推薦論文

強化学習を用いた遺伝的ネットワークプログラミングとそのエージェントの行動生成における性能評価

間普真吾[†] 平澤宏太郎[†] 古月敬之[†]

これまで，新しい進化論的計算手法として「遺伝的ネットワークプログラミング（GNP）」を提案してきた．GNP は有向グラフ構造の遺伝子で解を表現することが最も大きな特徴であり，そのため様々な機能を本来的に備えている．たとえば，エージェントの過去の行動を暗黙的に記憶できたり，ノードの再利用機能により，非常にコンパクトなグラフ構造を生成することが可能になったりしている．本論文では，GNP の性能向上のため，有向グラフの遷移に状態と行動を定義し，強化学習を GNP に組み込むことで，解の探索能力の向上を実現している．また，代表的なエージェントのベンチマーク問題を用いたシミュレーションを行いその性能を検証している．

Genetic Network Programming Using Reinforcement Learning and Its Performance Evaluation on Making Agent Behavior

Shingo Mabu,[†] Kotaro Hirasawa[†] and Jinglu Hu[†]

A new evolutionary method named “Genetic Network Programming, GNP” has been proposed. GNP represents its solutions as graph structures which have some useful functions inherently. For example, GNP has the implicit memory function which memorizes the past action sequences of agents, and GNP can re-use nodes repeatedly in the directed graph flow, so very compact graph structures can be made. In this paper, the search ability of solutions is improved by defining states and actions in the directed graph flow and applying reinforcement learning to GNP. In the simulations using two typical benchmark problems of agents, it is clarified that the proposed method improves the ability of GNP.

1. はじめに

生物は，学習，進化，発生などのメカニズムに基づいて高次で動的な環境にうまく適応してきた[1]．これまで，生物の進化のメカニズムを模倣した進化論的計算手法が提案されており，代表的な手法に遺伝的アルゴリズム（GA）[2]，遺伝的プログラミング（GP）[3] がある．GA は解を文字列などで表現し，主に最適化問題への応用がなされている．GP は解を木構造で表現することにより，様々な問題への適用が可能となったため進化論的計算手法の進展に大きく寄与している．また，解の表現方法の違いにより，問題に適した特長を備えることができると考えられる．

最近，GA，GP など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）” が提案されている[4]．GNP は解をグラフ構造で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．詳しくは 2 章で説明する．

文献 4) の GNP は，一般的な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い，その結果に基づいて遺伝的操作を施し，これを 1 世代として繰り返していくものである．一方，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，進化を用いず，Q 学習[5] を用いて GNP の構造を学習する手法[6] も提案されている．本論文では，強化学習と進化を組み合わせた新しい GNP のアルゴリズム（GNP with Reinforcemet Learning，GNP-RL）を提案し，その性能を検証している．強化学習の役割は

† 早稲田大学大学院情報生産システム研究科
Graduate School of Information, Production, and Systems, Waseda University

本論文の内容は 2004 年 3 月の火の国情報シンポジウムにて報告され，火の国情報シンポジウムプログラム委員長により情報処理学会論文誌への掲載が推薦された論文である．

学習速度と探索精度の向上であり，進化の役割は大域探索の効率を向上させることである．また，本論文でGNP に適用する強化学習のアルゴリズムは，これまでの方式[6] に大きな改良を加え，より一般的な枠組みとなるよう工夫している．具体的には，有向グラフ上で，1) 新しく状態と行動の定義を行い，2) ノード間の接続，ノードの内容を強化学習によって決定し，3) Q テーブルの大きさをコンパクトにすることでメモリの使用量や計算時間の短縮が可能な進化が実現できる，ということである．

また，文献 6) の手法は，強化学習のみを用いてプログラムを生成するため，進化アルゴリズムは使用されていない．つまり，GNP のオンライン学習方式を提案するため，GNP を一度進化手法の枠組みから外し，強化学習のみでノード遷移のルールを学習させることを目的としてきた．しかし，各状態（ノードの分岐点）から他のすべてのノードへの遷移を考えて，その中から 1 つの最適な遷移を強化学習で選択しているため，Q テーブルが非常に大きくなり，大量のメモリを消費してしまう問題があった．本論文で提案するGNP では，有向グラフ上で状態と行動の再定義を行い，グラフ構造（Q テーブルに対応）をコンパクトにする．つまり，構造をコンパクトにし，その後強化学習によって最適なノード遷移を選ぶという処理を行う．これは大域探索を進化に任せ，局所探索を強化学習で行うことをねらいとしている．たとえば，本論文のタイルワールドを用いたシミュレーションでは，同ノード数で比較したとき，1 個体あたりの Q 値（各状態における各行動の価値）の数は，文献 6) の手法が 12,980 個に対し，提案手法は 60–240 個（可変）となり，メモリの使用量，計算量の点で効率が良くなっている．

強化学習と進化を組み合わせた手法には，文献 7), 8) などがある．文献 7) では，学習用のノードを GP の終端ノードに使用し，その内容を強化学習で決定している．文献 8) では，Q テーブルを GP によって生成することを目的としており，たとえば，(TAB($*xy$)($+z5$)) という解は，$x*y$ と $z+5$ の 2 つの軸からなる 2 次元の Q テーブルを表している．一方，提案手法は GNP のグラフ構造の接続とノードの内容を適切なものに設定し，優れたプログラムを生成することを目的としている．

