1C1-1 Actor-Critic を用いた遺伝的ネットワークプログラミング
Genetic Network Programming with Actor-Critic

畠山 裕之・早稲田大学

Hiroyuki Hatakeyama, Waseda University

間普 真吾・早稲田大学　　平澤 宏太郎・早稲田大学　　古月 敬之・早稲田大学

Shingo Mabu, Waseda University　　Kotaro Hirasawa, Waseda University　　Jingle Hu, Waseda University

Key Words: *Evolutionary Computation, Reinforcement Learning, Khepera robot*

「Genetic Network Programming, GNP」と称する新しいグラフ構造の進化論的計算手法が開発されている。GNP はグラフ構造で解を表すため表現能力と性能の点で優れている。さらに，タスク実行後の進化に加えてタスク実行の間学習ができるので，効率的に解を求める強化学習を取りいれた GNP（GNP-RL）も提案されている。

この論文では，GNP-RL の新しいタイプである Actor-Critic(GNP-AC)を用いた GNP を提案する。元々，GNP-RL は離散的な変数の学習を行うが，GNP-AC は連続変数の学習を目的としている。今回は提案する方法を Khepera シミュレータのコントローラに適用し，壁伝い問題を使用して評価を行う。

1. はじめに

「Genetic Network Programming, GNP」と称する新しいグラフ構造の進化論的計算手法が開発されている[1][2]。GNP はグラフ構造で解を表すため表現能力と性能の点で優れている。例えば，GNP はネットワークのノードを再利用するため，コンパクトな構造を作ることができる。しかし，従来の GNP は進化に基づいている。つまり GNP プログラムがある程度実行された後にその適合度に基づいて評価し進化を行うため，多くの試行を行う必要がある[1][2]。この問題に対処するための，進化と強化学習を結合する，GNP-RL が提案されている[3-5]。エージェントがタスクを実行している間，RL が実行されるので，GNP-RL は進化の操作以外にタスク実行中の判断と処理およびその評価によりより良い解を探索できる。GNP-RL の目的は RL のオンライン学習能力による集中探索と進化による広域探索を組み合わせることである。しかしながら，既に提案されている GNP-RL では離散的なセンサ入力(例えば，「右」のような目標の位置の方向)を判断する。そこで，GNP-AC では例えば「32 度」のような連続したセンサ入力を判断するための新しいノードを GNP に取り入れる。本論分では，強化学習の 1 つの手法である Actor-Critic を GNP に適用(GNP with Actor-Critic，GNP-AC)し Khepera シミュレータを利用して提案手法の性能評価を行う。提案手法は進化により GNP の構造，すなわち，ノードの間の接続を決定し，Actor-Critic により効率的に判定ノードのパラメータと処理ノードの車輪の速度を決定する。本論分は以下のように構成している。セクション 2 で，提案手法のアルゴリズムを説明し，セクション 3 ではシミュレーションの結果を示す。セクション 4 は結論である。

2. Genetic Network Programming with Actor-Critic

2-1. GNP-AC の基本構造

図 1 は GNP-AC の基本構造を示している。GNP-AC は複数のノードからなる有向グラフであり，ノードの再利用を容易に行うことが可能である。また，ノードはそれぞれ異なった最小基本命令を保持しておりスタートノード，判定ノード，処理ノードに分類される。スタートノードは GNP の個体がプログラムを開始するときだけ実行されるノード，判定ノードは環境から情報を受け取りその情報を判定する関数を持つノード，処理ノードは環境に対して行動を行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，図 1 内のおよび d_i および d_i^A，d_i^B…はノードの処理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，人間の脳の機能を再現すべく導入されたが，現在は GNP プログラム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。

2-2. GNP-AC の遺伝子構造

図 1 に GNP の最小単位であるノード $i(0 \leqq i \leqq n-1)$の遺伝子構造を示している。この遺伝子構造は，遺伝子の種類，および内容に関する部分(node gene)とノード間の接続に関する部分(connection gene)に大きく分けられる。node gene において，Ki はノード i の種類を示すコードである。さらに，IDi 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。Vi は状態価値関数を示し，d_i は先に述べたノードの実行遅れ時

