7G－4

遺伝的プログラミングによるロボットの行動制御プログラムの生成

高橋 健介　　藤本好司

龍谷大学理工学部数理情報学科

1　はじめに

従来、GP(あるいはGA) を応用した実験では、解きたい問題のための問題と解のサンプル (環境) をいくつか用意し、それら全ての環境をGPが解いた時点で問題解決を成し遂げたということが多かった。このこと自体は方法として自然かつ合理的である。

だが、GPは(GAと比較して)より柔軟な自動プログラミングとしての性格が強い。GPはただ問題を解くために使うより、便利な自動プログラミングのためのツールと見なした方が将来性に溢れている。GPでより汎用的なプログラムが生成できれば、GPは自動プログラミングのツールとして大きな意味を持つことになるだろう。

本研究では、それが育った環境条件にとらわれること無くプログラムを動作させるためにはどのような進化を行なえば良いかを考える。このことは進化論から言えば矛盾している。自然界における生物は周りの環境により適応しようとして生体組織を変化させる。これが一般的に進化と呼ばれている。つまり、進化とは周囲の環境が持つ特徴に合わせていくということである。従って、進化したプログラムは異なる環境でも進化した環境と共通した特徴を持っていれば適応できるということが期待される。

2　問題設定

本研究では、ある平面上に設計した迷路にロボットを設置し、ロボットを目的地まで向わせることを目的とした行動プログラムをGPに生成させる。

このような問題設定の報告はいくつかある。ここでは、プログラムにいくつかの環境を与えて進化させ、完全に問題を解ける状態になってからそのプログラムがどの程度の適応を他の環境に向けて示すかを検証する。もしそれが可能であれば適応のメカニズムを解明して他の環境でも動作するようなプログラムを意図的に生成することを試みる。

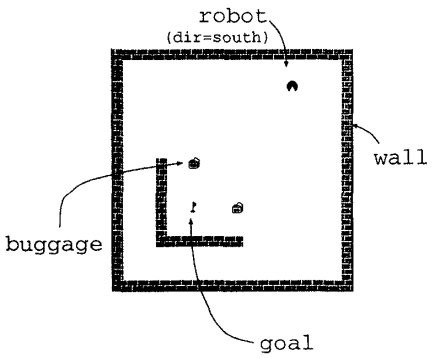

図 1: 環境の例

環境 (迷路) の仕様とロボットの仕様を述べる。ロボットは迷路のどこか1箇所に配置され、前進や左右転回を行なうことで目的地を目指す。こうしたアクションにはエネルギーが必要で、ロボットはエネルギーの保つ限り行動を行なうことが出来る。迷路の大きさは標準で20×20で、この格子1つ1つに障害物などが設置されている。障害物には動かせる物 (荷物:buggage) や動かせない物 (壁:wall) があり、パズル性が加えられている。また、ロボットにはセンサーがあり、前後左右4方向の至近の障害物までの距離と、格子1つ分前方の障害物の種類が認識できる。さらに目的地の具体的な座標とロボットの現在座標をいつでも知ることが出来る。

一見情報を与えすぎの印象を受けるが、未知の環境でプログラムを正常に動作させるには、必ず目的地の情報と現在位置の関係の情報が必要である。なぜなら、目的地を知らされないままの進化は、GPは用意された環境における目的地までの経路をプログラムという形を借りて生成するからである。また、自分の現在位置が分からないということは、現在位置と目的地との関係が分からないということであり、より汎用的なプログラムを生成する為にはこのような情報が必要である。

上記の仕様で環境を構築した例を図1に示す。

Generations of Robust Programs for Robot control with GP

Kensuke Takahashi, Fujimoto Yoshiji
Ryukoku University
1-5 Yokoya, Seta Ooe, Ohtsu, Shiga 520-21,Japan

reproduct	crossover func/any	mutation (copymiss)
10%	15%/50%	25% (0.3%)

表 1: 遺伝的操作のパラメーター

環境の数	1 個	2 個	3 個	4 個	5 個
成功率	100%	80%	90%	50%	20%
平均世代数	21.4	52.1	60.2	96.4	147.0

