進化型学習に基づく積み木問題の自律分散的解法

浜 克己, 皆川 雅章*, 嘉数 侑昇**

This paper describes a new approach to nonlinear block stacking problem in traditional AI field. In our problem setting, we see the problem as a planning in dynamic environment and attempt to solve it in a distributed manner. Each block is considered as an autonomous agent having sensors and effectors and capable of moving in the blocks world. As the behaviors of the agents change the environment, the agents are required to resolve conflict among other agents. In our approach, motions of the agents are controlled using recurrent neural network (RNN). Based on sensory inputs to RNN, the agents select their alternative actions. The RNN implemented in each agent is trained through evolutionary learning and both the structure and connection weights of the network are altered appropriately using two kinds of mutations according to the degree of success of the planning. Actions of agents are triggered by activation level which is calculated on the basis of received signals from other agents and environment. Based on the proposed methodology, computational experiments are carried out and some experimental results are shown.

Key Words: Artificial Intelligence, Robot, Neural Network, Block Stacking Problem, Evolutionary Learning, Task Planning, Autonomous Distribution, Genetic Algorithm, Recurrent NN

1. 緒 言

積み木問題は，従来よりAI分野における代表的なロボット作業計画問題の一例として取り上げられてきた．この積み木問題とは，初期状態と目標状態が与えられたときに，目標を達成するための各積み木の移動動作の並びを求めるものである．従来法では，探索空間(積み木群の状態，前提条件，オペレータ)は述語論理に基づき記号表現されていた[1]．また，与えられた問題は副問題に分解されて副目標（サブゴール）群が生成され，各目標の前提条件群が逐次満足されるまで探索を繰り返し解を得るものであった．これらの探索はプラン生成の結果とみなせるが，状態に基づく方式と計画に基づく方式に大別され[2]，戦略的視点からは手段—目標分析，階層的計画，最小拘束戦略などが提案され，成功をおさめてきた．

Katsumi HAMA, Masaaki MINAGAWA*,
Yukinori KAKAZU** 函館工業高等専門学校,
*札幌学院大学社会情報学部, **北海道大学工学部

しかし，従来法は，問題領域やタスクの持つ特有の性質への強い依存性，問題環境の変化に対する柔軟性の低さなどが指摘されている．そのため，問題特有の機構を有するプランナーが必要とされ，そのプランナーには対象問題に対する完全かつ正確な知識を持つことが要求された．

一方，このような解探索の問題領域への依存性を軽減することを1つの目的として，近年，Evolutionary Computation(EC)[3]，あるいは Evolutionary Algorithms(EA)[4]の分野が急速に発展している．この分野には，その基礎となる3つの代表例がある：Genetic Algorithms(GA)，Evolution Strategies (ES)，Evolutionary Programming(EP)．歴史的に見ると，古典的なGAでは，領域に独立な様式で作用する交差や突然変異の遺伝子操作を伴う，固定長の2進表現による問題に依存しない遺伝子表現を使用した[5]．この一般性は，広い範囲のアプリケーションに対し，ロバストな適応システムの設計に関する強調にも反映された．ESは，難しい実数値パラメータの最適化問題を解くことができるシステム構築に強い焦点を置くことによって発展された[6]．自然表現は実数値の遺伝子ベクトルであり，主に実数値パラメータを摂動するために設計された突然変異オペレータにより操作された．また，これも一般性が初期の中心テーマであったEPが取った方向は，個体群を環境の刺激に対応できる有限状態マシンとして表現し，構造および挙動の変化に影響するようにオペレータ（主に突然変異）を展開する考えであった[6]．これら3つのアルゴリズムの類似および差異をまとめたものを表1に示す[4]．これらECに関する従来の問題領域は，評価関数が時間不変で，各個体の評価が集団内の他のメンバーから独立に計算できる範囲であった．これに対し，現在解が要求される重要な3つの問題クラスがある．1つ目は，適応度がEAのあらゆる探索活動に独立なエージェントの振る舞いで定義される場合の自律的に変化する関数，2つ目は，ある特定の問題を集合的に解く個体グループを進化することが目標の協調的振る舞いの進化，そして3つ目は，適応度関数が進化過程自身によって直接影響される場合の生態学問題である．また，進化技法をある特定の問題クラスに適用する際に必要な決定として，EAによって探索されるべき空間の仕様がある．これは，問題空間内の点群と内部表現空間内の点群との間の写像を定義することによって達成されるが，このとき，一般的表現から問題特有の表現までの間で，表現形式をいかに選択するかが問題となる．集団進化のダイナミクスのモデリングに関しては，まず集団サイズの動的な調整が必要となる．さらに，一

表1 進化計算法の特徴比較

	GA	*ES*	*EP*
Representation	Binary-valued	Real-valued	Real-valued
Self-adaptation	None	Standard deviations and covariances	Standard deviations and correlation coefficients
Fitness	Scaled obejective function value	Objective function value	(Scaled) obejective function value
Mutation	Background operator	Main operator	Only operator
Recombination	Main operator	Different variants, important for self-adaptation	None
Selection	Probabilistic, preservative	Deterministic, extinctive	Probabilistic, extinctive

