Actor-Critic を用いた遺伝的ネットワークプログラミングの 小型移動ロボットの行動生成における性能評価

間 普 真 吾*・平 澤 宏太郎*
畠 山 裕 之*・古 月 敬 之*

Performance Evaluation of Genetic Network Programming with Actor-Critic for Creating Mobile
Robot Behavior

Shingo MABU*, Kotaro HIRASAWA*,
Hiroyuki HATAKEYAMA* and Takayuki FURUZUKI*

Genetic Network Programming (GNP) has been proposed as a new graph-based evolutionary algorithm. GNP represents its solutions as graph structures which contribute to improving the expression ability of the programs. GNP with Reinforcement Learning (GNP-RL) was also proposed as an extended algorithm of GNP and its effectiveness has been confirmed. Because GNP-RL executes reinforcement learning during task execution in addition to evolution after task execution, it can search for solutions efficiently. In this paper, GNP with Actor-Critic (GNP-AC) is proposed to enhance the effectiveness of GNP-RL. Actor-Critic can adjust numerical values appropriately during task execution, i.e., online learning, and use them for determining actions. To confirm the effectiveness of the proposed method, GNP-AC is applied to the controller of the Khepera simulator and its generalization ability is evaluated.

Key Words: evolutionary computation, reinforcement learning, actor-critic, Khepera robot, Genetic Network Programming

1. はじめに

進化論的計算の代表的な手法である遺伝的アルゴリズム (GA)¹⁾, 遺伝的プログラミング (GP)²⁾は, それぞれ主にストリング構造および木構造を用いて解を表現するものであり, 最適化問題, 関数生成, ロボットプログラムの生成などさまざまな問題に適用されてきた. 遺伝的ネットワークプログラミング (Genetic Network Programming, GNP)^{3), 4)}は GA, GP を拡張した新しい進化論的計算手法としてを提案しているもので, 解を有向グラフ構造で表現することでさまざまな機能を本来的に備えることが可能であり, 主に動的な問題に対して有効であることを明らかにしてきた. 具体的には, 1) ノードの再利用が可能であるためグラフ構造の一部をサブルーチンとして繰り返し使用でき GP における自動定義関数 (ADF)⁵⁾のような機能をもつこと, 2) プログラムをコン

パクトにすることが可能であり, GP でしばしば問題となるブロート (プログラムの構造が極端に大きくなってしまいう現象) を起こすことがないこと, 3) ノード遷移はノード間の接続関係に従って順次実行されていくためノード遷移中に過去の判定や行動を暗黙的に記憶でき, 時間の流れを考慮した行動決定ができることである. さらに, 進化と強化学習⁶⁾を組み合わせることでより効率的なプログラムを生成する方式として, Genetic Network Programming with Reinforcement Learning (GNP-RL) を提案してきた^{3), 7)~9)}. GNP-RL では, タスク終了後に適合度に基づいて行なわれる進化に加え, タスク実行中の判定・処理に対する評価に基づいて強化学習を行なうことができる. したがって強化学習 (オンライン学習) による集中化探索, および進化 (オフライン学習) による広域探索が効率的に行なわれる. しかしながら, すでに提案されている GNP-RL では離散化した入力が必要であるため (たとえば, センサの値が 500 以上であるか 500 未満であるか), 入力空間の分割は設計者が行なう必要があった.

本論文では, 強化学習の一手法である Actor-Critic を GNP に適用し (GNP with Actor-Critic, GNP-AC), 入力情報を適切に判定するパラメータおよび行動決定に用いるパラメータを学習する方式を提案する. Actor-Critic は強化学習の特

* 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 July 30, 2007)

(Revised February 26, 2008)

Fig. 1 Basic structure of GNP-AC

長である将来の報酬を予測した学習が可能であることに加え、連続値を効率的に扱える手法であり⁶⁾パラメータをオンラインで逐次調整できる特長があるため、適切な状況判断が可能かつプログラムの生成を効率的に行なうことが可能である。これまでの GNP-RL は、ノード遷移中に実行する判定および処理の内容を選択する手法であるが、本手法では判定および処理に使われるパラメータの生成、およびその調整を行なえるという点が異なる。さらに提案手法を Khepera ロボットのコントローラに適用し、Khepera シミュレータ¹⁰⁾を用いたシミュレーションを行ないその汎化能力の評価を行なった。本論文は以下のように構成している。第 2 章で提案手法のアルゴリズムについて説明し、第 3 章ではシミュレーションの条件および結果について示す。第 4 章は結論である。

