適応的ノード選択による遺伝的プログラミングの効率改善

玉 秀列[†] 宮下和雄[†,††] 西原清一[†]

遺伝的プログラミング（GP）は，問題を解決するための計算機プログラムを探索する進化型探索アルゴリズムである．GPを用いて問題解決を行うためには，GPが生成するプログラムの中に用いられるノードとして，対象問題における十分な情報が利用できる必要がある．反面，与えられたノード集合が冗長である場合は，GPにおける探索空間が大きくなり，GPの探索性能は低下することになる．本論文では，GPによる問題解決において冗長なノード集合から有用なノードを獲得するための新しいアプローチに関して述べる．我々は，冗長なノード集合から無用なノードを削除するために，ノードの重みに基づいた適応的な突然変異手法を提案し，記号当てはめ式問題を用いた実験で，提案した手法が有用なノードを効果的に選択しGPの効率を改善することを示した．

Improving Efficiency of GP by Adaptive Node Selection

Sooyol Ok,[†] Kazuo Miyashita[†,††] and Seiichi Nishihara[†]

Genetic Programming (GP) is an evolutionary search algorithm which searches a computer program capable of producing the desired solution for a given problem. For the purpose, it is necessary that GP system has access to a set of features that are at least a superset of the features necessary to solve the problem. However, when the feature set given to GP is redundant, GP suffers substantial loss of its efficiency. This paper presents a new approach in GP to acquire relevant nodes from a redundant set of nodes. We propose the adaptive mutation based on node weighting mechanism for eliminating irrelevant nodes from the redundant node set. We show empirically that the proposed method is effective for find relevant nodes and improving efficiency of GP in the experiments on symbolic regression problems.

1. はじめに

遺伝的プログラミング（GP）は，生物の進化過程を模倣した探索アルゴリズムである遺伝的アルゴリズム（GA）の拡張形である．GAは，「環境により適合した個体が生き残り，適合しない個体は淘汰される」という進化論に従って，評価関数により与えられる適合度を基に個体の選択を行い，それらに遺伝オペレータ（再生，交叉，突然変異など）を作用させることで次世代を生成する．この世代交代プロセスの繰返しにより，より良い解の反復探索を行う．GAにおける個体は，遺伝子型と表現型という2つの側面を持っている．遺伝子型は，生物における染色体構造に相当し，個体の個々の性質を決定する部分であり，遺伝オペレータによる操作の対象となる．表現型は，環境内での遺伝子型の発現型であり，主に個体評価の対象とされる．

GAに対するGPの拡張は，この遺伝子型に構造的表現を取り入れたという点である．GPでは，染色体表現としてグラフ構造や木構造といった構造的表現を取り扱えるように拡張されている．構造的な遺伝子型を扱えることにより，GPではより広い探索空間における探索が可能になる．このため，GPを用いることにより，より複雑な問題に対しても進化型の探索処理が適用できると考えられる．GAが主に最適化を目指すのに対して，GPは記号処理的なプログラム生成を目的としている．

GPを実際にさまざまな応用問題に適用するためには，次のような基本要素を設計しなければならない[1)]．

- 関数ノード：関数として使う記号，LISPのS式での関数．
- 終端ノード：終端（ターミナル）として使う記号，LISPのS式でのアトム．
- 適合度関数：問題に応じてプログラムの適応度を評価するための関数．

† 筑波大学工学研究科電子・情報工学系
Institute of Information Sciences and Electronics (IISE), University of Tsukuba
†† 独立行政法人産業技術総合研究所
National Institute of Advanced Industrial Science and Technology (AIST)

- パラメータ：交差，突然変移の起こる確率，集団サイズなど．
- 終了条件：実行を終了するための条件．たとえば，最大世代にまで達したとき，あるいは目的とするプログラムが得られたときなど．

GPにおける染色体表現は，関数ノードと終端ノードの組合せによって構成された木構造であり，実行可能なプログラムを表現する．木を構成するノードは，対象問題領域の解を表現するために適当に設計された固定の記号集合である．GPは，それらのノードで構成される階層的なプログラムの形やサイズをダイナミックに変えることにより，構成可能なプログラムの空間の中で，より良いプログラムを適応的・確率的に探索する手法である．GPを用いて与えられた問題を効率的に解決するためには，木構造を構成する各ノードを適切に設計することが重要であり，その効果的な設計には対象とする問題に関する十分な事前知識が必要とされる．GPを適用しようとしている問題領域に関する事前知識が不足している場合には，GPの適用を繰り返すことにより，問題解決のために有用なノードを試行錯誤的に選ぶ必要がある．これは非常に時間のかかる面倒なプロセスであり，問題領域に関する十分な事前知識を持たない現実的問題にGPを応用する際に解決すべき大きな障害の1つになる．

一般に，GPで種々の問題を解く状況において，必要最小限のノードをあらかじめ最適に決定することは非常に困難である．そのため，GPの適用にあたっては，ノードを冗長に設計し，解探索を行うプログラム空間を大きく設定する方が解の見逃しを防ぐという点でより現実的である．しかしながら，それにより，選択されたノード集合には無用なノードが多く含まれることになり，GPの探索効率は拡張した仮説空間でむだな探索を行うことで低下してしまう．GPの性能を改善するためには無関係なノードを取り除いて，有用なノードだけを自動的に抽出することが望ましい．

GPにおいて適切なノードを決定する過程は，他の機械学習パラダイムにおける特徴選択，あるいは特徴部分集合選択[2)~4)]と同様の意味を持つ．機械学習においては，学習する概念に無関係な特徴が学習アルゴリズムに与えられていると，概念の学習に必要な訓練事例数が爆発的に増大することが知られている．特徴選択は，与えられた特徴集合の中から，概念の学習に有用な少数の特徴を識別・選択するものであり，機械学習分野では活発に研究されている．