定数の最悪個体をいかに除くかという個体削除戦略，どの個体が親となって子孫を残すかという親個体選択問題，そして適応的突然変異や適応的交差実現のための再生メカニズムが，探索/利用の問題のバランスに影響を与える．このように，ECはその急速な変化に伴い今後解決すべき課題も多いが，それ以上に魅力的な数々の特徴を有する．

これらを背景に，積み木問題の解法としてGAに基づく方式もいくつか試みられている．Kozaのアプローチでは，LISPコードのS式が人工染色体として表現され，ブロック移動に関してそのプログラムが探索されるが，一般的な例は示されていない[7]．平野は，ブロック群の各状態をノード，実行可能な遷移をグラフの有向エッジとして表し，状態遷移グラフに基づく染色体表現を使用した[8]．各エッジに割付けた番号が人工染色体の遺伝子としてコード化され，最小遷移系列が解として探索されるが，このアプローチでは，ブロック数の増加に伴いグラフ生成自体が困難な問題となる．皆川らは，問題を座標空間で表現し，人間の試行錯誤過程を模倣する探索メカニズムを導入したヒューリスティックな探索手法を試みた[9]．ブロック移動の基本的な動きをプロダクションルール(基本ルール)として与え，この並びを染色体としたメタルール長の最小化問題として扱った．この方式では，多くのGAオペレータが必要である．

これに対し，本研究ではNeural Networks (NN)とECを組合せるこれらとは異なる方式でのアプローチを試みる．NNの構造決定を目的として，その実現のためにGAを用いる試みは以前より盛んに行われている．従来法の多くは，フィードフォワード型の固定アーキテクチャに対し，Back Propagation (BP)に代表される勾配降下方式の最適化手法を導入して，そのユニット間の荷重の変更にのみGAを用いていた．その後，BP学習ルールを利用しながら，ネットワーク構造と結合荷重を含むパラメータの両方を対象に，その最適化のためにGAを使用する方式が提案された[10]~[13]．また，1991年以降，それまでのフィードフォワード型のアーキテクチャに加え，時間依存問題に対して有望なリカレント型をはじめとする相互結合型のアーキテクチャについても，進化方式によるアプローチが試みられた[14],[15]．その後もGAをベースとする多くの方式が提案されているが[16],[17]，最近GAを用いる方式の問題点として，ネットワークに対する直接的操作のしにくさや再結合による個体性の軽視の点が指摘され，突然変異オペレータのみを用いるEPやESを用いる方式が提案されている[18],[19]．

これより，本研究ではEPをベースとする方式を導入する．即ち，ここで，個々の積み木を自律的な移動物体（エージェント）と見なし，人工生物における捕食行為の進化方式にヒントを得て[20]，積み木問題を自律分散的に解く新しい方法を提案する[20],[21],[23]．各エージェントはそれぞれセンサ，エフェクタなどを有し，その動作はリカレント型のニューラルネットワーク（RNN）で表現される挙動関数にしたがって制御される．各エージェント内に実現される同一のRNNはその状況に応じて異なる行動を取る．ネットワーク構造と結合荷重は，プランニングの成功の度合いに応じて進化的に学習される．それぞれのエージェントは，各時点において環境や他のエージェントからの入力情報により計算される活動レベルによって最も望ましい行動を獲得する．また，競合解消のために行動の実行要求度が最大のエージェントのみがその動作を実現させる．問題の解はこれらの行動列として求められる．

以下，本手法の定式化と数値実験を通して，その適用可能性と学習性能について考察する．

2．問題空間の記述

以下の前提を置く．

(1) 対象問題空間を2次元平面空間とする．

(2) 各積み木の大きさは同一とする．

(3) 各積み木の状態はその中心座標で表す．

対象となる問題空間を，記号空間ではなく座標空間を用いて表現する(図1)．これに基づき，ある時間ステップにおける状態 q_i を以下のように記述する．

$$q_i=\{b_i,\ h_i,\ v_i\} \tag{1}$$

b_i：積み木（$b_i \in B$，B：積み木の集合）

h_i：水平位置($h_i \in H$, H：水平位置集合)

v_i：垂直位置($v_i \in V$, V：垂直位置集合)

積み木の状態遷移関数 δ は，次のようにある状態 q_i を他の状態 q_j に写像する．

$$q_j = \delta(q_i,\ m),\ q_j,\ q_i \in Q \tag{2}$$

Q：状態の有限集合

$M=\{m\}$：行動パターン系列

$m = \sigma_1\sigma_2\cdots\sigma_i\cdots\sigma_n (0 < n < T$，$T$：最大時間ステップ)

$\sigma_i=\{b_i,\ a_i\}(b_i \in B,\ a_i \in A$，$A$：実行可能行動集合)

