進化学習型遺伝的ネットワークプログラミングと そのタイルワールド問題への適用

間 普 真 吾*・平 澤 宏太郎*・古 月 敬 之*

Genetic Network Programming with Evolution and Learning
and Its Application to the Tileworld Problem

Shingo MABU*, Kotaro HIRASAWA* and Jinglu HU*

A new evolutionary algorithm named "Genetic Network Programming, GNP" has been proposed. GNP represents its solutions as directed graphs, which realizes better expression ability than GA and GP which use string and tree structures, respectively. The aim of developing GNP is to deal with dynamic environments efficiently by using the distinguished expression ability and the inherently equipped functions of the network structure. However, since GNP is based on evolution, the programs cannot be changed until one generation ends. In this paper, we propose the extended algorithm, "GNP with Evolution and Learning" which combines evolution and reinforcement learning in order to adapt to dynamic environments quickly. The tileworld is used as a simulation environment and the results show some advantages of the proposed method.

Key Words: Genetic Algorithm, Genetic Programming, Genetic Network Programming, Q-learning, tileworld

1. はじめに

これまで、進化論的計算手法に関する研究が数多く行なわれて来た。代表的な手法に遺伝的アルゴリズム (GA)¹⁾、遺伝的プログラミング (GP)^{2),3)} がある。GA は解を文字列で表現し、主に最適化問題への応用がなされてきた。GP は解を木構造で表現することにより、解の表現能力が向上し、より複雑な問題への適用が可能となり、進化論的計算手法の進展に大きく寄与している。解の表現方法はシステムの性能に大きな影響を持つため重要な問題であるといえる。

最近、筆者らは GA, GP などの計算手法を拡張した新しい手法 "遺伝的ネットワークプログラミング (Genetic Network Programming, GNP)" を提案している⁴⁾。GNP では有向グラフで解を表現することにより、他手法と異なる機能を本来的に備えることができ、より効率的な解の探索が可能である。

従来の進化論的計算手法は、進化のアルゴリズムに基づいている。すなわち適合度を計算するためにある期間の試行を行ない、その結果に基づいて選択、交叉、突然変異といった

遺伝的操作を施し、これを 1 世代として繰り返していくものである。GNP は以上のような一般的な進化のアルゴリズムに基づいている。これに対し、タスク試行中の情報を効果的に用い、環境変化に迅速に対応できるシステムを構築するため、強化学習の分野で代表的な Q 学習⁵⁾ を適用した GNP のオンライン学習^{6)~8)} も提案されている。進化型 GNP は、たくさんの個体を用いてタスクを実行し、プログラムの変更をタスク終了後に行なうが、学習型 GNP は一つの個体を用い、タスク実行中に変更を行なうため、これらの 2 つの手法は区別して考えられてきた。つまり、進化型 GNP は、一般的な進化論的計算手法の枠組み内での提案であり、学習型 GNP は、有向グラフ構造をオンラインで学習させる手法を提案するため、進化手法を用いず、強化学習のみで有向グラフ遷移のルールを学習させている。しかし、学習型 GNP は、各状態からほかのすべてのノードへの遷移を考え、強化学習によって最適な遷移を決定しなければならないので、Q-table の規模が膨大になり探索が遅くなる欠点がある。そこで、本論文では学習と進化を統合した新しい方式 "進化学習型 GNP" を提案する。これは、進化により大域探索を行ない、有向グラフの規模 (Q-table) をコンパクトにし、学習によって局所的な探索を行なうことで、解の探索能力と探索速度の面で優れたシステムを構築することを目的とする。

進化と学習の組み合わせについて文献 9) では、学習用のノードを用意し、そのノードの内容、すなわちエージェントの行動内容を Q 学習によって決定するとともに、学習用

* 早稲田大学大学院情報生産システム研究科
北九州市若松区ひびきの 2-7

* Graduate school of Information, Production, and Systems, Waseda University, 2-7, Hibikino, Wakamatsu-ku, Kitakyushu
(Received August 29, 2003)
(Revised August 23, 2004)

Fig. 1 Basic structure of GNP with evolution and learning

のノードの決定も進化によって自動的に行なっている。また文献 10) では、進化型 GP の計算の中で Q-table を生成している。たとえば、状態変数として x, y, z が存在したとき、得られた GP のプログラムが $(TAB(*xy)(+z5))$ であれば $x*y$ と $z+5$ を軸とする 2 次元の Q-table を表わす。