しかしながら，GPにおける無関係なノードの問題は，現在まであまり研究の対象とされてこなかった．GPの進化過程においては，評価の低いプログラムは自然と淘汰され，集団内のプログラムにおける冗長なノードの割合は徐々に減少していく[5)]とされていた．しかしながら，実際には真に有用な特徴量の規定が困難な現実の複雑な問題（たとえば，株価の変動予測問題など）に対してGPを適用する際には，冗長なノードの存在によってもたらされるGPによる探索の非効率さは深刻な問題になる．

本論文では，GPにおいて有用なノードの選択を加速化するための新しいアプローチに関して述べる．まず，次章では機械学習における特徴選択の研究との関係について述べる．その後，GPの進化過程で無用なノードを削除する新しいアプローチに関して詳しく述べる．最後に，我々が提案したアプローチの有効性を確かめるために無用なノードを含む2つの記号当てはめ式問題の実験を行い，提案した方法をこれらの問題に適用した結果に対する詳しい分析を行う．

2. 機械学習における特徴選択

機械学習を困難にする要因の1つに，特徴空間の次元数の問題がある．一般に，機械学習を行う場合，利用者は訓練事例を記述するための特徴を増やしすぎる傾向にある．これは，事例を記述する特徴の数を増やせば，それだけ学習に有効な情報が多く手に入り，学習率が上昇すると期待できることによるものである．しかしながら，事例記述のための特徴の数を増やせば増やすほど，学習すべき概念とは無関係な特徴が混入する可能性も増大する．また，特徴空間の次元の増大は探索空間の拡大を生じ，適切な概念の獲得に必要な計算量を増加させる結果（次元の呪い）に陥る．一般的に，概念学習に必要な訓練サンプル数は，特徴数のとともに指数的に増えていくのに対して，特徴量の増大とともにデータ収集のコストは高くなるので，実際に得られる訓練サンプル数は限られている場合が多い．したがって，不用意に特徴空間の次元数を増やせば，かえって機械学習の性能は低下してしまう．このことから，事前に適切に特徴選択を行い，事例記述に用いる特徴数を削減することは機械学習の重要な課題の1つである．

特徴量の評価選択に関しては，これまでにもパターン認識や統計などの分野で研究が行われている．これまで機械学習方式においては特徴選択のために主として2つのアプローチが提案されてきた[2)]．1つは明示的なヒューリスティック探索による特徴部分集合選択法，もう1つは特徴重み付けによる特徴選択法である．

ヒューリスティック探索アプローチでは，探索空間

の各状態を可能な特徴の組合せの部分集合で記述する．ヒューリスティック探索アプローチには，特徴部分集合に対する探索方向の観点から，順方向選択手法と逆方向排除手法が提案されている．さらに，学習プロセスとは独立した評価基準を用いて，学習に対して事前に適切な特徴部分集合を選択するフィルタ手法や，学習プロセス自体を用いて最適な特徴部分集合の評価選択を行うラッパー手法という分類も可能である．フィルタ手法では適切な帰納バイアスの設計が重要であるが，John らによって提案されたラッパー方法[6]では，属性集合の候補（利用可能属性集合の部分集合）を，学習アルゴリズムそのものを用いて評価することを繰り返すことによって，最適な属性集合を求められる．属性集合の候補は，その属性のみから記述されたデータを学習アルゴリズムに入力し，学習結果の精度を交差解析することによって評価される．

特徴重み付け法による選択では，学習過程において学習目標（概念）と特徴との関連深さの度合いを反映する重み値が各々の特徴に割り当てられる．一般的に，ヒューリスティック探索アプローチは，その探索結果が人間が理解できることが重要であったり，結果が他のプログラムの入力になったりする場合には自然な手法である．一方，重み付けアプローチはオンラインの段階的な特徴選択が簡単に実現できるため，純粋に学習性能のみを考慮したとき，重み付けアプローチが有利であることが多い．一般に長時間にわたる GP の問題解決過程を考慮すると，GP において有用なノードを選択するには効率的なオンライン学習の方が好ましい．したがって，我々は GP におけるノード選択方法として，ノードの重み付けに基づく選択手法を用いることにした．次章では，GP の進化過程において無用なノードを削除するアプローチに関して詳しく説明する．

3. GP におけるノード選択

GP において，ノード集合の設定は解の探索性能を大きく左右する最も重要な要素である．しかしながら，生成されるべき望ましいプログラムにとって冗長，あるいは無関係なノードが存在することがもたらす問題は，いまだ研究者による十分な議論が行われていない．Koza は，GP の基本的な枠組をまとめた最初の本[7]の中で，GP の進化過程においてはランダムな突然変異によってツリー集団中の冗長なノードは徐々に減少していくので，無関係なノードの存在は一般的に GP の探索効率を悪化させるだけで，GP による問題解決にとって深刻な問題ではないと述べている．しかしながら，現実の問題においては，ノードとして利用可能な情報の中に，問題解決に関連する可能性はあるが，その関連性が不明確なものが数多く存在する．このような現実の問題に対して GP を適用する際には，冗長なノードによって生じる非効率性は深刻な問題になる．本章では，GP を用いた問題解決過程において，有用なノードの選択を加速化するためのノードの重み付けに基づいた新しい突然変異手法について述べる．

3.1 ノードの重み付け方法

機械学習における一般的な特徴重み付け方法では，概念の学習に有用な特徴の重みは学習過程の中で段階的に増加される．我々は，GP において有用なノードを抽出するために，GP の処理過程で個体（プログラム）の適合度評価が行われる際に，より適合度の高いプログラム中で用いられているノードに対して，その重みを増加させる処理を反復的に繰り返して，各ノードの重みの更新を行う．提案したノードの重み付けの方法では，ノードの重みは以下の式を用いて更新される．

$$W_n(g) = \sum_{i \in S_g} (fit(i) * freq_n(i)) + W_n(g-1)$$

ただし，

$W_n(g)$：世代 g でのノード n の重み

$fit(i)$：個体 i の適応度（非負の値）

$freq_n(i)$：個体 i におけるノード n の出現回数

S_g：世代 g における上位 10%の適応度を持つ個体の集合

適合度の高い（集団全体の上位 10%）プログラムに含まれているノードには，そのプログラムの質（適合度）とプログラム中のノードの出現頻度に従って重みが加えられる．このようにして，ノードがプログラムの適合度に与える影響の大きさに応じて，ノードの有用性が繰り返し評価され，ノードの重みの値が更新される．

