人　工　知　能　105－2
（1996．8．26）

カオス的想起に基づく並列進化モデルとそのA-NETマルチコンピュータへの実装

木幡直樹　　山口亨　　馬場敬信
宇都宮大学 工学部 情報工学科

本稿はマルチエージェントモデルにおける並列の進化モデルを実現することを目的とする。マルチエージェントモデルは複数の知的エージェントが動作し、ファジィ連想記憶上でのカオス的想起に基づき新たな知識を発想するもので、その一連の処理において並列処理が有効である。そこで、提案のモデルの並列処理をA-NETマルチコンピュータ上で実現する。本モデルの適用例としてロボットの2台並走を取り上げ、シミュレーションにより並列化の有効性を示す。

Evolutionary Parallel Computation based on Chaotic Retrieval and its implementation on A-NET Multi-Computer

Naoki KOHATA　　Toru YAMAGUCHI　　Takanobu BABA
Department of Information Science, Faculty of Engineering, Utsunomiya University

This paper proposes a realization of evolutionary parallel computation on multi-agent model. On multi-agent model, the plural intelligent agents work concurrently and each agent creates new knowledge based on chaotic retrieval on fuzzy associative memories. On a series of its processes, parallel processing is effective for the proposed model. Therefore, we realize a parallel processing of this model on A-NET(Actors NETwork) multi-computer. We apply the model to multi-agent robots which move together, and show the usefulness of its parallel processing based on simulation results.

1　はじめに

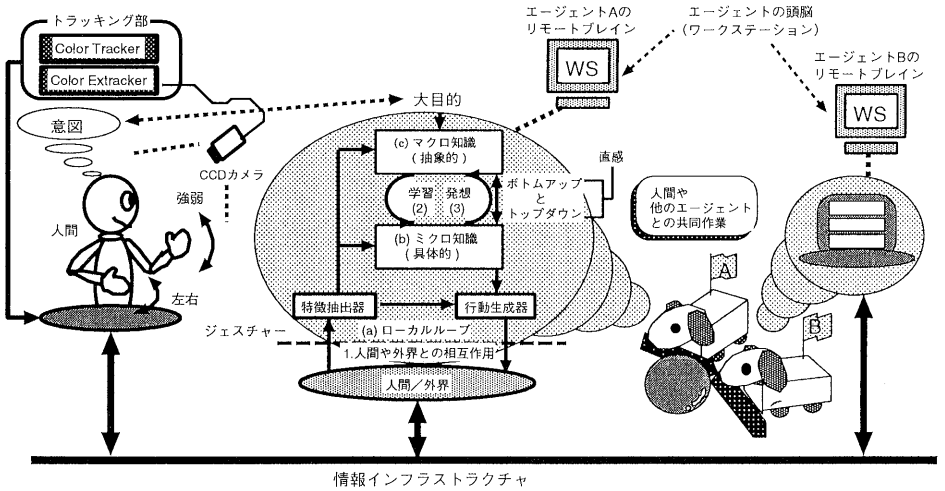

図 1: 直感に基づくエージェントモデル

近年、人工生命の進化・淘汰のモデルに関する研究 [1] が進められている。さらに著者らは進化モデルの一例として直感に基づく知的エージェントを用いたマルチエージェントモデルを提案し、並列処理による知能進化の研究を実施している [2]。このマルチエージェントモデルは図1に示すように複数の知的エージェントが人間や外界、他のエージェントと相互に作用しながら学習や発想を繰り返すことに

より進化し、大目的を達成するものである。マルチエージェントモデルにおけるエージェントには、環境が変化しても現在の知識を基に新しい知識を発想するなどして状況変化に自律的に対応するため、図1に示すような直感（すなわち、ボトムアップ処理とトップダウン処理を融合した並列処理）に基づくエージェントモデルを用いる。これは Russmussen モデル [3] を拡張したものであり、カオス力学系に基づくミクロな知識の発想を行ない、それをマクロな知識を使って評価する機能を有する。この「直感」や発想機能はファジィ連想推論及びファジィ連想記憶上でのカオス的想起により実現される。以上のような複数のエージェントの一括処理や「直感」的な処理等の実現のためには、並列処理が必然といえる。