一方、本論文での GNP の進化学習は、ネットワーク遷移の経路を生成するものである。すなわち、進化によってネットワークの大枠を決定し、学習によって詳細な経路を決定するものである。したがって、進化と学習の適用に関しては各種の検討がなされているが、本論文では、GNP の枠組みの中でその最適化のための進化と学習の統合について述べている。

本論文の構成は以下のとおりである。第 2 節では GNP の基本構造、進化学習型 GNP を説明する。第 3 節ではシミュレーションに用いるタイルワールド環境について述べ、第 4 節でシミュレーション結果を述べる。第 5 節は結論と今後の課題である。

2. Genetic Network Programming (GNP)

2.1 GNP の基本構造

GNP は、Fig. 1 のように一個の初期ノードと複数の処理ノード (Processing node)、および判定ノード (Judgement node) が有向グラフ状に接続された構造をもつ。判定ノードは環境からの入力に対して決められた判定を行ない、その結果に応じて遷移の分岐を行なうノードである。処理ノードにはエージェントが行なうべき行動内容が記述されており、判定ノードとは異なり条件分岐はない。また、初期ノードは最初に行なう判定もしくは処理を決める役割のみを持つため具体的な機能は持たない処理ノードと考えてよい。

GNP は図のような有効グラフであり、初期ノードからブ

ランチの矢印の向きにしたがってノード遷移を開始し、初期ノードへ戻ることはなく、ノード遷移を次々と行なっていくことになる。このノード間の接続の組み合わせによりエージェントの行動系列が生成される。ノード同士が有向グラフ状に接続されることで、一度利用したノードの再利用が可能であり、このため GNP はノード数を固定して進化を行なうことができる。したがって、GNP ではプロットを起こす恐れがない。換言すると、エージェントがタスクを達成するために必要なノードへの接続は、学習・進化の過程で多くなり、必要なノードを何度も用いることができる。ノード再利用の機能により、極端に多くのノードを用いる必要がなく、実験的に適切なノード数の設定が可能となる。以上の特性により、GNP では解くべき問題向けの非常にコンパクトな構造が実現可能である。ただし、GNP の拡張アルゴリズムとしてノード数可変型 GNP¹¹⁾ も提案されている。

前述のように、GNP は初期ノードよりノード遷移を開始すると、初期ノードへ戻ることはなく、ノードの接続にしたがって次々と遷移をしていくため、決められた終端ノードが存在しない。どのノードの内容が実行されるかは、それまでのノード遷移に大きく影響を受けるため、暗に過去の記憶の働きをする。これが GNP の大きな特徴である。このことは GNP の判定ノードの組み合わせが単なる if-then ルールの集合体ではないことを示しており、ある判定ノードを用いるとき、そのノードによる遷移の分岐という事実に加え、この分岐が過去の情報に依存していることを意味する。

また、GNP は遅れ時間を設定することができる。遅れ時間は各ノードの特徴の一つとしてノード遺伝子に書き込まれるため、遅れ時間も考慮した進化が可能となり、より柔軟なシステムを構築することができる。

GNP の構造はノード遺伝子の組み合わせによって表現さ

Fig. 2 Example of the setting of N_{ij}

れる。Fig. 1は進化学習型のネットワークを表わしたものであるが、進化的 GNP との違いは Q 値が遺伝子の中に組み込まれているかどうかである。

K_i はノードの種類を表わし、 $K_i=0$ が処理ノード、 $K_i=1$ が判定ノードである。 ID_i は判定および処理の内容を表わす番号である。 d_i は判定および処理にかかる遅れ時間である。判定ノードの場合条件分岐があるため判定結果に応じて A, B, C, ... を使い分ける。 $d_{ij}^A, d_{ij}^B, d_{ij}^C, \dots$ はノード遷移にかかる遅れ時間である。 $N_{ij}^A, N_{ij}^B, N_{ij}^C, \dots$ はノード i からノード j への接続が可能かどうかを表わし、 $N_{ij}=0$ であれば不可能、 $N_{ij}=1$ であれば可能となっている。 $Q_{ij}^A, Q_{ij}^B, Q_{ij}^C, \dots$ は、ノード i からノード j への遷移の価値を表わす値である。条件分岐がある場合、判定結果が“B”であれば $d_{ij}^B, N_{ij}^B, Q_{ij}^B$ を参照する。処理ノードは条件分岐がないため $d_{ij}^A, N_{ij}^A, Q_{ij}^A$ のみを用いる。以降これらの A, B, C, ... を状態と呼ぶことにする。