2. Genetic Network Programming with Actor-Critic (GNP-AC)

2.1 GNP-AC の基本構造

Fig. 1 は GNP-AC の基本構造を示している。GNP-AC は 1 個の開始ノードおよび複数の判定ノード、処理ノードが接続された有向グラフ構造をもつ。開始ノードは最初の実行する判定ノードまたは処理ノードへ接続する役割を持ち、判定ノードは環境から受け取る情報を判定する関数を持ちその結果によってノード遷移の分岐を行なうノードであり、処理ノードは環境に対して行動を行なう関数が定義されたノードである。また、これらの関数をノード関数と呼び、初期化時に設計者の手によって用意される。さらに、Fig. 1 内の d_i および $d_i^A, d_i^B, \dots$ はノードの実行遅れ時間とノード間の遷移遅れ時間を示している。これらの遅れ時間は当初、人間の脳の処理遅れを再現すべく導入されたが、本論文では GNP が限られた時間内に何らかの処理を実行することを保証し、システムの信頼性を高める理由で使用されている^{3), 4)}。

2.2 GNP-AC の遺伝子構造

Fig. 1 に GNP の最小単位であるノード i ($0 \leq i \leq n-1$ ^(注 1)) の遺伝子構造を示している。この遺伝子構造は、

ノードの種類、ノードの内容に関する部分 (node gene) とノード間の接続に関する部分 (connection gene) に分けられる。node gene において、 K_i はノード i の種類を示すコードである (0: 開始ノード, 1: 判定ノード, 2: 処理ノード)。さらに、 ID_i はノード i が保持するノード関数の内容を示しており、このノード関数は設計者が用意する LIBRARY に定義されている。 V_i は Actor-Critic の状態^(注 2) 価値関数を示し、 d_i は先に述べたノードの実行遅れ時間を示す。 $\mu_{i1}, \sigma_{i1}, \dots$ はそれぞれのノードがもつ確率密度関数のパラメータである。詳しくは 2.3 節で説明する。判定ノードのみがもつ n_{i1}, n_{i2}, n_{i3} はノード i で GNP が判定するセンサの番号を示している。

connection gene において、 $C_i^A, C_i^B, \dots$ はノード i から接続するノード番号が記されている。同様に $d_i^A, d_i^B, \dots$ は上記ノード間の遷移遅れ時間を示す。上側の添字 $A, B, \dots$ は判定結果に対応しており、処理ノードは条件判定を行わないため常に C_i^A および d_i^A を参照するが、判定ノードの場合判定結果が「A」であるとき接続先は C_i^A 、遅れ時間は d_i^A となり、「B」であれば接続先は C_i^B 、遅れ時間は d_i^B となる。本論文では簡単のためノード遷移による遅れ時間は 0 としている。また、判定ノードの d_i を 1 単位時間、処理ノードの d_i を 5 単位時間としている。さらにエージェントが 10 単位時間以上行動するとそのエージェントは 1 ステップを終了すると定義した。

2.3 GNP-AC のノード遷移規則

GNP-AC の処理は開始ノードから始まり、遷移先は判定ノードの判定結果とノード間の接続関係に基づいて決定される。

2.3.1 判定ノード

実行中のノード i が判定ノードのとき、以下のようにして遷移先を決定する。本論文では、Khepera robot を用いて評価を行なうため、Khepera robot の制御に用いる判定内容およびノード遷移を例にして説明する。

Fig. 2 に判定ノードの例を示す。Khepera robot は 8 個の距離センサを搭載しており、 n_{i1}, n_{i2}, n_{i3} はノード i で GNP が判定するセンサ番号を示している。本論文では、各判定ノードが 3 個のセンサを同時に判定するが、そのセンサ番号の初期値は個体の初期化時にランダムに設定する。まず、判定ノード i は各センサ値 ($x_{n_{i1}}, x_{n_{i2}}, x_{n_{i3}}$) を得た後確率密度関数に基づいて重み ($w_{i1}, w_{i2}, w_{i3}, w_{i4}$) を生成する。その後センサ値と重みの線形結合である出力 y_i を計算し最終的な判定を行なう。Fig. 2 においては $y_i (= 0.125) \geq 0$ であるから判定結果は「A」となりつぎのノード番号は C_i^A となる。もし、 $y_i < 0$ であれば判定結果は「B」となる。

