修士論文要旨

遺伝的プログラミングを用いた強化学習における階層構造の自律生成

（知能情報システム学）名田　茂洋

１.緒言

今日、ロボット工学分野での進歩は目覚ましいものがある。そのため、今までのように一定の環境で動作するだけでなく、様々な環境の変化に対応しながら動作するロボットが求められている。その必要性に答えるためロボットが環境の変化を認識し、その変化に対応しながら学習するための手法について研究が盛んに行われている。本研究はこれまでに行われてきた研究が用いている手法よりも環境の変化に適応した学習ができないかということを考え行った。これまでの研究では、遺伝的プログラミングを用いて強化学習の階層構造を自律的に生成することが行われている。しかし、この研究では、遺伝的プログラミングの遺伝的操作として交叉という方法しか用いられていないため、よく似た環境に移ったときに必要となる遺伝子が淘汰されている場合があり、その場合には初期遺伝子を新たにハンドコーディングする必要があった。本研究では自律的に階層構造を生成できる遺伝的プログラミングと報酬を得て学習を進めていく強化学習の組み合わせという枠組みにおいて、交叉だけでなく突然変異を用いることで環境の変化に応じた学習が必要となったときに初期状態から学習するのでは無く、それまでに学習した結果を活かしていける手法について研究を行った。ここでは、その学習の枠組み、実験環境、考察について述べる。

２.処理概要

2.1 強化学習

本研究で用いた強化学習はQ学習と呼ばれる学習法である。この学習では状態価値関数Qを報酬ｒにより更新することで、どの状態行動対に価値があり、どの状態行動対に価値が無いのかを学習する手法である。状態価値関数の更新式を式(1)に示す。

$$Q(s_t, a) \leftarrow Q(s_t, a) + \alpha [r + \gamma \max_{a} Q(s_{t+1}, a)] \qquad (1)$$

式(1)は、ある状態s_tにおいて行動ａを行い、その後最適と思われる政策に従って行動した場合の報酬期待値を示している。本研究では、Q学習を階層化し、１つのサブタスクでの状態数を少なくすることで学習を速くし、状態数が大きくなっても対応できるようにする。階層化によって状態数を減らすことが出来る理由として、サブタスクを生成することが挙げられる。サブタスクとは１つの仕事を分割した幾つかの部分的な仕事であり、サブタスクを達成するまでのロボットが見る状態数を減らしたり、サブタスクを達成した時に報酬を与えることでサブタスクが担当する部分の仕事の学習速度を速めたりすることができる。

2.2 遺伝的プログラミング

強化学習における学習の階層構造を決定する遺伝的プログラミングの部分について説明

する。遺伝的プログラミングでは、まず初期集団として幾つかの木構造をランダムで発生させ、そして、各木構造がどれだけ上手に仕事をこなす事が出来るかを表す適合度を計算する。そして集団の中で、適合度が高い木構造を残し、その他の木構造に対して突然変異、交叉といった処理を行って新たな木構造を作り、適合度を計算する。この処理を予め決めた世代数だけ繰り返し、適合度が高い木構造を探索する。遺伝的プログラミングで行う突然変異という処理は、図 1 のように木構造の中のノード y を決められた一定確率で別のノード x に変えて別の木構造を生み出す処理のことである。

交叉とは、ある 2 つの木構造の部分木を別の木構造の中の部分木と入れ替える処理である。図 2 では x を交叉点とする部分木と終端ノード y を交叉した例について示している。この例のように交叉では、異なる 2 つの木構造を用いて、元の木構造とは異なる 2 つの木構造を生み出すことができる。本研究では、交叉だけでなく突然変異という処理を用いることで、それまで学習していた環境では必要なく淘汰されてしまったノードであっても環境の変化によって必要となった場合には取り出せるようになっている。これによって人間が予め環境の変化を予想し、それに応じたノードを用意しておけば、環境の変化に応じて強化学習ができるようになる。また、淘汰されてしまう場合においても淘汰されてしまうまでは学習をおこなっているので別の環境で必要となった場合にも途中がから学習を行うことが可能となる。

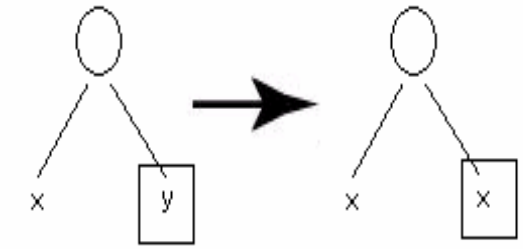

図 1 突然変異

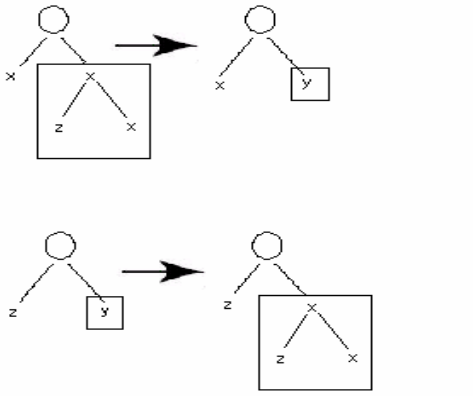

図 2 交叉

[処理概要]