ここで、進化的 GNP は各状態から一本のブランチが伸びておりつぎのノードを示している。一方、進化学習型 GNP は各状態ごとに $N_{ij}=1$ である遷移を複数設定する [Fig. 2] ことによって複数のブランチを作る。これは、進化によって複数の遷移候補を選び、学習によって一つに決定するという学習と進化の連携を行なっているためである。ただし、各状態ですべての N_{ij} の値が“0”であると、つぎのノードへ遷移できないので少なくとも1つは“1”となるように設定する必要がある。

2.2 ネットワーク遷移のルール

Fig. 3は学習進化的 GNP のノード遷移の手続きを表わしている。判定ノードの場合、割り当てられた判定を行ない、その結果から A, B, C, ... のいずれかを参照する。たとえば、現在のノードが i で、判定結果が“B”であった場合 $N_{ij}^B=1$ である遷移を選び、その中からあらかじめ設定した確率 $1-\varepsilon$ で最大の Q 値を持つ遷移 $i \rightarrow j$ ($1 \leq j \leq n-1$ (注1), $j \neq i$) を選び、 ε の確率でランダムに選ぶ。これは本論文が ε -greedy 方針にしたがっているからであるが、これについては学習フェーズの節で詳しく説明する。処理ノードの場合、割り当てられた処理を行ない、 $N_{ij}^A=1$ である遷移を選び、その中から $1-\varepsilon$ の確率で最大の Q 値を持つ遷移を選び、 $1-\varepsilon$ の確率でランダムに選ぶ。

(注1) 総ノード数が n 個であるとき各ノードには 0 から $n-1$ の番号が付いている。

Fig. 3 Flowchart of node transition

Fig. 4 Flowchart of evolution and learning

2.3 進化フェーズ

Fig. 4は学習と進化の大まかな流れを示しており、図中の Learning phase における判定/処理の詳細な流れは、Fig. 3で示している。

本論文では、進化フェーズでネットワークの構造 (Q-table) をコンパクトにし、学習フェーズでの Q 学習が効率よく行なわれるようにしている。つまり従来の学習型 GNP では各状態からそのほかのすべてのノードへの遷移を考え、その中

Fig. 5 Crossover of GNP with evolution and learning

から1本をえらぶという処理が必要であったが、提案手法では、各状態からいくつかのノードへの遷移を進化フェーズで選び(絞込み)、その中の1本を学習フェーズで選ぶことになる。よって、進化フェーズは大域探索を行ない、ある程度よい性能が得られるノード遷移の組み合わせを選び、学習フェーズでは、絞り込まれた遷移の中から最適なものを選んでいくことで局所探索を行なうことになる。

本節では進化オペレータの適用法について述べ、次節でQ値の学習法について述べる。

GNPでは交叉、突然変異を用いてプログラムの進化を実現する。各世代ごとに前世代の個体数と同数の個体を生成し、入れ替えを行なう。各世代で交叉および突然変異によって生成される個体数をそれぞれ交叉個体数、突然変異個体数と呼び、常に一定数の新しい個体を生成する。また、適合度の高い個体を優先的に残す、エリート保存選択も用いている。

2.3.1 交叉オペレータ

交叉は2個の親個体間で行なわれ、2個の子個体を生成する[Fig. 5]。交叉のアルゴリズムを以下に示す。

- (1) トーナメント選択を2回行ない、親となる2個体を選択する。
- (2) 各ノードごとに交叉率 P_c で交換を行なうノードを選択する。
- (3) 2個体間において、(2)で選択された同ノード番号の遺伝子を交換する。
- (4) 生成された2個体を次世代の個体とする。

Fig. 6 Mutation of GNP with evolution and learning

Fig. 5は簡単のため処理ノード3個のみのネットワークの例であるが、判定ノードでの交叉を考える場合は、判定結果、B, C, ... に関する遺伝子も同時に交換する。また、交叉を行なう場合、交換されるノードに付随するQ値は保持し、次世代の初期値として使用する。

2.3.2 突然変異オペレータ

突然変異は1個体で行なわれ、1個の子個体を生成する[Fig. 6]。突然変異オペレータの作用する遺伝子は N_{ij} である。

- (1) トーナメント選択を1回行ない、親となる個体を選択する。
- (2) 各ノードは確率 P_m で突然変異ノードとして選ばれる。各突然変異ノードのすべての $N_{ij}^A, N_{ij}^B, N_{ij}^C, \dots$ について0または1に決められた規則(注2)にしたがって変更する。
- (3) 生成された個体を次世代の個体とする。