3.2 適応的突然変異

一般的に，GP における突然変異処理は，個体中からランダムに突然変異点を選択することから始まる．この突然変異点は，プログラム中の関数ノードでも終端ノードでもよい．一般的なランダムな突然変異では，選択した突然変異点以下のプログラムに含まれるサブツリーを削除し，新しくランダムに生成されたサブツリーをその突然変異点に付け足す．従来の GP では，通常すべてのノードに対して，同一の確率で突然変異点が選択される．我々は無用なノードを削除するために新しい突然変異操作を用いることにし，それを適応的突然変異と呼ぶ．提案した適応的突然変異では，ノードの重みに応じて突然変異点が選択される．

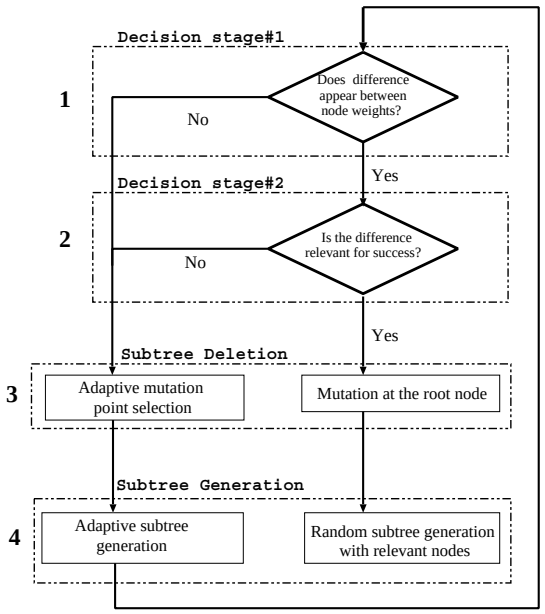

図 1 適応的突然変異
Fig. 1 Adaptive mutation.

すなわち，重みの小さなノードほど高い確率で突然変異点として選択される．そして選択されたノード以下のサブツリーはプログラムから削除される．したがって，GP の進化過程において，ある程度の割合で突然変異が起きる場合には，適応的突然変異によって無用なノードがプログラムに含まれる可能性が徐々に低くなっていく．適応的突然変異の概念図を**図 1** に示す．我々が提案した適応的突然変異は以下の 4 ステップから成る．最初の 2 ステップは有用なノードの選択に関係し，残る 2 ステップはサブツリーの削除と生成過程である．各ステップに関して，以下に詳しく説明する．

(1) ノード候補の抽出

問題解決にとって有用なノードと無用なノードを識別するために，ノードの重みに基づいてノード集合を 2 つのカテゴリにクラスタリングする．クラスタリングを行うに際して，まずノード重みに正規化を行い，さらに平滑化のために値域（$[0, 1.0]$）を 10 等分に分割する．そして分割された各領域に対して，個々のノード重みの値に応じて，以下の式に基づき各領域におけるノードの分布を世代ごとに更新する．

$$H(i)=\begin{cases} +0.5 & \left(i=\left\lfloor 10\dfrac{W_n(g)}{\sum_{n=1}^{N} W_n(g)}\right\rfloor \text{のとき}\right) \\ +0.2 & \left(i=\left\lfloor 10\dfrac{W_n(g)}{\sum_{n=1}^{N} W_n(g)}\right\rfloor \pm 1 \text{のとき}\right) \\ +0.1 & \left(i=\left\lfloor 10\dfrac{W_n(g)}{\sum_{n=1}^{N} W_n(g)}\right\rfloor \pm 2 \text{のとき}\right) \end{cases}$$

ただし，

i：領域インデックス（0 から 9）

N：ノード集合のサイズ

$W_n(g)$：世代 g でのノード n の重み

$H(i)$：領域 i におけるノードの累積分布

上の式において，+0.1，0.2，0.5 などの数値や $i = \lfloor 10 \times W_n(g) / \sum_{n=1}^{N} W_n(g) \rfloor \pm 2$ などの条件は，ノード分布の形状を制御するために適当に定められたものである．これらの値や条件を変えることにより，ノード識別のタイミングを調整することが可能である．このようにして世代ごとにノード重みの分布に対する累積ヒストグラムを求めると，世代を経るに従い累積ヒストグラムに双峰性が現れる．我々は，そうした双峰性の出現を，ノードの重み値を用いて有用なノードと無用なノードの識別を行うのに個体集合が十分に進化した兆候であり，ノードにはその有効性を判別するための証拠が重みとして十分に蓄積されたと考える．

上記のクラスタリングのアルゴリズムを簡単な例を用いて説明する．たとえば，第 1 世代での 10 個のノードに対してその重みを次のとおりと考える：$W_1(1) = 3$, $W_2(1) = 2$, $W_3(1) = 4$, $W_4(1) = 5$, $W_5(1) = 6$, $W_6(1) = 7$, $W_7(1) = 14$, $W_8(1) = 26$, $W_9(1) = 0$, $W_{10}(1) = 6$．その場合，ノードの重みの合計 $\sum_{n=1}^{N} W_n(g)$ は 73 になる．さらに各ノードにおいて，$W_n(g) / \sum_{n=1}^{N} W_n(g)$ を求めると次のようになる．