本研究の処理手順は次のような手順で行っている。

(1) 初期集団として 10 個体木構造を発生させる。この時の木構造は全て深さが 4 である。

(2) (1)で発生させた木の各ノードは階層型強化学習のサブタスクに相当するためロボットは遺伝的プログラミングで作られた木構造に従い行動する。木構造のノードとなるサブタスクは非終端ノードとして Root、Capture、Deliver、Find_Battery、Visible_Battery、Visible_Nest、Find_Nest の 7 種類を用意し、終端ノードとして Avoid、Wander、Approach_Battery、Approach_Nest、Turn の 5 種類を用意した。また終端ノードは、MoveUp、MoveDown、MoveRight、MoveLeft の 4 種類の行動が選択できるようになっている。

(3) (2)で行った行動により起こった状態変化に応じて報酬と罰を与える。受け取った報酬と罰から式(1)に従って Q 値の更新を行う。

(4) (3)で各ノードが受け取った報酬の合計を適合度とし、適合度が一番多いものを優秀な木構造として残し、他の木構造に対して突然変異および交叉処理を行う。

(5) (2)から(4)の処理を最大世代数行い最も適合度の高い木構造を学習していた環境下での最優秀個体として選び出す。

(6) 環境を変化させ、(5)までの処理で淘汰されてしまった遺伝子がある場合においても初期個体をハンドコーディングすることなく環境の変化に適応して学習できることを確かめる。

3.実験

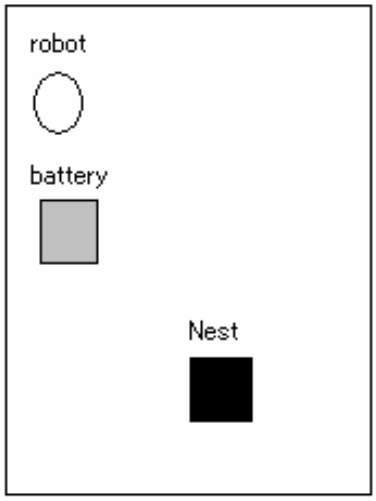

図3実験環境

本研究では、学習は全てシミュレータで行っている。図 3 で示すように本研究では二輪ロボットが、バッテリーを巣に持っていくという仕事をシミュレーションしている。このフィールドは壁に囲まれておりロボットが外にはみ出ることはない。また、ロボットは視覚センサを有し、そのセンサはロボットの中心から前方に-30°から+30°までの範囲を見ることが出来る。視覚に捕らえている対象との距離情報は正確にはわからないが対象との距離が近いか遠いかという情報がわかるようになっている。対象が見える方向は前方、前方左、前方右の 3 種類が識別できるようになっている。またフィールド内には壁を置くことができ、ロボットの視覚は壁により遮られるものとする。ロボットが行える行動は 4 種類で前進、後退、右折、左折である。また、ロボットはバッテリーのある座標に来た時点でバッテリーを取ったものとしている。罰を与える条件は以下のようにサブタスクごとに与えている。

[非終端ノード]

Capture…バッテリーを捕えることができれば+10 の報酬

Deliver…バッテリーを巣に持ち帰ることができれば+10 の報酬

Find_Battery…バッテリーを見つければ+10 の報酬

Visible_Battery…バッテリーを持ち帰れれば+10 の報酬

バッテリーを見失うと-10 の罰

Find_Nest…バッテリーを持っている状態で巣を見つければ+10 の報酬

Visible_Nest…バッテリーを持っている状態で巣に帰れば+10 の報酬

バッテリーを持っている状態で巣を見失うと-10 の罰

[終端ノード]

Avoid…壁が見えている状態で壁を視界から外せば+10 の報酬

壁が見えていない状態で選択されると 1 つ上位のノードに-10 の罰

Wander…バッテリーを持っていないときにバッテリーを視界に入れたら+10 の報酬
バッテリーが見えている状態で選択されれば 1 つ上位のノードに-10 の罰
バッテリーが持っていて巣が見えている状態で選択されると 1 つ上位のノードに-10 の罰

Turn…壁が見える状態で見えなくすれば+10 の報酬
バッテリーが見えている状態で選択されると 1 つ上位のノードに-10 の罰
バッテリーを持っていて巣が見えている状態で選択されると 1 つ上位のノードに-10 の罰

Approach_Battery…バッテリーが見える状態からバッテリーに着けば+10 の報酬
バッテリーを見失うと-10 の罰
バッテリーが見えない状態で選択されると 1 つ上位のノードに-10

Approach_Nest…バッテリーを持っていて且つ巣が見えている状態で巣に着けば+10
上記以外の状態で選択されれば 1 つ上位のノードに-10 の罰
バッテリーを持っている状態で巣を見失ったら-10 の罰

4. 結果

結果については､遺伝的プログラミングの処理による木構造の変化が比較的小さいものは学習が収束するに到った。

5. 考察

本研究で行った手法では、環境の変化に適応するために学習速度を犠牲にしてしまっていることがわかった。これは、環境の変化に対応するために遺伝的プログラミングに木構造の組み換えを盛んに行うと強化学習のQテーブルが安定した環境で学習が行ない難いためだと考えられる。環境の変化に適応しやすくすることを考えて、遺伝的プログラミングによる各ノードの組み換えを活発に行うと各ノードが持つQ値の収束が困難となる。逆に各ノードの組み換えをあまり行わないと仕事を達成できるかどうかは遺伝的プログラミングの部分で最初に作る初期集団に大きく依存してしまうためハンドコーディングとあまり変わらない結果となってしまう。今後の課題としては、遺伝的プログラミング部における階層構造の組み換えの度合いと強化学習部の学習が収束する速度の関係を調べていく必要がある。