一方、馬場らは A-NET マルチコンピュータ [4] の開発·研究を進めている。A-NET(Actors NETwork) は、並列オブジェクト指向の概念を核としたトータルアーキテクチャであり、並列オブジェクト指向言語 A-NETL(A-NET Language) により並列プログラムを自然に記述できる。そこで、本稿ではマルチエージェントモデルの並列処理を A-NET マルチコンピュータ上で実現する。その適用例としてロボットの 2 台並走を取り上げ、シミュレーションにより並列化の有効性を示す。

2 ファジィ連想記憶システムとカオス的想起

ここでは、発想機能の実現に用いたファジィ連想記憶システム及びカオスファジィ連想記憶システムについて述べる。

2.1 ファジィ連想記憶システム

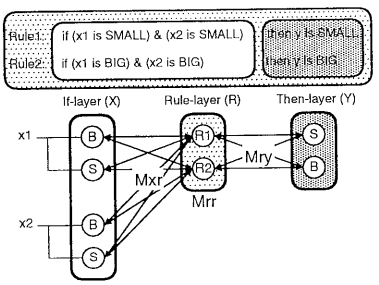

図 2: ファジィ連想記憶システムの構造とファジィルール

ファジィ連想記憶システムは、ファジィ知識を連想記憶上に構築することにより、ファジィ連想推論を実現したシステムである。ファジィ知識としてファジィルールを用いる場合、ファジィ連想記憶システムは図2のように構成される。ファジィ連想記憶システムでは、連想記憶ネットワークとして BAM(Bidirectional Associative Memory)[5] を用いているため、双方向の想起が可能であり、活性値の伝搬によるボトムアップ・トップダウン処理により入力条件に最もマッチしたパターンを想起できる (BAM の連想マトリクスとして、正規化連想マトリクスを用いている)。また、親和性の高い明示的な知識表現を有しているため、反響動作後のネットワークの活性値から知識を抽出することができる。

2.2 カオスファジィ連想記憶システム

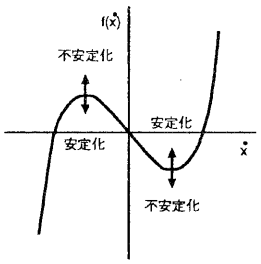

図 3: 非線形抵抗とその特性変動

図 4: エネルギー曲面上でのカオス的遍歴

カオスファジィ連想記憶システムは、ファジィ連想記憶システムの記憶の想起過程にカオス的想起を応用したシステムである。カオス的想起手法としては、カオス的最急降下法（CSD 法）を用いている。カオス的最急降下法とは、ニューラルネットワークのエネルギー曲面での運動を記述する力学方程式の散逸項に図 3のような周期的に変動する非線形抵抗を導入し、カオス的にエネルギー極小解の間を遍歴させる方法である。図 4はエネルギー曲面におけるカオス的遍歴のイメージ図である。ホップフィールド型ニューラルネットワークを例とした場合、CSD 法は式 (1)、式 (2) によって表される。

$$m\ddot{u}_i + f(\dot{u}_i, \omega t) = \epsilon \sum_j w_{ij} a_j + \eta I_i + \begin{cases} 0(-\beta \le u_i \le \beta) \\ -\alpha(u_i > \beta) \\ \alpha(u_i < -\beta) \end{cases} \quad (1)$$

$$f(\dot{u}_i, \omega t) = [d_0 \sin(\omega t) + d_1]\dot{u}_i + d_2 \dot{u}_i^2 sgn(\dot{u}_i) \quad (2)$$

ここで、u_i, a_i, I_iは、それぞれi番目のノードの内部状態、出力及び外部入力、w_{ij}はノードiからjへの結合の重み、$f(\cdot)$は非線形抵抗を表す関数、m, ϵは正の係数、ωは非線形抵抗の特性を変動させるための角速度、αはエネルギー勾配の補正値、βはノードの内部状態の絶対値の範囲、ηは外部入力に対する重み係数、d_0, d_1は線形抵抗の係数、d_2は非線形抵抗の係数、$sgn(\cdot)$は符号関数である。但し、式(1)の外部入力の項(ηI_i)は本来のCSD法に新たに加えられた項である。また、系のエネルギーEと、$a_i([0,1])$、w_{ij}はそれぞれ式(3)～(5)で表されるものとする。