$$W_1(1)/\textstyle\sum_{n=1}^{N} W_n(1) = 0.041 \quad (3/73)$$
$$W_2(1)/\textstyle\sum_{n=1}^{N} W_n(1) = 0.027 \quad (2/73)$$
$$W_3(1)/\textstyle\sum_{n=1}^{N} W_n(1) = 0.053 \quad (4/73)$$
$$W_4(1)/\textstyle\sum_{n=1}^{N} W_n(1) = 0.068 \quad (5/73)$$
$$W_5(1)/\textstyle\sum_{n=1}^{N} W_n(1) = 0.082 \quad (6/73)$$
$$W_6(1)/\textstyle\sum_{n=1}^{N} W_n(1) = 0.095 \quad (7/73)$$
$$W_7(1)/\textstyle\sum_{n=1}^{N} W_n(1) = 0.191 \quad (14/73)$$
$$W_8(1)/\textstyle\sum_{n=1}^{N} W_n(1) = 0.356 \quad (26/73)$$
$$W_9(1)/\textstyle\sum_{n=1}^{N} W_n(1) = 0.000 \quad (0/73)$$
$$W_{10}(1)/\textstyle\sum_{n=1}^{N} W_n(1) = 0.082 \quad (6/73)$$

したがって，

$$\lfloor 10 \times W_1(1)/\textstyle\sum_{n=1}^{N} W_n(1) \rfloor = \lfloor 0.41 \rfloor = 0$$
$$\lfloor 10 \times W_2(1)/\textstyle\sum_{n=1}^{N} W_n(1) \rfloor = \lfloor 0.27 \rfloor = 0$$
$$\lfloor 10 \times W_3(1)/\textstyle\sum_{n=1}^{N} W_n(1) \rfloor = \lfloor 0.53 \rfloor = 0$$
$$\lfloor 10 \times W_4(1)/\textstyle\sum_{n=1}^{N} W_n(1) \rfloor = \lfloor 0.68 \rfloor = 0$$
$$\lfloor 10 \times W_5(1)/\textstyle\sum_{n=1}^{N} W_n(1) \rfloor = \lfloor 0.82 \rfloor = 0$$
$$\lfloor 10 \times W_6(1)/\textstyle\sum_{n=1}^{N} W_n(1) \rfloor = \lfloor 0.95 \rfloor = 0$$

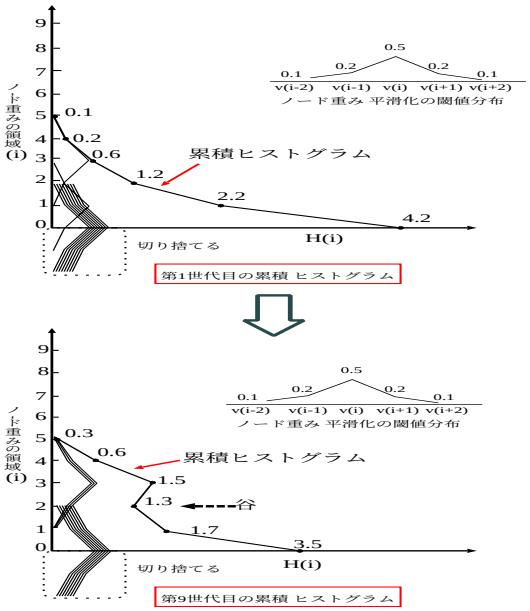

図 2 クラスタリング手法
Fig. 2 Clustering method.

$$\lfloor 10 \times W_7(1) / \sum_{n=1}^{N} W_n(1) \rfloor = \lfloor 1.91 \rfloor = 1$$
$$\lfloor 10 \times W_8(1) / \sum_{n=1}^{N} W_n(1) \rfloor = \lfloor 3.56 \rfloor = 3$$
$$\lfloor 10 \times W_9(1) / \sum_{n=1}^{N} W_n(1) \rfloor = \lfloor 0.00 \rfloor = 0$$
$$\lfloor 10 \times W_{10}(1) / \sum_{n=1}^{N} W_n(1) \rfloor = \lfloor 0.82 \rfloor = 0$$

が得られる．

以上の結果から，式 $H(i)$ を用いて各領域におけるノード重みの分布を求めると次のようになる．

$H(0) = 0.5 \times 8 + 0.2 = 4.2$
$H(1) = 0.2 \times 8 + 0.5 + 0.1 = 2.2$
$H(2) = 0.1 \times 8 + 0.2 + 0.2 = 1.2$
$H(3) = 0.1 + 0.5 = 0.6$
$H(4) = 0.2$
$H(5) = 0.1$

さらに，たとえば第 9 世代においてノードの重みが以下のように変化したと考える：$W_1(9) = 13$, $W_2(9) = 32$, $W_3(9) = 574$, $W_4(9) = 14$, $W_5(9) = 602$, $W_6(9) = 14$, $W_7(9) = 29$, $W_8(9) = 570$, $W_9(9) = 6$, $W_{10}(9) = 43$. これらの値に基づき，上の同様に $H(i)$ を求めると，$H(0) = 3.5$, $H(1) = 1.7$, $H(2) = 1.3$, $H(3) = 1.5$, $H(4) = 0.6$, $H(5) = 0.3$ となる．

このようにして各世代ごとに求められるノード重みの分布に対する累積ヒストグラムを図示すると，**図 2** のようにある世代の累積ヒストグラムからはグラフに双峰性が現れ，ノードをその重みの大きさに応じて 2 つのグループにクラスタリングする際の指標として用いることができる．

(2) **ノードの有用性の判定**

上で選ばれた大きな重みを持つカテゴリに属するノードが，本当に問題解決のために有用なノードかどうかを判断する．そのために，適合度の高いプログラムの中に，選択されたノード候補が実際に利用されているかどうかを調査する．もし，大きな重みを持つカテゴリに属するノード集合と，適合度の最も高い上位 1%のプログラム集団に含まれているノード集合が一致したとき，そのノード集合は有用なノードとして判定される．この判定は，GP における進化過程では，十分に進化した適合度の高いプログラムには有用なノードだけが含まれている，という仮定に基づいている．

