修 士 論 文

自律移動ロボット群による協調行動の学習に関する研究

北陸先端科学技術大学院大学
知識科学研究科知識社会システム学専攻

後藤 昭夫

2002 年 3 月

修 士 論 文

自律移動ロボット群による協調行動の学習に関する研究

指導教官 藤波 努 助教授

北陸先端科学技術大学院大学
知識科学研究科知識社会システム学専攻

050032 後藤 昭夫

審査委員主査 藤波 努 助教授
審査委員 中森 義輝 教授
審査委員 林 幸雄 助教授
審査委員 佐藤 賢二 助教授

提出年月: 2002 年 2 月

目 次

第 1 章　序論 **1**
1.1　研究の背景と目的 1
1.2　本論文の要旨 3

第 2 章　設計手法 **4**
2.1　人工ニューラルネットワーク (ANN) 4
2.1.1　ANN の概要 4
2.1.2　ネットワークモデル 6
2.2　遺伝的アルゴリズム (GA) 6
2.2.1　GA の概要 6
2.2.2　GA の基本動作 7

第 3 章　協調行動の学習 **10**
3.1　ANN の設計 10
3.2　GA の設計 10
3.2.1　コーディング手法 10
3.2.2　適応度関数 12
3.2.3　選択, 交叉, 突然変異 13

第 4 章　シミュレーションによる実験 **14**
4.1　シミュレーションプログラムの概要 14
4.2　ロボットの概要 14
4.3　ロボットの入出力 15
4.4　シミュレーションの環境 15
4.5　適応度計算時の初期状態 15
4.6　GA の学習パラメータ 16
4.7　結果 16
4.7.1　学習結果の評価方法 17
4.7.2　学習結果の成功と失敗 17
4.7.3　適応度関数の設計の違いによる相違 20
4.7.4　隠れ層の素子数の違いによる相違 21

4.7.5 1 個体における適応度の計算回数 21
4.7.6 3 台で学習を行った結果を用いて台数の増加 22

第 5 章 考察 **25**
5.1 ANN の構造の評価 25
5.2 適応度関数の評価 26
5.3 利き腕の発現 27
5.4 初期状態の依存 27

第 6 章 謝辞 **28**

付 録 A UNDX による交叉の実装 **31**
A.1 単峰性正規分布交叉 (UNDX) 31
A.2 UNDX を用いた学習結果と検討 32

第1章　序論

1.1　研究の背景と目的

単純な自律移動ロボットを複数台つかうことにより、そのロボットが 1 台では困難な作業 (例えば箱押し作業 [11][6]) を効率よく進めて行くことことができる。また、複数のロボットで作業を行うことにより、1 台のロボットが壊れてしまった場合でも暇な他のロボットがそのロボットの作業を引き継ぐような、柔軟なシステムを構築できる。しかし、このようなロボット群のシステムを構築する際に、ロボットの台数や作業目標が増加することによって、ロボットのおかれる状態数が増加してしまう。そのため、高度な作業を行うロボット群のシステムを人間の手によって全て設計することは困難となる。この問題を解消するために、人間は基本的な内部構造と学習手法の設計を行うだけで、後は作業内容や環境にあわせてロボット自身により学習して行くように設計する必要がある。また、その内部構造や学習手法をできるだけ簡潔な設計を用いることで、実装などの面で有利となる。本研究では、協調行動としてフォーメーションタスクを設定し、単純な内部構造としてニューラルネットワーク (ANN), 学習手法として遺伝的アルゴリズム (GA) を用いた協調行動の学習について検討を行う。

複数台のロボットによってフォーメーションの形成を行うことは、ロボット群のグループ行動の制御や、人工生命の観点などから非常に重要な問題である。Reynols[10] らは、近傍の仲間との衝突を回避する, 近傍の仲間と速度をあわせようとする, 近傍の仲間のそばにとどまろうとするという非常に単純な 3 つの行動原理を用いることで、鳥の群れを表現を行っている。これと似た振る舞いを用いて Mataric ら [5] は、ロボットによる群れの形成に成功している [9]。

Balch ら [1] は、フォーメーションの形成を、ロボットの集団の中心点との位置関係, リーダーとの位置関係, 近くのロボットとの位置関係の 3 つをコントロールすることにより行っている。このらのロボットは GPS などのグローバルな情報を用いてフォーメーションを形成している。しかし、通信コストの増大などの面より、各ロボットが他の全てのロボットの状態を知っているのは非現実的であり、局所的な情報のみでフォーメーションを形成することが望ましい。Jakob ら [4] は、センサーなどのローカルな情報を基本として、さらに局所的な通信と各ロボットにユニークな ID づけを行うことによってフォーメーションの維持を行っている。Desai ら [3] は、近くのロボットとの角度 ψ と距離 l をコントロールする ψ-l *control* と、2 台の近くのロボットとの距離をコントロールする l-l *control* によってフォーメーション形状の柔軟な維持を行っている。

本研究におけるフォーメーションタスクでは、次の条件を設けてフォーメーションの形成を行う。

- 各ロボットは局所的な情報のみ得られる。
- 各ロボットはユニークな情報を持っていないため、他のロボットを一意的に判断することはできない 。
- 各ロボット同士の通信は行わない。
- グループのリーダーは存在しない。

このような条件のもとで、ロボット群がフォーメーションを形成するのに適切かつ単純な設計手法についてあきらかにして行く。また、フォーメーションを形成する際の初期状態の依存性についても検討する。

本研究では内部構造として、センサーの入力とモーターの出力を単純にニューラルネットワーク (ANN) によって結合した構造を用いる。センサーとモーター間に ANN をはさむことで、ANN の特徴であるノイズへの強さや汎化能力をロボットに持たせることができる。ロボットのセンサーとモーター間の ANN の結合重みの学習では、あるセンサー入力から得られた ANN からのモーター出力がタスク全体において正しかったかどうかを一意に決めることは難しい。そのためバックプロパゲーションのような教師あり学習を使うことができない。そこで、ANN の結合重みの学習には進化論的手法である遺伝的アルゴリズム (GA) をもちいる。

ロボットの行動や形状を進化論的な手法 (例えば GA や GP) によって進化させながら、環境に適応するロボットを学習させる手法は、進化ロボティクスと呼ばれている。本研究で用いる GA+ANN は、進化ロボティクスにおいてよく用いられる手法である [8]。しかし、進化ロボティクスに用いられる ANN の構造は様々な方法がとられているが、一般的によいという構造はない [7]。本研究では、ネットワークの構造は、フィードフォワード (Feedforward) 型, SRN(Simple Recurrent Network) 型,2 層再帰結合 (2-Layer Recurrent) 型, さらにフィードフォワード型の入力層に 1 ステップ前の入力情報とあわせて入力するネットワーク形状 (Past-Input Feedforward) を用いて、フォーメーションタスクに適したネットワークの形状について検討を行う。また、GA の適応度関数を 3 種類設計し、学習を行う際にどれだけフォーメーションを形成するためのバイアスを与えればよいかについて比較検討を行う。

以上のように、本研究ではフォーメーションタスクの学習に適した、設計を明かにするために、ANN の構造, 適応度関数の設計について比較検討を行う。また、各設計の初期値依存性について調べることにより、その設計の評価を行う。