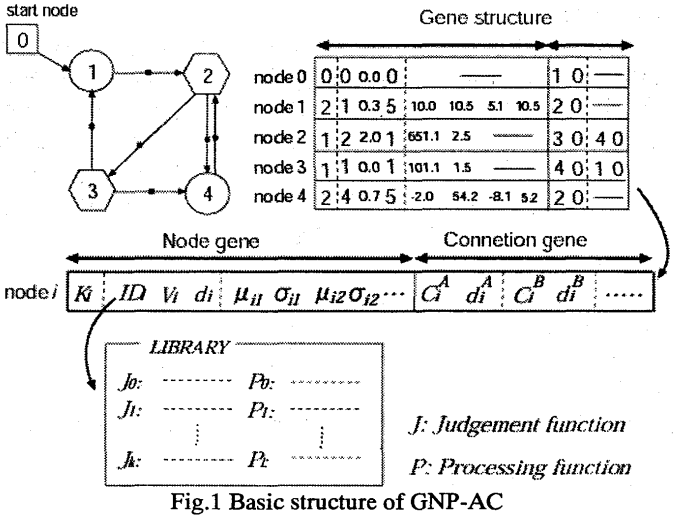


Fig.1 Basic structure of GNP-AC

間を示す。μ_{i1}, σ_{i1}, ･･･はそれぞれのノードが持つ確率密度関数のパラメータである。詳しくはノードの説明の際説明する。

また，connection gene においては，C_i^A, C_i^B・・・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・・・は上記ノード間の遷移遅れ時間を示す。このブランチ数 k は Ki，もしくは IDi に依存し，処理ノードでは k=1 となり，判定ノードでは，環境から得られる情報を基にした判定結果の数だけブランチが存在する。例えば，判定結果が「A」なら接続先は C_i^A，遅れ時間は d_i^A となり，「B」なら接続先は C_i^B，遅れ時間は d_i^B となる。

本論分では簡単化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1ステップを終了する。

2-3. GNP-AC のノード遷移規則

GNP-AC の処理はスタートノードから始まり，遷移先は判定ノードと判定結果に基づいて決定される。

<判定ノード>

現在のノード i が判定ノードなら，以下のようにして遷移先を決定する。ここで，Khepera robot を用いて評価を行うため，Khepera robot での判定内容およびノード遷移を例にとり説明する。図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，IDi はノード i で GNP が判定するセンサ番号を示している。各判定ノードには3個のセンサを最初に割り振っておく。まず，GNP は各センサ値 (x_{i1},x_{i2},x_{i3}) と，後述の確率密度関数で計算される重み $(w_{i1},w_{i2},w_{i3},w_{i4})$ を入力する。その後，x_{i1},x_{i2},x_{i3} と重みとの線形結合である出力 y_i を計算し最終的な判定を行う。図2において y_i(=0.125)≧0 であるから判定結果は「A」となり次のノード番号は C_i^A となる。もし，y_i<0 なら判定結果は「B」となる。

<処理ノード>

現在のノード i が処理ノードなら判定ノードと同様に確率密度関数によって v_{i1} と v_{i2}(Khepera robot の両輪の速度に相当する)を計算する。ここで，v_{i1} と v_{i2} は整数でなければならないため少数第1位を四捨五入して整数としている。図3では v_{i1}=5，なので右車輪の速度 v_R は5となる。左車輪のも同様の操作を行う。そして，遷移先は C_i^A となる。

2-4. 進化

図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作成する。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かなり高くなるため，良い個体を得ることが難しい。したがって，今回は子を作る際，トーナメント選択の後，別々の個体で交叉と突然変異の操作を行う。また，図5および図6に交叉と突然変異の例を示す。

<交叉>

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 $i(0≦i≦n-1)$ を交叉確率 Pc に基づ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。

