進化型学習に基づく積木問題の自律分散的解法*

浜 克己^{*1}, 皆川 雅章^{*2}, 嘉数 侑昇^{*3}A Distributed Approach to Block Stacking Problem
Based on Evolutionary Learning

Katsumi HAMA, Masaaki MINAGAWA and Yukinori KAKAZU

We describe a new approach to a nonlinear block stacking problem in a traditional AI field. In our problem setting, we see the problem as one of planning in a 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 block 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 a recurrent neural network (RNN). Based on sensory inputs to the 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 levels which are calculated on the basis of received signals from other agents and the 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⁽¹⁾。この積木問題とは、積木群の位置に関して初期状態と目標状態が与えられたときに、目標状態に到達するための各積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⁽²⁾。また、与えられた各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⁽³⁾と計画に基づく方式⁽⁴⁾に大別され、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て成功をおさめてきた。

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。このため、従来法では問題固有のプランナーが必要となり、そのプランナー

は対象問題に対する完全かつ正確な知識を持つことが前提となっている。

近年、積木問題に対する一解法として、遺伝的アルゴリズム（Genetic Algorithms, 以下GAと記す）による探索も試みられている。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、積木移動に関してそのプログラムが探索されるが、一般的な問題例に対するプログラム生成結果は示されていない⁽⁵⁾。平野は、積木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⁽⁶⁾。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、積木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⁽⁷⁾。積木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。

これらは問題の解決に際してGAを直接的に適用したアプローチであるが、いずれもプランナーの存在を前提としたトップダウン方式に基づくものであった。

* 原稿受付 1995年5月11日。

*¹ 正員、函館工業高等専門学校（〒042 函館市戸倉町14-1）。*² 正員、札幌学院大学（〒069 江別市文京台11）。*³ 正員、北海道大学（〒060 札幌市北区北13条西8丁目）。

一方、マルチエージェントロボットシステムの関連分野として、プランニングを複数の主体（エージェント）で行う分散プランニングの研究も行われている⁽⁸⁾。この分散化に伴い、柔軟性や頑健性などの利点と同時に整合性や同期などの解決すべき問題も存在するが、環境が動的に変化したり問題領域が複雑な場合には、プランナーを必要としないボトプアップ的なアプローチが必要である。対象となる積木問題は、積木の移動に際して当該積木間で目標達成に関して相互干渉が存在する、非線形プランニング問題の一つと考えられる。したがって、この場合も個々の積木を自律的な移動物体（エージェント）と見なすことにより、各エージェントに自律性を持たせ、局所的な相互作用のみを利用する自律分散的解法の適用が可能となる。また、このような方式では、各エージェントに内在させる機能の設定が重要となるが、人工生物の集団捕食の挙動を進化させるシミュレーションでは、動作を決定する挙動関数をニューラルネットワーク（Neural Networks, 以下NN）で構成し、それを対象問題向きに進化させるアプローチが取られている⁽⁹⁾。

このようなNNの構造決定を目的として、その実現のために、進化型計算（Evolutionary Computation, 以下EC）⁽¹⁰⁾⁽¹¹⁾の一つであるGAを用いる試みは以前より盛んに行われている。初期の多くは、固定のフィードフォワード型アーキテクチャに対し、良適応の荷重集合の計算だけにGAを用いていたが、その後ネットワーク構造をも同時に最適化する方式へと移行した⁽¹²⁾⁽¹³⁾。さらに相互結合型のアーキテクチャについても進化型のアプローチが適用され⁽¹⁴⁾⁽¹⁵⁾、その後もGAをベースとする多くの方式が試みられている⁽¹⁶⁾⁽¹⁷⁾。しかし、最近、GAによるネットワークのオペレーションに対し、直接的な操作のしにくさや再結合（例えば交叉）による個性の軽視の点が問題として指摘されている。そこで、GAに代わり、ECを構成する他の代表的凡例として、突然変異を主たるオペレーションとする進化型プログラミング（Evolutionary Programming, 以下EP）や進化戦略（Evolution Strategy, 以下ES）を用いる方式が提案されている⁽¹⁸⁾⁽¹⁹⁾。

前述の人工生物における捕食行為の進化方式の例⁽⁹⁾では、使用するNNのアーキテクチャに関しても、いくつかのタイプについてその適応性が調査されている。その結果として、ランダム結合を許すリカレント型のNN（Recurrent NN, 以下RNN）が有効な接近法であるとの結論が得られている。このタイプは、積木問題のように以前の応答がその後の挙動に影響を与える時間依存の処理に対しても必要であると思われる。

以上より、本研究では各積木をエージェントと見なし、EPによりRNNを対象問題向きに進化させる方式を採用して、積木問題を自律分散的に解く新しい方法を提案する⁽²⁰⁾⁽²¹⁾。各積木はそれぞれセンサ、エフェクタなどを有し、その動作はRNNで表現する挙動関数にしたがって制御される。各積木は同一のRNNを所有するが、その置かれた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。積木群は、各時点において環境や他の積木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出力が最大の積木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ことになる。