Fig. 6も簡単のため処理ノード3個のみのネットワークの例である。判定ノードでの突然変異を考えると、判定結果、B, C, ... に関する遺伝子別に独立して同様の操作を行なう。また、新たに発生したノード遷移(0から1に変化した N_{ij})に対応するQ値は0に設定する。

2.4 学習フェーズ

GNPにおけるQ学習はタスク実行中に判定/処理ごとに行なわれ、Q値が更新される。そのため、進化を1世代ごとに行なうことに加え、判定/処理ごとにも学習を行なうことで効率化を図ることができる。また、進化型計算は大域探索に優れているが、Q学習を応用することで、局所探索の精度も増すことができる。

強化学習では、学習が不十分な場合において、Q値が最大となる行動を選択しつづけると、現時点での政策より優れた政策が存在していても政策の改善を図ることができない。そこで、改善を図る手段の一つとして ϵ -greedyがある。これは ϵ の確率でランダムに行動を選択させ、 $1-\epsilon$ の確率でQ値が最大となる行動を選択するというものであり、Q学習に

(注2) たとえば、決められた数もしくはそれ以下を1に設定し残りを0にする。または、特定の確率にしたがって0か1を決定する方法などが考えられる。本論文では $N_{ij}^A, N_{ij}^B, N_{ij}^C, \dots$ についてそれぞれ2個を1に設定し、そのほかを0に設定することにする。

において、環境の探索と、それまでに獲得した Q 値、すなわち経験の活用とのバランスをとるための一般的な手法である。本論文では、Q 学習の行動選択に ϵ -greedy を用いている。

2.4.1 状態と行動の定義

Q 学習ではエージェントが知覚した情報によって状態 s が決定され、エージェントの実際の行動を a と考える。一方、GNP では判定および処理を行なった後の状態を s と定義し、ノード遷移を行動 a と定義する。Fig. 7 はノード遷移の例を示しているが、記号“●”が状態を、“→”が行動を示している。

2.4.2 GNP の Q 学習

Fig. 7 を用いて Q 値の更新方法について述べる。いま、時刻 t でノード $i (1 \leq i \leq n-1)$ に遷移してきたところだとする。ノード i は判定ノードであるので、割り当てられた判定を行なう。この例の場合、判定結果は A, B, C の三つをとり得、今回は B であったとする。このとき、状態 s_t は s_i^B であり、 $Q_{i1}^B, \dots, Q_{in}^B$ のうち、対応する N_{ij}^B が 1 である遷移を選び、その中で最大の Q 値を持つ遷移を行動 a_t として選択する。ただし、 ϵ の確率でランダムなノード（ただし $N_{ij}^B=1$ ）への遷移が選択される。

● case1: 遷移先が処理ノードの場合

Q_{ij1}^B を持つ遷移が選択された場合、処理ノード j_1 に移り、割り当てられた処理を行ない、報酬 r_t を得る。時刻は $t+1$ となり、状態 s_{t+1} は判定ノードと異なり分岐がないため、確定的に s_{j1}^A になる。 $Q_{j11}^A, \dots, Q_{j1n}^A$ のうち、対応する N_{j1k}^A が “1” である遷移を選び、その中で Q_{j1k1}^A が最大であればこれを Q_{ij1}^B の更新に用いる。また、行動 a_{t+1} は $1-\epsilon$ の確率で最大の Q 値をもつノード k_1 への遷移となり、 ϵ の確率でランダムな遷移（ただし、 $N_{j1k}^A=1$ ）が選択される。

このとき、Q 値の更新式はつぎのようになる。

$$Q_{ij1}^B = Q_{ij1}^B + \alpha [r_t + \gamma Q_{j1k1}^A - Q_{ij1}^B] \quad (1)$$

● case2: 遷移先が判定ノードの場合

Q_{ij2}^B を持つ遷移が選択された場合、処理ノード j_2 に移り、割り当てられた判定を行なう。判定ノードでは実際の行動は行なわれないので報酬 r_t は零である。時刻は $t+1$ となり、状態 s_{t+1} は判定の結果に応じて $s_{j2}^A, s_{j2}^B, s_{j2}^C$ のどれかになる。判定結果が A であったとき、 s_{t+1} は s_{j2}^A になる。 $Q_{j21}^A, \dots, Q_{j2n}^A$ のうち、対応する N_{j2k}^A が “1” である遷移を選び、その中で Q_{j2k2}^A が最大であればこれを Q_{ij2}^B の更新に用いる。また、行動 a_{t+1} は $1-\epsilon$ の確率で最大の Q 値をもつノード k_2 への遷移となり、 ϵ の確率でランダムな遷移（ただし、 $N_{j2k}^A=1$ ）が選択される。