2.3.2 処理ノード

実行中のノード i が処理ノードのとき判定ノードと同様に

(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。

(注 2) GNP-AC ではノードを「状態」と定義する。

Fig. 2 An example of a judgment node

確率密度関数によって v_i を生成する。 $ID_i = 0$ であれば右車輪の速度 v_R を v_i に設定し、 $ID_i = 1$ であれば左車輪の速度 v_L を v_i に設定する。ここで、 v_R および v_L は整数でなければならないため v_i の小数第1位を四捨五入して整数としている。 Fig. 3 では $v_i = 5$ なので右車輪の速度 v_R は5となる。ノード遷移先は G_i^A である。

2.4 進化フェーズ

Fig. 4 は GNP-AC 全体のフローチャートである。各世代においてエリート個体は次世代に持ち越し、他の個体は交叉と突然変異によって生成する。GNP では、交叉と突然変異を重複して実行すると変化の割合が大きくなるため、良い個体を得ることが難しい。したがって、今回は子個体を生成する際、トーナメント選択の後、交叉と突然変異の操作を個別に行なう。また、Fig. 5 および Fig. 6 は交叉と突然変異の例を表わしている。

2.4.1 交叉

今回、交叉には一様交叉を用いた。その手順を以下に示す。

- (1) トーナメント選択により親となる2個体を選択する。
- (2) 各ノード i ($0 \leq i \leq n-1$) を交叉確率 P_c に基づ

Fig. 3 An example of a processing node

Fig. 4 Flowchart of GNP-AC

き交叉ノードとして選択する。

(3) 選択された、対応する交叉ノードを2つの親個体間で交換する。

(4) 次世代の新しい個体とする。

Fig. 5 は処理ノード3つからなる個体の交叉例である。

2.4.2 突然変異

突然変異ではノードの接続先、ノードパラメータおよび判定ノードに入力するセンサの番号変更を行なう。手順は以下のとおりである。

(1) トーナメント選択により1個体を選択する。

(2) 突然変異操作

(a) 接続: 変異確率 P_m に基づきランダムに接続を変

Fig. 5 Crossover

Fig. 6 Mutation (connection)

更する。

(b) パラメータ (μ_i, σ_i) : 変異確率 P_m に基づきランダムに μ_i と σ_i をランダム^(注3)に変更する。

(c) センサ番号: 変異確率 P_m に基づきランダムに判定ノード内のセンサ番号 n_{i1}, n_{i2}, n_{i3} を変更する。

(3) 次世代の新しい個体とする。

2.5 学習フェーズ

GNP-AC はノード遷移中にパラメータ (V_i, μ_i, σ_i) の学習を行なう。進化と学習を用いる目的は学習による集中化探索と進化による広域探索を組み合わせることである。

2.5.1 Actor-Critic

一般的な強化学習の枠組みでは状態 s はエージェントが得ることのできる情報により決定され、行動 a はエージェントが実際に取ることのできる行動を表わしている。GNP-ACにおいて状態 s はノードを表わし、各ノードで判定・処理に用いる値 w_{ik} ($1 \leq k \leq 4$), v_i を決定することを行動とする。GNPのグラフ構造は非常にコンパクトであることに加え、ノードを状態と定義することによって、強化学習における状態数を少なくすることが可能である(本論文では50)。したがって学習を効率的に行なうことが可能になる。Fig. 7の処理ノードにおける遷移を例にして学習過程について説明する。学習

Fig. 7 An example of node transition

フェーズでは、すべての判定および処理に対して報酬を与え、この報酬に基づいてノードパラメータ (V_i, μ_i, σ_i) を更新する。つまり、処理ノードで用いる車輪の速度、および判定ノードで用いる重みを生成する確率密度関数のパラメータを適切に学習することがActor-Criticの目的である。時刻 t における詳しい更新方法は以下のとおりである。

(1) 時刻 t において、GNP-AC は2.3章において説明した方法により w_{ik} または v_i を決定する。

(2) GNP-AC は w_i または v_i を使用して判定または処理を行ない、その結果として報酬 r_t (判定ノードの場合は $r_t = 0$) を得る。そして次ノード $j = C_i^A, C_i^B, \dots$ (処理ノードの場合は C_i^A) に移る。

