3E4-03

遺伝的プログラミングにおける進化計算の高速化について

Improvement for Evolution Speed in Genetic Programming

荒関 仁志[*1]
Hitoshi Araseki

[*1]日本大学大学院総合社会情報研究科
Graduate School of Social and Cultural Studies, Nihon University

Genetic Programming is used for problem solution of various fields. It is known that building blocks in individuals have played the role important for evolution speed. However, the question of how to emerge the building blocks is still open. The most common argument in favor of learning is that some aspects of the environment are unpredictable, so it is positively advantageous to leave some decisions to learning rather than specifying them genetically. In biological evolution, it has been shown that there is a close interaction between evolution and learning. It is called Baldwin effect.

In this paper, in order to improve evolution speed, Genetic Programming with learning process was proposed and an inheritance strategy for propagating the building blocks to offspring. The result of our experiment clearly shows that our approach is superior to standard Genetic Programming.

1. はじめに

遺伝的プログラミング (GP: Genetic Programming) は、直接プログラムを進化させることが可能なため、その進化能力の高さから様々な分野への応用が試みられ大きな成果が得られている。

GP における進化メカニズムは、Schema 理論によって説明されるが、それによると、進化メカニズムは、集団内に「有効な積木構造 (Building Blocks: 以下積木構造)」が生成された後、その積木構造が次世代に継承され、それらを組合わせることで、より適合度の高い個体を生成することができると考えられている (積木仮説)。

しかし、通常の GP(Koza's GP[1]、以下 SGP:Standard Genetic Programming) では、評価値空間での選択淘汰を行うため、Schema 理論が指摘するような積木構造を直接的に取り扱うことはできないため、効率的な解探索を行っているとは言い難い。したがって、Schema 理論が指摘する積木構造を集団の中から効率的に取り出すことが可能ならば、進化能力の改善が期待できる。

一方、生物学的進化においては、進化過程だけではなく学習過程も重要な役割を担っているとの指摘がある。これは Baldwin 効果として知られ、近年盛んに研究が行われている [9, 10, 11]。Baldwin 効果とは、進化と学習の相互関係を、学習のメリットとコストのバランスで説明するものである。進化におけるこの学習効果は、解空間での効率的な探索を支援することができると期待できる。

ここでは、GP の進化過程における「効率的な積木構造の探索」を目的として、GP に学習過程を導入した進化モデル (LGP: Learning Genetic Programming) を提案する。この LGP では、従来の進化演算 (SGP) によって生成された個体構造内に、進化に有効な積木構造が存在するかどうかを学習過程により検査する機能を付加することで、効率的な積木構造の探索・保存・継承を実現するものである。

実験の結果、SGP に比べ、進化能力に関して明らかな改善が見られた。さらにこの学習過程は、積木構造の集団への伝搬にも大きな効果があることも分かった。

A: 連絡先：荒関 仁志，日本大学大学院総合社会情報研究科，〒359-0003 埼玉県所沢市中富南 4-25，Tel: 042-996-4177，Fax: 042-996-4163，ara@gssc.nihon-u.ac.jp

2. GP と Schema 理論

GP における Schema 理論 [12, 13] では、進化のメカニズムとして、集団内に進化に必要な複数の積木構造が生成され、それらが進化演算 (交叉や突然変異など) によって破壊されることなく次世代へ継承され、さらに、それら複数の積木構造が組合わさることで、より高い適合度を持つ個体が生成されると考えられている。しかし、実際の SGP の進化計算では、例え個体構造内に有効な積木構造が存在していても、評価値空間で個体を選択淘汰する限り、それを直接探索することは出来ない。したがって、進化計算を効率的に行うには、この積木構造をどのように集団内から探索し、継承するかが重要な問題となる。

現在までに、積木構造を直接的に生成･探索する試みが幾つかなされている。その 1 つは、Koza による ADFs(Automatically Defined Functions)[2] である。これは SGP の進化過程とは別に、積木構造 (関数) を生成する過程を用意し、その積木構造を各個体が利用することで進化能力を改善しようとする試みである。実験の結果、進化速度に大きな効果があることが分かった。しかし、この ADFs を生成するためには、予め積木構造の形や種類を決めておかねばならず、ADFs を利用する場合には、与えられた問題に関するある程度の知識を必要とするため汎用的な取り扱いとは言い難い。