(3) **サブツリーの削除**

突然変異操作では，まず突然変異点を決定し，突然変異点の以下のサブツリーを削除する．提案した適応的突然変異の場合，有用なノードが見つかったかどうかによって，以下の 2 つのタイプの突然変異点選択方法から該当するものが適用される．

- **有用なノードがまだ見つかっていない場合：**
  ノードの重みに反比例して突然変異点が選択される．
- **有用なノードが見つかった場合：**
  すべてのツリーのルートを突然変異点として選択する．これは有用なノードだけで全集団を再初期化するためである．この処理は GP のプロセスにおいて，1 度だけ行われる．

(4) **サブツリーの生成**

突然変異点が選択された後，新しいサブツリーを生成し，突然変異点に挿入する．適応的突然変異では，有用なノードが見つかったかどうかによって 2 つのタイプのサブツリー生成方法がある．

- **有用なノードがまだ見つかっていない場合：**
  ノード重みの比例して選択したノードを用いてサブツリーを生成する．
- **有用なノードが見つかった場合：**
  有用なノードだけを用いてサブツリーを生成する．

有用なノードが定められた後は，適応的突然変異は従来の GP のランダム突然変異とまったく同様な操作を行う．ただし，有用なノードが選択された後の進化過程は，選択された有用なノードのみを用いて続けられる．

4. 実　　験

提案した適応的突然変異手法の有効性を検証するた

表 1 GP パラメータ
Table 1 GP parameters.

Objective	Evolve a function that fits the data points of the fitness cases
Terminal set	$x_1 \ldots x_{33}$
Function set	$+, -, *, \%, \sin, \cos, \exp, rlog$
Fitness cases	The given sample of 200 data points $\{x_1(i), \cdots, x_{33}(i)\}$ each terminal's value is in the interval $[-1, +1]$, (1) $y = x_1^3 + x_1^2 + x_1$ (2) $y(i) = \sin(x_4) + \sin(2x_4) + \sin(3x_4) + \sin(4x_4) + \sin(x_5) + \sin(2x_5) + \sin(3x_5) + \sin(x_{13}) + \sin(2x_{13})$
Raw fitness	The sum, taken over the 200 fitness cases, of the absolute value of difference between value of the dependent variable produced by the S-expression and the target value y_i of the dependent variable.
Hits	Number of fitness cases for which the value of the dependent variable produced by the S-expression comes within 0.01 of the target value of the dependent variable.
Parameters	Population size = 2000, Generation = 200 for the function (1), Generation = 1000 for the function (2).
Crossover Probability	80%
Reproduction Probability	10%
Mutation Probability	10%
Success Predict	An S-expression scores 200 hits

めに，記号当てはめ問題を用いて実験を行った．記号当てはめ問題はある未知関数 f に対する入力を $\{x_1, x_2, \cdots, x_n\}$，出力を $y = f(x_1, x_2, \cdots, x_n)$ とし，GP は，いくつかの入力値と出力値のペアを与えられて，f を同定，もしくは近似する式を与えられた関数と終端記号の組合せにより生成する．本実験では，記号当てはめ対象として 1 変数の関数と 3 変数の非線形関数の 2 種類を用意し，各々の問題に対してまったく問題とは無関係なノードを含む 33 個の終端ノードをあらかじめ GP に与えて実験を行った．本実験では，通常の GP の応用において，より重要な問題になる終端ノードに注目して実験を行ったが，本研究で提案した手法は，無用な関数ノードの削除にもまったく同様に応用することができる．

本実験で実際に用いた記号当てはめ対象である 2 つの未知関数式は，以下のとおりである：

- $f_1(x_1, \ldots, x_{33}) = x_1^3 + x_1^2 + x_1$
- $f_2(x_1, \ldots, x_{33}) = \sin(x_4) + \sin(2x_4) + \sin(3x_4) + \sin(4x_4) + \sin(x_5) + \sin(2x_5) + \sin(3x_5) + \sin(x_{13}) + \sin(2x_{13})$

各未知関数に対して，各変数の値を $[-1.0, 1.0]$ の範囲でランダムに選択した 200 組の訓練データを用意し，そのデータを GP に与えることで記号当てはめを行う．本実験で用いた GP のパラメータは，**表 1** に示すとおりである．本実験の実装は，汎用的な GP ツールである lil-gp1.0 [8] を一部修正して行った．

4.1 単純な記号当てはめ問題

単純な記号当てはめ問題として，Koza[7] の中で，冗長な無効終端ノードの GP に対する影響を調べるために用いられたものと同じ以下の式を用いた．

$$y = x_1^3 + x_1^2 + x_1$$

この問題では，用意された終端ノード集合に含まれる終端ノードのうち，実際に式に用いられる有効終端ノードは 1 個だけで，他の 32 個の終端ノードは記号当てはめすべき対象の式とは無関係な値を持つ．各終端ノードは独立で，その値は $[-1.0, 1.0]$ の範囲でランダムに選択される．この実験で用いたその他のパラメータは表 1 に示している．本論文で示した結果は，ランダムシードを変更し，100 回実験を試行した平均結果である．

多くの無効ノードが含まれる問題に対して GP を適用する際に，適応的突然変異方法が有効ノードの選択に効果的であるか否かを調べるために，本実験では次のような 3 つのタイプの実験を行い，その結果を比較した．

(1) 無効ノードを含まない標準 GP（Standard GP(1)）
この実験は有効終端ノード x_1 だけが与えられたランダム突然変異の操作を行う標準的な GP を用いた実験である．

(2) 無効ノードを含む標準 GP（Standard GP(2)）
この実験は有効終端ノード x_1 に無効終端ノード $x_2 \ldots x_{33}$ の 32 個を加えた終端ノード集合を用いたランダム突然変異の操作を行う標準的な GP を用いた実験である．

(3) 無効ノードを含む適応的 GP（Adaptive GP）
この実験は有効終端ノード x_1 に無効終端ノード x_2 ～x_{33} の 32 個を加えた終端ノード集合で適応的突然変異の操作を行う GP を用いた実験である．