(3) GNP-AC では以下の手順で状態価値関数 V_i と確率密度関数のパラメータ (μ_i, σ_i) を報酬 r_t に基づいて更新する。

$$\delta_t = r_t + \gamma V_j - V_i$$

$$V_i \leftarrow V_i + \alpha \delta_t$$

$$\mu_i \leftarrow \mu_i + \beta_\mu (g_i - \mu_i) \delta'_t$$

$$\sigma_i \leftarrow \sigma_i + \beta_\sigma \left(\frac{(g_i - \mu_i)^2}{\sigma_i^2} - 1 \right) \delta'_t$$

ここで、

γ : 割引率

$\alpha, \beta_\mu, \beta_\sigma$: 学習率

$$P(g_i) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp \left[-\frac{(g_i - \mu_i)^2}{2\sigma_i^2} \right]$$

$$(g_i, \mu_i, \sigma_i) \in \{(w_{i1}, \mu_{i1}, \sigma_{i1}), \dots, (w_{i4}, \mu_{i4}, \sigma_{i4})\}$$

($ID_i = 0$ (判定ノード) のとき)

$$(g_i, \mu_i, \sigma_i) = (v_i, \mu_{i1}, \sigma_{i1})$$

($ID_i = 1$ (処理ノード) のとき)

$$\delta'_t = \begin{cases} 0 & (\delta_t = 0) \\ 1 & (\delta_t > 0) \\ -1 & (\delta_t < 0) \end{cases}$$

(4) $t \leftarrow t+1, i \leftarrow j$ とし (1) に戻る。

3. シミュレーション

提案手法の性能評価のために、Khepera robot¹⁰⁾を用いた

(注3) ただし、パラメータの取り得る値の範囲は以下のとおりである。判定ノード: $\mu_i \in [0, 1]$ (w_{i1}, w_{i2}, w_{i3}), $\mu_i \in [-1, 0]$ (w_{i4}), $\sigma_i \in (0, 1]$, 処理ノード: $\mu_i \in [-10, 10]$, $\sigma_i \in (0, 1]$

Table 1 Function Set

symbol	content
J_0	judge whether the output y_i of liner combination network is zero or more, or less than zero
P_0, P_1	determine the speed of the right and the left wheels, respectively

Fig. 8 The sensors and wheels of the Khepera robot

壁伝い問題¹¹⁾を検討する。

3.1 Khepera robot

Khepera robot (Fig. 8) は 8 個の赤外線センサにより距離を測定する。各センサは距離を 0 から 1023 の値で返し、0 は障害物までの距離が近く、1023 は障害物に近い (1023 ではほぼセンサが障害物に接触している) ことを表す。中間の値はセンサと障害物のおおよその距離を与える。右車輪と左車輪を回す 2 つのモータは別々に回転し、+10 から -10 までの整数をとる。+10 ならば全速前進、-10 ならば全速後進となる。

3.2 GNP-AC の設定

GNP-AC で用いるノードを Table 1 に示す。判定ノード J_0 は判定結果として A または B を返す。たとえば入力センサの番号が 1, 3, 5 であり、 $(w_{i1} \times x_1 + w_{i2} \times x_3 + w_{i3} \times x_5)/1023 + w_{i4} \geq 0$ である場合 ($w_{i1}, w_{i2}, w_{i3}, w_{i4}$ はノード i の確率密度関数により生成される) 判定結果は A となる。 $(w_{i1} \times x_1 + w_{i2} \times x_3 + w_{i3} \times x_5)/1023 + w_{i4} < 0$ である場合判定結果は B となる。進化と学習におけるパラメータを Table 2 に示す。次世代の個体として突然変異を用いて 355 個体、交叉を用いて 240 個体作成し、エリート個体を 5 個体残す。ノード数については処理ノードが 10 個、判定ノードはセンサ番号の初期の組み合わせ (0, 1, 2), (1, 2, 3), ..., (7, 0, 1) 8 種類についてのおおの 5 個ずつ設け 40 (8 × 5) 個としている。学習と進化のパラメータ ($\alpha, \beta_\mu, \beta_\sigma, \gamma$) は実験的に良いものを選択した。確率密度関数の平均と標準偏差の初期値は可能な範囲内でランダムに与えている。ノード間の初期接続についてもランダムに与えている。前述したように 1 ステップは 10 単位時間 (判定ノードは 1 単位時間、処理ノードは 5 単位時間を消費する) としている。10 単位時間を消費するまで GNP は判定および処理を行ない、その結果決定された車輪の速度を用いて Khepera robot を動かし、報酬を得てつぎのステップへ移行する。