1.2 本論文の要旨

以降の章の構成について述べる。2 章は、本論文を読むにあたり必要とされる知識、具体的にはニューラルネットワーク、遺伝的アルゴリズムについて述べる。3 章では、本研究において用いたロボットの内部構造の設計と学習方法、特に適応度関数について述べる。4 章では、3 章で述べた設計の実装とシミュレーションによる結果について述べる。最期に 5 章において学習結果の検討とまとめについて述べる。付録に追実験として、GA の交叉に UNDX を用いたときの結果について述べる。

第2章　設計手法

本章では、本研究において用いたニューラルネットワーク (ANN) と遺伝的アルゴリズム (GA) について説明する。

2.1　人工ニューラルネットワーク (ANN)

2.1.1　ANN の概要

人工ニューラルネットワーク (ANN) は生体の脳を模倣しており、生体の神経細胞 (Neuron) をモデル化した人工ニューロンを互いにつなぎ合わせてネットワークを構成したものである。生体ニューロンの主な構成要素は、図 2.1 のように樹状突起 (dendrite)、細胞体 (cellbody)、軸索 (axon)、そしてシナプス (synapse) となっている。樹状突起は、他のニューロンからの活性度を細胞体に伝達する役割を果たしており、細胞体は入力された活性度を合計する役割を担っている。樹状突起から受け取った累積的な刺激が、ある一定の閾値を超えたときにそのニューロンは興奮し、軸索から他のニューロンへスパイクの形で情報を伝達する。このスパイクは、軸索に沿って伝播する活性電位のことである。そして軸索は、シナプスを介して他のニューロンと結合する。シナプスにはニューロンの活性度を増大させる興奮性のものと、逆に活性度を減少させる抑制性のものとが存在する。

人工ニューロンは生体ニューロンを抽象化し、図 2.2 のように多入力 1 出力の情報処理素子としてモデル化できる。あるニューロン i は、他の n 個のニューロンから入力信号 $x_1, x_2, \cdots, x_i, \cdots, x_n$ を受け取る。この入力信号に、ニューロン $j(j = 1, 2, \cdots, n)$ との結合荷重 w_{ij} を重ね合わせ、ニューロンの累積的な入力値 u_i は次のようにモデル化される。

$$u_i = \sum_{j=1}^{n} w_{ij} x_j \tag{2.1}$$

このとき、w_{ij} が正の場合は興奮性、負の場合は抑制性となる。u_i が閾値 θ_i を超えるとニューロンが興奮し、出力信号 y_i を出す。この入力と出力の関係は次式のようになる。

$$y_i = f(u_i - \theta_i) = f(\sum_{j=1}^{n} w_{ij} x_j - \theta_i) \tag{2.2}$$

ここで、f は活性化関数と呼ばれるものであり、生体ニューロンの非線形性を模倣している。主な活性化関数には、図 2.3(a) の線形閾値素子 (Step Function) や、図 2.3(b) のシグモイド関数 (Sigmoid Function) が用いられる。

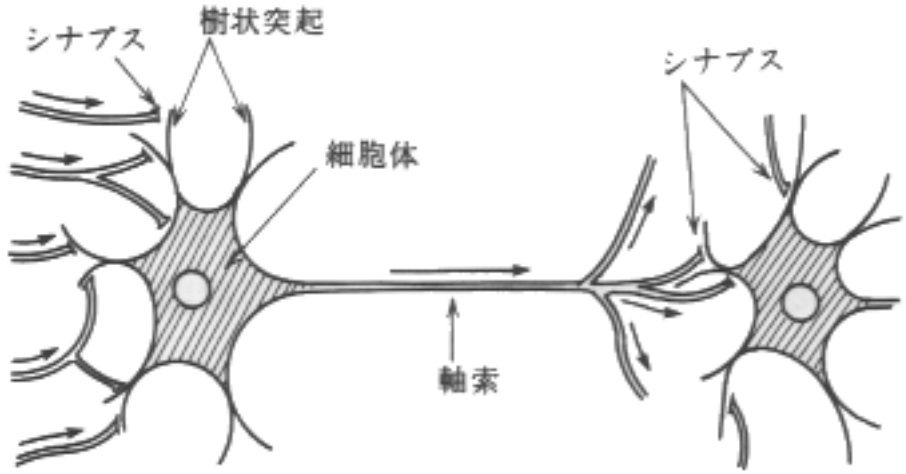


図 2.1: 生体ニューロンの構成

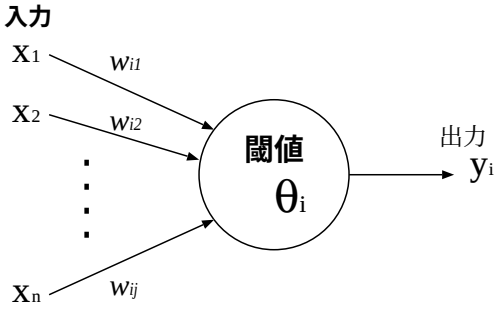
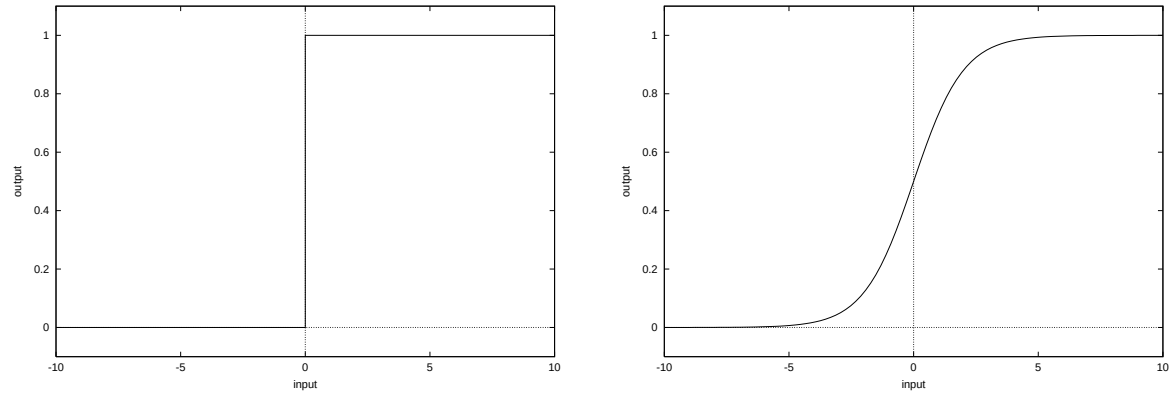


図 2.2: 閾値素子のモデル



(a) Step Function

(b) Sigmoid Function

図 2.3: 活性化関数

2.1.2 ネットワークモデル

人工ニューロンを用いて構成するネットワークモデルは、非常に多くの提案がなされている。ネットワークの構造として、よく用いられるものは階層型 (Feedforward)、再帰結合型 (recurrent)、相互結合型 (Hopfield) である。