この ADFs の考え方を、より汎用的な取り扱いにするために、Angeline と Pollack により GliB(Geneic Library Builder) が提案されている [3]。GLiB は、個体内から部分木を適当に取り出し、その構造を積木構造と仮定してライブラリに登録する。各個体は進化の過程で、このライブラリ内の積木構造を利用することで進化能力の向上を目指したものである。しかし、実験の結果、適当に取り出された部分木には、有効な積木構造は少なく、進化能力に関しては、SGP に比べて優位な差を見つけることはできなかった。

同様な取り組みとして、Araseki と Inoi による GOLiB (Genetic Object Library Builder) がある [4]。GOLiB では、評価値が高い個体 (エリート個体) 全体 (葉ノードを除く) を積木構造としてライブラリに登録し、各個体がこの積木構造を利用することで進化能力の改善を計るものである。実験の結果、

SGP に比べて最適解を探索する能力は改善されたが、個体サイズが急激に増加することが分かった。このため進化計算の途中で、計算機資源 (CPU やメモリ) を急激に消費することになり、結果的に計算速度の改善は小さいものとなっている。

この個体サイズの増大は bloat 現象と呼ばれ、個体内に評価とは無関係な構造 (intron) が増えることが原因とされている。この bloat 現象も進化速度を阻害する大きな要因の一つとして、その抑制方法の研究が盛んに行われている [5, 7, 8]。

上記３つの研究から、積木構造と進化能力に関して以下ことが理解できる。

1. 評価値の高い個体中には、積木構造が存在する可能性が高い。
2. 進化能力を改善するためには、積木構造が進化演算から破壊されないようにするメカニズムが必要となる。
3. bloat 現象を避けるためには、個体内の intron の生成を抑制する必要がある。

3. Baldwin 効果

生物学的進化においては、ラマルク的な獲得形質の遺伝的な仕組みがなくても、集団における個体の学習が集団全体の進化に方向性を与え、進化のスピードを促進することが可能であると考えられている。これは Baldwin 効果として知られ、近年盛んに進化システムへの応用研究が行われている [9, 10, 11]。Baldwin 効果では、進化と学習の相互関係を、学習のメリットとコストのバランスで説明することで、解の効率的な探索が可能となると考えられている。

Baldwin 効果では、一般的には以下の２つの過程を経て、学習により獲得された有効な形質 (積木構造) が、生得的な性質へと進化して行くと考えられている。

第 1 段階 学習過程により、有効な構造 (以下積木構造) を獲得した個体 (学習個体) が、多く子孫を残す。

第 2 段階 学習に掛かるコストにより、積木構造を生得的に獲得している個体 (非学習個体) が多くの子孫を残す。

SGP では、進化演算の結果生成される個体 (子孫) には、複数の部分木が内在しており、各部分木の評価値の総和として個体の評価値が決定される。したがって、例え積木構造が「ある部分木」として存在していても、それが直接個体の評価値として発現することは少ない。SGP の進化過程では、積木構造の生成と共に、積木構造の組合せと並びの順序を同時に満足する必要があるため、複雑な問題を解く場合には、解の探索空間 (評価値空間) が広大となり、最適解を見つけるまでには多くの計算量が必要となる。

したがって、集団内から積木構造を効率的に探索することが可能で、かつその構造を破壊せずに次世代へ継承することができれば、より少ない計算量で最適解を見つけ出すことが可能になると考えられる。そこで広大な解の探索空間でも効率良く解を見つけ出す方法として、学習過程を利用することを考える。

4. 学習過程を備えた遺伝的プログラミング

ここでは、SGP に学習過程を付加した進化モデル (LGP: Learning Genetic Programming) を提案する。ここで学習過程とは、進化演算処理によって個体構造が決定された後に、個体構造内に積木構造が存在するかどうかを探索するメカニズムのことを言う。

LGP には、SGP の進化過程 (交叉、突然変異、選択) に加え、以下の学習過程等が備わっている。

1. 学習過程では、個体構造の全ての部分木を評価し、最も評価値が高い部分木をその個体の積木構造とする。積木構造以外の部分は評価対象外とする (intorn、図 1 を参照のこと)。
2. 学習過程を持った親からは、学習過程を持つ可能性のある子供 (個体) が生成される。ただし、子供が学習個体になるか、非学習個体になるかは、その評価値によって決定される。すなわち、積木構造が発見された場合にはその個体は学習個体となり、発見されなかった場合には非学習個体となる。
3. 学習過程で決定された積木構造は、交叉や突然変異などの進化演算からは破壊されない。この積木構造が進化演算 (特に交叉) により、非学習個体に継承されても、学習個体と同様に進化演算によって破壊されない。
4. 学習過程では、個体内にある部分木長に比例したコストが評価値に加算される。
5. 交叉演算における親からの部分木切り出し処理を行う場合、親の個体内に積木構造が存在する時には、その積木構造のみを切り出し、子供に継承する (intron 構造の削除処理)。