$$E = -\sum_{i,j} w_{ij} a_i a_j \tag{3}$$

$$a_i = \frac{1}{1+\exp(-u_i)} \tag{4}$$

$$w_{ij} = \sum_k (2a_i^k - 1)(2a_j^k - 1) \tag{5}$$

ここで、a_i^kは、k番目の記憶させるパターンのノードiの値である。系の持つ時定数に対し、比較的遅い周期で正抵抗および負抵抗の値を変動させると、正抵抗が大きくなる位相で状態は安定化し極小に漸近する。逆に負抵抗が大きくなる位相では、系は不安定化し極小から飛び出そうとする。このような変動を適当に行なうことによって、図4のようにエネルギー曲面の極小と極小の間をカオス的に遍歴させることができる。

カオスファジィ連想記憶システムは、次の2つの機能を有している。

1. 記憶させたパターンの中で入力パターンに近い範囲にある記憶パターンを動的に想起する。

2. 記憶させていないが意味的に有効である新たなパターンを想起する。

機能2は、カオス的な記憶の探索では記憶させていないパターンが多数想起されるが、その中に意味的に有効なパターンが含まれている可能性があるということである。これは、カオスファジィ連想記憶システムでは、ネットワークの各ノードにファジィラベルで表現される意味を持たせることで知識を明示的に表現しているため、想起されたパターンが有効かどうか判断できるからである。

以上のような機能により、カオスファジィ連想記憶システムは連想メモリ上に記憶した既存の知識を用いて新たな知識を生成できるという特徴を持つ。このため、カオスファジィ連想記憶システムを用いて発想(支援)が可能である[1]。

3 A-NETマルチコンピュータ

馬場らは、並列オブジェクト指向の概念を核としたトータルアーキテクチャA-NET(Actors NETwork)[4]の開発・研究を進めている。A-NETは、並列オブジェクト指向言語A-NETLとプログラミング支援環境、1000台規模のMIMD分散メモリ型並列計算機を統一的に扱うことを前提に、研究が進められている。A-NETマルチコンピュータの構成を図5に示す。この計算機の処理要素であるノードプロセッサは、要素プロセッサ(PE)とローカルメモリ、メッセージの経路選択を行なう専用のトポロジ独立なルータから成り、ネットワークトポロジは、ハイパーキューブなど標準的なもの6種類が用意されている他、任意のトポロジを定義して用いることができる。また、オブジェクトを多様なトポロジに最適に割り付けるアロケータを備えている。

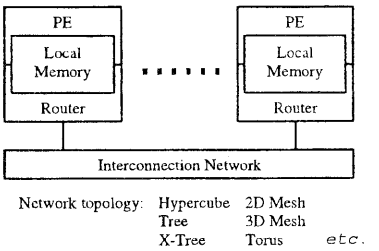

図5: A-NETマルチコンピュータの構成

A-NETL(A-NET Language)は、各問題内の並列性を自然な形で記述できる並列オブジェクト指向言語である。A-NETLにおける並列処理の単位はオブジェクトである。オブジェクトは、データ構造部と手続き部から成り、前者は状態変数宣言部、後者はメソッド宣言部と呼ばれる。通常一つのプログラムにおいて、複数のオブジェクトを定義し、それらを各ノードプロセッサに割り付ける。各オブジェクトは、同期的または非同期的にメッセージをやり取りして、互いに協調しながら並列に処理を行なう。これらのオブジェクトは、プログラムの分割の最小単位であり、一つのファイルに一つずつ定義される。また、オブジェクトが受け取るメッセージの型は、過去型、現在型、未来型の3種類があり、これらのメッセージと同期機構を使い分けることにより、実行順序を制御している。

4 進化モデルの適用例：2台並走ロボット

人間は、2人並んで歩く場合に、普通互いに歩調を合わせながら歩く。曲るときには外側の人は足を早め、内側の人はゆっくりと歩く。マルチエージェ

ントを用いた進化モデルの適用例として、このような歩行時の協調的な対応をカオス的に想起する2台並走ロボットを取り上げる。図6に示すように、ロボットは相手の動きとそれに対する自分の状態を把握し、対応をカオス的に想起する。想起された知識の中で適切な並走が可能な知識を残し、不適切な知識を捨てることで知能の進化・淘汰が行なわれる。本適用例におけるロボット制御部の構成を図7に示す。