Fig. 9 に今回のシミュレーションで用いた環境を示す。本環境 (実寸) は 1m × 1m で、この中に障害物 (壁) を配置している。また、本環境は著者らが作成したものであるが、ロボッ

Table 2 Simulation conditions

the number of trials	10
the number of generation	1000/trial
the number of individuals	600 (mutation: 355, crossover: 240, elite: 5)
the number of nodes	50 (judgment: 40, processing: 10)
parameters of evolution	$P_c = 0.1, P_m = 0.01$
parameters of learning	$\alpha = 0.9, \gamma = 0.1, \beta_\mu = 0.01, \beta_\sigma = 0.01$

トが同じ場所を回り続けることで適合度を上げることを避け、できるだけ速い速度で直進と右左折を行なう壁伝い行動を学習させることを目的として作成したものである。ステップ数があらかじめ決められたステップ数 (今回は 1000 ステップ) に達するとタスクを終了し適合度を計算する。報酬と適合度の計算式は文献 11), 12) を参考につぎのように設定した。

$$\text{Reward} = \frac{v_R + v_L}{20} \times \left(1 - \sqrt{\frac{|v_R - v_L|}{20}} \right) \times C, \quad (1)$$

$$\text{Fitness} = \left(\sum_{\text{step}=1}^{1000} \text{Reward} \right) / 1000, \quad (2)$$

ここで、

$$v_R, v_L : \text{それぞれ右と左の車輪速度,}$$

$$C = \begin{cases} 0: & \text{センサの値が} \\ & \text{ひとつでも 1000 以上,} \\ & \text{または} \\ & \text{すべてのセンサの値が 100 以下} \\ 1: & \text{それ以外.} \end{cases}$$

報酬はロボットができる限り速く直線的な経路で移動するほうが高くなるよう設定している。適合度は全ステップにおける報酬の平均とした。C はロボットが壁に沿って動いているときだけ報酬を得るための条件である。

3.3 比較手法

今回、比較に用いた手法は Actor-Critic なしの GNP である。この節では比較に用いた手法について簡単に説明する。

比較対象の手法は Actor-Critic を用いておらず、したがってパラメータ μ_i, σ_i を持っていない。各判定ノードは 1 つのセンサ情報を判定し^(注 4)、処理ノードは両輪の速度の決定

(注 4) 提案手法、比較手法ともに 8 個のすべてのセンサ情報を利用可能である。ただし、提案手法は各センサ入力に対する重みパラメータをもつことで複数のセンサ入力を 1 つの判定ノードで判定可能であり、比較手法は各判定ノードで 1 つのセンサ情報を判定し、複数の判定ノードの組み合わせによって複数のセンサ情報が利用可能である。したがって、両手法が得られる入力情報は同じでありその利用方法のみが異なる。

Fig. 9 Training environment

Fig. 10 Fitness curves

を行なう。判定ノードではセンサの値が閾値より大きいかわからないかで分岐が決定する。処理ノードではノード内に設定された値を車輪の速度とする。判定ノードの初期閾値および処理ノードの初期速度はランダムに与える。GNP は交叉により判定ノードの閾値、処理ノードの速度および接続先を交換し、突然変異によりこれらをランダムに変更する。世代数は提案手法と同じ 1000 世代であるがこれは同時間（同ステップ数）だけロボットを動作させたときの性能評価をするためである。

3.4 シミュレーション結果

Fig. 10 にロボットの初期位置を各世代ごとにランダムに設定した場合の適合度曲線を示している。Fig. 10 の適合度曲線は 1000 世代を 1 回の試行とする独立した試行を 10 回行ない、エリート個体の平均を示している。適合度は理論上 1.0 が最高であるが、この値はロボットの両車輪とも最高 (+10) でまっすぐ走り、常に壁に沿って走っている場合にのみ得ることができる値である。GNP-AC は GNP without Actor-Critic よりも適合度が緩やかに上昇し始めるため、初期世代では GNP-AC のほうが低い適合度になっている。これは、Actor-Critic を用いると、学習のための時間が必要となるためである。最終世代においては同程度の結果となった。ただし、壁に沿っていない行動に対する報酬は常に 0 と設定しており、最終的な適合度は両手法とも訓練環境に対しては壁伝い行動を学習できていることを示している。Fig. 11 は最終世代において最も適合度の高かった GNP-AC のプログラムの走行経路を示す。この結果より壁に沿って適切な経路を走行していることがわかる。