階層型の結合では、図 2.4(a) のように複数のニューロンにより層を構成し、さらに階層に対して結合がなされている。

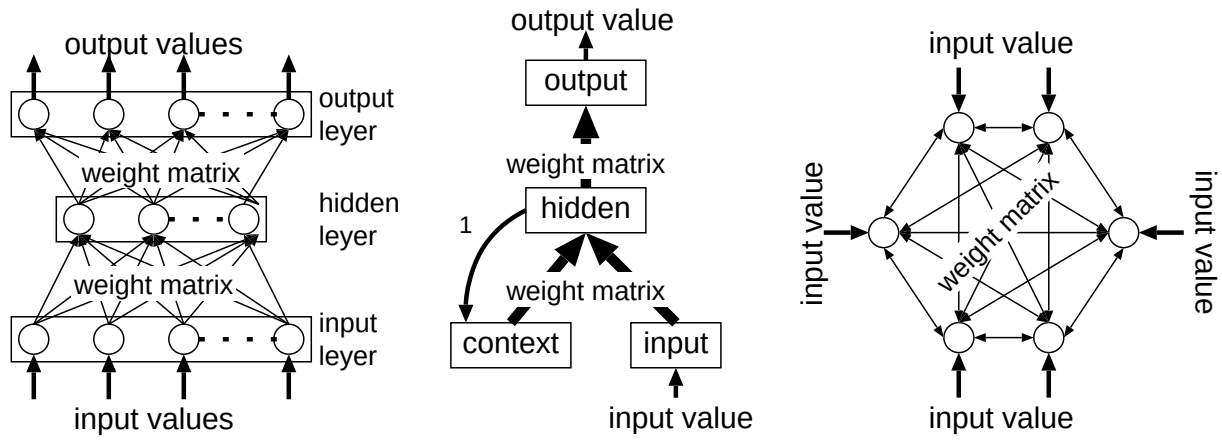
再帰結合型では、図 2.4(b) のように、隠れ層 (hidden leyer) のニューロンを context 層として次の計算時に与える。これによりネットワークは、離散時間遅れのデータを扱うことが出来る。

相互結合型のネットワークは、図 2.4(c) のように各ニューロンがそれぞれに結合している構成となっている。

2.2 遺伝的アルゴリズム (GA)

2.2.1 GA の概要

遺伝的アルゴリズム (GA) は、生物の進化を模倣した確率的な最適値探索アルゴリズムである。自然界においては、生活環境に適応できない個体は死滅していき、環境に適した個体は生き残り子孫を増やしていくことが出来る。そして、それを繰り返していくことによってその集団の中にすぐれた遺伝子が広がり群れ全体が環境に適応するように繁栄する。このメカニズムを取り入れ、問題に対してよい個体を生成しようとするのが GA であ



(a) 階層型　(b) 再帰型　(c) 相互結合型

図 2.4: ネットワーク構造

る。GA を構成する要素は以下の通りである。

GA の最小構成要素は遺伝子 (gene) であり、遺伝子の並びを染色体 (chromosome) と呼ぶ。染色体は個体 (individual) を特徴付けるものである。個体について、表現型 (phenotype) と遺伝子型 (genotype) があり、表現型は固体の形質や特性を、遺伝子型は染色体の構造を表す。表現型から遺伝子型へ写像することをエンコード化 (encoding)、逆に遺伝子型から表現型へ逆写像することをデコード化 (decoding) という。

このような個体の集まりを個体群 (population) といい、ある世代 (genetype) を形成している個体群の中で、環境への適応度 (fitness) の高い個体が高い確率で生き残るように選択 (selection) していく。さらに、交叉 (crossover) や突然変異 (mutation) により、次の世代の個体群が形成される。このような世代の交代が繰り返されて、世代交代が繰り返されるたびにより最適解に近い個体が増えていき、やがて最適解が得られるであろうというのが GA の基本的な考え方である。

2.2.2 GA の基本動作

GA の基本的な動作を図 2.5 に示す。

1. **初期個体群の生成 (Initialization)**
   個体群の各個体の遺伝子をランダムに決定する。

2. **個体の評価 (Evaluation)**
   与えられた適応度関数によって、各個体の適応度を決定する。この適応度が高い個

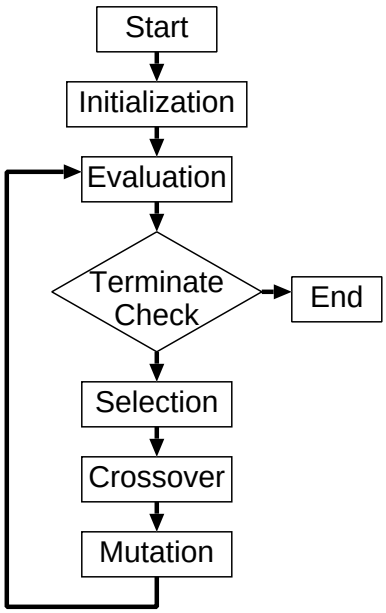


図 2.5: GA の処理手順

体ほど、優れた固体として自分の遺伝子を次世代に残す可能性が高くなる。本研究では、問題とするロボットの動作を GA で学習するために適応度関数をどのように設計すればよいかについて比較・検討を行うために、複数の適応度関数を用意した。

3. **終了判定 (Terminate Check)**
   あらかじめ決めていた世代数だけ世代交替を行ったか、または他の終了条件を満たしているかによって計算を終了するかどうか判定する。

4. **選択 (Selection)**
   個体群より次世代の個体の親となる個体を決定する。この選択を行う方法にはさまざまな方法が提案されている。その中で最もよく知られている方法はルーレット戦略である。ルーレット戦略は適応度比例戦略とも呼ばれ、各個体の適応度に比例した確率で子孫を残す方法である。まずある個体の適応度 F_i が全体で占める割合によって選択する確率を決定する。ある個体 i が、選ばれる確率 $P_{selecti}$ は次式となる。

$$P_{selecti} = \frac{F_i}{\sum_{j=1}^{n} F_j} \tag{2.3}$$

   この式によって与えられた確率にしたがって、個体群の中から次世代の親となる個体を決定する。この方法は、ルーレット上の占有面積が大きいものほど次世代に子孫を残せる。

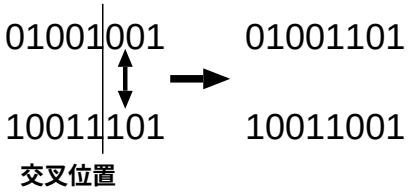


図 2.6: 交叉

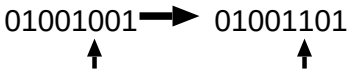


図 2.7: 突然変異

本研究では、ルーレット戦略とあわせて、集団中で最も適応度の高い個体をそのまま次世代に残すエリート戦略もおこなう。

5. **交叉 (Crossover)**
   交叉は、選択によって選ばれた 2 つの親の染色体を組み合わせて次世代となる子の染色体を作る操作である。

   交叉は、図 2.6 のように、親の染色体の中から交叉位置を決定し、その位置から遺伝子を入れ替えた染色体を子の染色体とする。基本的な交叉方法として、交叉位置が 1 つしかない単純交叉、交叉位置が複数存在する複数点交叉がある。

6. **突然変異 (Mutation)**
   突然変異は、個体の遺伝子を一定の確率で変化させる操作である。交叉とは異なり、図 2.7 のように個体中の遺伝子を強制的に変更することによって、交叉によって得られない探索を行う。