また，グラフ構造を持つ他の進化手法に PADO[9] があるが，PADO には外部メモリがある点が GNP と異なる．PADO のノードには 2 つの機能があり，1 つが行動部で，もう 1 つが分岐決定部である．分岐決定部ではスタックメモリの内容に基づいてノード遷移の分岐を行う．PADO のノード遷移はメモリの内容に基づいて行われ，ある閾値時間以内に終端ノードにたどり着いたときは，メモリを初期化することなく再び初期ノードから遷移を開始する．一方，GNP の特徴は，処理部と判定部をそれぞれノードとして分けて用意し，各ノードを必要なときに柔軟に使い分けることを目的としており，またノード遷移に特定のメモリを使用することなく，過去のノード遷移に蓄えられた暗黙的なメモリ機能を最大限に活用することを前提としている．さらに，PADO は画像認識や音声認識などの静的な問題に適用されているが，GNP は動的な問題を扱うことを主な目的としている．

また，実際の問題では，センサの能力が不完全であったり，ノイズなどの影響があったりすることで，不完全性や不確実性が存在することが多い．GNP はグラフ構造を持つことによって部分観測マルコフ決定過程に適用することが可能となる．たとえば，GNP は有限オートマトンと異なり，必要な情報（ノード）のみを選択することが可能なため，グラフ構造がコンパクトになることに加え，状況に応じた情報の組合せにより，環境の状態を適切に推定できる．また前述のように過去の判定と処理の履歴を暗黙的に記憶することができるため，センサの情報が不十分な場合でも過去の履歴を用いて状態を決定できる．たとえばエージェントの前後左右 4 方向の状況を判別できるセンサがあったとき，履歴を用いない手法の場合現在のみの情報に基づくため，センサの情報が同じであれば異なった状態でも同じ状態と判定してしまう．履歴を用いる手法では，センサの情報が同じであっても過去に “左折した” などの記憶を用いて状態を区別することができる．

本論文は以下のように構成されている．2 章では，GNP の基本構造と提案手法について説明する．3 章では，シミュレーション環境について説明し，その結果を示す．4 章は結論である．

2. Genetic Network Programming

はじめに，GNP との比較のため，GP について簡単に説明する．GP は非終端ノードに *if-then* 型の判定機能を与え，終端ノードに具体的な行動内容を割り当てれば，決定木として利用できる．GP の処理は根ノードから始まり，終端ノードへ向かって *if-then* の条件分岐を行っていく．そして最終的にたどり着いた終端ノードの内容が実行される．しかし，GP は制限がなければ，進化を繰り返すことによって極端に木の深さが増し，大量のメモリと実行時間を費やしてしま

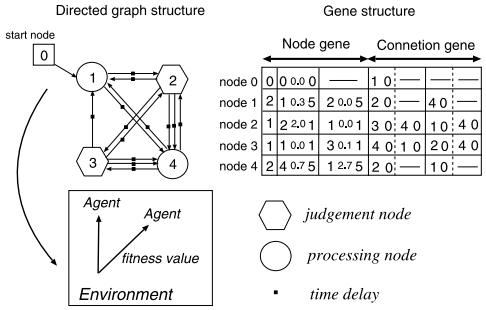

図 1 GNP の基本構造
Fig. 1 Basic structure of GNP.

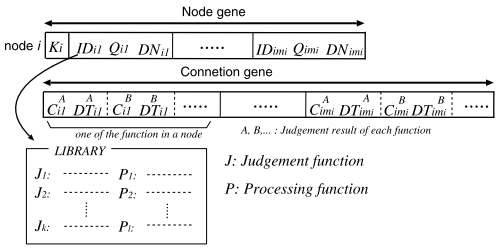

図 2 GNP の遺伝子構造
Fig. 2 Gene structure of GNP.

うブロートの問題もある．

次節から 2.3 節で簡単に GNP の特徴を説明し，2.4 節で，提案手法特有のノード遺伝子について説明する．2.5 節以降では，提案手法のノード遷移のルール，進化オペレータ，学習法について説明する．

2.1 GNP の構成要素

図 1 は GNP の基本構造を表している．GNP のプログラムは複数の判定ノードと処理ノードから構成されており，判定ノードは *if-then* 型の条件判定を行って次のノードを決定し，処理ノードはエージェントの行動を決定する．ただし処理ノードは判定ノードと異なり，条件判定による分岐は行わない．GNP はノード数をあらかじめ設定しておくことができるため，GP で問題となっているブロートを起こすことがない．また，GNP はグラフ構造を持つため，ノードを再利用する機能を本来的に持っており，タスク処理に必要なノードを繰り返して用いることができる．よって，極端に多くのノードを用意する必要がなく，ノードの再利用の機能によりコンパクトな構造が実現できる．また，図 1 の有向グラフは同図中の Gene structure によって表現されるが詳しくは 2.4 節で説明する．

2.2 メモリ機能

GNP のノード遷移は初期ノードから始まるが，終端ノードは存在しない．そのため，ノード遷移はノード間の接続関係と判定ノードでの条件分岐によって決定され，タスク終了までその遷移を続ける．換言すると，実行されるノードの決定は過去のノード遷移に大きく影響を受けるため，それが過去のエージェントの行動の暗黙的な記憶機能として働くことになる．したがって GNP は，現在の情報に加え，過去の記憶も基にした処理が可能になっている．また，ノード遷移は終了条件を満たしたとき，たとえば，制限時間に達したときや，与えられたタスクを完了したときに終了する．

2.3 遅れ時間

GNP は判定ノードまたは処理ノードの実行に要する時間，ノード間の遷移に要する時間を遅れ時間として設定できる．実世界では，状況を判断し，次の行動に移行し，実際に行動を行う際に時間を費やすため遅れ時間を考えた枠組みは有効である．たとえば，ある人が歩いていて目の前に水溜りがあったとする．このとき，水溜りを認識し（判定ノードの遅れ時間），次の行動に移行し（判定ノードから処理ノードへの遷移時間），水溜りを避ける（処理ノードの遅れ時間）ことになる．遅れ時間はそれぞれのノードの遺伝子に書き込まれており，そのノードの特徴を表す情報の 1 つとなっているため，GNP は遅れ時間も考慮した柔軟なプログラムが生成できる．

