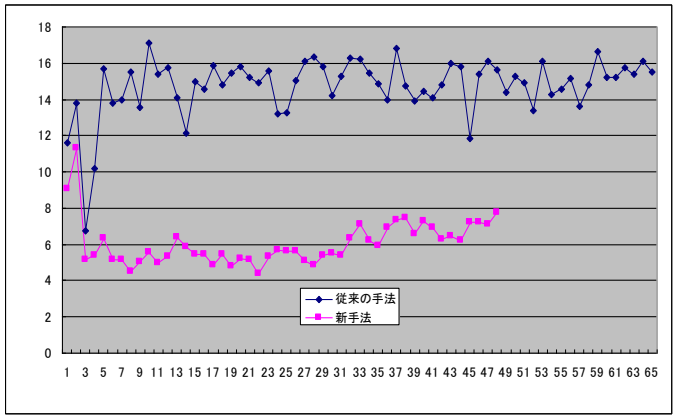

図 4 世代毎の平均ノード数

4. 考察と課題

今回のシミュレーションにおいて, イントロンを抽出しそれらを除去し, 一定の効果が得られることがわかった. 新手法では平均適合度においてばらつきが激しく, 構造を守るイントロンを除去することでコードを破壊してしまった面も見られる. イントロンを除去する頻度を調整することでこれらの状況に対応していきたい. また, この他のタスクでもシミュレーションを行い, イントロン発現の検出を行っていきたい. これからの課題として, イントロン発現の基準を設け, より明確な指標を用いて世代毎のイントロンをどの様に扱うかを考えていきたい.

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 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.