この状態遷移は，行動パターンにしたがって各ステップ毎に行われる．これより，積み木問題は，与えられた目標状態の到達に伴う状態遷移を終結するまでの行動パターン系列を見つける問題とみなされ(図2)，以下のように記述される．

$$Find\ M=\{m \mid \delta(q_0, m) = q_G\} \tag{3}$$

q_0：初期状態（$q_0 \in Q$）

q_G：目標状態（$q_G \in Q$）

このような行動のコーディングにおいて，提案するアプローチでは従来法と次のような違いがある．

1）各エージェントはサブゴールを持たない．

2）行動系列はトップダウン的に与えられるのではなく，分散処理の結果の一例としてコード化される．

また本研究は，座標(h, v)における積み木の有無に応じてその名前または空集合(ϕ)を返す関数 *assign* を

$$assign(h,\ v)=\begin{cases} b(b \in B) \\ \phi \end{cases} \tag{4}$$

と定義し，座標空間における物理的な整合性を保証するために，次のような各条件を設定する．

1）干渉なしの条件

同一位置は複数の積み木で占有されないこと．

$$for\ \forall(p_i,\ p_j) \in q (p_i \neq p_j) \Rightarrow h_i \neq h_j,\ v_i \neq v_j \tag{5}$$

$p_i=(X,\ h_i,\ v_i) \in q$, $p_j=(Y,\ h_j,\ v_j) \in q$

X，Y：積み木の名前

2）安定条件

各積み木は台か他の積み木の上に置かれること．

$$for\ \forall\ p_i=(X,\ h_i,\ v_i) \in q \Rightarrow assign(h_i,\ v_i-1) \neq \phi\ (v_i > 1) \tag{6}$$

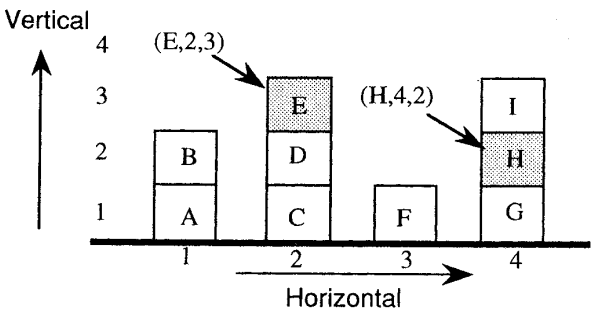


図1 積み木の状態表現

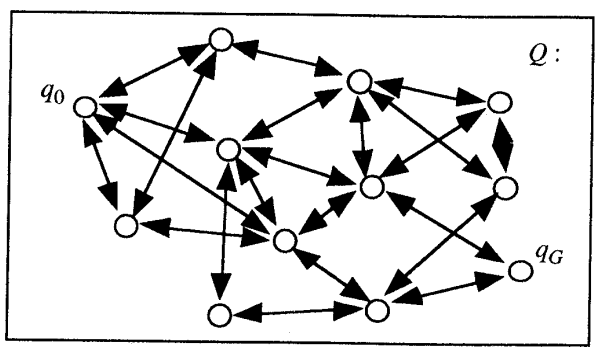


図2 有限状態遷移

3）目標達成条件

特定の積み木の目標は，その下に置かれるべき積み木の目標がすべて達成されている時にのみ達成されること．

$$for\ \forall\ w(0 < w < v) \Rightarrow Y = Z \qquad (7)$$

$$Y = assign(h,\ w)((Y,\ h,\ w) \in q)$$

$$Z = assign(h,\ w)((Z,\ h,\ w) \in q_G)$$

3．自律エージェントと挙動関数

各積み木は，センサ，エフェクタ，並びに通信機能を持つ自律エージェントとして見なされる(図3)．また，エージェント内部にはリカレントニューラルネットワーク（RNN）を所有し，このRNNで表現されるエージェントの挙動関数は，センサ入力を受けてエフェクタ（モータ）出力を生成する．すべてのエージェントが同一のRNNを所有するが，それぞれは動的に変化する環境状況などに応じて異なる行動をとることができる．ネットワークの入力は，積み木が利用できる基本的なセンサ入力であり，以下のようにコード化した5つの情報が各時間ステップで使用される．

$$S = \{s_1,\ s_2,\ s_3,\ s_4,\ s_5\} \qquad (8)$$

$for\ i \in B,\ s_j = \{0,\ 1\}$

$s_1 = 1$：*if the position on block i is cleared*

$s_2 = 1$：*if the goal of block i is cleared*

$s_3 = 1$：*if the position under the goal of block i is filled*

$s_4 = 1$：*if block i is at the goal position*

$s_5 = 1$：*if block i is requested to remove*

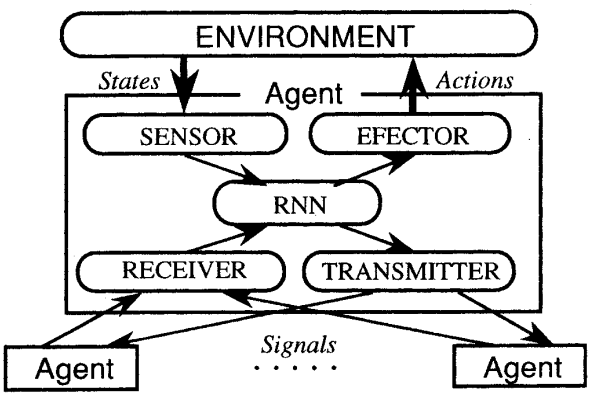


図3 自律エージェントの構造

ネットワークの出力は，単一のエージェントが実行可能な基本行動群である．a_1 から a_4 に相当する4つの出力ユニットは，それぞれ以下のようなユニークな行動に対応し，各エージェントは *Winner Take All* の考えにしたがって，対応する出力ユニットが活性度最大となる行動を唯一つ獲得する．