<突然変異>

1) トーナメント選択により1個体を選択する。

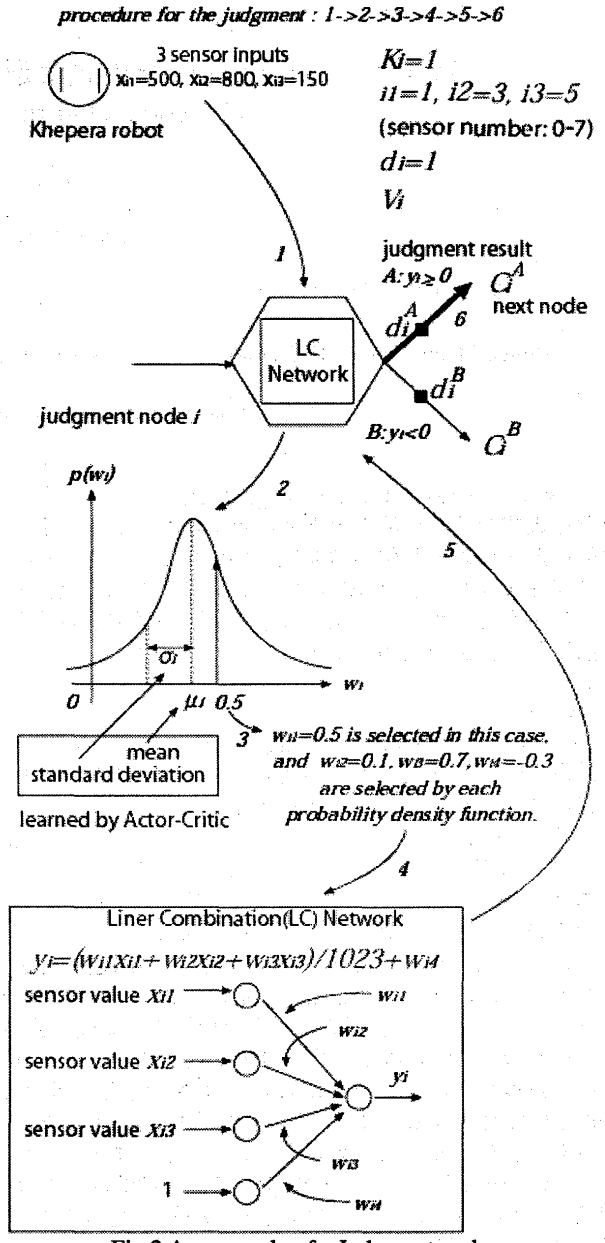


Fig.2 An example of a Judgment node.

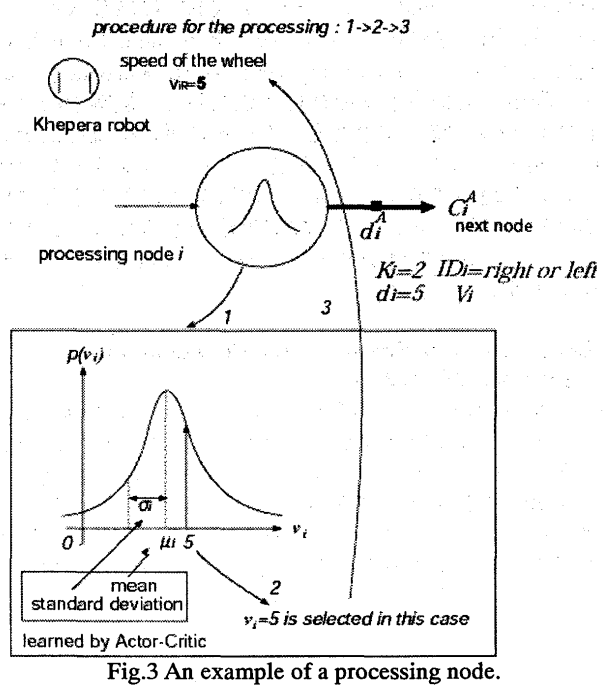


Fig.3 An example of a processing node.