このとき、Q 値の更新式はつぎのようになる。

$$Q_{ij2}^B = Q_{ij2}^B + \alpha [Q_{j2k2}^A - Q_{ij2}^B] \quad (2)$$

一般的な Q 学習ではすべての状態と行動の対に Q 値が割り当てられているが、GNP では判定ノードと処理ノードのそれぞれに Q 値が割り当てられていることが異なる。Fig. 8 は一続きの判定とそれに続く処理を “One Iteration” と定義

Fig. 7 Example of node transition

Fig. 8 Example of the flow of One Iteration

することを示している。エージェントが処理を行なった後与えられる報酬は、その処理に導いたすべての判定も同様に受け取るべきだと考え、One Iteration 中の処理および判定ノードには報酬を割り引かず分配している。言い換えれば、判定ノードへの遷移で割引を行なうと、判定ノードが多くなればなるほど分配される報酬が少なくなり、結果的に判定ノードが使われなくなる。したがって case2 では、割引率 γ は “1” に設定し、割引を行なわない。

3. シミュレーション

3.1 タイルワールド

提案手法の性能を検証するために、タイルワールド環境¹²⁾におけるエージェントのプランニング問題のシミュレーションを行なう。Fig. 9 はタイルワールドの例であり、複数のエージェント、障害物、タイル、穴、床からなる 2 次元格子状の

Fig. 9 Tileworld

仮想世界である。エージェントは複数のセンサ機能と行動機能を持ち、それらを駆使してすべてのタイルをできるだけ早く穴に落とすことを目的とする。そのため、限られたセンサ機能と行動機能を、エージェントの状況に応じて使い分けることが要求される。与えられたセンサ機能、行動機能はそれぞれ単独ではタスク達成のための直接的な機能として成り立たず、それらの工夫された組み合わせによってのみタスク達成が可能となっている。

3.2 ノード関数

GNPで構成されるエージェントは Table 1 のノードを持っている。処理ノードはその処理内容を実行した後状態が一意に決定されるので、接続情報にしたがって次ノードへ遷移する。判定ノードは割り当てられた判定を行ない、その結果によって状態が決定される。ノード JF, JB, JL, および JR による判定結果は各方向の情報にしたがって、床、壁、タイル、穴、およびエージェントのいずれかをとり、状態が決定される。また、TD, HD, THD, および STD は前、後、左、右、なし、のいずれかを判定結果としてとることになり、これが Fig. 7 の A, B, C, ... のようにそのノードの状態となる。

3.3 シミュレーション結果

シミュレーションは、進化学習型 GNP (GNP with EL), 進化型 GNP (GNP) および GP を用い、それぞれ 2 種類の環境内 (シミュレーション I: 静的環境, シミュレーション II: 動的環境) で行なった。実行条件は Table 2 のとおりである。

進化学習型 GNP のエリート個体数が常に 5 に設定されているのは、すべての個体はタスク実行中にもそのプログラムを変更し続けているためエリート個体を多くに残すことによってシステムを安定させるねらいがある。そのほかの手法でもシミュレーション II においてはエリート個体を 5 としている。これはシミュレーション II は環境が動的に変動し、タイルと穴の出現場所が確率的であるため、したがってシステムを安定させるために 5 個を残している。また、本論文では簡単化のため、ノード間の遅れ時間 d_{ij} を 0, 判定ノードの遅れ時間 d_i を 1, 処理ノードの遅れ時間 d_i を 5 と設定している。また、エージェントが一連の行動を行なう

単位時間を 1 ステップと定義し、1 ステップあたりに可能な処理時間の上限を設定すれば、エージェントは 1 ステップ中に何らかの判定または処理が必ず行なわれる。ここでは 1 ステップの最大時間を 5 と定義し、エージェントは 1 ステップに 4 回以内の判定と 1 回の処理、または 5 つの判定を行なうことができるものとする。

また、参考のため木構造を持つ GP によるシミュレーションも行なっている。ルートノードおよび非終端ノードに GNP の判定ノードを用い、終端ノードに処理ノードを用いた GP を生成する。エージェントの行動決定の手続きは、ルートノードから条件分岐を開始し、非終端ノードによる条件分岐を続け、辿り着いた終端ノードの内容を実行することになる。これを 1 ステップとして繰り返していく。シミュレーションは深さを固定した GP を用い、よい結果を示す深さを実験的に求めた。遺伝的操作には交叉と突然変異を用いているが、深さを固定するため Poli らの一点交叉¹⁴⁾を用いている。また木構造の初期化に“full method²⁾”と“ramped half-and-half method”の二つを用いてさまざまな形と大きさの木を作った。