本論文では簡単のため，ノード遷移の遅れ時間を 0，判定ノードの遅れ時間を 1，処理ノードの遅れ時間を 5 と設定した．さらに，エージェントの行動の単位を 1 ステップと定義し，1 ステップは 5 以上の時間を消費した時点で終了するものとした．つまり，エージェントは 1 ステップ中に 4 回以下の判定と 1 回の処理，または 5 回の判定が行えることになる．

2.4 GNP の遺伝子構造

GNP の構造は，以下で説明する遺伝子の集合で決定される．**図 2** はあるノード $i(0 \leq i \leq n-1$☆$)$ の遺伝子構造を示している．**図 3** は遺伝子とノードの関係を分かりやすく図示したものである．

まずノード遺伝子（Node gene）について，K_i はノードの種類を表し，$K_i = 0$ は初期ノード，$K_i = 1$ は判定ノード，$K_i = 2$ は処理ノードを表す．ID_{ip} $(1 \leq p \leq m_i$☆☆$)$ は処理および判定のコード番号であり，LIBRARY の中にその内容が記載されている．図 3 の例では，すべてのノードの m_i は 2 であるた

☆ ノード数が n 個のとき，それぞれのノードには 0 から $n-1$ の番号がついている．

☆☆ m_i（$1 \leq m_i \leq M$ M：ノード内容の候補の最大数を示す）はノード i の処理または判定内容の候補の数であり，GNP は m_i 個のノード内容のうち 1 つを選択できる．また，m_i の値は突然変異によって変化する．

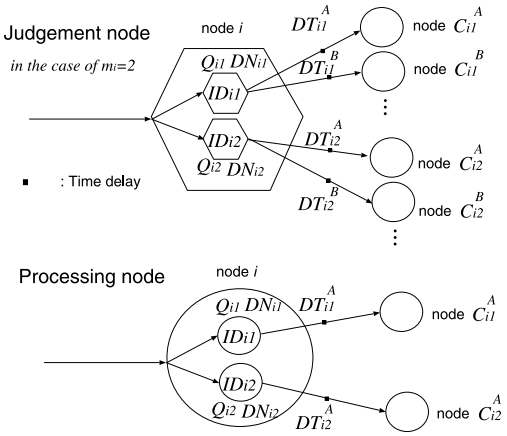

図 3 強化学習を用いた GNP の判定ノードと処理ノード
Fig. 3 Judgement node and Processing node of GNP-RL.

め，GNP は ID_{i1} もしくは ID_{i2} の内容を選択できる．Q_{ip} は Q 値であり，すべての状態行動対に割り当てられている．提案手法では “状態（State)” は実行中のノードを示し，“行動（Action)” はノード内容 (ID_{ip}) の選択である．DN_{ip} は処理または判定の遅れ時間である．

次に接続遺伝子（Connection gene）について説明する．まず，Connection gene 中の遺伝子情報を分割する実線はノードの判定または処理の内容を分ける線とし（つまり，Node gene の ID，Q および DN の組を分ける実線と対応)，破線は各ノード内容における判定結果を分ける線とする（処理ノードは判定を行わないので存在しない)．$C^A_{ip}, C^B_{ip}, \ldots$ は次ノードの番号を示しており，$DT^A_{ip}, DT^B_{ip}, \ldots$ は次ノードに遷移するときの遅れ時間である．ここで，判定ノードは，参照する接続遺伝子（connection gene）の上の添え字 A，B などを判定結果に従って決定する．たとえば，ある判定ノードに接続ブランチ A と B が存在したとする．このとき，もし判定結果が “B” であれば GNP は C^B_{ip} と DT^B_{ip} を参照し次のノードへ遷移する．ただし，処理ノードは判定による分岐がないので C^A_{ip} と DT^A_{ip} のみを参照する．

2.5 ノード遷移のルール

GNP では，まず初期ノードの接続に従って次のノードに遷移する．もし遷移したノード i が判定ノードのときは，$Q_{i1}, \ldots, Q_{im_i}$ の中から ε-greedy 方策に基づいて 1 つの Q 値を選択する．つまり，$1-\varepsilon$ の確率で最大の Q 値を選択し，ε の確率で 1 つの Q 値をランダムに選択する．その後選ばれた Q 値に対応する ID_{ip} が選ばれ，GNP はその判定内容を実行し，判定結果に基づいて次ノードを決定する．たとえば，Q_{i2}，ID_{i2} が選択され，判定結果が “B” であったときは，次のノードは C^B_{i2} となる．

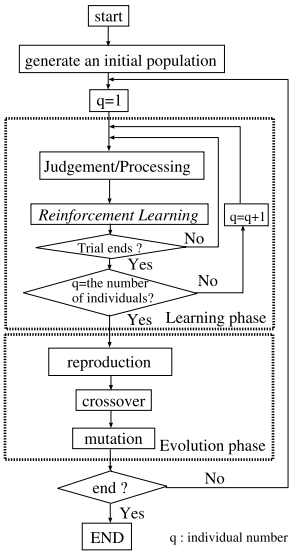

図 4 GNP の実行の流れ
Fig. 4 Flowchart of GNP.