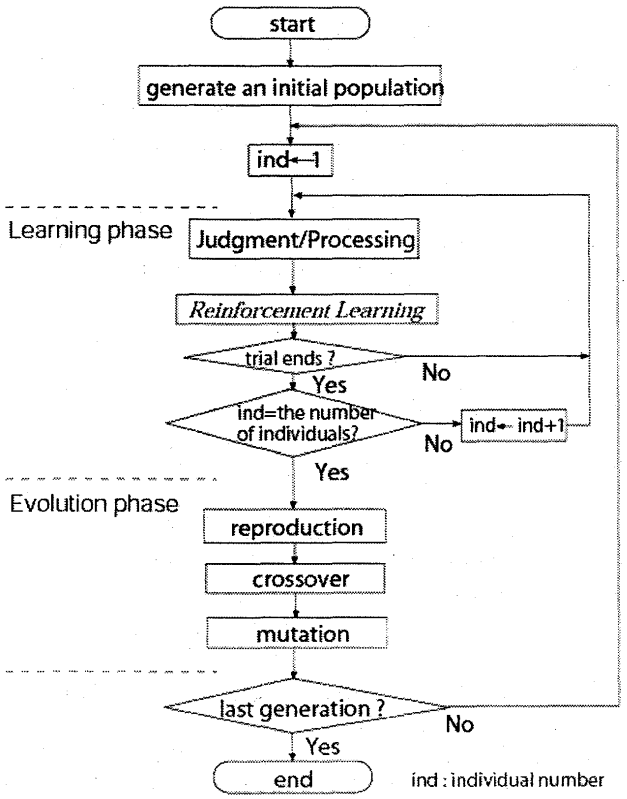


Fig.4 Flowchart of GNP-AC.

2) 突然変異操作
1)接続:変異確率 Pm に基づきランダムに接続を変更する。
2)パラメータ(μ_{ik}, σ_{ik}):変異確率 Pm に基づきランダムに μ_{ik}, σ_{ik} を変更する。
3)センサ番号:変異確率 Pm に基づきランダムに判定ノードのセンサ番号を変更する。
3) 次世代の新しい個体とする。

2-5. 学習

GNP-AC はノード遷移中に学習(V, μ_{ik}, σ_{ik})を行う。進化と学習を用いる狙いは学習による集中探索と進化による広域探索を組み合わせることである。

<Actor-Critic>

本論分では連続した変数の学習が可能な Actor-Critic[6]を GNP の学習に用いる。

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基づき，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している。GNP-AC において状態 s は現在のノードを表し行動は各ノードの値 w_i, v_i を表している。図 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。

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(Vi, μ_i, σ_i)を更新する。時刻 t における詳しい更新方法は以下の通りである。

Step1. 時刻 t において GNP-AC は 2.3 章で説明した方法で w_i および v_i を計算する。

Step2. GNP-AC は w_i または v_i を使用して判断または処理を行い，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。そして次のノード j= C_i^A または C_i^B・・・(処理ノードの場合は C_i^A)に移る。

Step3. GNP-AC では以下の手順で，状態価値 Vi と確率密度関数を表すパラメータ(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

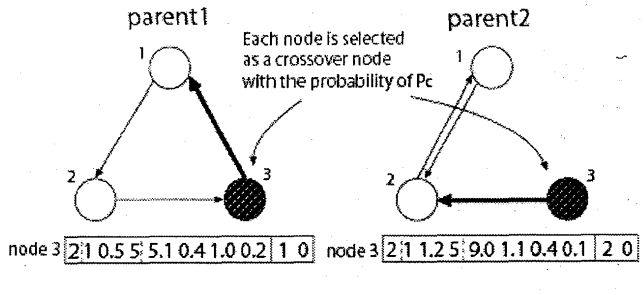
$$P(g_i) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left[-\frac{(g_i - \mu_i)^2}{2\sigma_i^2} \right]$$

$$g_i \leftarrow \{w_{i1}, w_{i2}, w_{i3}, w_{i4}, v_{iR}, v_{iL}\}$$

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

Step4. t←t+1, i←j とし，ステップ 1 に戻る。

もし判定ノードであればすべての重み w_{ik}(k=1,2,3,4)は μ_i と σ_i が更新されることによって更新される。判定結果から次のノードは C_i^A, C_i^B, ・・・の中から選択される。処理ノードの場合には車輪の速度 v_{iR} と v_{iL} が μ_i と σ_i により計算されるので μ_i と σ_i の更新により車輪速度も更新される。次のノードは C_i^A となる。