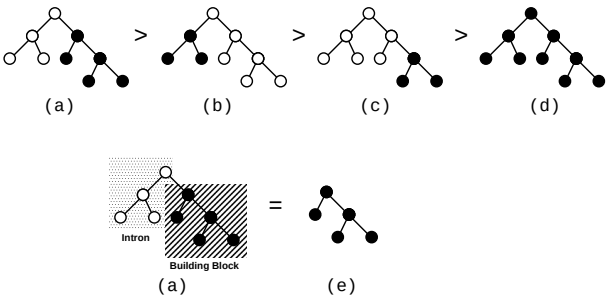

図 1: 積木構造と intron 構造

図 1 には、個体内の各部分木の評価値が $(a) > (b) > (c) > (d)$ の順でランク付けされていることを示している。ただし、(d) は部分木ではなく個体 (木) である。学習過程の結果、個体の評価値として、評価が最も高い (a) が採用される (●が評価対象構造、○が評価対象外構造)。この個体 (d) は、(e) の部分木構造と評価値空間では等価となることを示している。

LGP での解の探索は、以下のような２段階の探索を行う。

進化過程: 交叉や突然変異などの進化演算により、解探索空間での大局的な探索が行われる。

学習過程: 個体の内部構造での探索 (個体内探索) を行うため、局所的な解探索が行われる。

これらの２段階の解探索を組み合わせることで、効率の良い解探索が期待できる。

計算量に関しては、SGP での個体の評価値計算 f は、予め各部分木 $(f_1, f_2, ..., f_m)$ の評価値を計算し、その総和として行われる。

$$f = \{f_1 \odot f_2 \odot ... \odot f_m\}$$

ただし、ここで $\odot = \{+, -, \times, \div\}$ はバイナリ演算子となる。

同様に LGP での個体評価計算も各部分木 $(f_1, f_2, ..., f_n)$ を評価し、その中で評価値が最適なものを選択するので、計算量は SGP と同じオーダ $O(n)$ となる。ここで n は個体内のノード数である。

$$f = \max\{f_1, f_2, ..., f_m\}$$

5. 進化実験

5.1 実験パラメータ

ここでは、LGP を使ってパリティ問題 (Even-4-Parity) を学習する実験を行った。実験では、GPL と SGP 共に表 1 に示すパラメータを使っている。

表 1: 実験に使用したパラメータ値

パラメータ名	パラメータ値
集団数	4,000
突然変異率	10%
選択則	MGG(Minimal Generation Gap)
初期学習個体の割合	50%

選択則に採用した MGG は、進化演算において多様性を維持する能力が高く、進化計算の高速化が期待できることが知られている [14] 。ここでの MGG では、集団の中から適当に 2 個体を親として取り出し、交叉と突然変異を行い、子供 1 個体を生成する。親個体と子供個体を含めた 3 個体の間で評価値を計算し、評価値の高い 2 個体を集団に戻す処理を行う。

実験に際しては、各々60 回の実験を行い、その平均を実験結果とした。実験結果に表示される SGP は選択方法に MGG を採用した SGP であり、LGP が学習過程を持った GP となる。

評価関数は、Adjusted Fitness[1] を使用した。実際の評価関数 (f) は以下のような関数となる。

$$f = \frac{1}{1 + \Sigma_i \mid r_{i0} - r_i \mid c(n)}$$

ここで、r_{i0} は教師信号となる。学習コストの計算式は、

$$c(n) = e^{n^2/\sigma^2}$$

とする。ここで、n は部分木のノード数である。実験では、σ^2 は定数として取り扱う。

5.2 Even-4-Parity 問題

LGP を使って、Even-4-Parity 問題を学習させる実験を行った。ここでは、表 2 にある演算子を SGL および LGP で使用した。

表 2: 実験に使用した演算子

演算子名	値
終端記号	0, 1, d0, d1, d2, d3
演算子	and, or, nand, nor
σ^2	50,000

図 2 には、正解を学習した回数を正解率としてを示す。ここでは、学習過程を持たない SGP と、学習過程を持った GP(LGP with cost)、さらに比較として学習コストなし LGP(LGP without cost) を表示している。また、学習過程を持った GP(LGP with cost) の学習個体と非学習個体の割合を Learning rate として表示する。横軸は評価回数を示す。実験の結果、SGP で