処理ノードのときも，判定ノードと同様の手順でノード内容を 1 つ選択する．たとえば，ID_{i2} が選ばれたとき，その内容を実行後，次のノードは C^A_{i2} となる．

2.6 進化フェーズ（Evolution Phase）

図 4 は GNP のシステム全体の流れを示しており，この節では進化フェーズ中の遺伝的オペレータについて説明する．

GNP は交叉と突然変異を用いて進化を行うが，まず全個体中から良い適合度を示したエリート個体を保存し，その他の個体を，交叉と突然変異で生成した新しい個体と入れ替える．

2.6.1 交　　叉

交叉は 2 個の親個体間で行われ，2 個の子個体を生成する（**図 5**)．このとき，交換されると選択されたノードの遺伝子がすべて入れ替わる．交叉の手順は以下のとおりである．

(1) トーナメント選択を用いて 2 個の親個体を選択する．
(2) 親個体中のそれぞれのノード $i\,(0 \le i \le n-1)$ を確率 P_c で交叉ノードとして選択する．
(3) 親個体間で，同じノード番号の交叉ノードの遺伝子を交換する．
(4) 生成された 2 個の子個体が次世代の個体となる．

図 5 は 3 つの処理ノードからなる構造の簡単な交叉の例を示している．

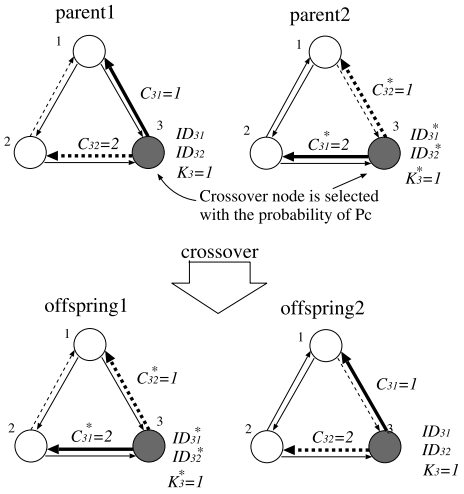

図 5 交叉
Fig. 5 Crossover.

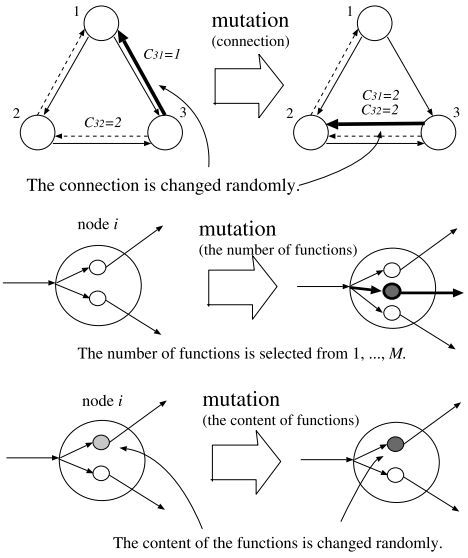

図 6 突然変異
Fig. 6 Mutation.

2.6.2 突 然 変 異

突然変異は 1 個体で行われ，新しい 1 個体が生成される（**図 6**）．

(1) トーナメント選択を用いて 1 個の個体を選択する．

(2) 突然変異操作

(a) 接続：すべてのノードのすべての接続をそれぞれ確率 P_m で選択し，選択された接続をランダムに選ばれた他のノードへの接続に変更する．

(b) ノード内容の候補の数：それぞれのノード i を確率 P_m で選び，そのノード内容の候補の数 m_i を $1, \ldots, M$ のいずれかの値に再設定する．このとき，もし m_i が変更前の値より大きくなったときはその数になるまで新しいノード内容を $LIBRARY$ から選択し，追加する．逆に小さくなったときは，ランダムにノード内容を削除する．

(c) ノードの内容：すべてのノード内のすべてのノード内容についてそれぞれ確率 P_m で選択し，選択された内容をランダムに他の内容に変更する．つまり，ID_{ip} と DN_{ip} が変更されることになる．

(3) 生成された個体が次世代の個体となる．

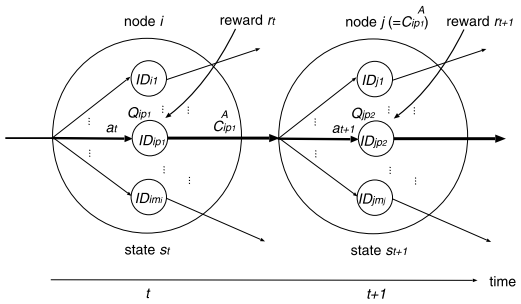

図 7 ノード遷移の例
Fig. 7 An example of node transition.

2.7 学習フェーズ（Learning phase）

本節では，GNP が用いる強化学習のアルゴリズムについて説明する．提案手法は適合度に基づいて毎世代行われる進化に加え，タスク実行中に得られる情報を基にしてオンラインで行われる強化学習を用いることで解の探索能力を向上させている．

2.7.1 Sarsa

一般的に，強化学習では状態 s はエージェントが得ることのできる情報に基づいて決定され，行動 a はエージェントの実際の処理や行動を表している．一方，GNP では，状態を実行中のノードとし，行動をノード内容の決定と定義している．強化学習のアルゴリズムには Sarsa を用いており，**図 7** に従って具体的なノード遷移と Q 値の更新の手順を説明する．

(1) 時刻 t において，すべての Q_{ip} を参照し，ε-greedy 方策に基づいて 1 個の Q 値を選択する．ここでは Q_{ip_1} とそれに対応するノード内容 ID_{ip_1} が選ばれたとする．

(2) ID_{ip_1} の内容を実行し，報酬 r_t を得，次のノード j は $C^A_{ip_1}$ となる．

(3) 時刻 $t+1$ において，Q_{jp} の中からステップ (1) と同様の手順で 1 個の Q 値を選択する．こ

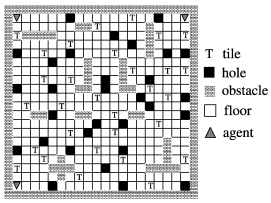

図 8 タイルワールド問題
Fig. 8 Tileworld problem.

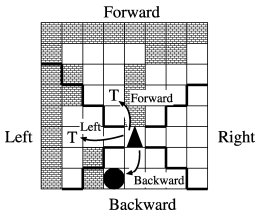

図 9 エージェントが認識できる 4 方向
Fig. 9 Four directions agents perceive.