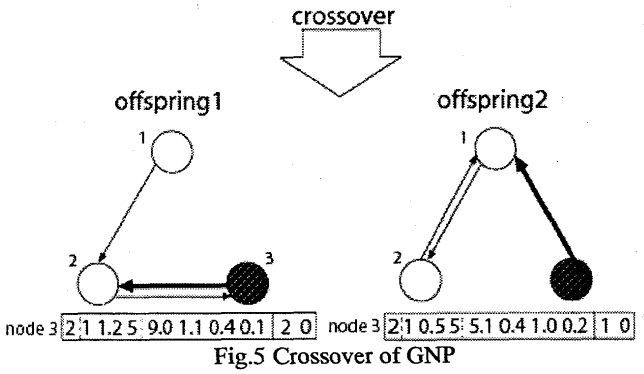


Fig.5 Crossover of GNP

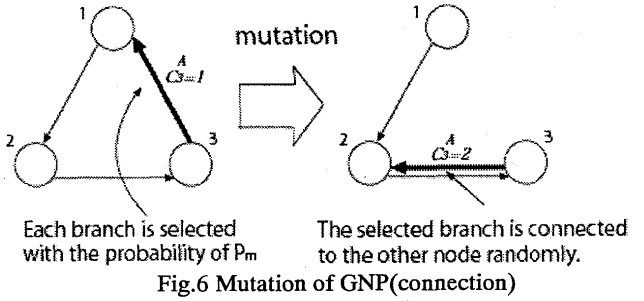


Fig.6 Mutation of GNP(connection)

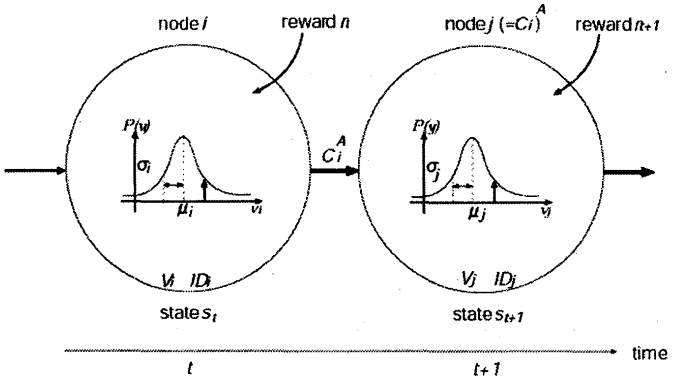


Fig.7 An example of node transition.

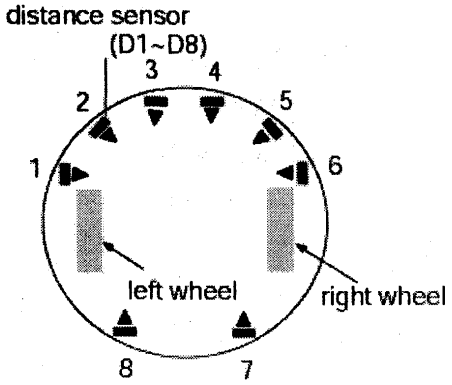


Fig.8 Khepera robot

3. シミュレーション

今回は連続変数の強化学習を評価するために，Khepera robot[7]を用いた壁伝い問題を検討する。

3-1. ロボットの設定

Khepera robot(図 8)は 8 個の赤外線センサを搭載し反射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遠く，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。2 つのモータの右車輪と左車輪は別々に回転し，+10 から-10 までの値をとる。+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 で用いるノードを表 1 に示す。判定ノード{J1, ･･･, J8}は判定結果として{A, B}を返す。例えば，J1 の判定ノードで入力センサが 1 (x_{i1})，3 (x_{i2})，5 (x_{i3})であり，$(w_{i1}x_{i1}+w_{i2}x_{i2}+w_{i3}x_{i3})/1023+w_{i4}\geqq 0$ であった場合(w_{i1}, w_{i2}, w_{i3}, w_{i4} はノード i の確率密度関数によって計算される)，判定結果は A となる。もし$(w_{i1}x_{i1}+w_{i2}x_{i2}+w_{i3}x_{i3})/1023+w_{i4}<0$ となった場合，判定結果は B となる。進化と学習におけるパラメータを表 2 に示す。次世代の個体としてトーナメント選択と突然変異を用いて 355 個体，トーナメント選択と交叉を用いて 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両輪の速度決定を行う処理ノードが 10 個，判定ノードはセンサ番号の組み合わせ{(0,1,2),(1,2,3),･･･,(7,0,1)}8 種類について各々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(α, γ)はシミュレーションにより決定した。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 は判定と処理を繰り返し環境内で Khepera robot を動かし，報酬を得て次のステップへ移行する。