同図の上位ネットワークには数パターンの状況変

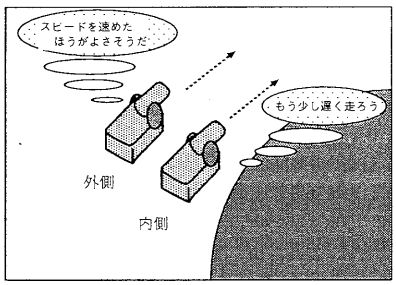

図 6: 2台並走イメージ図

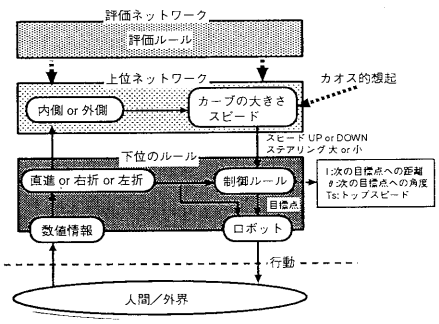

図 7: 2台並走におけるロボット制御部

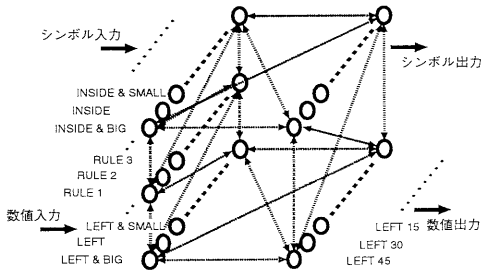

図 8: CFS（概念ファジィ集合）

化に対応する為の定性的な知識を持っており、下位の層の出力値はこれに依存し変化する。つまり上位ネットワークと下位のルールを図8のようなCFS(概念ファジィ集合)[6] を用いて構成する。同図において、下部が下位のルールであり、上部が上位ネットワークである。また下位のルールでは、左側で数値入力、右側で数値出力を行う。数値入力は'右に大きく'や'左に小さく'などのラベルと真理値に変換される。また、出力にはスピードとステアリングの値が出力される。上位ネットワークにある定性的な知識としては、単独で動くのか、2台で並走するのか、2台並走ならば次曲がる方向に対して自分が内側になるのか外側になるのかというものがあり、それらの情報によって出力値を状況に応じて変化させる。

5 進化モデルの並列化

2台並走ロボットを例にとり、提案の進化モデルの並列化の方法を示す。

5.1 並列連想処理

連想記憶上に構築された知識を想起するための連想推論を並列化する方法を述べる。ここでは、図9に示すようなネットワークを基に説明する。

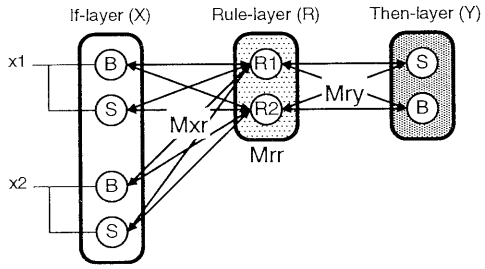

図 9: ネットワーク化したファジィルール

各レイヤ間（If(X)-Rule(R)、Rule(R)-Then(Y)）の連想推論は式 (6)、(7) による活性値の伝搬（反響動作）により行なわれる。但し、以下の式中の A、Bは If(X)-Rule(R)、Rule(R)-Then(Y) のいずれかのレイヤのペアに対応する。

$$\begin{aligned} B(t) &= \phi(MA(t)), \\ A(t+1) &= \phi(M^T B(t)) \end{aligned} \tag{6}$$

ここで、$A(t) = (a_1(t), a_2(t), \cdots, a_n(t))^T$, $B(t) = (b_1(t), b_2(t), \cdots, b_p(t))^T$は、ステップ t における各レイヤの活性値ベクトルであり、$\{0,1\}$ の値をとる。ϕは各ノードが持つ関数でシグモイド関数を用いる。シグモイド関数は式 (7) を用いる。

$$OUT_i = \frac{1}{1 + exp^{-\lambda \cdot NET_i}} \tag{7}$$

ここで、OUT_iはノード i の出力値で、NET_iはノード i における重みづけされた入力値の総和である。ま

た、λは、関数の勾配を決定する定数である。また、各レイヤ間のノードの結合の重みを表す連想マトリクス Mは、記憶したいパターンペアが $(A_1^T, B_1^T)^T \cdots (A_m^T, B_m^T)^T$とすると、式 (8) で求められる。

$$M = \sum_{i=1}^{m} \beta B_i A_i^T, \quad M^T = \sum_{i=1}^{m} \beta A_i B_i^T \qquad (8)$$