こでは Q_{jp_2} が選択されたとする．

(4) このとき，以下の更新が行われる．
$Q_{ip_1} \leftarrow Q_{ip_1} + \alpha[r_t + \gamma Q_{jp_2} - Q_{ip_1}]$
α: learning rate, γ: discount rate

(5) 次に，$t \leftarrow t+1$，$i \leftarrow j$，$p_1 \leftarrow p_2$ とし，ステップ (2) に戻る．

以上の例ではノード i が処理ノードであったが，判定ノードのときは，判定結果に従って次のノードを $C^A_{ip}, C^B_{ip}, \ldots$ の中から選択することになる．

3. シミュレーション

本章では，タイルワールドと迷路問題の 2 つのベンチマーク問題について説明し，そのシミュレーション結果を示す．

3.1 タイルワールド

タイルワールド[10] は 2 次元の格子状の空間の中に複数のエージェント，障害物，タイル，穴，床が存在する．エージェントは複数のセンサを備え，移動することができ，制限時間内にできるだけ多くのタイルを穴に落とすことを目的とする．**図 8** は本シミュレーションで用いた環境である．エージェントに与えられたセンサと行動は非常に単純なものであるため，それらの工夫された組合せ，つまり判定ノードと処理ノードの組合せが要求される．シミュレーションで用いるノードは，処理ノードとして，MF：前へ進む，TL：左を向く，TR：右を向く，ST：その場にとどまる，判定ノードとして，それぞれ前後左右を判定するノード {JF, JB, JL, JR} および，最も近いタイルへの方向，最も近い穴への方向，最も近いタイルから最も近い穴への方向，2 番目に近いタイルへの方向を判定するノード {TD, HD, THD, STD} である．判定ノード {JF, JB, JL, JR} はそれぞれ，tile，hole，obstacle，floor，agent のいずれかの判定結果を返し，{TD, HD, THD, STD} はそれぞれ，forward，backward，left，right，nothing のいずれかを返す．また，各方向の判定の例を**図 9** に示している．この判定結果はエージェントが北（図上方向）を向いているときの例である．またこれらの判定結果は，図 2 および図 3 中で，ノード遷移時に参照すべき C_{ip} と DT_{ip} の上側の添え字 $(A, B, \ldots)$ を決定する．たとえば，判定ノード JF の判定結果は A: tile，B: hole，C: obstacle，D: floor，E: nothing となる．

表 1 シミュレーション条件
Table 1 Simulation conditions.

number of individuals	300 (mutation: 179 (SGNP, GP), 175 (GNP-RL), crossover: 120, elite: 1 (SGNP, GP), 5 (GNP-RL))
number of nodes	60 (judgment: 40, processing: 20)
parameters of evolution	$P_c = 0.1$ (GNP-RL, SGNP), $P_m = 0.01$ (GNP-RL), P_m =0.1 (SGNP, GP) tournament size=6
parameters of RL (GNP-RL)	$\alpha = 0.1$, $\gamma = 0.3$, $\varepsilon = 0.1$ $M = 4$(tileworld) $M = 2$(maze)
Q learning	α=0.1, γ=0.99, ε=0.1

3.1.1 適合度と報酬

本シミュレーションでは，1 回のタスクは制限時間（300 ステップ）に達すると終了するとし，適合度と報酬はそれぞれ以下のように設定した．

適合度 = 制限時間内に落としたタイルの数
報酬 = 1（タイルを穴に落としたとき）

3.1.2 シミュレーション結果

本シミュレーションでは比較のため，提案手法（GNP-RL），進化のみの GNP（Standard GNP, SGNP），GP，メモリつき GP☆，および Q 学習を用いて行った．シミュレーションの条件は**表 1** に示している．表 1 に示すように，進化によって毎世代生成される個体の数は 300 個体とし，そのうち，交叉，突然変異およびエリート保存によって生成される個体数を固定して進化させる．たとえば，GNP-RL は，交叉によって 120 個体，突然変異によって 175 個体を生成

☆ 過去 1 回前，2 回前，および 3 回前の処理ノードの内容を非終端ノードで判別できる．

し，前世代のエリート個体を 5 個体残すことになる．交叉は 300 個体の中からトーナメント選択（トーナメントサイズ：6）を 2 回行って 2 個の親個体を選んだ後，遺伝子を交換することで新しい 2 個体を生成する．これを繰り返し 120 個の新しい個体を生成する．突然変異は 300 個体の中からトーナメント選択を 1 回行って 1 個の親個体を選び，突然変異の操作を行うことで新しい 1 個体を生成する．これを 175 回行うことで 175 個の新しい個体を生成する．また，GNP-RL のみエリート個体を 5 個残し，他の進化手法では 1 個体のみ残した．SGNP および GP のエリート個体は，タスク中にプログラムの変更は起こらないため，エリート個体を 1 個残せば次世代でも同様の行動を示すが，GNP-RL はタスク中に強化学習によってプログラムが変更されるため前世代のエリート個体が現在の世代でも良い結果を示すとは限らない．そのためエリート個体を 5 個残すことで安定して良い適合度が得られるようにした．

ここで，GP の非終端ノードには GNP の判定ノードを用い，メモリつき GP はそれに加え，過去 1 回前，2 回前および 3 回前の処理の内容をそれぞれ判定できる 3 種類のノードを用いた．終端ノードには GNP の処理ノードを用いた．GP の 1 ステップは，根ノードから終端ノードに向かって条件分岐を行い，たどり着いた終端ノードの内容が実行されるまでとする．また，交叉には文献 11) で提案されている一点交叉を用い，様々な形と大きさの木を作って比較するため，初期個体の生成に “ramped half-and-half method” [3] を用いた．Q 学習は，GNP のすべての判定ノードによる判定結果の組合せで状態を決定し，各状態と行動（GNP の処理ノードの内容）の組に対して Q 値を割り当てた．