$$A = \{a_1,\ a_2,\ a_3,\ a_4,\ a_5\} \qquad (9)$$

a_1：*move to the goal*

a_2：*move to work area*

a_3：*request to remove*

a_4：*no operation*

a_5：*request to perform action*

ここで，a_1，a_2 は実際の移動を伴う行動であり，a_3 は行動対象となる積み木のすぐ上の位置，および目標位置に存在する積み木に対して，その場所から移動することを要求する．これは入力の s_5 に対応する．また，a_4 はネットワーク学習は進行するが，エージェントは何もせずに現在位置に留まることを意味する．これらに対し，a_5 は各エージェントが行動を実行するためのトリガとして使用され，各ステップにおけるエージェント間の競合解消を行うためにも重要な出力である．分散方式の考えからは，その決定においてエージェント間に通信機能などを持たせる必要があるが，ここでは単にそれらの出力値を比較し，最大の値を持つエージェントに実行権を与えた．

また，作業領域（*work area*）とは，目標位置に指定されていない水平座標上で，前述の安定条件を満たす空間位置を示し，以下のように表現される．

$$q_j = \{(X,\ h_j,\ v_j) \mid h_i \neq h_j,\ \min_{v_j} assign(h_j,\ v_j) = \phi\} \qquad (10)$$

q_j：作業領域の候補状態

$q_i = (X,\ h_j,\ v_j)$：現在の状態

これらの入出力情報を関係づける適切な挙動関数を実現するために，我々はユニット間にランダムな結合を許し，動的な非線形ユニットを構成することのできるRNNを使用する．そのアーキテクチャは以下の条件にしたがって構成される（図4）．

1）入力ユニットへの他のユニットからのリンク，並びに出力ユニットからの接続リンクは存在しない．

2）ユニット間のリンクには実数値荷重を使用する．

3）各ユニットは非同期的に状態遷移を行う．

隠れユニットの個数は一定とするが，その中にリンクを全く持たない孤立ユニットが存在する場合がある．入力を除く各ユニットの出力は，結合しているすべてのユニットからの重みづけ和のある関数として算出される．時刻$t+1$におけるユニットu_iの状態$R_i(t+1)$は，以下のようなシグモイド関数によって計算される．

$$R_i(t+1)=\frac{1}{1+exp\{-R_i(t)\}},$$

$$R_i(t)=\sum_{j=1}^{N_i} w_{ji}R_{ji}(t) \tag{11}$$

$R_i(t)$：時刻tにおけるユニットu_iの状態

$u_{ji}(1\leqq j\leqq N_i)$：$u_i$に信号を出力する任意ユニット

$R_{ji}(t)(1\leqq j\leqq N_i)$：時刻$t$のユニット$u_{ji}$の状態

$w_{ji}(1\leqq j\leqq N_i)$：$u_{ji}$から$u_i$への結合荷重

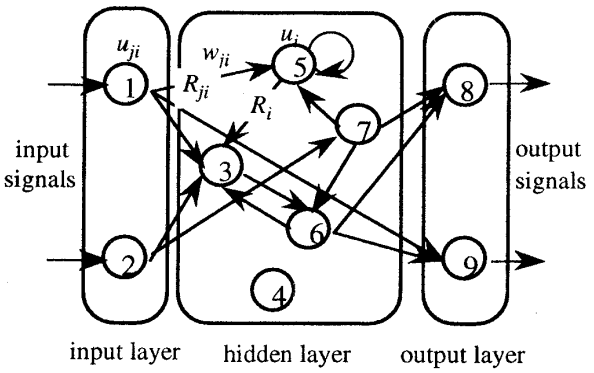


図4 積み木エージェントの外部インタフェース

4．進化学習

各エージェント内に実現されるRNNの進化学習は，EPに基づいて行われる．図4に示すようなRNNの集合が母集団を形成するが，それらの初期生成手順は以下の通りである．

1）前述のネットワーク条件に基づき，一定の確率で各ユニット間に接続の有無を設定する．

2）ネットワーク構造が選択された後，ユニット間に存在するすべてのリンクには，$[-W,\ W]$の範囲で一様ランダムに荷重が割り当てられる．

これらのネットワーク群に対して，以下の各オペレーションが適用される．

4.1 淘汰と再生

最終的には，初期状態から目標状態までの最適な行動系列$M=\{m\}$を求めることが要求される．そこで，各RNNの有用性を測るための適合性値には，目標状態の達成度に加え，実際に移動を伴う真の行動数も考慮する．前述の目標達成条件を満足しながら，それぞれの適合性値V_ηを以下のように与える．

$$V_\eta=c_1\frac{G_\eta}{N_B}+c_2\frac{Z_\eta}{N_S} \tag{12}$$

η：$RNN(\eta\in P,\ P$：母集団$)$

G_η：水平位置毎に台上から上方向の積み木を順次調べ，目標状態を満足し続ける積み木の総数

N_B：対象となる積み木の総数

Z_η：移動に基づく真の行動数

N_S：最大時間ステップ数

c_1，c_2：選択係数$(c_1+c_2=1)$

世代交代毎に各RNNが評価され，適合性値の低い一定の割合を淘汰し，生き残った親から子の生成を行う．また，EPをネットワーク推定に使用している一例としての文献(8)を参