Table.1 Function set.

symbol	
J1, ･･･, J8	judge whether the output yi of liner combination network (Section 2-3) is more than threshold or not Pattern{(0,1,2),(1,2,3),(2,3,4),(3,4,5),(4,5,6,),(5,6,7,),(6,7,0),(7,0,1)}
P	determine the speed of the right and the left wheels

Table2. Simulation conditions.

the number of individuals	600(mutation:355 crossover:240 elite:5)
the number of nodes	50(judgment:40 processing:10)
parameters of evolution	Pc=0.1, Pm=0.01
parameters of learning	a=0.1, γ=0.9 $ß_{\mu}$=0.01, $ß_{s}$=0.01

図 9 に今回のシミュレーションで用いた環境を示す。この環境(実寸)は 1m×1m で，この中に障害物(壁)を配置している。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)になると適合度を計算する。報酬と適合度の計算式は次の通りである。

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

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

ここで

v_R, v_L：それぞれ右車輪または左車輪

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

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 を用いていない GNP である。この章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 を用いていない。そのため μ_i や σ_i といったパラメータを持っていない。各判定ノードは 1 個のセンサ情報を持ち，処理ノードは両輪の速度の決定を行う。判定ノードの場合センサの値が閾値より大きいかそうでないかで分岐が決定する。処理ノードでは車輪の速度があらかじめノード内に設定された速度とする。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。

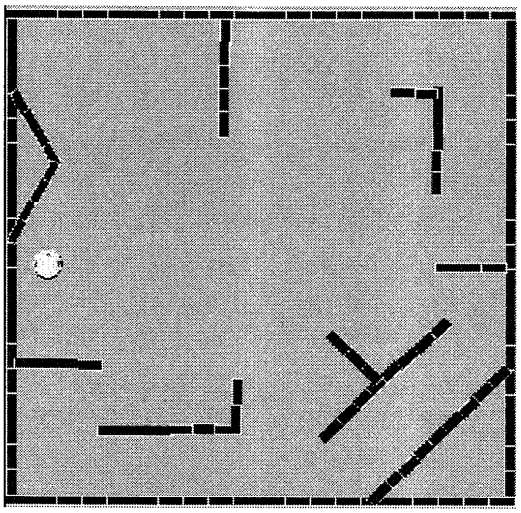


Fig.9 Environment

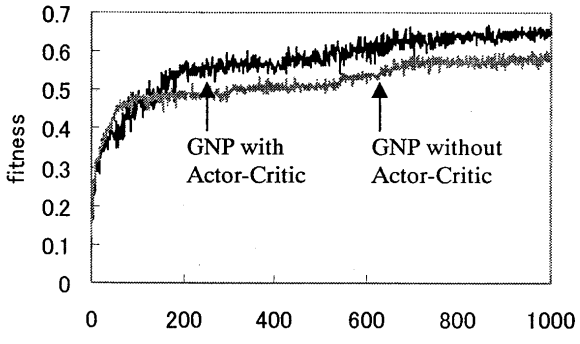


Fig.10 Fitness curves

3-4. シミュレーション結果

図 10 にロボットの初期位置を各世代毎にランダムに設定した場合の適合度曲線を示している。図 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 回行い，そ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。

また，図 11，12 に最終世代において最も適合度の高かった個体の走行経路を示す。この結果をみると GNP with Actor-Critic と GNP without Actor-Critic は同じくらいの結果を示している。これは互いに最も良い個体は同程度の適合度を示しているからである。

次に表 3 に汎化能力の結果を示す。今回用いた汎化能力は最終世代において最も良かった個体を抽出し，初期位置を毎回変更するという方法で求める。汎化能力の測定のためのシミュレーション回数は 10000 回である。表の結果は 10000 回の測定結果の平均値，標準偏差，t 検定結果である。その結果，P 値は 0.031 となり GNP-AC は Actor-Critic を用いていない GNP に対して優位性が認められた。

4. まとめ

今回，判定ノードに 3 つのセンサ情報を入力とする線形結合を使用し，その重みおよび処理ノードの車輪速度の決定に確率密度関数を適用する新しい GNP-AC の方式を提案した。提案方法は複数のセンサを単一の判定ノードで用いること