図 10 は 30 回の独立した試行で平均化した各手法の最良個体の適合度を表しているが，GP の結果は様々な木の深さで実行した結果，深さ 5 が最も良い結果を示したためその結果を載せている．また，Q 学習は個体数が 1 個であるため，300 ステップの試行を 300 回繰り返し，これを 1 世代相当とした．また，その間の最良の適合度を図 10 に示した．この結果から，提案手法が最も良い適合度を示していることが分かる．強化学習を用いた手法のほうが使わないものより良い結果を残すことは当然のように思われるが，提案手法の目的は与えられた制限ステップの中でより多くの情報を利用し学習を効率的に行うことであるため，強化学習をうまく GNP のプログラムの生成に利用できたと考えられる．

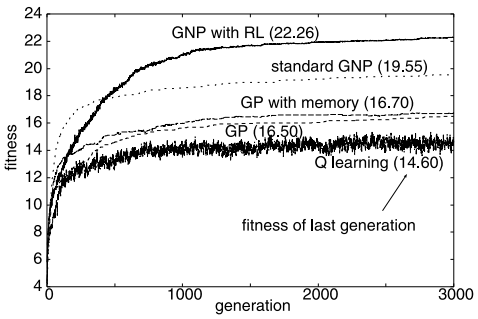

図 10 適合度曲線（タイルワールド）
Fig. 10 Fitness curves of Tileworld problem.

また，この問題では GP の非終端ノードの引数の数が比較的多く（5 個），木の深さが大きくなるにつれてノード数が極端に多くなってしまう結果になった．そのため，深さを増すにつれて GP は良い結果を示したが実行時間と消費メモリが増大してしまった．たとえば，深さ 6 のときはメモリと実行時間を非常に多く使いながらも実行できたが，深さ 7 ではメモリ不足により実行できなかった☆．一方，GNP は良いプログラムを少ないノード数で生成することができた．メモリつき GP は GP より良い適合度を示したが，その差はわずかであった．

また，Q 学習は最終的に平均して 14.6 個のタイルを落とせるプログラムを獲得できたが，進化手法がより良い結果を示した．本問題は判定結果の組合せが多いため Q テーブルのサイズが大きく，Q 学習が十分進まなかったと考えられる．具体的には 8 つのセンサ（判定内容）を持ち，それぞれ 5 つの判定結果を持つため，状態数が $5^8 = 390{,}625$ となり，Q 値の数は $5^8 \times 4$（行動の数）$= 1{,}562{,}500$ となる．

次に強化学習のパラメータが性能に及ぼす影響を検証する．**図 11** はそれぞれ学習率 α，割引率 γ および ε を変えたときの適合度の変化を示したものである．学習率は，0.7 に設定したときに最も良い結果を得ることができた．大きな差はなかったものの，GNP における強化学習では 0.1 のような小さい値に設定すると学習が進みにくくなり，0.9 のような大きい値にすると 1 回の更新の幅が大きくなり，良い解が見つかりにくくなることが分かった．したがって，0.7 は学習速度および得られる適合度の点でバランスがとれているといえる．割引率は 0.3 のときに最も良い性能を示した．これより，GNP はあまり長く先読みをした学

☆ マシンスペック ： Pentium4 2.5 GHz，DDR-SDRAM PC2100 512 MB.

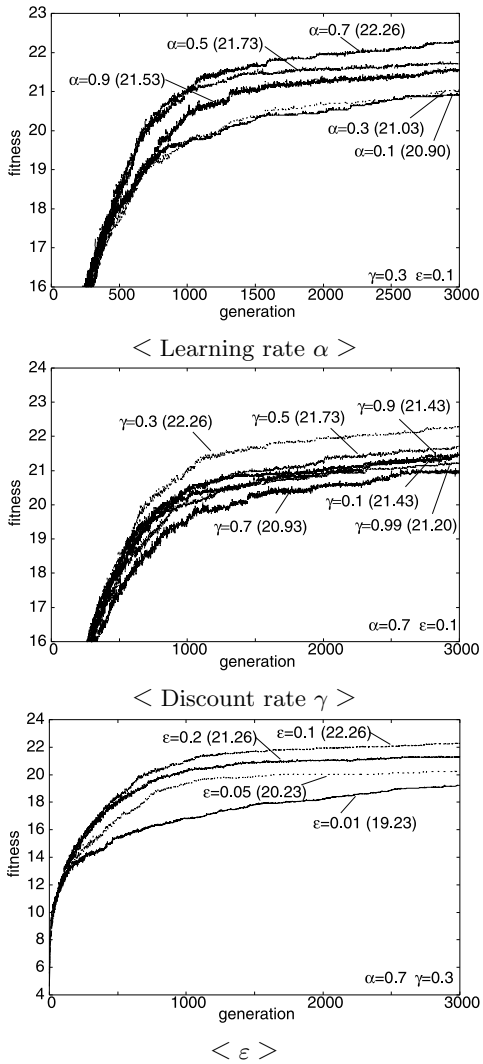

図 11 強化学習パラメータの性能への影響
Fig. 11 Effect of the parameters of RL on the performance.

習はせずに，数ステップ先までを考慮した細かいルールを作ることが重要だと分かる．最後にランダムな行動をとる確率 ε について，0.1 が最も良い性能を示した．0.01 など小さい値に設定すると，探索がなかなか行われずにより良い解が見つからない結果となり，逆に 0.2 など大きな値に設定するとランダムな行動をとる確率が大きくなり，結果的に適合度が下がってしまう結果となった．

3.2 迷路問題

図 12 はシミュレーションで用いた迷路問題の環境を示している．エージェントはゴールに到達することを目的としているが，ゴールの前にはドアがあるため，ドアを開けるために環境の左上にある鍵をとる必

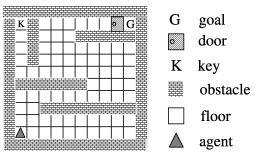

図 12 迷路問題
Fig. 12 Maze problem.