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) 各積木の大きさは同一とする。
- (3) 各積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 t における状態 q_t を以下のように記述する。

$$q_t = \{b_i, h_i, v_i \mid i = 1, 2, \dots, N_B\} \quad (1)$$

b_i : 積木 ($b_i \in B$, B : 積木の集合)

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)

N_B : 積木数

積木の状態遷移関数 δ は、状態 q_t を次の状態 q_{t+1} に写像する。

$$q_{t+1} = \delta(q_t, m), \quad q_t, q_{t+1} \in Q \quad (2)$$

Q : 状態の有限集合

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

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

Fig. 1 The block state description

この状態遷移は、実施すべき行動にしたがって各ステップ毎に行われる。これより、積木問題は、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系列 M 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。

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

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) 各積木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、本研究では積木間の論理的な二者間関係に基づく状態表現を用いていないので、座標 (h, v) における積木の有無に応じてその名前あるいは空集合 (ϕ) を返す関数 $assign$ を定義し、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木で占有されないこと。

$$\begin{aligned} \text{for } \forall (p_i, p_j) \in q_t \quad (p_i \neq p_j) \\ \Rightarrow h_i \neq h_j, v_i \neq v_j \end{aligned} \quad (4)$$

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

X, Y : 積木の名前

- 2) 安定条件

各積木は台か他の積木の上に置かれていること。

$$\begin{aligned} \text{for } \forall p_i = (X, h_i, v_i) \in q_t \\ \Rightarrow assign(h_i, v_i - 1) \neq \phi \quad (v_i > 1) \end{aligned} \quad (5)$$

- 3) 移動可能条件

移動すべき積木の上には他の積木が置かれていないこと。

$$\begin{aligned} \text{for } \forall p_i = (X, h_i, v_i) \in q_t \\ \Rightarrow assign(h_i, v_i + 1) = \phi \quad (v_i \geq 1) \end{aligned} \quad (6)$$

- 4) 目標達成条件

特定の積木の目標は、その下に置かれるべき積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。

$$\text{for } \forall w \quad (0 < w < v) \Rightarrow Y = Z \quad (7)$$

$$Y = assign(h, w) \quad ((Y, h, w) \in q_t)$$

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

3. 自律エージェントと挙動関数

各積木は、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扱われる（図3）。また、積木内部にはRNNを所有し、それによって表現され

Fig. 2 The finite state transition

る挙動関数は、センサ入力を受けてエフェクタ（モータ）出力を生成する。すべての積木が同一の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。

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

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

$s_1 = 1$: 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

ネットワークの出力は、単一の積木が実行可能な基本行動群で構成される。 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する。

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

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

Fig. 3 The structure of an autonomous agent

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木のすぐ上の位置と目標位置に存在する積木に対して、その場所から移動することを要求する。これは入力 s_5 に対応するものである。 a_4 はネットワーク学習は進行するが、積木は何もせずに現在位置に留まることを意味する。各積木は、以下のような *Winner Take All* の考えにしたがい、この中から対応する出力ユニットの活性度が最大となる行動 (Action) を唯一つ獲得する。

$$Action = a_j \text{ such that } \max_{j \in N_a} o_j, \quad o_j \in R \quad (10)$$

o_j : ネットワーク出力

N_a : 行動パターン番号, $N_a = \{1, 2, 3, 4\}$

また、 a_5 は各積木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積木間の競合解消のためにも利用される。分散方式の考えからは、実行可能な積木の決定に際してもそれに関連した通信機能などを持たせる必要があるが、ここでは単に出力値を比較し、行動獲得と同様の方式を用いて最大の値を持つ積木に実行権を与えた。

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。

$$q_j = (X, h_j, v_j) \text{ such that } h_i \neq h_j \text{ and } \min_j v_j \text{ for assign } (h_j, v_j) = \phi \quad (11)$$

q_j : 作業領域の候補状態

$q_i = (X, h_i, v_i)$: 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、ここで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。ネットワークアーキテクチャの構成条件は以下の通りである (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。

$$R_i(t+1) = \frac{1}{1 + \exp\{-R_i(t)\}}, \quad R_i(t) = \sum_{j=1}^{N_i} w_{ji} R_{ji}(t) \quad (12)$$

$R_i(t)$: 時刻 t におけるユニット u_i の状態

Fig. 4 The block agent external interface

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

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

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

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 (集団サイズ $Psize$) 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) 確率 $Pconnect$ で各ユニット間の接続を決定する。
- 2) 接続ユニット間に存在するすべてのリンクには、 $[-W, W]$ の範囲でランダム (一様乱数) に荷重が割り当てられる。