3.3.1 シミュレーション I

本節では、Fig. 9 の環境を用いてシミュレーションを行なった。この環境内には、それぞれ 30 個のタイルと穴が配置されており、エージェントは 150 ステップでできるだけ多くのタイルを穴へ落とすことが求められている。エージェントはたまかな情報しか得られないため、判定ノードと処理ノードの工夫された組み合わせを見つける必要がある。実行結果を Fig. 10 に示す。それぞれ、各世代における最良個体の適合度を、乱数系列を変えた 30 試行で平均化したものである。Fig. 10 を見ると、進化学習型 GNP が適合度、解の探索速度ともに最もよい結果を残していることがわかる。

Q 学習を用いた手法が用いない場合よりよくなるというのは自明のようにも思われるが、本論文では、エージェントが同じ時間 (ステップ数) だけ行動を行なったとき、その経験をどれだけ解の探索に利用できるか、ということに重点をおいている。提案手法は学習フェーズの実行分だけ計算量が増えることになるが^(注 3)、タスク中には利用可能な情報 (センサの情報、取った行動の良否等) がたくさん存在するため、動作中に得られる情報を即座に反映させてプログラムの修正を行ない、よりよい解を得ることは有意義であると考えられる。

しかし、進化学習型 GNP と進化型 GNP の、最終世代 (3000 世代目) で得られた最良個体の適合度に関する分散分析表を Table 3(a) に示したところ、p 値は 12.13% となり、最終世代でのデータにおいて進化型 GNP と進化学習型 GNP の間に統計的に有意な差はないという結果となってしまう。本シミュレーションの環境は、各試行において、タ

(注 3) 約 1.45 倍 (500 世代あたり GNP: 110 秒, GNP with EL: 159 秒, 実行環境: Pentium4 2.5GHz, DDR-SDRAM PC 2100 512MB)

Table 1 Node Function

Judgement node		Processing node	
symbol	content	symbol	content
JF	judge FORWARD	MF	move forward
JB	judge BACKWARD	TR	turn right
JL	judge LEFT side	TL	turn left
JR	judge RIGHT side	ST	stay
TD	direction of the nearest TILE from the agent		
HD	direction of the nearest HOLE from the agent		
THD	direction of the nearest HOLE from the nearest TILE		
STD	direction of the second nearest TILE		

Table 2 Simulation Conditions

		GNP	GNP with EL	GP
the number of individuals	crossover	120	120	120
	mutation	180 (176 in II)	176	120 (116 in II)
	inversion	—	—	60
	elite	1 (5 in II)	5	1 (5 in II)
the number of nodes		60 (5 per each)	60	(depth 4-6)
Crossover rate P_c		0.1	0.1	—
Mutation rate P_m		0.01	0.01	0.1
Inversion rate P_i		—	—	0.1
Learning parameters	step size α	—	0.1	—
	discount rate γ	—	0.9	—
	ε	—	0.01	—

Fig. 10 Result of Simulation I

Fig. 11 Initial condition of Tileworld II

Fig. 12 Result of Simulation II

イルと穴の数および位置が固定されており、不確実性のある変動は起こらない環境であった。よって進化的な環境においては良い結果を示し、統計的には提案手法との間に評価できるほどの大きな差が生じなかった。そこで、シミュレーション II では、より動的で複雑な環境を用意し、より一般的な行動の獲得を目的とした実験を行なった。

また、本シミュレーションで用いた GP の非終端ノードの引数の数は比較的多く 5 個である。そのため深さが増すほどノードの数が極端に多くなる。たとえば、深さ 4 で full method により初期化された木 (GP-full4) は 781 個、GP-full5 は 3,906 個、GP-full6 は 19,531 個のノードを持つ。ノード数が増すほど解の表現能力は増すが、プログラムの実行時間、メモリの使用量は極端に多くなった。たとえば、GP-full6 は計算時間が長く実行が困難であり、また GP-full7 はメモリ不足のため実行不可能であった。ノード数が増すことで表現能力の向上は見られたが、GNP よりよ

い適合度を得ることはできなかった。

3.3.2 シミュレーション II

つぎに Fig. 11 の環境に対するシミュレーションを行なった。本節で用いる環境は、初めは少しのタイルと穴しか存在しないがあらかじめ設定した確率にしたがってタイルと穴