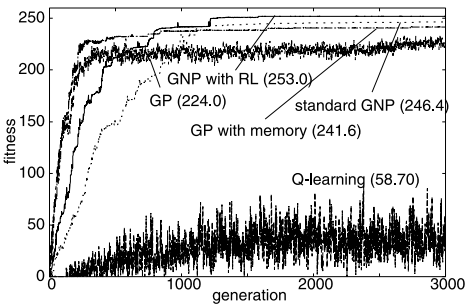

図 13 適合度曲線（迷路問題）
Fig. 13 Fitness curves of Maze problem.

要がある．本シミュレーションで使うノードは，処理ノード {MF, TL, TR, RD☆} と判定ノード {JF, JB, JL, JR} であり，判定ノードは，floor，obstacle，key，goal のいずれかを判定結果として返す．

3.2.1 適合度と報酬

制限ステップは 300 であり，適合度と報酬は以下のように設定した．

適合度 = 余った時間（ゴールに着いたとき），
それ以外は 0.

報酬 = 1（ゴールに着いたとき）

3.2.2 シミュレーション結果

図 13 はタイルワールドのシミュレーションと同様に，独立した 30 回の試行で平均化した各手法の適合度を示している．

結果より，GP およびメモリつき GP（深さ 5）が世代の早い段階では最も良い結果を示している．これはタイルワールド問題に比べ，非終端ノードの種類および引数が少ないためある程度良い解の探索が速く行

☆ MF，TL，TR のうちランダムに 1 つ選ぶ．タイルワールドでは RD を用いない場合でも，すべての手法でタイルを 1 つ以上落とせたため，ある程度の適合度（または報酬）が得られ，その結果進化（または学習）が進んだ．一方，迷路問題ではゴールするまで報酬が得られないため，ランダムな行動をとるノードを用いて環境中を効率的に探索できるようにした．実際，提案手法は強化学習で ε-greedy を採用しており，タスク中にランダムな行動がとれ，解を効率的に探索できるが，SGNP および GP ではタスク試行中の解の探索はできず，なかなかゴールを発見できない結果となったため，処理ノードに RD を導入したところ良い適合度を示した．また，シミュレーションを同じ条件下で行うため，すべての手法で RD を用いることにした．

われたと考えられる．しかし GNP の特徴はノードの再利用とメモリ機能であるため，それらの機能を駆使して良いプログラムを生成するのに少し時間はかかるが，最終的には良い結果を得ることができている．さらに，提案手法は進化のみの GNP よりも，学習速度，良い解の生成の両方の点で優れた結果を示している．具体的には，提案手法は 30 回の試行のすべてで最適方策（適合度 = 253）を得ることができた．GP も最適方策を獲得できていたが，適合度 200 前後の結果となる試行もあり，それが平均的な適合度を下げてしまう結果となった．しかし，メモリつき GP は GP と比べ，平均して約 18 ステップ早くゴールできるようになった．これは過去の処理内容を判定できることにより，エージェントの周りがたとえ同じだと判定されても過去の処理により区別できているからだと考えられる．また，前述のように非終端ノードの種類とその引数の数が少ないためメモリつき GP の解が比較的効率良く探索できたことも考えられる．今後，より複雑な問題（ロボット制御など）で GNP とメモリつき GP を比較していくことは重要だと考えられる．Q 学習の報酬はゴールしたときに “1” あるいは “余った時間” を与えてシミュレーションを行ったが，前者がわずかに良い適合度を示したため，その結果を図 13 に表示した．図 13 より，Q 学習はゴールにたどり着くことはできているが他手法と比べて多くのステップ数が必要となった．

3.3 解の探索速度に関する考察

本節では，解の探索速度に関する考察を行う．タイルワールドおよび迷路問題における性能比較は，世代を横軸にとって行っているが，これはエージェントが環境内で同一時間動作したとき，その経験をどれだけプログラムの生成に利用できるかということを検証することが本論文の目的だからである．つまり，すべての手法について同じステップ数を与えて比較を行っている．この点で，提案手法は 3,000 世代の実行で，進化のみの GNP より早くより良い適合度を得られている．

ここで，進化が提案手法の性能に及ぼす影響を考察するため，個体数 IND を 10，50，100，300，500 個と変え，個体数 × 世代数 = 900,000 となるように世代数を変えて（たとえば 300 個体なら 3,000 世代）シミュレーションを行った（**図 14**）．つまり，エージェントを同ステップ数動作させたときの個体数による性能の違いを検証している．もし個体数が少ないほど性能がよければ，強化学習をより多く行った方が良く，進化の意味がないことになる．しかし，個体数が 10 個のとき，いずれの問題でもあまり良い結果は得られなかった．タイルワールドでは 300 個体のとき最も良い適合度が得られ，迷路問題では最終的な適合度はほぼ同じであったものの 300 個体が最も早く最適解を発見できた．500 個体が 300 個体より性能が劣っているのは，解の候補は多いが世代数が少ないため，進化が十分ではなかったと考えられる．したがって，およそ 300 個体のとき個体数と世代数のバランスが良く，進化が効率良く行われたと考えられる．

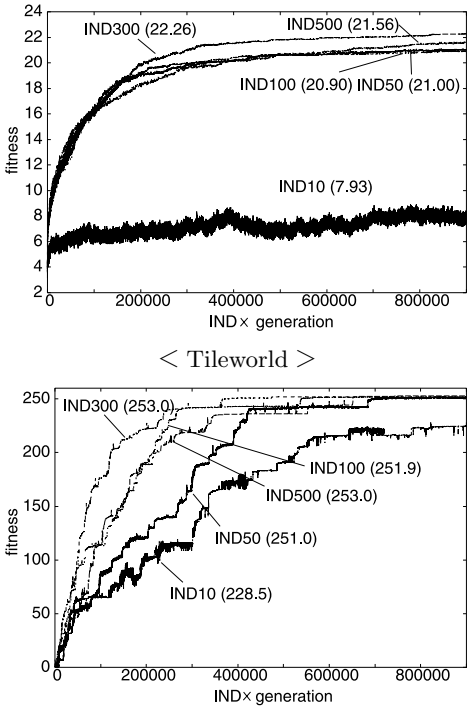

図 14 個体数の性能への影響
Fig. 14 Effect of the number of individuals on the performance.