第3章　協調行動の学習

3.1　ANNの設計

本研究において結合荷重の学習を行う、ニューラルネットワーク (ANN) の構造について説明をする。

ANN の各素子は、線形閾値素子を使用した。

GA によって学習を行う ANN のネットワークの構成として、大きくフィードフォワード (Feedforward) 型と再帰 (Reccurent) 型の 2 種類を用いる。

- **フィードフォワード (Feedforward) 型**
  Feedforward 型のニューラルネットワークには、図 3.1 のように 2 種類の構造を用いる。

  図 3.1(a) は、非常に一般的な 3 層からなるネットワークである。図 3.1(b) は、(a) のネットワーク構成に 1 ステップ前の入力情報をもう一度、入力層に与える。(b) の構造を用いることにより、瞬間的な時間変化を用いた処理が可能になると考えられる。

- **再帰型 (Reccurent)**
  再帰型のネットワーク構成は図 3.2 のように 2 種類。再帰構造を持つことで、時間変化をうまく使う個体があらわれることが期待される。

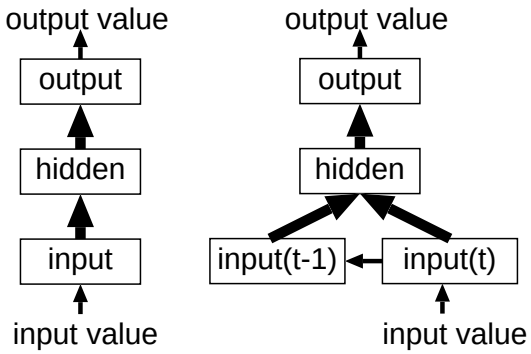
  図 3.2(a) の構造は、再帰型ニューラルネットワークにおいて基本的な Simple Reccurent Network(SRN) である。図 3.2(b) は、2 層の再帰型構造のネットワークである。出力層は 2 つのモーター出力のほかに、隠れ層にあたる冗長なニューロンを持たせている。

3.2　GAの設計

本研究における遺伝的アルゴリズム GA の基本的な設計について説明する。

3.2.1　コーディング手法

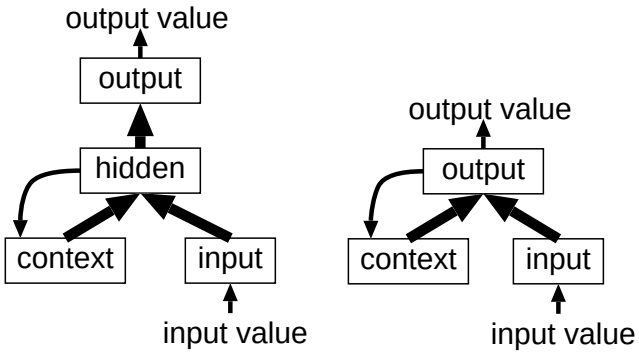
遺伝子型から表現型へのデコード方法について説明する。本研究では、実数値 GA の初歩的な方法を用いる。



(a) Feed-forward type

(b) Past-Input Feedforward type

図 3.1: Feedforward type



(a) Simple Recurrent Network tyep

(b) 2-Layer Recurrent Network type

図 3.2: Reccurent type

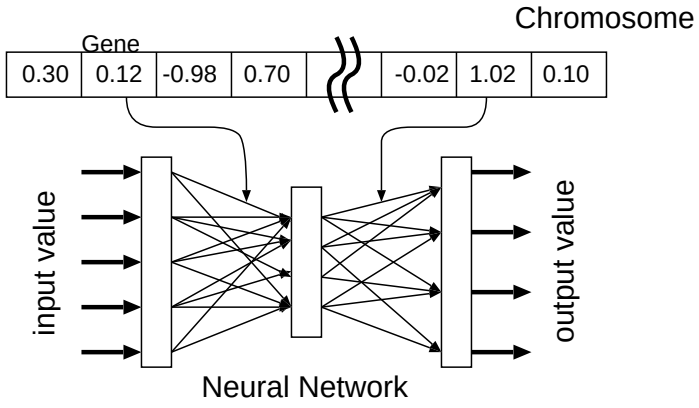


図 3.3: 遺伝子型から ANN の結合重みへのデコード

ANN の構造に関しては、本研究では GA の学習対象としない。そのため、学習時には ANN の結合数は変化しないので、各ニューロン間の結合と染色体の遺伝子を一意的に対応づけることができる。遺伝子型から表現型へのデコードを行うときには、各結合に対応する遺伝子の実数値を直接結合重みの値として用いる。(図 3.3)

3.2.2 適応度関数

本研究で用いた適応度関数は、a) 個体 (ロボット) の振る舞いに関する適応度関数と b) ロボット群の振る舞いに関する適応度関数の 2 つに別けることができる。フォーメーションを形成するためにどの程度選択圧を与えれば学習できるかについて比較するために、ロボット群の適応度関数 b) を 3 種類設計を行う。

本研究において適応度関数は、単位時間 ($dt = 0.1$ 秒) 毎にどのような状態にあるかについて評価し、その評価によってその時間における点数をあたえる。そして全試行時間 50 秒におけるその点数の合計を個体の適応度として計算する。

a) 個体 (ロボット) の振る舞いに関する適応度関数

各ロボットが個体としてうまく振舞えたかについては、徘徊 (Wandering)、発見 (Find)、接近 (Approach) について適応度を計算する。

- **徘徊 (Wandering)**
  Wandering 適応度はロボットが前の位置から動いたときに点数を与える。この Wandering は、ロボットが停止せずに動き回るときに適応度が高くなるようにする。また、この Wandering とあわせてロボットが衝突した時はロボットが停止すると言う条件を含めることで衝突回避 (Avoid) が暗黙的に記述される。

- **発見 (Find)**
  Find は、ロボットが他のロボットをカメラに捕らえたときに適応度を加点する。これによりロボットが他のロボットを見つけたときに適応度が高くなるようにしている。また、カメラの位置で中央に見つけるほど適応度が高くなるようにする。
- **接近 (Approach)**
  Approach はロボットが他のロボットに一定に距離内 (近接センサーの近距離内) に近づいたときに点数をあたえる。

b) ロボット群の振る舞いに関する適応度関数

ロボット群の振る舞いに関する適応度関数は、目的とするフォーメーションにどれだけ近いかについて評価する。

1. **フォーメーションについてあいまいに記述**
   各個体が直線に並んだときに高得点を与える。各ロボットの向きについては考慮しない。
2. **フォーメーションについて明確に記述**
   各個体が直線に並んだときに得点をあたえ、さらに各ロボットの向きがそろった場合にさらなる高得点を与える。
3. **フォーメーションまでの過程を含めて記述**
   3 台のロボットのうち 2 台が他のロボットを見つけて、さらに直線になり、同じ方向を向くというような段階にそって高得点を与える。このとき、2 台が他のロボットを見つけるという適応は、Find とほとんど同じであるため、この適応度関数の際は、Find による点数は与えない。

本研究では、個体の適応度関数 a) とロボット群の適応度関数 b) の 3 種類をそれぞれをあわせて、上記の番号順に適応度関数 1, 適応度関数 2, 適応度関数 3 と呼ぶ。

3.2.3 選択, 交叉, 突然変異

次世代の子孫の選択方法は、ルーレット戦略とエリート戦略の両方をあわせた戦略を用いる。エリート戦略によって残す個体数は 10 個体とし、残りの次世代は今の世代よりルーレット戦略により両親を決定する。