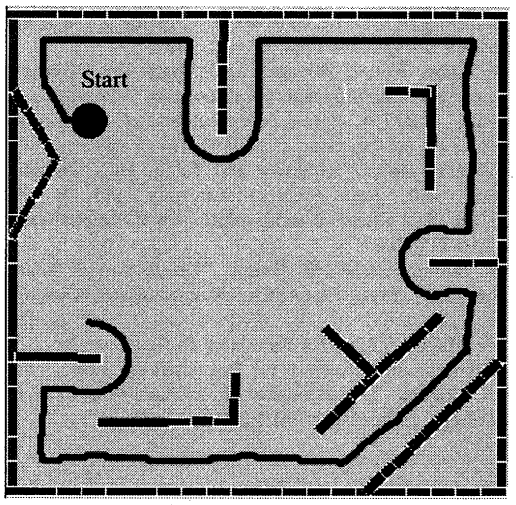


Fig.11 Track of the Khepera robot (GNP with Actor-Critic)

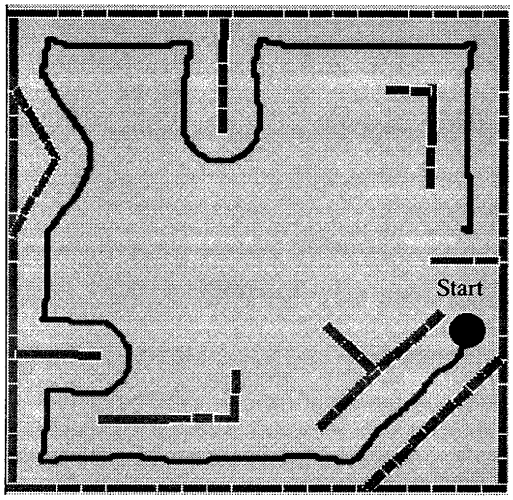


Fig.12 Track of the Khepera robot (GNP without Actor-Critic)

Table3. Result of t-test between GNP with Actor-Critic and GNP without Actor-Critic of generalization

	GNP with Actor-Critic	GNP without Actor-Critic
mean	0.499	0.493
standard deviation	0.0597	0.0486
p value		0.031

で環境を総合的に判断できるようになる。

提案した方法を Khepera シミュレータに適用し，GNP-AC は Actor-Critic を用いていない GNP より高い適合度が得られることを明らかにした。また，Actor-Critic を用いた GNP は汎化能力の点でも優れていることを明らかにした。

今回は初期位置を変更する場合の汎化能力を調べたが，環境が全く異なる場合の汎化能力についても調べる予定である。

参考文献

(1) 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, Vol.35, No1, pp. 179-193, 2006.
(2) S. Mabu, K. Hirasawa and J. Hu, ``A Graph-Based Evolutionary Algorithm: Genetic Network Programming and Its Extension Using Reinforcement Learning", Evolutionary Computation, MIT Press. (To appear)
(3) Shingo Mabu, Kotaro Hirasawa, Jinglu Hu and Junichi Murata,

``Online Learning of Genetic Network Programming", in 2002 Congress on Evolutionary Computation, pp. 321-326, 2002.
(4) Shingo Mabu, Kotaro Hirasawa and Jinglu Hu, ``Genetic Network Programming with Learning and Evolution for Adapting to Dynamical Environments", in 2003 Congress on Evolutionary Computation, pp. 69-76, 2003.
(5)Shingo Mabu, Kotaro Hirasawa and Jinglu Hu, ``Genetic Network Programming with Reinforcement Learning and its Performance Evaluation", in 2004 Gnenetic and Evolutionary Computation Conference, part II pp. 710-711, 2004.
(6) R. S. Sutton and A. G. Barto, ``Reinforcement Learning - An Introduction", MIT Press Cambridge, Massachusetts, London, England, 1998.
(7) Olivier Michel, ``Khepera Simulator Package version 2.0: Freeware mobile robot simulator written at the University of Nice Sophia-Antipolis by Olivier Michel. Downloadable from the World Wide Web at http://wwwdi.epfl.ch/lami/team/michel/khep-sim/.