Table 3 Analysis of Variance of the best individuals at the last generation

(a) Simulation I						
Summary						
Group	Sample size	Total	Mean	Variance		
GNP with EL	30	601	20.03	2.309		
GNP	30	579	19.30	4.217		
Analysis of Variance						
Source	Sum of square	Degree of freedom	Meansquare	F value	P value	Critical value (F distribution)
Between groups	8.067	1	8.067	2.472	0.1213	4.007
Within groups	189.3	58	3.263			(level of significance 5%)
Total	197.3	59				

(b) Simulation II						
Summary						
Group	Sample size	Total	Mean	Variance		
GNP with EL	30	544	18.13	6.602		
GNP	30	492	16.40	13.56		
Analysis of Variance						
Source	Sum of square	Degree of freedom	Meansquare	F value	P value	Critical value (F distribution)
Between groups	45.07	1	45.07	4.471	0.03879	4.007
Within groups	584.7	58	10.08			(level of significance 5%)
Total	629.7	59

Fig. 13 Ratio of used nodes

が出現したり消滅したりする。今回は、各ステップごとにタイルと穴がそれぞれ 20% の確率で 1 個出現し、タイルワールド内にあるすべてのタイルと穴がそれぞれ 1% の確率で消滅するものとした。この環境はシミュレーション I よりも動的であり、エージェントはできるだけ早くタイルを穴へ落とす一般的な知識の獲得が求められる。また、制限時間は 300 ステップと設定した。

最良個体の適合度を 30 回の試行で平均したものを **Fig. 12** に示した結果、進化学習型 GNP が最もよい適合度と探索速度を示している。本シミュレーションでも GP はノード数が増加するとそれだけ表現能力が増し、適合度が改善されたが GNP がよりよい結果を示した。また GNP は少ないノードで効率的なプログラムを生成している。さらにシミュレーション I と同様に、最終世代で得られた最良個体の適合度に関する分散分析表を Table 3(b) に示したところ、p 値は 3.879% となり、進化型 GNP と進化学習型 GNP の間には統計的に有意な差があることがわかった。この結果から、提案手法は、より動的で不確実性の高い環境においてよい性能を示すことがわかった。これは、従来の GNP が、タスク終了後に得られる適合度のみに基づいてプログラムを生成するのに対し、進化学習型 GNP は、タスク試行中に得られる情報を逐次利用し、環境の変動も考えた学習ができているからだと考えられる。進化型 GNP も進化学習型 GNP も、大域的な探索には進化を用いているため、プログラムの生成の大部分は進化による影響が大きく、圧倒的な差は生じていないが、進化学習型 GNP は Q 学習によって局所的な探索を行なうことができるため、平均的によりよい適合度を得ることができる。また、この局所的な探索（調整）は実際に応用する際により精度の高いシステムを作る上で有意義だと考えられる。

Fig. 13 は初期世代と最終世代で進化学習型 GNP の最良個体が使ったノードの使用比率を表わしている。x 軸はノード番号とノードの種類を表わしており、y 軸はノードの比率を表わしている。初期世代ではネットワーク構造はランダムに初期化されているのに加え、環境が常に変動するため、多くのノードを用いて探索を行なっている。しかし、最終世代では必要なノードだけを多く使う構造となっている。たとえば、MF, JF, TD, THD のノードは頻繁に使われており、こ

の結果から、主にエージェントは前を見つタイルと穴の方向を調べ前進するという直観的にもわかりやすい行動が得られている。ネットワーク構造が非常にコンパクトになっているため、Q-tableの大きさも重要な状態と行動のみを抽出したコンパクトなものになっている。したがって、学習が迅速に行なわれる要因となっている。このように必要なノードを進化により獲得しコンパクトな構造を生成するのはGNPの進化の特徴であるため、従来のGNPでもこのような特徴的なノード比率を実現することはできる。しかし、提案手法では問題解決に重要なノードの組み合わせ（ノード遷移）をより細かく調節してタスクを実行できるようになるため、よりよい解が探索できている。

4. 結 論

本論文では、進化論的計算とQ学習を用いることで学習速度および探索精度の向上を行なった。進化のみの計算では、試行中にプログラムの修正を行なうことができないが、Q学習を行なうことで、毎判定/処理の結果を有効にプログラムの生成に役立てることができるようになった。また、進化論的計算は長期的に多様化と淘汰を繰り返すことによって大域探索に優れ、Q学習は即時報酬を基にした漸近的な学習が可能であるため両者の組み合わせによって、より効率的な解の探索が可能になった。