ここでβは連想係数である。

連想推論では各レイヤ間で反響動作を行なわせることにより推論を行なっている。この反響動作はネットワーク全体で同時に行なうことができ、逐次的に行なう必要はない。そこで、If(X), Rule(R), Then(Y) の各レイヤの活性値ベクトルを求める式 (6)、(7) による活性値演算処理を各レイヤ同時に行なうという並列化を実現する。この並列連想処理の流れ図を図10に示す。A-NETLプログラムでは、各レイヤに相

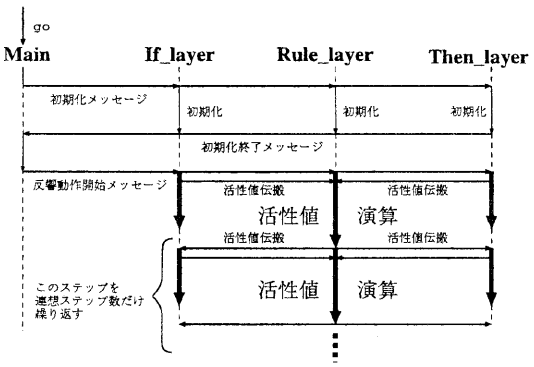

図 10: 並列連想処理の流れ図

当するオブジェクトを用意して活性値演算処理を並列に行なう。各レイヤに相当するオブジェクトは、それぞれ活性値ベクトルと連想マトリクスを用いて計算を行ない、最後に式 (7) で表されるシグモイド関数を通すことで、自身の新たな活性値ベクトルを得る。新たな活性値ベクトルはペアになっている相手のレイヤ宛てにメッセージとして送り（活性値伝搬)、それを受け取った相手のレイヤはそれを用いて次のステップの活性値演算処理を開始する。以上のような処理が互いにメッセージをやり取りしながら連想ステップ数だけ並列に繰り返される。

図9のネットワークの連想推論を実行する逐次型と並列型の A-NETLプログラムを作成し、シミュレータにより処理時間の比較を行なったところ、反響動作による連想推論処理に関しては並列型は逐次型の約 3.6 倍高速となった。

5.2 2台並走ロボットの並列化

上述の並列連想処理の手法に基づき、2台並走ロボットの並列化の方法を述べる。図11に2台並走ロボットの並列処理の流れ図を示す。これは2台並走

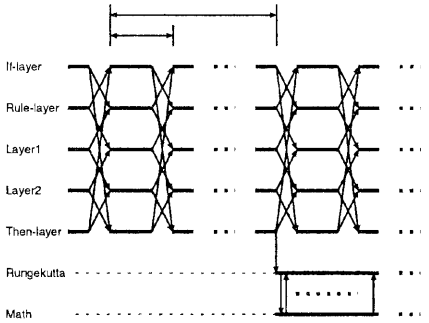

図 11: 2台並走ロボットの並列処理の流れ図

ロボットの制御部を実現する図8のネットワークの並列処理を行なう流れ図である。図8は5つのレイヤから成る階層的な連想記憶ネットワークであるので、同様の並列連想処理を行なうことができる。図11において、If-layer～Then-layerの各オブジェクトは図8で示される5つのレイヤに相当し、各レイヤ単位の連想推論処理を行なう。また、Rungekutta、Math はロボットの動きを計算する運動方程式の数値計算のためのオブジェクトである。図11はロボット1台分の処理を表しているので、同様のオブジェクトを2台分用意し、互いに同期をとりながら同時に処理することで2台並走ロボットの並列化が実現される。