考に，ここでは子の生成の際に交差は行わず，リンク荷重とネットワーク構造に対応する2種類の突然変異を用いる．手順は以下である．

1）生き残った親を淘汰された部分にコピーする．

2）実行されるべき突然変異の重大度を決定する．

3）コピーされた親に以下の突然変異を施す．

4.2 突然変異

(1) 突然変異の重大度

対象となる親 η に対する突然変異の重大度として，以下に示すアニーリングに類似の温度 T_η を設定する．

$$T_\eta = 1 - \frac{V_\eta}{V_{max}} \tag{13}$$

V_{max}：対象タスクに対する最大の適合性値

これより，初期段階のように低い適合性値を持つネットワークが高い確率で突然変異を受けることになる．

(2) パラメトリック突然変異

パラメトリック突然変異は，ネットワーク内のパラメータ（リンク荷重）を変更する．ガウス性雑音によってネットワークの各荷重 w を摂動することにより達成され，実際の荷重修正は以下のように行われる．

$$w = w + N(0,\ \alpha T_\eta) \forall\ w \in \eta \tag{14}$$

α：比例定数

$N(\mu,\ \sigma^2)$：ガウス型ランダム変数

(3) 構造的突然変異

構造的突然変異は，ネットワーク内のユニット間のリンクの存在(ネットワーク構造)を変更する．この突然変異は，親から子への挙動的な連続性を保存するために用いられる．このため，入出力ユニットとの結合状態や変更するリンク数などについても十分検討する必要があるが，ここでは簡単のためにリンクの存在を変更する確率を以下のように修正することとする．

$$P_\eta = P_m T_\eta \tag{15}$$

P_m：基本突然変異率

5．計算機実験

以下では，提案した方法論に基づき行った実験結果について示す．特に，非線形計画問題に対する分散解法や進化学習の有効性，並びに小さな問題で獲得される解を大きな問題の解探索に利用することの利点について検証する．前述のように，各ステップにおいて，各々の積み木エージェントは4つの実行可能な行動の中からネットワーク出力が最大となる行動を1つ選択するものとする．しかし，ネットワーク学習が不十分な段階では，上に積み木が乗っている場合でも移動行動が選択される可能性があるため，最小限の禁止行為を設定した．また，競合解消のために，各時点では行動の実行要求出力が最大の積み木のみがその動作を実現するものとする．ただし，この積み木が禁止行為を活性化している場合には，次の活性レベルを持つ積み木が行動を起こすものとする．適合性値は最大時間ステップまでの目標達成率等で与えられる．それぞれの問題で共通となる実験条件を以下に示す．

- RNN の初期生成
  ユニット間の結合確率：0.5
  結合荷重の範囲：$W = 1.0$
- ユニット数：入力 = 5，隠れ = 10，出力 = 5
- 垂直位置数：V =制限なし
- 集団サイズ(ネットワーク数)：50
- 淘汰率：0.5，基本突然変異率：$P_m = 0.3$
- パラメトリック突然変異比例定数：$\alpha = 1.0$
- 適合性値選択係数：$c_1 = 0.9$，$c_2 = 0.1$

まず，非線形を含む小さな問題について実験を行うため，図5に示す3つの例題を用いた．それぞれの例題で使用した水平位置数と最大時間ステップ数を表2に示す．水平位置数の違いによる目標達成までの世代数や行動

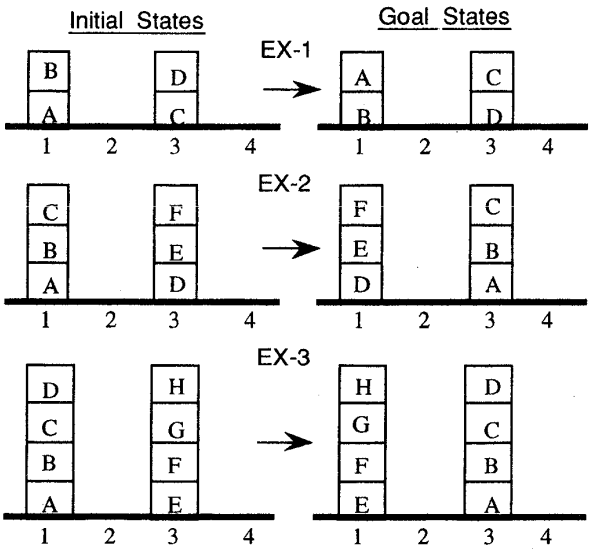


図5 積み木問題テスト例題(1)

表2 実験条件(1)

	The number of horizontal locations	The maximum time steps
EX-1	4	25
EX-2	4	100
EX-3-1	4	150
EX-3-2	5	150