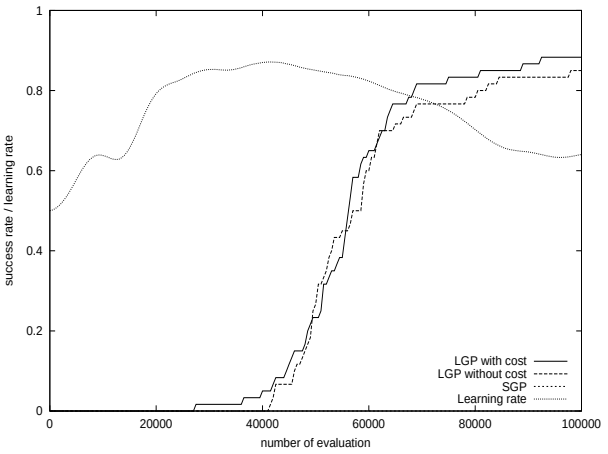

図 2: Even-4-Parity 問題

は 60 回の実験で正解を得ることは出来なかった。したがって、LGP(LGP with cost と LGP without cost) は、SGP に比べて顕著な進化速度 (能力) の改善があると考えられる。また、学習個体と非学習個体との比 (Learning rate) は、Baldwin 効果で示されているように、進化の初期にその割合が大きくなり進化速度を加速し、積木構造の探索に貢献していることが分かる。したがって、進化速度の改善に大きな役割を果たしていると考えられる。ただし、学習過程はそのコストの有無に関わらず、この学習コストパラメータでは進化速度が同じであることがわかる。

ちなみに、Koza の ADFs の計算 [1] では、90%の正解を得るためには約 144,000 回の評価計算を必要としている。

図 3 には、個体サイズの変化を示す。図 2 の進化能力には差がなかった、コスト無し LGP(LGP without cost) と、コストあり LGP(LGP with cost) であるが、個体サイズ比較では顕著な差が見られる。図 3 から、学習コストの効果は、個体サイズの抑制効果を持っていることが分かる。

6. 実験の結果と考察

SGP に学習過程を付加することで、積木構造を効率良く探索する能力を獲得することがことが分かった。

また、学習コストパラメータ (σ^2) は進化速度を直接改善することは少ないが、個体サイズ (bloat 化) の抑制に強く関係していることが分かった。したがって、今後の研究によって適切な学習コストパラメータを設定することができれば、進化能力を損なわずに個体サイズをさらに抑制することが期待できる。

図 2 から、学習個体の増加は、進化の初期段階で積木構造を新たに探索する必要がある場合に起こっている。一方、非学習個体が増加する部分では、学習過程で探索された積木構造が、個体に頻繁に利用されていると考えられる。ただし、進化の初期に有効な積木構造を探索で出来なかった場合 (正解を得られなかった約 10%の実験) には、常に学習個体の割合が高くなる

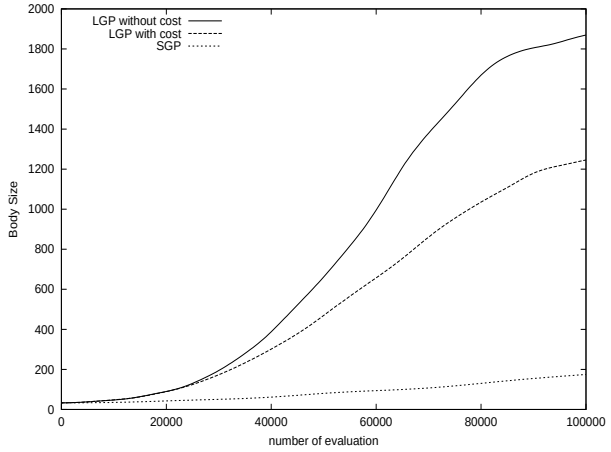

図 3: 個体サイズの変化

傾向が見られ、進化の最後まで、進化に有効な積木構造を探索していることがわかる。ただし、進化の後半では、個体サイズが大くなるため、学習コストが大きくなり、有効な積木構造を効率良く探索できないようである。また、学習コストパラメータは、解決する問題の難しさに関係しており、学習コストパラメータ (σ^2) を極端に小さくすると正解を探索することができない。