図 3 は適応的 GP の実験において世代にともなう終端ノードの重みの変化を示したものである．このグラフでは 33 個の終端ノードの重みの総和が 1.0 になるよう正規化されている．このグラフから，適応的 GP

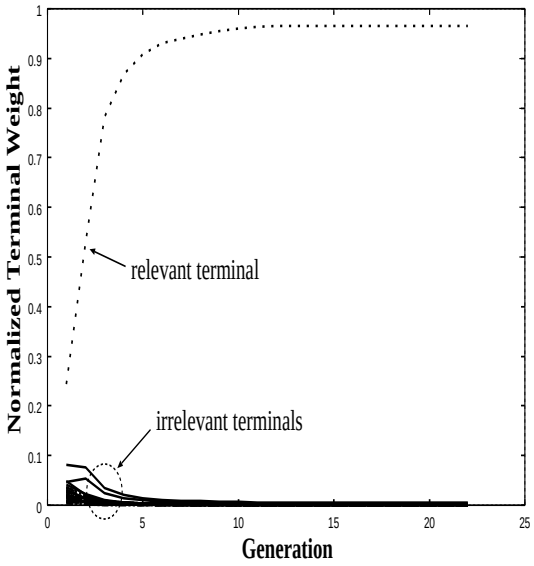

図 3 簡単な記号当てはめ式の問題に対する終端ノード重みの変化
Fig. 3 Change of terminal weights in simple regression problems.

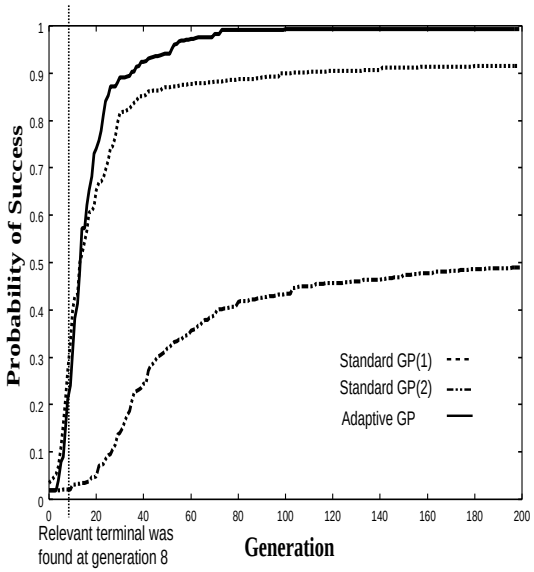

図 4 簡単な記号当てはめ式問題における成功率
Fig. 4 Success rate in simple regression problems.

は無効終端ノードを含む 33 個の終端ノード集合の中から，比較的早い世代（実際には 8 世代目）で有効終端ノードを抽出できたことが分かる．すなわち，適応的突然変異におけるノード選択圧力は，ノード集合に多数の無効ノードが含まれている際に，進化の方向性に対して大きな影響を与えることが可能であると考えられる．**図 4** は，上で言及した 3 タイプの GP 実験において解の成功確率を示している．このグラフから，適応的 GP は有効ノードを選択する前までは，有効ノードだけ含まれている標準 GP(1) よりも解探索効率の点で劣っているが，有効ノード集合が選択された後では，適応的 GP の性能は標準 GP(1) に匹敵することが分かる．また，適応的 GP や標準 GP(1) と標準 GP(2) の結果を比較すると，多くの無効ノードを含む標準 GP(2) の結果がかなり悪いことが分かる．

有効ノードが 1 つだけである簡単な記号当てはめ式問題の実験からは，適応的突然変異手法が効果的に有効ノードを見つけて，GP の性能向上に大きく寄与することが分かった．しかし，有効ノードが複数ある場合，適応的突然変異手法がすべての有効ノードを効率的に選択できるかどうかはこの実験では確認できていない．そこで我々は，有効終端ノードを 3 つ持つより複雑な記号当てはめ式の問題に対して実験を行った．

4.2 複雑な記号当てはめ式問題

2 番目の実験として，次のような式を未知関数とするより困難な記号当てはめ問題に対して，上と同様の実験を行った．

$$y = \sin(x_4) + \sin(2x_4) + \sin(3x_4) + \sin(4x_4) + \sin(x_5) + \sin(2x_5) + \sin(3x_5) + \sin(x_{13}) + \sin(2x_{13})$$

この問題での終端ノードの値の設定と GP パラメータは，以前の簡単な記号当てはめ問題と同様である．本問題では，与えられた終端ノード集合に含まれる 33 個の終端ノードのうち，有効終端ノードは x_4，x_5，x_{13} の 3 個である．さらに，3 個の有効終端ノードは，個々の変数値の関数値に対する影響度が異なるため，無効終端ノードと有効終端ノードを区別するのが困難である．我々は，この問題でも前記と同様の 3 タイプの実験に対して，ランダムシードを変えて 100 回つづ実験を行った後，その平均結果を比較した．**図 5** は，適応的 GP による世代に対する終端ノードの重みを変化を示している．このグラフから，有効終端ノードと無効終端ノードの重みの差が世代が経つにつれ，明らかになることが分かる．本実験では，適応的 GP が約 90 世代で有効終端ノードのすべてを選択するのに成功することが確認された．**図 6** は，3 タイプの GP による解の成功確率を示している．無効ノードを含まない標準 GP(1) での成功確率を比べることにより，本問題が前記の単純な記号当てはめ問題よりも難しいことが分かる．このグラフから，初期段階では適応的 GP は無効終端ノードを含む標準 GP(2) と同程度の性能であるが，世代が進むにつれて，適応的 GP の性能が改善され，有効終端ノードだけを含む標準 GP(1) の