交叉方法は 2 点交叉を行い交叉点はランダムに決定する。

突然変異は、突然変異確率にしたがって最大 10%の遺伝子が ± 1 の範囲で変化する。

第4章　シミュレーションによる実験

4.1　シミュレーションプログラムの概要

本研究で用いるシミュレーターは、C++によって記述し、g++で作成した。動作環境は、g++によるコンパイルが可能なUNIX(またはPOSIX互換)上で動作する。また、学習結果のGUI表示を行うプログラムは、GUIのライブラリにGtk+を使用しているため、実行にはX-WindowSystemとGtk+ライブラリがインストールされている必要がある。

シミュレーター上における物理的な計算のうち衝突に関する計算は、撃力ベース法[2]を用いた。また、微分方程式はオイラー法によって計算する。

4.2　ロボットの概要

シミュレーター上で動作するロボット(図4.1)について説明する。このロボットは、AAI社のGaia-2を参考にして設計している。

ロボットには、前方に5つ、後方に2つの近接センサー(Gaia-2では超音波センサーであるため以降ソナー(Sonar)と呼ぶ)がついている。さらに視覚センサーとしてCCDカメラを取り付けている。実際のGaia-2では、4本のタイヤに別々の回転を与えることが可能であるが、実際には2本の左右の対は、同じ回転数で動作させることが望ましい。シミュレーターでは、簡略化のため、ロボットには左右に1本づつタイヤを取り付けてある。

シミュレーター上のロボットは、正方形の形をしており、Gaia-2とは大きく形状が異なっている。そのためロボットが他の物体に衝突したときの物理的な影響がまったく異なるものとなる。しかし、本研究での学習時において、衝突した(バンパーセンサがONになる)場合に停止するようしているため、ロボットの行動に衝突による行動制御(たとえば衝突時の衝撃を用いて方向転換を行うなど)は省かれるので、問題がないと判断し、簡単な正方形にした。また、シミュレーションにおいて物体の辺の数が増えることにより、計算数が増加するので、計算時間の短縮のためシミュレーター上のロボットは正方形にしている。

本研究ではGaia-2を用いたシミュレーション結果の実環境における評価は行わない。

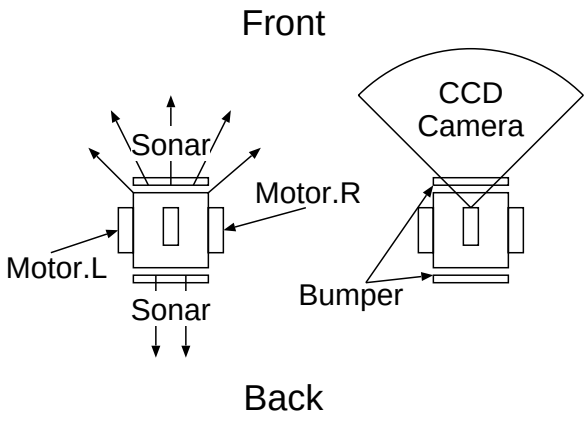


図 4.1: シミュレーション上のロボット

4.3 ロボットの入出力

ロボットの内部構造である ANN の入力には、7 つのソナーと CCD カメラからの信号を与える。

ソナーからの入力信号 (0〜255) の値を、一定の値 (本研究では 30) 以下だと 0、以上では 1 を ANN の入力に与える。CCD カメラからの入力は、カメラの左右の視界を 8 領域に分解し、その領域各々で他のロボットが見えない場合 0 を、見えるときに 1 を ANN の入力を与える。

ANN の出力である 2 つのニューロンの出力値は、左右各々のモーターの ON/OFF とする。

4.4 シミュレーションの環境

ロボットが動作する実験環境として、$10m \times 10m$ の正方形のフィールドを想定した。この広さは、ロボットの大きさに対して十分に大きく、ソナーのレンジが最大値を出力することが可能である。

4.5 適応度計算時の初期状態

個体の適応度評価を行うときの初期状態は、Field 中心から 3 台が正三角形をなすように配置する。(図 4.2) また、各ロボットの初期の方向は、ランダムに決定する。

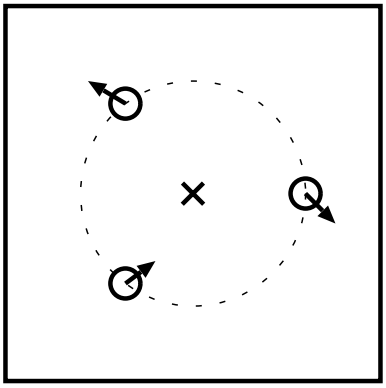


図 4.2: 初期状態

個体数	100
最大世代数	300
交叉率	0.5
突然変異率	0.1

表 4.1: GA の学習パラメーター

4.6 GA の学習パラメータ

GA によるニューラルネットワークの重み学習における、学習パラメータは表 4.1 のように、個体数 100, 最大世代数 300, 交叉率 0.5, 突然変異率 0.1 を用いた。

4.7 結果

シミュレーション上で、相手の後ろについてまっすぐ並ぶ Column Formation(図 4.3) について学習を行った結果を示す。

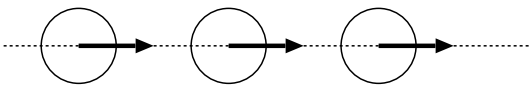


図 4.3: Column Formation

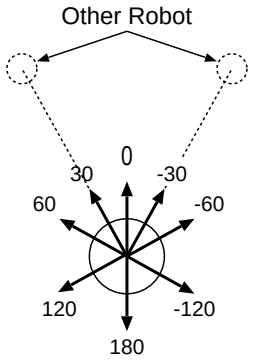


図 4.4: 学習結果評価時の方向

4.7.1 学習結果の評価方法

学習結果の、初期状態依存性とフォーメーション形成能力についての評価を行うために次のようなテストを行う。

タスクを行うフィールドは $10m \times 10m$ の広さで初期位置は学習時と同じものを用いる。図 4.4 ロボットの初期角度は、各ロボットでフィールドの中心を向いている状態を $\pm 0^\circ$ として、$\pm 30^\circ, \pm 60^\circ, \pm 120^\circ, 180^\circ$ の 8 状態を考え、全体として 512 状態の初期角度をもちいて初期状態の依存性について評価を行う。

各初期状態からフォーメーションを形成できたかの評価値は、評価時間 100 秒以内に、以下の 3 つの状態を全て満たしている状態を 50 秒以上形成することができた回数を用いる。

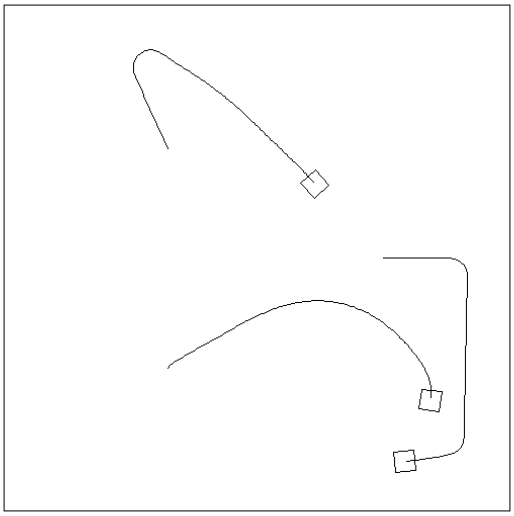
- 2 台が他のロボットを中心に見ている
- 直線に並んでいる
- 同じ方向を向いている