このことから、さらに進化速度を改善するためには、学習コストパラメータも進化過程で常に一定な値を取るのではなく、進化の初期には、小さな値を取ることで、小さな積木構造を効率良く探索することを考え、進化速度が鈍った場合には、大きな値を取ることで個体サイズを一時的に増加させ、新たな積木構造を個体内から探索する機会を増やすなどの取り扱いが必要であると考える。

さらに学習過程は、集団内に積木構造が探索されると、その積木構造を急速に集団内に伝搬させる能力がある。すなわち、高評価値を持つ積木構造が個体内に存在するならば、個体内のどこに位置しようとも必ず高評価値を個体評価値として返すために効率的な個体評価が可能となっている。また、この積木構造は進化演算により破壊されないため、次世代に確実に継承されることで、高評価を持つ積木構造が集団内へ急速に拡散することが可能となっている。

7. まとめ

遺伝的プログラミングに学習効果を付加した進化システム (LGP) を提案した。LGP によって、GP の進化計算においても、学習過程が重要な役割を果たすことが分かった。すなわち、従来の GP の進化計算では大局的な探索が主であるために、積木構造の効率的な探索が難しかったが、学習過程を持つことで局所探索 (個体内探索) が可能となり、進化能力の改善を計ることが可能となった。また、この学習過程の計算量は、従来の GP の評価値計算と同等 ($O(n)$) であるため、様々な分野への応用も期待できる。

LGP の学習コストパラメータ (σ^2) は、積木構造の探索能力と個体サイズの抑制に関係することが分かった。しかし、今回の実験では、近年の bloat の抑制研究 [5, 7, 8] に比較すると、遥かに大きなサイズの個体を生成しており、計算機資源が無駄に消費されたことは否定できない。今後は、この学習パラメータと個体サイズの関係を詳細に調査する必要がある。

参考文献

[1] J. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection.*, MIT Press, 1992

[2] J. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs*, MIT Press, 1994

[3] P. J. Angeline and J. B. Pollack, *Competitive Environments Evolve Better Solutions for Complex Tasks*, Proceeding of the Fifth International Conference(GA93), Margan Kaufmann Publisher Inc., 1993

[4] H. Araseki and K. Inoi, *Protect the Effective Object from Destructive Crossover in Genetic Programming*, Intelligent Autonomous Systems, IOS Press, pp.735-742(1998)

[5] D. Andre and A. Taller, *A Study in Program Response and the Negative Effects of Introns in Genetic Programming*, GP'96, 1994

[6] P. Nordin, F. Francone and W. Banzhaf, *Explicitly Defined Intron and Destructive Crossover in Genetic Programming*, Advances in Genetic Programming II, MIT press, 1996

[7] R. Poli and W. B. Langdon, *On the Search Properties of Different Crossover Operators in Genetic Programming*, Appeard in the Proceeding of Genetic Programming '98, pp.293-313(1998)

[8] W. B. Langdon, *Size Fair and Homologous Tree Genetic Programming Crossovers*, Proceeding of Genetic and Evolutionary Computation Conference (GECCO99), pp.1092-1097(1999)

[9] G. E. Hinton and S. J. Nowlan, *How learning Can Guide Evolution*, Complex Systems, Vol.1, pp.495-502(1987)

[10] 鈴木 麗璽, 有田 隆也, **進化と学習の相互作用—繰り返し囚人のジレンマゲームにおける *Baldwin* 効果—**, 人工知能学会, Vol.15, No.3 pp.495-502(2000)

[11] Anna Esparica-Alcazar and Ken Sharman, *Phenotype Plasticity in Genetic Programming: A Comparision of Darwinian and Lamarckian Inheritance Schemes*, Second European Workshop, EuroGP'99, pp49-64(1999)

[12] 伊庭 斉志, **遺伝的プログラミング入門**, 東京大学出版 (2001)

[13] W. B. Langdon and R. Poli, *Foundations of Genetic Programming*, Springer(2002)

[14] 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, Vol.12, No.5, pp.734-744(1997)