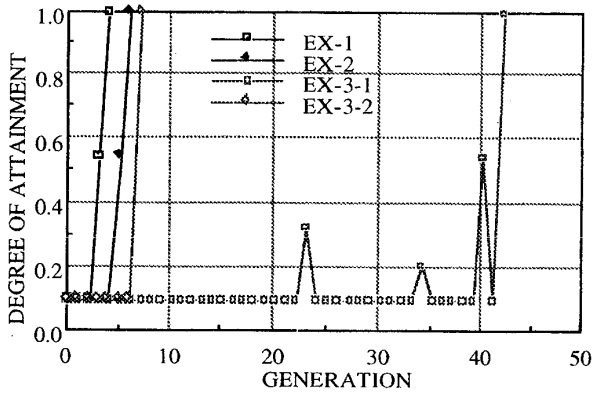


図6 適合性値の変化（例題 EX1～EX3）

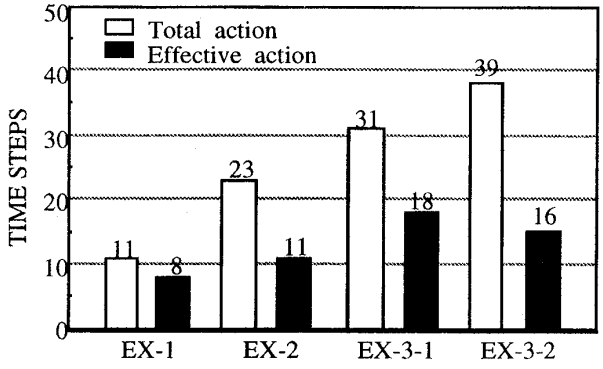
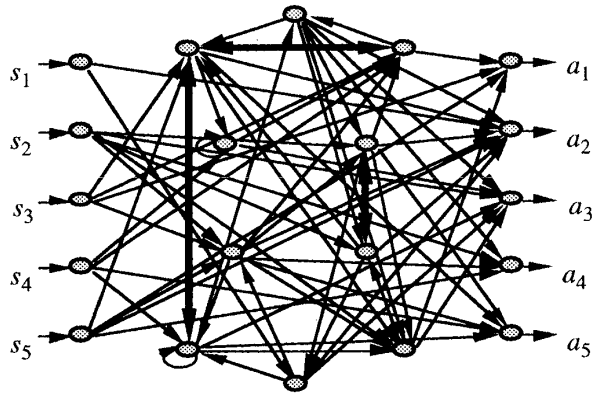


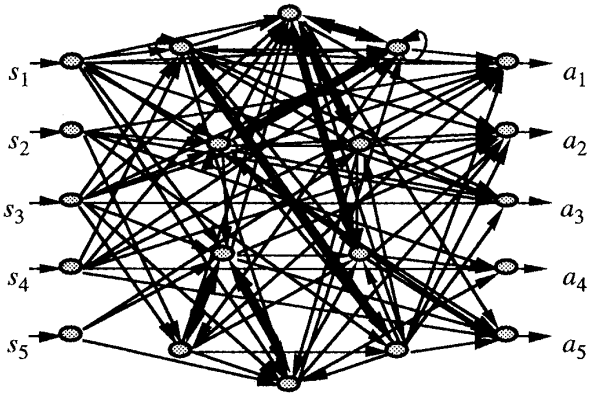
図7 目標達成までの時間ステップ数(図6の例題)

数の差異を調べるため，EX-3についてはその値を2通り用意した．それぞれの例題に対し，設定した最大時間ステップに達するまでの目標達成度を求めた．数回の試行による結果の中から最良の値のみを図6に示す．図中，横軸は世代数，縦軸は目標達成度を含む適合性値である．また，そのときの目標状態までの全行動数，および有効行動数を図7に示す．ここで，有効行動数とは，特定の積み木が連続的に行動を行った場合の途中の行動（行動前の状態に戻った時は全行動）を取り除いたものである．これらの結果より，水平位置数の増加は，作業領域の選択範囲を広げて目標状態までの学習を速めるとともに，よりよい行動系列の獲得に寄与していることがわかる．なお，EX-1，EX-2，EX-3の最適な行動数はそれぞれ，8，11，15である．また，学習によるネットワーク構造の成長過程を確認するため，EX-1の結果について初期状態と目標状態におけるネットワークの結合状態を図8に示す．ここで，太線は双方向結合を表している．この結果より，目標状態では実際に移動を伴う行動出力 a_1，a_2 へのリンク結合数が増加し，これらの行動が選ばれ易い状況になっていることがわかる．

次に，積み木の数をさらに増加させ，競合が発生し易いより複雑な環境での実験を行った．図9は，このような大きな問題として用意した追加例題である．ここでは，解探索に対し，水平位置数の違いのほかに，事前のより小さな問題で獲得された解の利用の有無による影響について調べた．これらの組合せ条件を表3に示す．図10と図11は，それぞれEX-4とEX-5に対して，条件の違いによる学習結果を示している．図中の各軸は図6と