4.7.2 学習結果の成功と失敗

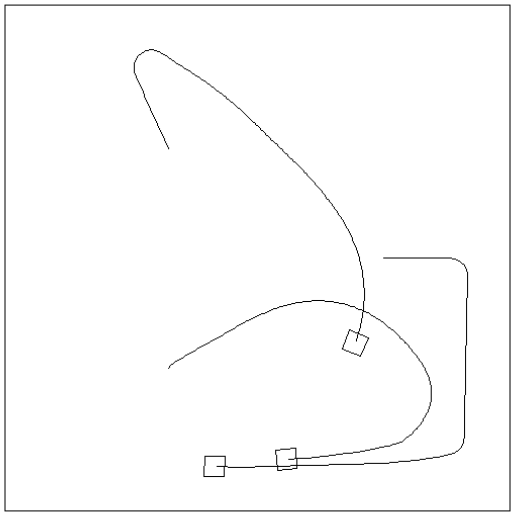
様々な条件で学習を行った結果、学習の成功例 (図 4.5) と失敗例 (図 4.6) を示す。

図 4.5 のように学習に成功した個体は、壁沿いに一定の方向にまわりながら他のロボットを追いかける。そして、先頭の個体に近づきながらフォーメーションを形成していく。このとき、フォーメーションの先頭になる個体は初期状態によってランダムに決定されている。

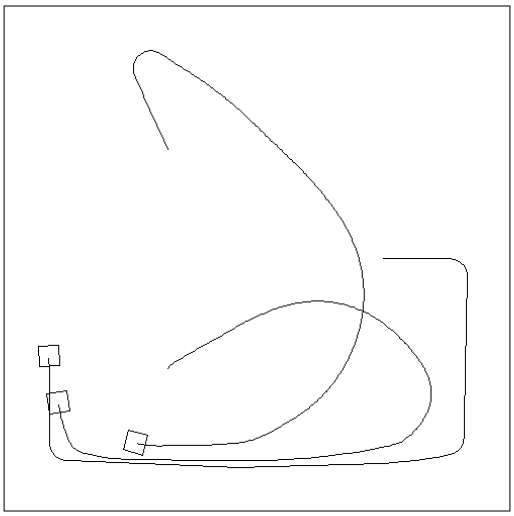
図 4.6 のように学習に失敗した個体は、両端になる 2 つのロボットは一定の場所で回転しながら、中に入るロボットは他の個体に近づきながら直線になって行く。このとき、直線に並び、さらに同じ方向に向くのはかなり限定された初期状態のみである。



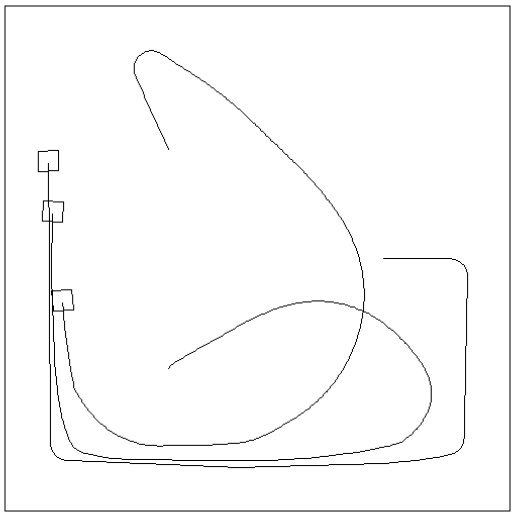
(a)



(b)

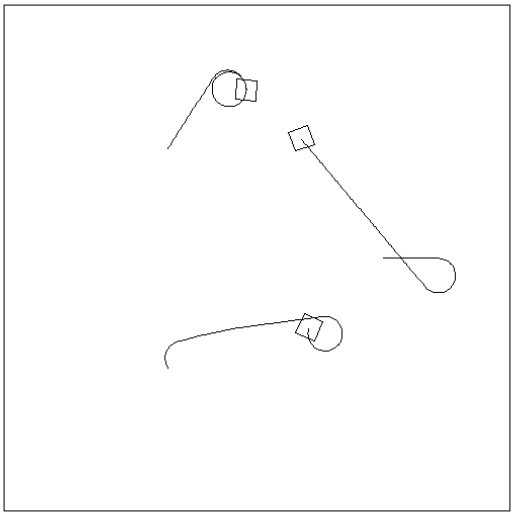


(c)

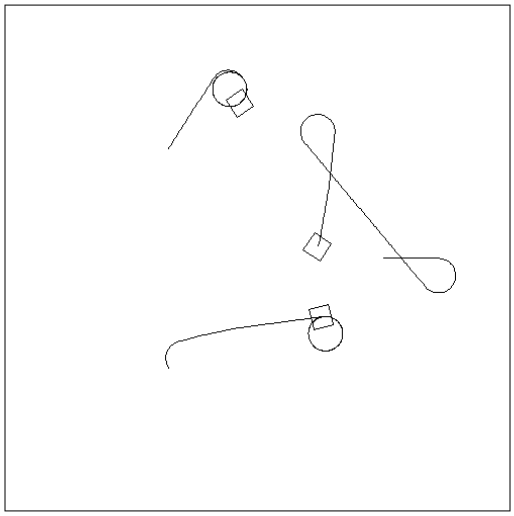


(d)

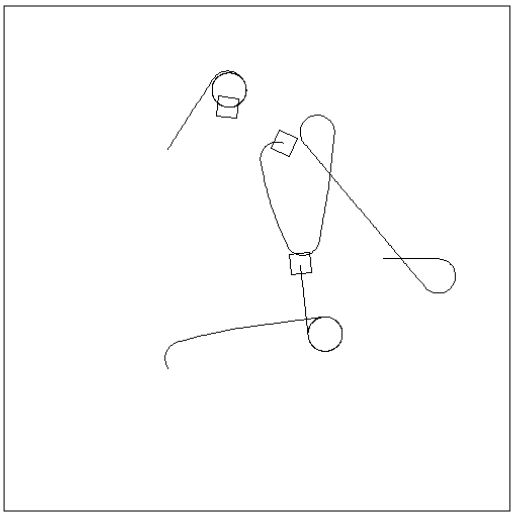
図 4.5: 学習の成功例



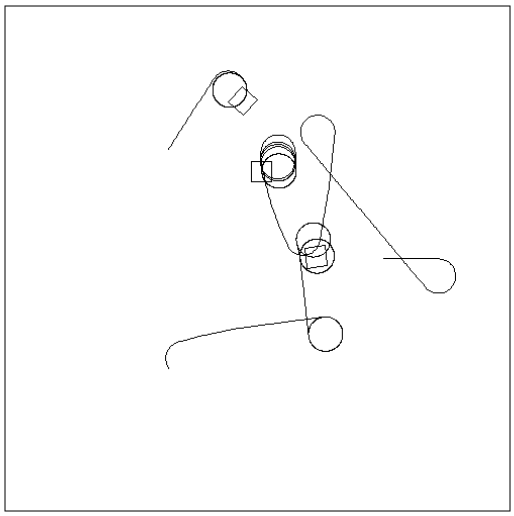
(a)