実際の生成ストリングは以下のように表現される。

$$str_i = (z_1, w_1), (z_2, w_2), \dots, (z_j, w_j), \dots, (z_{NL}, w_{NL}) \quad (13)$$

z_j : 結合状態 ($z_j \in \{0, 1\}$), w_j : 荷重値

NL : 結合可能な最大リンク数

for (z_j, w_j) such that $z_j = 1$

$$w_j = rand([-W, W]), \quad W \in R$$

これら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。

$$V_\eta = c_1 \frac{G_\eta}{N_B} + c_2 \frac{Z_\eta}{N_S} \quad (14)$$

η : RNN ($\eta \in P$, P : 母集団)

G_η : 目標状態を満足している積木の総数

N_B : 対象となる積木の総数

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、確率 P_{sel} で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、前述の適応例⁽¹⁴⁾に基づき、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。手順は以下の通りである。

- 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。

$$T_\eta = 1 - \frac{V_\eta}{V_{max}} \quad (15)$$

V_{max} : 対象タスクに対する最大の適合性値

これにより、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。

$$w = w + N(0, \alpha T_\eta) \quad \forall w \in \eta \quad (16)$$

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。ここではリンクの存在を変更する確率を以下のように修正する。

$$P_\eta = P_m T_\eta \quad (17)$$

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。具体的には、非線形プランニング問題に対する自律分散解法や進化学習の適応性、並びに積木数が10個未満の小規模問題向きに生成されたネットワーク構造を積木数を増加させた問題の解探索（プラン生成）に利用する場合の効果について検証する。なお、以下では問題の規模を表現する便宜上の記法として、M-N（M：積木数，N：許容水平位置数）も併用する。

Fig. 5 The test examples of block stacking (1)

Table 1 The experimental conditions (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

前述のように、各ステップにおいて、各々の積木は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。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木のみがその動作を実現する。ただし、その積木が移動可能条件などを満たしていない場合には、次にこの出力値の大きい積木が行動を起こすものとする。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： $P_{connect} = 0.3$

結合荷重の範囲： $W = 1.0$

・ユニット数：入力層5，隠れ層10，出力層5

・垂直位置数(V)制限なし

・集団サイズ（ネットワーク数）： $Psize = 50$

・淘汰率： $P_{sel} = 0.5$ ，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： $c_1 = 0.9$ ， $c_2 = 0.1$

5.1 実験1

まず、非線形な小規模問題について目標達成状況を確認するために、図5に示す3つの例題を用いて実験を行った。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1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

Fig. 6 The transition of fitness for EX-1 to EX-3

Fig. 7 The time steps up to the goal state for Fig. 6

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てみた。数回の試行による結果の中から行動数に着目した場合の最良値のみを図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、特定の積木が連続的に行動し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、小規模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。EX-1, EX-2, EX-3の最適な行動数は、それぞれ8, 11, 15でEX-3-1を除いてクリアしているが、積木の増加とともに求められた全行動数は最適行動数に比べて多くなっている。この原因の一つとして、移動可能条件や目標達成条件を満たし、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挙げられる。また、EX-3については、水平位置数の増加が作業領域の選択範囲を広げ、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。

5.2 実験2

次に、積木の数をさらに増加させ、競合が発生し易

Fig. 8 The test examples of block stacking (2)

Table 2 The experimental conditions (2)

	The number of horizontal locations	The maximum time steps	Utilization of the networks created for small scale problems
EX-4-1	6	250	None
EX-4-2			EX-3
EX-4-3			None
EX-4-4	8	400	EX-3
EX-5-1	8		None
EX-5-2			EX-3
EX-5-3			None
EX-5-4	10		EX-3

いより複雑な環境での実験を行った。図8は、このように問題規模を大きく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、小規模問題向きに生成されたネットワーク構造の利用の有無による影響についても調べた。これらの組合せ条件を表2に示す。図9と図10は、それぞれ積木数が12, 18の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と同様である。また、図11と図12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-4の場合、目標達成までの収束の点からは生成された小規模問題向きのネットワーク構造を利用したEX-4-2とEX-4-4が有利であるが、真の行動数の点では水平位置数を増加したEX-4-3でも比較的良好な結果を示した。EX-5については、ランダムな初期状態から始めた場合には水平位置数を増加した18-10問題でも解が得られなかったが、小規模問題向きのネットワーク構造を利用したEX-5-2とEX-5-4の18-8, 18-10問題のいずれの場合も目標状態に到達した。このように、規模が増大し、ランダムな初期集団からの学習では解が得られにくい問題に対しても、小規模問題に関して生成された適切なネットワーク構造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。これは、小