Fig. 11 Track of the robot controlled by GNP with Actor-Critic

Fig. 12 Test environments

つぎに訓練環境で得られた個体を未知の環境で動作させるシミュレーションを行なった。未知の環境においても適切な壁伝い行動を行なえるかを検証することが本論文の目的である。手順は以下のとおりである。

- (1) トレーニングの最終世代におけるエリート 5 個体を 1 セットとする。10 回試行するので 10 セット得られる。
- (2) 1 セットずつ初期位置をランダムに決定して適合度を計算する。
- (3) そのセットの中で最も良かった適合度をそのセットの結果とする。
- (4) step2, 3 を 1000 回繰り返す。
- (5) 1000 回 $\times$ 10 セットの 10000 個のデータを使用して t 検定を行なう。

データの信頼性を保証するため、この方法で十分な数のデータ (1000 回 $\times$ 10 セット = 10000) を生成した。また、提案手法はテストシミュレーション中は学習を行なわない。

Fig. 12 はテストシミュレーションを行なう 4 種類の環境を表わしている。テスト環境は訓練環境になかった行き止まりの配置や、細かい切返しが必要な地点などを多く設置し、訓練環境で得られた行動ルールが適切なものであるかを検証するものである。10000 回の測定結果の平均値、標準偏差および t 検定の結果を Table 3 に示す。すると、すべての環境において p 値は 0.01 以下となり GNP-AC は GNP without

Table 3 Results of t-test between GNP with Actor-Critic and GNP without Actor-Critic (test simulation)

	environment1		environment2	
	GNP with Actor-Critic	GNP without Actor-Critic	GNP with Actor-Critic	GNP without Actor-Critic
mean	0.578	0.532	0.588	0.508
standard deviation	0.023	0.018	0.012	0.014
p value	8.3×10^{-110}		0	
	environment3		environment4	
	GNP with Actor-Critic	GNP without Actor-Critic	GNP with Actor-Critic	GNP without Actor-Critic
mean	0.524	0.420	0.580	0.532
standard deviation	0.020	0.031	0.023	0.019
p value	0		1.9×10^{-120}	

Actor-Critic に対して有意性が認められた。これらの結果から、トレーニングでは GNP without Actor-Critic は GNP-AC と同程度の能力を示したにもかかわらず環境が変化すると適合度が下がるため GNP without Actor-Critic はトレーニング環境に過学習していると考えられる。以下、適合度の違いについてロボットの走行経路の点から説明する。Fig. 13 はテスト環境 4 のロボットの軌跡である。環境 4 の上部には幅の狭い袋小路がある。GNP without Actor-Critic ではその袋小路の一番奥まで入って、壁に忠実に沿っている。しかし GNP-AC は袋小路の入り口でこの通路が狭いということを判断し袋小路の中には進入していないことがわかる。同様に Fig. 14 は環境 3 のロボットの軌跡である。この環境では下部に奥の深い通路が狭くなる袋小路が設けてある。GNP-AC の軌跡は GNP without Actor-Critic のものより中に進入していないことがわかる。これは明らかに GNP-AC が早い段階で周りの状況、つまり通路が狭くなっているということを認識できているためである。環境 2 についても環境 3 および 4 と同様に袋小路が複数配置されていることに加え、壁沿いに細かい障害物が配置されており、これらを回避しながらできるだけ速く真つすぐに移動していることが適合度の差になっている。また、Fig. 15 では上記とは異なるパターンを見ることができる。Fig. 13 および Fig. 14 では、通路の幅および奥行きを判定を、左右のセンサによって行なう場合の結果である。一方、Fig. 15 の結果では、図の右部分において GNP without Actor-Critic は一番外の壁に沿って走っている。これに対し、GNP-AC は一番外の壁に沿うのではなく内側にある壁に沿って走っている。これは GNP-AC では正面のセンサと左側のセンサを活用して判断したために内側の壁に沿った経路を通ったと考えられる。この結果から、学習したパラメータを用いて複数のセンサ情報を同時に判断することで各種の環境に適応することが可能になったと考えられる。