(a) Initial state



(b) Goal state

図 8 ネットワークの結合状態の変化

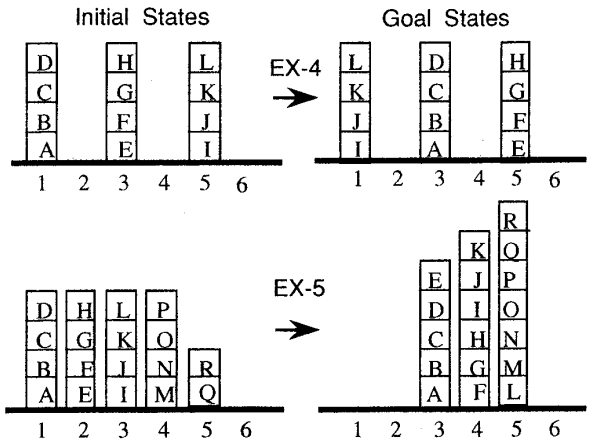


図 9 積み木問題テスト例題(2)

同様である．これらの結果より，追加例題の解探索では，小さな問題で獲得された解の利用が，水平位置数の増加に比べてより効果的

表 3 実験条件(2)

	The number of horizontal locations	The maximum time steps	Utilization of the solutions obtained in small scale problems
EX-4-1	6	250	Non
EX-4-2			EX-3
EX-4-3	8		Non
EX-4-4			EX-3
EX-5-1	8	400	Non
EX-5-2			EX-4
EX-5-3	10		Non
EX-5-4			EX-4

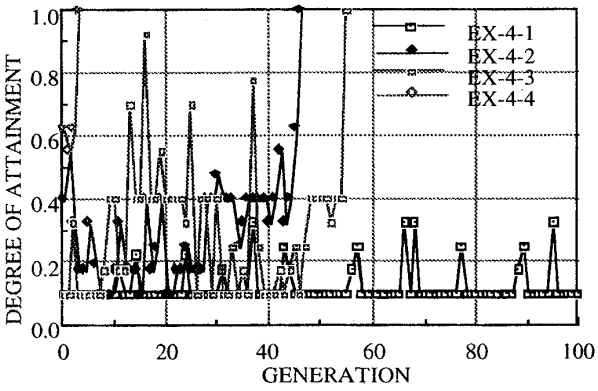


図 10 適合性値の変化（例題 EX-4）

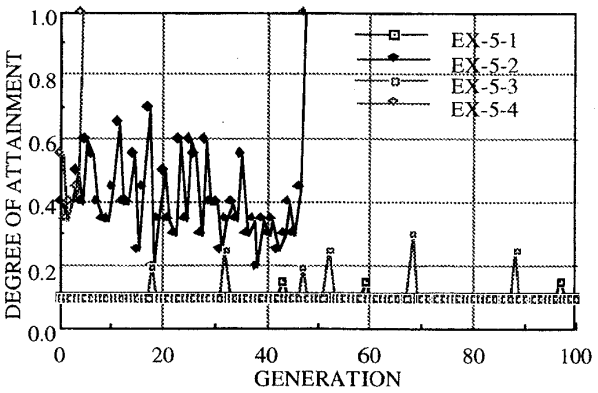


図 11 適合性値の変化（例題 EX-5）

であることを示している．また，図 12 と図 13 は，それぞれの場合に対する目標状態までの全行動数と有効行動数を表している．EX-5 については，水平位置数の増加による座標空間の広がりが，かえって目標状態までのステップ数を増加させている．この結果からも，大きな問題の解探索には，小さな問題で獲得される解の利用が有効であることを示している．

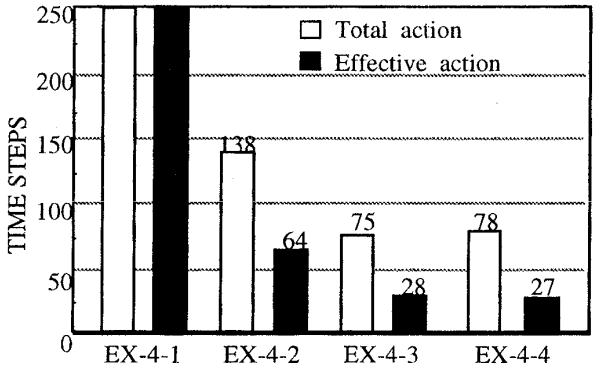


図12 目標達成までの時間ステップ数(例題 EX-4)

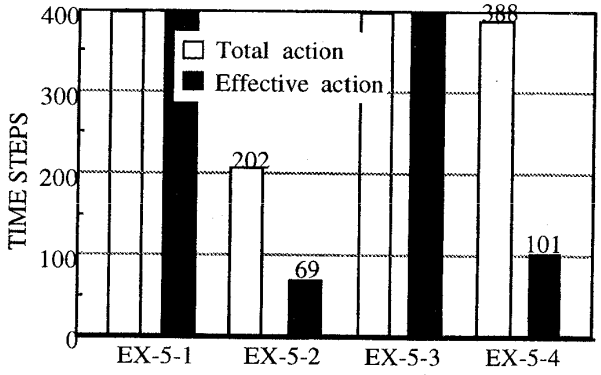


図13 目標達成までの時間ステップ数(例題 EX-5)

6. 考 察

実験結果より以下のようなことが推測される.

1) 小さい問題に対しては解の収束が速く, 適合性値が低い状態から急激に変化して目標状態をクリアしている様子がうかがえる. これは, ネットワーク学習が不十分な段階でも作業領域のランダム性がうまく働いているものと考えられる.