表 2 3,000 世代の実行にかかる時間［秒］
Table 2 Time[s] spent on executing 3,000 generations.

	GNP-RL	GNP	GP	Q-learning
Tileworld	1,048	581	4,600	3,288
Maze	61.0	30.5	214.4	63.7

最後に実時間での比較を行う．提案手法の計算時間は，強化学習を行う分だけ進化のみの GNP より多くなったが（**表 2**），タスク中には利用可能な情報（センサの情報，とった行動の良否など）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い，より良い解を得ることは有意義であると考えられる．GP はノード数が多いため進化に時間がかかったと考えられ，Q 学習は可能なすべ

ての判定を行うため，8 つの判定を行うタイルワールドでは時間がかかったが，迷路問題では判定の数が 4 つであり，その分計算時間が短くなったと考えられる．

4. 結　　論

本論文では，GNP の性能を向上させるため強化学習を用いた新しいアルゴリズムを提案した．強化学習を用いる目的はタスク試行中に得られる情報を最大限に活用することであり，シミュレーション結果から提案手法が進化のみの GNP より解の探索速度および精度の点で優れていることを確認した．

今後は提案手法の性能をさらに検証するため，ADF を用いた GP や他のグラフ構造を持つ進化手法との比較を行い，より複雑な問題へ適用していく予定である．

参 考 文 献

1) 有田隆也：人工生命，科学技術出版 (2000).
2) Holland, J.H.: *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975, MIT Press, Reading, Massachusetts (1992).
3) Koza, J.R.: *Genetic Programming, on the programming of computers by means of natural selection*, MIT Press, Cambridge, Mass. (1992).
4) 片桐宏伸，平澤宏太郎，胡　敬炉，村田純一：Genetic Network Programming とその応用システム，電気学会論文誌 C，Vol.122-C, No.12, pp.2149–2156 (2002).
5) Richard S. Sutton, A.G.B.: *Reinforcement Learning — An Introduction*, MIT Press, Cambridge, Massachusetts, London, England (1998).
6) 間普真吾，平澤宏太郎，胡　敬炉，村田純一：Online Learning of Genetic Network Programming and its Application to Prisoner's Dilemma Game, 電気学会論文誌 C，Vol.123-C, No.3, pp.535–543 (2003).
7) Downing, K.L.: Adaptive Genetic Programming via Reinforcement Learning, *Proc. 3rd Genetic and Evolutionary Computation Conference*, pp.19–26 (2001).
8) Iba, H.: Multi-agent reinforcement learning with genetic programming, *Proc. 3rd Annual Conference of Genetic Programming*, pp.167–172 (1998).
9) Teller, A. and Veloso, M.: PADO, Learning Tree-structured Algorithm for Orchestration into an Object Recognition System, Technical report library, Carnegie Mellon University (1995).
10) Pollack, M.E. and Ringuette, M.: Introducing the tile-world: Experimentally evaluating agent architectures, *Proc. conference of the American Association for Artificial Intelligence*, pp.183–189 (1990).
11) Poli, R. and Langdon, W.B.: Schema Theory for Genetic Programming with One-point Crossover and Point Mutation, *Evolutioinary Computation*, Vol.6, No.3, pp.231–252 (1998).

(平成 16 年 8 月 9 日受付)
(平成 17 年 10 月 11 日採録)

推　薦　文

筆者らは，新しい進化論的計算手法として有向グラフ構造の遺伝子で解を表現する「遺伝的ネットワークプログラミング（GNP）」を提案してきた．GNP に強化学習を組み合わせて，解の探索能力の向上を図ったが，メモリ使用量，計算量の点で問題があった．本論文では，GNP の性能向上のため，ネットワークの遷移に状態と行動を定義し，強化学習を GNP に組み込むことで，メモリ使用量と計算量を抑えて，解の探索能力の向上を実現している．また，代表的なエージェントのベンチマーク問題を用いたシミュレーションを行い，得られる解の精度の点で強化学習を組み込んだ本論文で提案しているアルゴリズムの有効性も示している．以上の理由により，本論文を推薦するものである．

（平成 15 年度火の国情報シンポジウムプログラム委員長
宇津宮孝一）

間普　真吾（学生会員）

1979 年生．2001 年 3 月九州大学工学部電気情報工学科卒業．2003 年 3 月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．電気学会，計測自動制御学会の各学生会員．

平澤宏太郎（正会員）

1942 年生．1966 年 3 月九州大学大学院工学研究科電気工学専攻修士課程修了．同年 4 月（株）日立製作所入社，日立研究所勤務．1989 年同研究所副所長．1992 年 12 月九州大学システム情報科学研究科教授．2002 年 9 月早稲田大学情報生産システム研究科教授，現在に至る．電気学会，計測自動制御学会，IEEE の各会員．工学博士．

古月 敬之

1962 年生．1985 年中国中山大学大学院修士課程修了．同年 7 月同大学電子工学科助手．1988 年 11 月同講師．1993 年 10 月渡日．1997 年九州工業大学情報工学研究科博士後期課程修了．同年 4 月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．2003 年 4 月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．電気学会，計測自動制御学会の各会員．情報工学博士．