4. 結 論

本論文では従来の GNP のアルゴリズムを改良し、センサ情報の判定に用いる重みと車輪速度の値をオンライン学習す

Fig. 13 Tracks in the environment4 **Fig. 14** Tracks in the environment3 **Fig. 15** Tracks in the environment1

る新しい方式を提案した。従来手法では各センサ情報を各判定ノードで個別に判定していたが、重みを学習する機能をもつことにより複数のセンサ情報を総合した判定を行ない、したがって環境の状態を適切に判断できるようになる。

性能評価は提案手法を Khepera シミュレータに適用し、壁伝い問題により行なった。トレーニングで得られた個体をテ

スト環境で実行した結果、提案手法は Actor-Critic を用いていない GNP に対し有意性が認められることを示した。また、判定ノードに3個のセンサ情報を用いることによりロボットが通路の幅や行先の状態を判断し通路が走行不可と考える場合があることを走行軌跡から明らかにした。

今後はロボカップサッカーなど複雑なマルチエージェントシステムへの応用を検討し、従来の進化論的計算手法、強化学習と比較を行なっていく予定である。

参 考 文 献

- 1) J.H. Holland: *Adaptation in Natural and Artificial Systems*, University of Michigan Press (1975), MIT Press (1992)
- 2) J.R. Koza: *Genetic Programming, on the programming of computers by means of natural selection*, Cambridge, Mass., MIT Press (1992)
- 3) S. Mabu, K. Hirasawa and J. Hu: A Graph-Based Evolutionary Algorithm: Genetic Network Programming and Its Extension Using Reinforcement Learning, MIT Press, 15-3, 369/398 (2007)
- 4) T. Eguchi, K. Hirasawa, J. Hu and N. Ota: A Study of Evolutionary Multiagent Models Based on Symbiosis, IEEE Trans. on Systems, Man and Cybernetics, Part B, 35-1, 179/193 (2006)
- 5) J.R. Koza: *Genetic Programming II, Automatic Discovery of Reusable Programs*, Cambridge, Mass., MIT Press (1994)
- 6) R. Sutton and A. Barto: *Reinforcement Learning — An Introduction*, MIT Press Cambridge, Massachusetts, London, England (1998)
- 7) S. Mabu, K. Hirasawa, J. Hu and J. Murata: Online Learning of Genetic Network Programming, Proc. of 2002 Congress on Evolutionary Computation, 321/326 (2002)
- 8) S. Mabu, K. Hirasawa and J. Hu: Genetic Network Programming with Learning and Evolution for Adapting to Dynamical Environments, Proc. of 2003 Congress on Evolutionary Computation, 69/76 (2003)
- 9) S. Mabu, K. Hirasawa and J. Hu, Genetic Network Programming with Reinforcement Learning and its Performance Evaluation, Proc. of 2004 Genetic and Evolutionary Computation Conference, part II, 710/711 (2004)
- 10) O. Michel: Khepera Simulator Package version 2.0: Free-ware mobile robot simulator written at the University of Nice Sophia-Antipolis by Olivier Michel. Downloadable from the World Wide Web at <http://diwww.epfl.ch/lami/team/michel/khep-sim/> (1996)
- 11) P. Nordin, W. Banzhaf and M. Brameier: Evolution of a world model for a miniature robot using genetic programming, Robotics and Autonomous Systems, 25-1-2, 105/116 (1998)
- 12) D. Floreano and F. Mondada: Automatic Creation of an Autonomous Agent: Genetic Evolution of a Neural-Network Driven Robot, Proc. of the Conference on Simulation of Adaptive Behavior (1994)

[著 者 紹 介]

間 普 真 吾 (正会員)

2001年九州大学工学部電気情報工学科卒業。03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。06年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修了。07年早稲田大学大学院情報生産システム研究科助教、現在に至る。電気学会、IEEEの各会員。博士(工学)。

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年同所副所長。91年大みか工場主管技師長。92年九州大学工学部教授。96年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科学研究科教授。02年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。

畠 山 裕 之

2005年呉工業高等専門学校専攻科卒業。07年早稲田大学大学院情報生産システム研究科修士課程修了。

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。03年早稲田大学大学院情報生産システム研究科助教、07年同研究科准教授、08年同研究科教授、現在に至る。電気学会会員。博士(情報工学)。