表 2: 実験結果

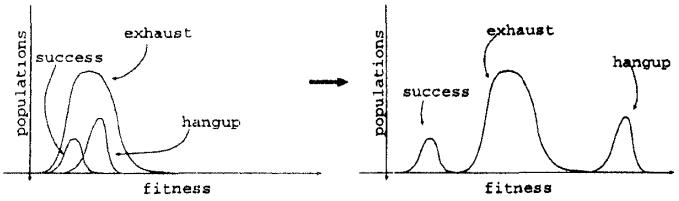

図 2: 適応度の修正

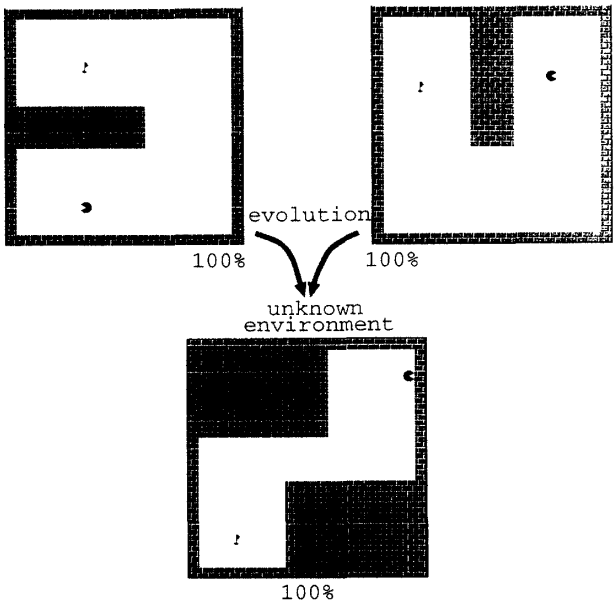

図 3: 結果例

3 進化方法

3.1 遺伝的操作

遺伝的操作には複製、交叉 (一点交叉)、突然変異がある。そして突然変異の 1 操作として遺伝子ノードのコピーミスが含まれる。これらの確率は表 1のように設定した。

3.2 集団間移民

本実験では個体集団を複数用意し、それぞれを独立的に進化させる。そしてある一定の世代間隔が経過したところでそれぞれの個体集団間で移民を行ない、それぞれの優れた遺伝子の交配を行なう。このような進化方法が GP には大変有効であることが実験中に明らかになった。

3.3 ペナルティー的進化

この問題を解くプログラムは、主に目的地までの到達率、エネルギーの消費率の 2 つの要素で評価されるが、プログラムの出来によって大きく 3 つに大別される。1 つは問題を解いたプログラム (success)、目的地にたどり着くまえにエネルギーが尽きたプログラム (exhausted)、そしてハングアップしたプログラム (hangup) である。プログラムの評価値には success でボーナスが、hungup でペナルティが与えられ、それぞれの個体の評価値に大きな差が開く (図 2)。これにより 1 つの集団内に 3 つの仮想的な集団が形成されることになり、良い個体はさらに良くなる機会を与えられ、悪い個体には淘汰の機会が与えられる。

4 結果

実験では進化に利用する環境の数を 1 個から 5 個へと段階的に増やしていき、それぞれの場合での適応の度合いをみた。その結果、表 2のような数値が得られた。

進化に使う環境を増やして行くに連れて面白い結果が得られた。個体はこれらの環境の共通した特徴を巧みに抜きだし、用意された環境全てを解くことに成功した。さらに、進化に使った環境と良く似た環境でも高い適応を示した。例えば左周りの行動を取ることによってゴールできる環境と右周りによってゴールできる環境を用意して個体を進化させると、図 3に示すような左周りと右周りを複合した迷路の新しい環境で 100% 解くことが出来たのである。

これによって言えることは、個体の性質は進化に使った環境に左右され、環境からより多くの特徴を抽出することが出来た個体ほど頑健性が増すと言うことである。環境を上手く設定することによって汎用性の有るプログラムも生成できると考えられる。

本研究は文部省科学研究費基礎研究 C(08680417) の補助を得て行なわれた。

参考文献

[1] John Koza , Genetic Programming , 1992

[2] 伊庭斉志, 遺伝的アルゴリズムの基礎, オーム社, 1994

[3] Advances in Genetic Programming , The MIT Press