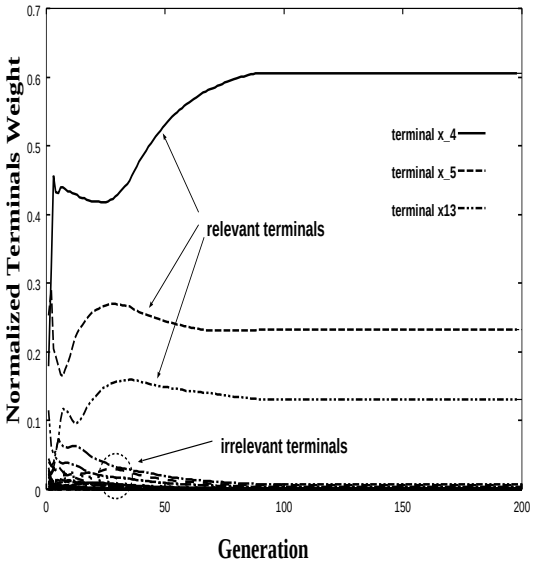

図 5 複雑な記号当てはめ式の問題に対する終端ノード重みの変化
Fig. 5 Change of terminal weights in harder regression problems.

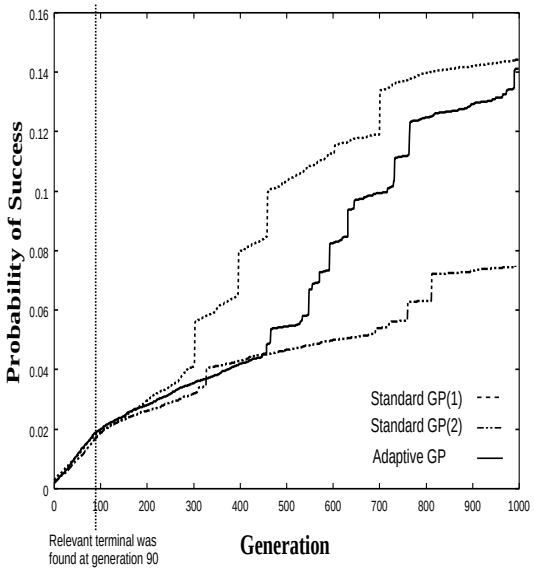

図 6 複雑な記号当てはめ式問題における成功率
Fig. 6 Success rate in harder regression problems.

性能に近づいていくことが分かる．この結果から，適応的 GP で用いられた適応的突然変異方法が多数の冗長なノードから複数の有用なノードを見つけるのに有効であることが確認された．

5. 議　　論

GP において，世代の進展にともなって集団内の個々の木のコードサイズが肥大化することにより探索空間が拡大し，その処理効率が悪化することが知られており，bloat 現象[9]などと呼ばれている．近年ではそうした事態を防ぐために，不要なノードあるいは部分木の除去，発生の抑制を図ることに関して，多くの研究事例が存在する（たとえば，除去バイアスの研究[10]など）．しかしながら，これらの研究で取り上げられている問題は，本論文の取り扱っているのとは別の冗長性に関する問題である．そこでは，個々のノードの特質としてではなく，他のノードや部分木との相対位置関係により，ノードの冗長性が決定される．そして，そのような関係性を維持したノード（部分木）が増殖することによる木の肥大現象を bloat と呼び，それを防ぐための交叉手法などの研究がさかんに行われている．すなわち，ノードの特質が問題解決に無関係であるか否かにかかわらず，他のノードや部分木との相対的位置関係により，そのノードが木の評価値に影響を与えない冗長な存在になりうることが bloat 現象の要因の 1 つである．本研究では，解決すべき問題と個々のノードとの関係を，他のノードや部分木との相対的位置関係とは独立に評価し，問題解決に本質的に関係しないノードを，GP の処理過程においてできるだけ早い段階で，遺伝的操作の対象から外すことにより，GP における探索効率の改善を図ることを意図しており，bloat に関する研究とは立場を異にする．しかしながら，本研究で得られた研究成果は，bloat に関する研究によって得られた効率改善手法と相反するものではなく，むしろ併用することにより，GP のさらなる効率改善が可能になるものと考えられる．

GP の効率化という点において他の関連深い研究としては，ADF（Automatic Defined Functions）[5]や MA（Module Acquisition）[11]などのような自動関数生成に関する研究がある．これらは有用な部分木を切り出して破壊的な交叉の影響を受けないように保護することにより，そのような有用な部分木が集団全体に広く使われることを促進する手法である．これは機械学習の研究においては，個々の特徴量を組み合わせて，概念獲得により有効な特徴量を新たに生成する特徴構築に近い技術と考えることができ，本研究の成果を組み合わせて適用することにより，より適切なモジュールの生成が可能になると考えられる．

従来の GP の研究，応用においては解決すべき問題と無関係なノードが多少木に含まれていても，それらは突然変異により自然に減少していくものであり，あまり深刻に対処する必要がないと考えられており，それが GP を利用する利点の 1 つであるとされていた．

さらに，そのような考えを支持する応用結果もいくつか発表されている[12]．それに対して，本研究は無関係なノードの存在がGPの効率を著しく損なうことがあり，そのような場合でも我々の提案する手法を適用することにより，GPの効率の効果的な改善が可能であることを実験的に示したもので，従来のGPの効率化に関する研究とは，問題意識や手法の点で大きく異なるものである．

また本来，GPにおける不要なノード，あるいは不要な部分木の定義は，本論文中の実験で扱われているように明確である場合は少なく，ある世代まではまったく使われていなかったノード，あるいは部分木が，その世代以降では重要な役割を果たすものとして使用される場合も多々あり，初めから“不要である”ことが自明の終端記号を除去することにどの程度の意味があるかも意見が分かれるところと考えられるが，本研究ではノードの重みの更新の式（$W_n(g)$）において，ノードの重みは木の適応度をベースに計算されているので，早い世代において木の適応度が低い段階でのノードの重みは大きく更新されず，ある程度世代を経て，重要なノードが揃い始め，木の適応度が高まった段階からノードの重みの更新も加速される．有効なノードを正しく選択することができるかどうかは，どの時点でノードの有効性を最終的に判断するかというタイミングに大きく関わってくると考えられるが，本研究では，大きな重みを持つカテゴリに属するノード集合と，適合度の最も高い上位1%のプログラム集団に含まれているノード集合が完全に一致したときに初めて，そのノード集合を有用なノードとして判定することにより，ノードの有効性の判断を保守的に行っている．それにより，上記のような場合でも本手法により正しく有効なノードの選択を行うことが可能であると考えられる．