Fig. 9 The transition of fitness for EX-4

Fig. 10 The transition of fitness for EX-5

Fig. 11 The time steps up to the goal state for EX-4

Fig. 12 The time steps up to the goal state for EX-5

規模問題向きに生成されたネットワーク構造が、12-6や18-8問題の解探索過程で生ずる類似の状況にうまく作用しているものと考えられる。また、一般には

(a) Initial network structure

(b) Final network structure

(c) Connection weights distribution

Fig. 13 The transition of network connections

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傾向にあるが、本方式のように作業領域のランダムな取扱いでは必ずしもそれが有効に使用されず、EX-5-4の18-10問題のように効果が低い場合も起こり得る。この結果からも、問題の規模が大きくなると、その解探索においては、水平位置数の増加以上に小規模問題向きに生成されたネットワーク構造の利用が有効であると考えられる。さらに、学習によるネットワーク構造の進化過程を確認するため、EX-5に対する小さな荷重を取り除いた初期状態と最終状態のネットワーク構造、並びに出力ユニットへの結合荷重の分布を図13に示す。ここで、実線は正荷重、破線は負荷重を表している。これらの結果より、最終状態では行動の実行要求出力 a_5

とともに実際に移動を伴う行動出力 a_1, a_2 への正の荷重値が増加し、これらが選ばれ易い状況になっていることがわかる。

6. 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一つである積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問題向きに訓練されることを示した。
- 3) 行動結果のみに基づく教師なし学習を行う自律分散方式の採用により、従来のトップダウン方式における問題点を軽減できることが確認された。
- 4) 積木数が増加しランダムな初期状態からでは探索が困難な問題に対しても、小規模問題向きに生成された適切なネットワーク情報を有効に利用することにより、問題解決に対して高い学習性能を持つことを示した。
- 5) 積木数が100個規模の問題、3次元異形状の場合については、近々に結果を報告したい。

参考文献

- (1) Tate, A. et al., A Review of AI Planning Techniques, in Readings in Planning, (1990), Morgan Kaufmann.
- (2) 岩井・片井・榎木・坂口・福森, 知識システム工学, (1991), 100-200, 計測自動制御学会.
- (3) Nilsson, N.J., Principles of Artificial Intelligence, Tioga, (1980).
- (4) Sacerdoti, E. D., A Structure for Plans and Behavior, Elsevier, (1977).
- (5) Koza, J. R., Genetic Programming : On Programming Computers by means of Natural Selection and Genetics, MIT Press, (1992).
- (6) 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, C.Q出版社, 177 (1992), 108-135.
- (7) 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—, 機講論, No.910-62, C (1991), 594-596.
- (8) 浅間, マルチエージェントロボットシステム研究の動向と展望, 日本ロボット学会誌, 10-4 (1992), 428-432.
- (9) Collins, R. J. and Jefferson, D. R., AntFarm : Toward Simulated Evolution, Artificial Life II, Addison Wesley, (1991), 579-601.
- (10) De Jong, K. and Spears, W., On The State of Evolutionary Computation, Proc. of ICGA-93, (1993), 618-623.
- (11) Back, T., Evolutionary Algorithms : Comparison of Approach, Computing with Biological Metaphors, Chapman & Hall, (1994), 227-243.
- (12) Harp, S. A., Samad, T. and Guha, A., Towards the Genetic Synthesis of Neural Networks, Proc. of ICGA-89, (1989), 360-369.
- (13) Miller, G. F., Todd, P. M. and Hegde, S. U., Designing Neural Networks using Genetic Algorithms, Proc. of ICGA-89, (1989), 379-384.
- (14) Torreale, J., Temporal Processing with Recurrent Networks : An Evolutionary Approach, Proc. of ICGA-91, (1991), 555-561.
- (15) Nagao, T., Agui, T. and Nagahashi, H., Structural Evolution of Neural Networks Having Arbitrary Connections by a Genetic Method, IECE, E76-D, (1993), 689-697.
- (16) Bergman, A., An Evolutionary Approach to Designing Neural Networks, Computing with Biological Metaphors, (1994), 298-308, Chapman & Hall.
- (17) Maniezzo, V., Genetic Evolution of the Topology and Weight Distribution of Neural Networks, IEEE Trans. on NN, 5-1, (1994), 39-53.
- (18) Angeline, P. J., Saunders, G. M. and Pollack, J. B., An Evolutionary Algorithm that Constructs Recurrent Neural Networks, IEEE Trans. on NN, 5-1, (1994), 54-65.
- (19) Utecht, U. and Trint, K., Mutation Operators for Structure Evolution of Neural Networks, PPSN III, (1994), 492-501.
- (20) 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, (1994), 365-369.
- (21) Hama, K. Minagawa, M. and Kakazu, Y., Distributed Approach to Block Stacking Problem Based on Evolutionary Learning, Intelligent Engineering System Through Artificial Neural Networks, 4, ASME PRESS, (1994), 321-326.