2) 積み木の増加とともに, 求められた有効行動数は最適行動数に比べて多くなっている. この原因の1つとして, 前述のように, 最小限の禁止行為以外はすべての行動を許したため, 実際に自分の目標位置が空いている場合でも作業領域に移動する可能性があることが考えられる.

3) 一般に, 水平位置数の増加は解の収束を速め, 同時に目標状態までのステップ数を減少させる. しかし, 作業領域のランダムな取扱いでは, それが必ずしも有効に使われず, 効果が低い場合も起こり得る.

4) ランダムな初期集団からの学習では解が得られにくい大きな問題に対しても, 事前のより小さな問題に対する適切な解を選択し利用することによって, 解探索が可能となることがわかる.

7. 結 言

以下をもって本論文の結論とする.

1) ロボット作業計画の1つである積み木問題を解くために, プランナーなしの自律分散的なアプローチを提案し, 計算機実験によりその適用可能性と学習性能を確認した.

2) 行動獲得のために積み木エージェント内部に実現される挙動関数をRNNで表現し, その構造がEPに基づく進化学習によって適切な形に訓練されることを示した.

3) 小さな問題で獲得された適切なネットワーク情報を, 大きい問題の初期集団内に適当に設定することにより, 問題解決における高い学習性能を持つことを示した.

4) 大規模問題, 3次元異形形状の場合については, 近々に結果を報告したい.

5) 行動結果のみに基づいて教師なし学習を行う自律分散方式の有効性が確認された.

参考文献

(1) 岩井・片井・椹木・坂口・福森: 知識システム工学, 計測自動制御学会編, pp.100-200 (1991).

(2) Minton, S.: Machine Learning Methods for Planning, Morgan Kaufmann Publishers, pp.1-31 (1993).

(3) De Jong, K. and Spears, W.: On The State of Evolutionary Computation, Proc. of ICGA-93, pp.618-623 (1993).

(4) Back, T.: Evolutionary Algorithms, Comparison of Approach, Computing with Biological Metaphors, Chapman & Hall, pp.227-243 (1994).

(5) Holland, J.H.: Adaptation in Natural and Artificial Systems, The University of Michigan Press, Ann Arbor (1975).

(6) Michalewicz, Z.: Genetic Algorithms+Data Structures=Evolution Program, Springer verlag (1993).

(7) Koza, J.R.: Genetic Proguramming, On Programming Computers by means of Natural Selection and Genetics, MIT Press (1992).

(8) 平野：積み木問題を遺伝的アルゴリズムで解く，インターフェース，CQ出版社，No.177, pp.108-135 (1992).

(9) 皆川・嘉数：ロボット作業計画の生成に関する研究——GAによるヒューリスティックアプローチ——，日本機械学会第69期全国大会講演会講演論文集，Vol.C, pp.594-596(1991).

(10) Harp, S.A., Samad, T. and Guha, A.: Towards the Genetic Synthesis of Neural Networks, Proc. of ICGA-89, pp.360-369 (1989).

(11) Miller, G.F., Todd, P.M. and Hegde, S.U.: Designing Neural Networks using Genetic Algoritms, Proc. of ICGA-89, pp.379-384 (1989).

(12) Schaffer, J.D., Caruana, R.A. and Eshelman, L.J.: Using Genetic Search to Exploit the Emergent Behavior of Neural Networks, Emergent Computation, pp.244-248 (1990).

(13) Wilson, S.W.: Perceptron Redux, Emergence of Structure, Emergent Computation, pp.249-256 (1990).

(14) Torreele, J.: Temporal Processing with Recurrent Networks, An Evolutionary Approach, Proc. of ICGA-91, pp.555-561 (1991).

(15) Nagao, T., Agui, T. and Nagahashi, H.: Structural Evolution of Neural Networks Having Arbitrary Connections by a Genetic Method, IECE, Vol.E76-D, pp.689-697 (1993).

(16) Bergman, A.: An Evolutionary Approach to Designing Neural Networks, Computing with Biological Metaphors, Chapman & Hall, pp.298-308 (1994).

(17) Maniezzo, V.: Genetic Evolution of the Topology and Weight Distribution of Neural Networks, IEEE Trans. on NN, Vol.5, No.1, pp.39-53 (1994).

(18) Angeline, P.J., Saunders, G.M. and Pollack, J.B.: An Evolutionary Algorithm that Constructs Recurrent Neural Networks, IEEE Trans. on NN, Vol.5, No.1, pp.54-65 (1994).

(19) Utecht, U. and Trint, K.: Mutation Operators for Structure Evolution of Neural Networks, PPSN III, pp.492-501 (1994).

(20) Collins, R.J. and Jefferson, D.R.: AntFarm, Toward Simulated Evolution, Artificial Life II, Addison Wesley, 579-601 (1991).

(21) 浜，皆川，嘉数：積み木問題の自律分散的解法に関する実験的考察，第4回FANシンポジウム講演論文集，pp.365-369 (1994).

(22) Hama, K. Minagawa, M. and Kakazu, Y.: Distributed Approach to Block Stacking Problem Based on Evolutionary Learning, Intelligent Enginnering System Through Artificial Neural Networks, Vol.4, ASME PRESS, pp.321-326 (1994).