6 シミュレーション

ここで図8のネットワークに基づき2台並走シミュレーションを行ない、逐次型プログラムの処理速度と並列型プログラムの処理速度を比較することによりその並列化の有効性を示す。 図12は並列型プログラムにおける、各オブジェクトを割り付けられたノードプロセッサの稼働状態を表すダイアグラムである。これは2台並走を行なうロボット2台分の実行ダイアグラムであり、ノードプロセッサの番号が5番以降のものが、図8のネットワークにおける各レイヤに相当するオブジェクトを割り付けられているものである。これよりレイヤ単位に行なう並列連想処理により、高並列な連想推論処理が可能であることが分かる。図13、図14に A-NET マルチコンピュータ上で2台並走シミュレーションを行った場合のロボットの軌跡を示す。図13は2台のロボットに対し、左へ大きく曲がり後に右へ大きく曲がるように指示を与えた場合のロボットの軌跡である。この場合、

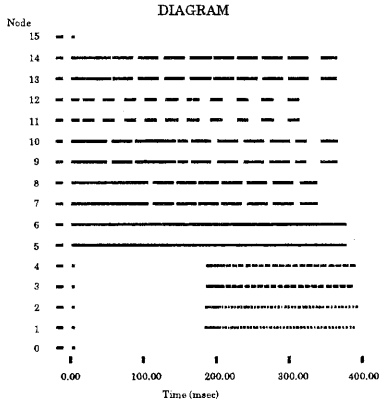

図 12: 各ノードプロセッサの稼働状態

適切な並走がなされていないので現在の上位ネットワークの知識は不適切な知識であり淘汰される。それに対し図 14は同様の指示を与えたものであるが、適切な並走がなされているので適切な知識として上位ネットワークに記憶される。また、逐次型プログラムと並列型プログラムとの処理時間の比較を図 15に示す。処理時間は、逐次型では約 4.33(sec)、並列型では約 0.39(sec) であり、並列型は逐次型の約 11倍処理が高速化されている。

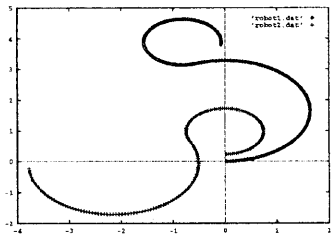

図 13: 2 台並走シミュレーション結果 1

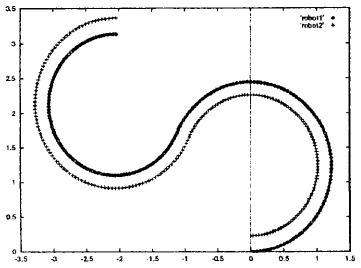

図 14: 2 台並走シミュレーション結果 2

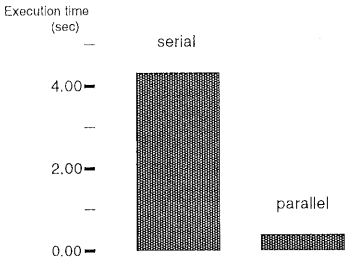

図 15: 逐次処理と並列処理との処理時間の比較

7 おわりに

本稿では、進化モデルの一例として直感に基づく知的エージェントを用いたマルチエージェントモデルにおける並列の知能進化を提案した。その適用例として 2 台並走ロボットを示し、その並列処理を A-NET マルチコンピュータ上で実現し、シミュレーションにより並列処理が本モデルの処理の高速化に有効であることを示した。

参考文献

[1] 山口：カオスと連想記憶, 日本ファジィ学会誌 Vol.7, No.3, 500-511 (1995).

[2] N.Kohata, T.Yamaguchi, Y.Wakamatsu and T.Baba : Evolutionary Parallel Computation based on Chaotic Retrieval and Creation, Proc. of Int. Conf. on Soft Computing(IIZUKA'96) (printing).

[3] J. Russmussen, Skills, Rules, and Knowledge : Signals, Signs, and Symbols, and Other Distinctions in Human Performance Models, IEEE Transactions on System, Man and Cybernetics, SMC-13-3, pp. 257-266 (1983).

[4] 馬場、吉永： 並列オブジェクト指向トータルアーキテクチャA-NET における言語とアーキテクチャの統合, 電子情報通信学会論文誌 D-I Vol.J75-D-I, No.8, 563-574 (1992).

[5] B. Kosko : Adaptive Bidirectional Assosiative Memories, Applied Optics, Vol. 26, No. 23, 4947-4960 (1987).

[6] 高木、山口、菅野：概念ファジィ集合とその連想記憶による実現, 第 7 回 日本ファジィ学会 ファジィシステムシンポジウム 講演論文集, 359-362 (1991).