6. まとめ

GPにおいて解探索性能を高めるためには，できるだけ探索空間を絞り込むために適切なノード集合を設計することが重要である．したがって，現実問題にGPを適用するうえでの大きな問題の1つは，ノードとして利用可能な冗長な情報の中から，問題解決に役立つ真に有効なノード集合を選定することである．

本論文では，冗長なノード，無用なノードが含まれているノード集合から，問題解決に有効なノード集合を決定するための新しい手法として適応的突然変異手法を提案した．本手法では，GPによる解探索過程において，オンラインで問題解決に有効なノード集合を自動的に獲得し，人間の専門家が持つ問題領域に関する知識を前提としない．本論文では，冗長なノード集合が与えられた記号当てはめ問題を用いて，提案した手法が世代の進展とともに適切な有効ノードを選択し，標準的なGPよりも解品質や効率の点で高い性能を示すことを確認した．また，冗長なノードとして終端ノードだけの場合についてのみ実験を行った．今後は関数ノードにも冗長なノードが含まれる場合に対しても本研究で提案した手法が有効であることを確認するとともに，本手法をより大規模な現実的問題[13]へと適用して，その効果を確認していく予定である．

謝辞 本論文の査読にあたり数々の適切なご指摘，ご助言を賜りました査読者の方々に感謝いたします．

参考文献

1) 伊庭斉志：遺伝的プログラミング，東京電気大学出版局 (1996).
2) Blum, A.L. and Langey, P.: Selection of Relevant Features and Examples in Machine Learning, *Artificial Intelligence*, Vol.97, pp.245–271 (1997).
3) Koller, D. and Sahami, M.: Toward optimal feature selection, *Proc. 13th International Conference on Machine Learning*, San Francisco, ICML, CA, pp.284–292, Morgan Kaufmann (1996).
4) Liu, H. and Setiono, R.: A probabilistic approach to feature selection—A filter solution, *13th International Conference on Machine Learning* (*ICML'96*), Bari, Italy, ICML, NJ, pp.319–327, Morgan Kaufmann (1996).
5) Koza, J.: *Genetic Programming II*: *Automatic Discovery of Reusable Programs*, The MIT Press, Cambridge, Massachusetts (1994).
6) John, G., Kohavi, R. and K, P.: Irrelevant features and the subset selection problem, *Machine Learning*: *Proc. 11th International Conference* (*ICML-94*), New Brunswick, ICML, NJ, pp.121–129, Morgan Kaufmann (1994).
7) Koza, J.: *Genetic Programming*: *On the Programming of Computers by Means of Natural Selection*, The MIT Press, Cambridge, Massachusetts (1992).
8) Michigan State University: *lil-gp 1. 0 User's Manual*, Cambridge, MA, USA (1995).
9) Blickle, T. and Thiele, L.: Genetic Programming and Redundancy, *Genetic Algorithms in Framework of Evolutionary Computation* (*Workshop at KI-94*), pp.33–38 (1994).
10) Soule, T. and Foster, J.A.: Removal Bias: A New Cause of Code Growth in Tree Based Evo-

lutionary Programming, *Proc. IEEE International Conf. on Evolutionary Computaion 98*, IEEE, pp.181–186 (1998).
11) Kinnear, K.E.J.: Alternatives in automatic function definition: A comparison of performance, *Advances in Genetic Programming*, Kenneth E. and Kinnear, J.(Eds.), pp.119–141, MIT Press (1994).
12) Gilbert, R., Goodacre, R., Shann, B., Taylor, J., Rowland, J. and Kell, D.: Genetic Programming-Based Variable Selection for High-Dimensional Data, *Genetic Programming 1998*: *Proc. 3rd Annual Conference*, pp.109–115, Morgan Kaufmann (1998).
13) Ok, S., Miyashita, K. and Hase, K.: Evolving Bipedal Locomotion with Genetic Programming—A Preliminary Report, *Proc. CEC-2001* (2001).

(平成 12 年 5 月 11 日受付)
(平成 13 年 4 月 6 日採録)

玉　秀列（学生会員）

1994 年韓国東亜大学産業工学科卒業．1998 年筑波大学大学院理工学研究科修士課程修了．現在，同大学院工学研究科博士課程在学中．人工生命，進化型計算等の研究に従事．

宮下　和雄（正会員）

1983 年東京大学工学部精密機械工学科卒業．1985 年同大学院工学系研究科修了．同年松下電器産業（株）入社．1990〜1992 年カーネギーメロン大学ロボティクス研究所客員研究員．1995 年通産省工業技術院電子技術総合研究所に入所．2001 年より独立行政法人産業技術総合研究所．現在，主任研究官．工学博士（大阪大学）．1999 年 1 月より筑波大学連携大学院助教授併任．分散協調問題解決，事例ベース学習，知的生産システム等に関する研究に従事．人工知能学会，AAAI，IEEE CS 各会員．

西原　清一（正会員）

1968 年京都大学工学部数理工学科卒業．同年，同大学大型計算機センター助手．1975 年より筑波大学電子・情報工学系．現在，同教授．工学博士．1982〜1983 年ヴァージニア工科大学，1998 年 IIASA．グラフィックスと CAD，組合せ探索アルゴリズム，知識処理，制約充足問題，複雑系の研究に従事．著書に「データ構造」（オーム社）等．1975 年情報処理学会論文賞．電子情報通信学会，人工知能学会，ACM，IEEE 各会員．