(b)



(c)



(d)

図 4.6: 学習の失敗例

Network Type	Success	Average(50[s] Over)	Max Time[s]
Feed Forward	4 / 10	36.8 / 512	84.2
Past-Input FF	3 / 10	26.3 / 512	86.3
Simple Reccurent	2 / 10	15.5 / 512	86.3
2-layer Reccurent	2 / 10	18.0 / 512	85.7

(Fitness Function No.1, Hudden layer=10, Loop=1)

表 4.2: 適応度関数 1 を用いた場合の学習結果

学習に成功した場合と失敗した場合では、初期値依存性の評価について比較すると、経験上、学習に成功した場合では 10 以上 (2%) の初期状態で 50 秒以上のフォーメーション形成に成功するが、失敗した場合では 50 秒以上フォーメーションを形成する初期状態はたいだい 10 未満である。以降の評価において、フォーメーション形成の成功, 失敗は上記の方法で行う。

4.7.3 適応度関数の設計の違いによる相違

適応度関数を 3 種類について学習を行った結果をしめす。ANN の形状は、feedforward,past-input feedforward,SRN,2-layer recurrent の 4 つである。隠れ層素子数は 10 である。各表における Success はランダムシードを変えて 10 回学習を行い、学習に成功した数である。Average は、学習に成功した個体で 50 秒以上フォーメーションを形成できた初期状態の平均である。

適応度関数 1

表 4.2 は適応度関数 1 について学習を行った結果である。

いずれの ANN の形状においても、学習に成功している最終個体は 50%にも満たない。また、50 秒以上フォーメーションを維持できる初期状態は 10%にも満たない。

適応度関数 2

表 4.3 は、適応度関数 2 について学習を行った結果である。

適応度関数 1 にくらべて、全体的に初期状態の適応が上がっている。しかし、初期配置の状態でフォーメーションを形成できる割合は、50 秒以上フォーメーションを維持できる初期状態は最大 10%程度である。

Network Type	Success	Average(50[s] Over)	Max Time[s]
Feed Forward	6 / 10	60.8 / 512	91.6
Past-Input FF	1 / 10	67.0 / 512	84.7
Simple Reccurent	3 / 10	47.7 / 512	86.4
2-layer Reccurent	5 / 10	33.8 / 512	93.3

(Fitness Function No.2, Hudden layer=10, Loop=1)

表 4.3: 適応度関数 2 を用いた場合の学習結果

Network Type	Success	Average(50[s] Over)	Max Time[s]
Feed Forward	6 / 10	47.7 / 512	86.7
Past-Input FF	2 / 10	43.0 / 512	86.9
Simple Reccurent	4 / 10	33.5 / 512	85.3
2-layer Reccurent	4 / 10	31.3 / 512	85.7

(Fitness Function No.3, Hudden layer=10, Loop=1)

表 4.4: 適応度関数 3 を用いた場合の学習結果

適応度関数 3

表 4.4 は、適応度関数 3 について学習を行った結果である。

学習の成功率は、適応度関数 2 とほとんど変わらない。逆に、Average を見ると学習に成功した個体のランダムな初期状態への適応は少なくなっている。

4.7.4 隠れ層の素子数の違いによる相違

ネットワークの隠れ層の数によって、学習できるか出来ないか。また、学習した結果、どのような違いがあるかについて比較を行った。ANN の隠れ層素子数を 6 個に減らして学習を行った結果を示す。(表 4.5)

同様の学習条件 (表 4.4) と比較すると、隠れ層の素子数が少ないほうが成績が良い。

4.7.5 1 個体における適応度の計算回数

ある世代における 1 個体に対して、適応度の計算を 1 回だけ行っているため、計算時の初期配置によって、その個体の適応度が大きく変動する。そのため、個体の適応度計算回

Network Type	Success	Average(50[s] Over)	Max Time[s]
Feed Forward	10 / 10	53.9 / 512	87.2
Past-Input FF	5 / 10	50.4 / 512	86.0
Simple Reccurent	2 / 10	53.0 / 512	84.9
2-layer Reccurent	8 / 10	54.0 / 512	91.6

(Fitness Function No.3, Hudden layer=6, Loop=1)

表 4.5: 隠れ層の違い

Network Type	Success	Average(50[s] Over)	Max Time[s]
Feed Forward	4 / 10	64.0 / 512	85.1
2-layer Reccurent	7 / 10	59.1 / 512	93.4

(Fitness Function No.2, Hudden layer=6, Loop=5)

表 4.6: 隠れ層の違い

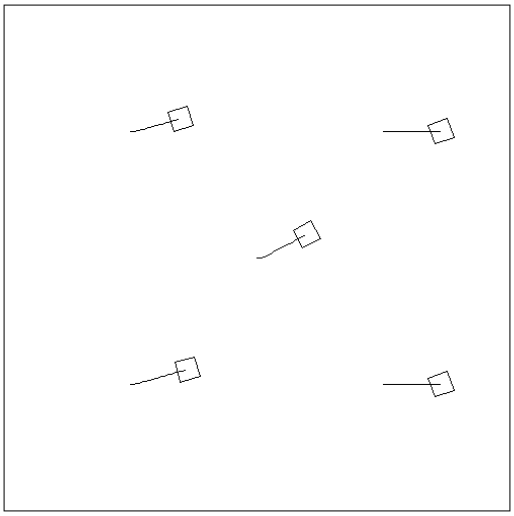
数を 5 回にして計算を行い、初期位置に依存しにくい適応度を用いて評価することで、初期位置に依存しない個体が現れるかどうかについてシミュレーションを行った。この結果を同じ学習条件 (Robot 3,Fitness No.2) と比較を行った結果を表 4.6 に示す。

学習に成功した個体の、ランダムな初期状態への適応は少し上昇している。

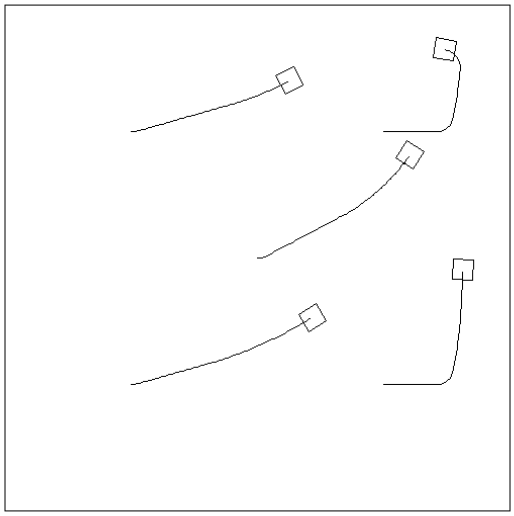
4.7.6 3 台で学習を行った結果を用いて台数の増加

学習時に存在しなかった環境において、学習したロボットがいかに適応できるかについて評価を行うために、ロボットの台数を増やした場合について実験を行った。学習によって得られた ANN の重みを使い、ロボットの台数を 5 台に増やして行動させた結果を図 4.7, 図 4.8 に示す。