本シミュレーションでは、基本的な学習進化型GNPの能力を検証するためのベンチマーク問題としてタイルワールド問題を使用した。今後はより複雑で、動的な環境を用いることで、提案手法の能力を検証していく予定である。また、学習フェーズではQ学習を用いてQ値の学習を行なったが、Profit Sharingやモンテカルロ法などの手法を用いて学習させ、比較することも考えている。さらに、最新のGPなど、ほかの進化手法や学習法との比較も重要であると考えられる。

参 考 文 献

- 1) John H. Holland: *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975, MIT Press (1992)
- 2) John R. Koza: *Genetic Programming, on the programming of computers by means of natural selection*, Cambridge, Mass., MIT Press (1992)
- 3) John R. Koza: *Genetic Programming II, Automatic Discovery of Reusable Programs*, Cambridge, Mass., MIT Press (1994)
- 4) 片桐, 平澤, 胡, 村田: Genetic Network Programming とその応用システム, 電学論 C, **122-C-12**, 2149/2156 (2002)
- 5) Richard S. Sutton and Andrew G. Barto: *Reinforcement Learning - An Introduction*, MIT Press Cambridge, Massachusetts, London, England (1998)
- 6) 間普, 平澤, 胡, 村田: 遺伝的ネットワークプログラミングのオンライン学習, 電学論 C, **122-C-3**, 355/362 (2002)
- 7) S. Mabu, K. Hirasawa, J. Hu and J. Murata: Online learning of Genetic Network Programming, in *Proc. of Congress on Evolutionary Computation*, 321/326 (2002)
- 8) 間普, 平澤, 胡, 村田: Online Learning of Genetic Network

Programming and its Application to Prisoner's Dilemma Game, 電学論 C, **123-C-3**, 535/543 (2003)

- 9) Keith L. Downing: Adaptive Genetic Programming via Reinforcement Learning, in *Proc. of the 3rd Genetic and Evolutionary Computation Conference*, 19/26 (2001)
- 10) H. Iba: Multi-agent reinforcement learning with genetic programming, in *Proc. of the Third Annual Conference of Genetic Programming*, 167/172 (1998)
- 11) 片桐, 平澤, 胡, 村田: ノード数可変型 Genetic Network Programming, 電学論 C, **123-C-1**, 57/66 (2003)
- 12) M. E. Pollack and M. Ringuette: Introducing the tile-world: Experimentally evaluating agent architectures, in *Proc. of the conference of the American Association for Artificial Intelligence*, 183/189 (1990)
- 13) Claudia V. Goldman and Jeffrey S. Rosenschein: Emergent coordination through the use of cooperative state-changing rules, in *Proc. of the Twelfth National Conference on Artificial Intelligence*, Seattle, WA, 408/413 (1994)
- 14) Riccardo Poli and William B. Langdon: Schema Theory for Genetic Programming with One-point Crossover and Point Mutation, *Evolutionary Computation*, **6-3**, 231/252 (1998)

[著 者 紹 介]

間 普 真 吾 (学生会員)

2001年九州大学工学部電気情報工学科卒業。2003年同大学大学院システム情報科学府電気電子システム工学専攻修士課程修了。現在、早稲田大学大学院情報生産システム研究科博士後期課程在学中。

平 澤 宏 太 郎 (正会員)

1964年九州大学工学部電気工学科卒業。66年九州大学大学院工学研究科修士課程電気工学専攻修了。同年(株)日立製作所入社日立研究所勤務。89年8月同所副所長。91年8月大みか工場主管技師長。92年12月九州大学工学部教授。96年5月同大学大学院システム情報科学研究科教授。2000年同大学大学院システム情報研究科教授。2002年9月早稲田大学大学院情報生産システム研究科教授、現在に至る。電気学会、情報処理学会、IEEEの各会員。工学博士。

古 月 敬 之 (正会員)

1985年中国中山大学大学院修士課程修了。同年、同大学電子工学科助手。88年同講師。93年来日。97年九州工業大学情報工学研究科博士課程修了。同年九州大学ベンチャービジネスラボラトリ非常勤研究員を経て、同年8月同大学大学院システム情報科学研究科助手。2000年同大学大学院システム情報研究科助手、2003年早稲田大学大学院情報生産システム研究科助教授、現在に至る。主として学習ネットワーク理論、システム同定とその応用に関する研究に従事。電気学会会員、情報工学博士。