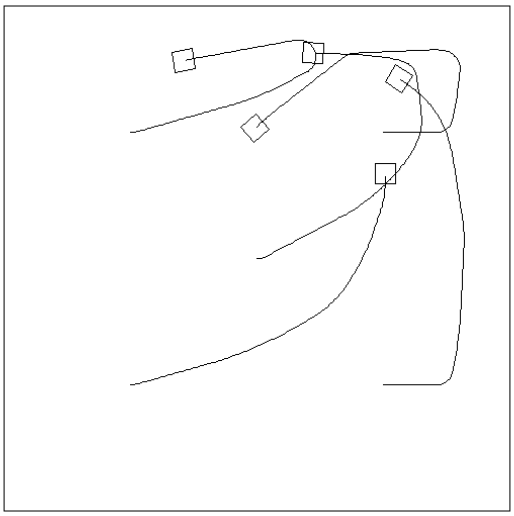
結果、5 台用いてもフォーメーションを形成できる。学習時において 2 台以上のロボットがカメラに写る状態がないにもかかわらず、4 台のロボットが見えても、フォーメーション形成できている。



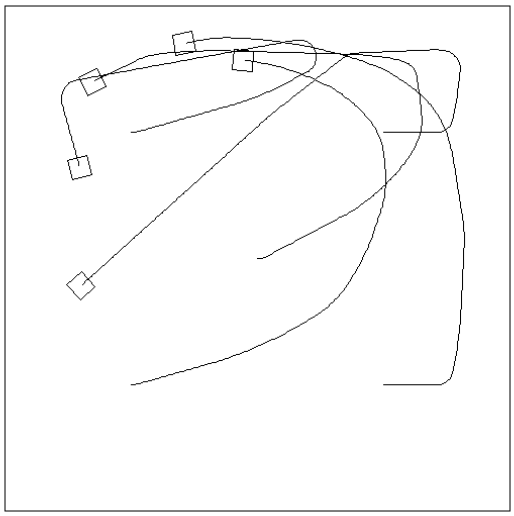
(a)



(b)

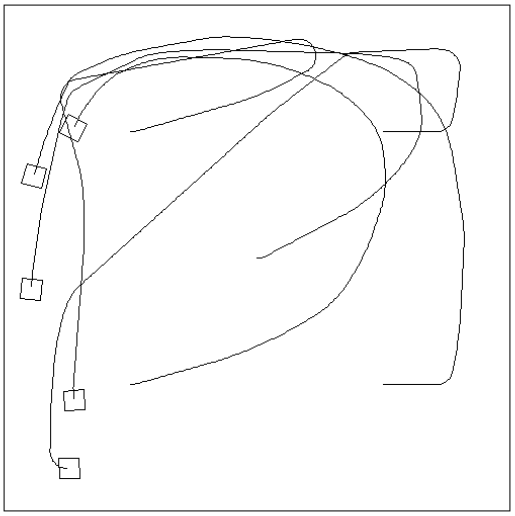


(c)

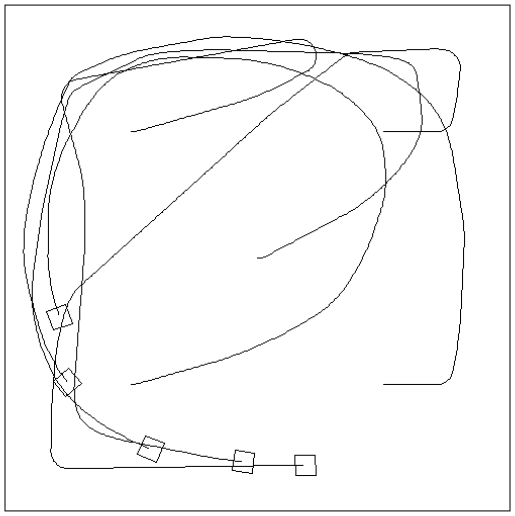


(d)

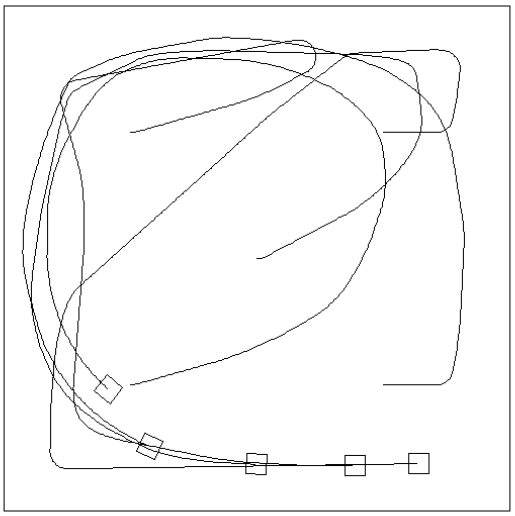
図 4.7: 学習結果を用いて、5 台のロボットによって試行 (1)



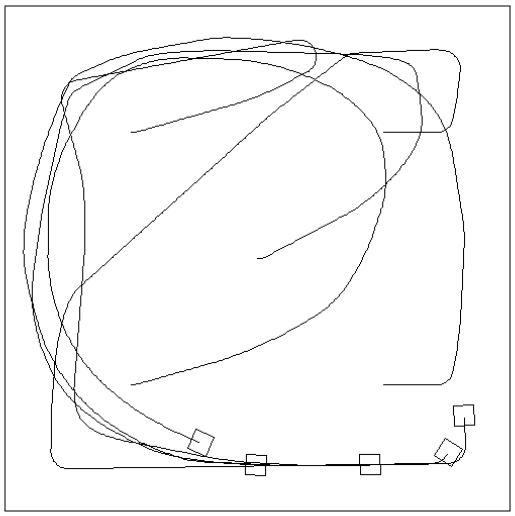
(a)



(b)



(c)



(d)

図 4.8: 学習結果を用いて、5 台のロボットによって試行 (2)

第5章　考察

シミュレーションを行った結果、内部構造にANNと学習方法としてGAを用いた非常に簡素な手法によって、一部の初期状態に関してはColumnフォーメーションを形成することができた。また、学習した結果を5台のロボットについて適応した結果、フォーメーションを形成することができ、汎化能力を確認できた。しかし、すべての初期状態に適応する個体は現れなかった。

5.1　ANNの構造の評価

本研究で用いたネットワーク構造のでは、もっとも単純なフィードフォワード型がフォーメーションを形成することに適していた。この理由として、column formationは過去の情報を参照しなくとも反射的な行動のみによって形成できるために、過去の情報を扱う扱わないにかかわらず、他のネットワーク形状にくらべて探索空間が狭いフィードフォワード型が一番よく学習できたと考えられる。また、隠れ層の数はできるだけ少ない方がよいという結果が得られたが、これも同様に探索空間が小さくなったためだと考えられる。この仮定の裏づけとして、図5.1のようにネットワークの形状に関係なく、結合の数(各ネットワークの形状と結合数は表5.1を参照)によって成功率が変化している。そのため、他のネットワーク形状において本来はfeedforwardよりよい性能を持つにもかかわらず、GAの探索能力不足のため、成績が悪くなってしまっている可能性がある。

	Number of Hidden layer	
	6	10
Feed Forward	108	180
Past-Input FF	198	330
Simple Reccurent	138	270
2-layer Reccurent	126	250

表 5.1: ネットワークの結合数

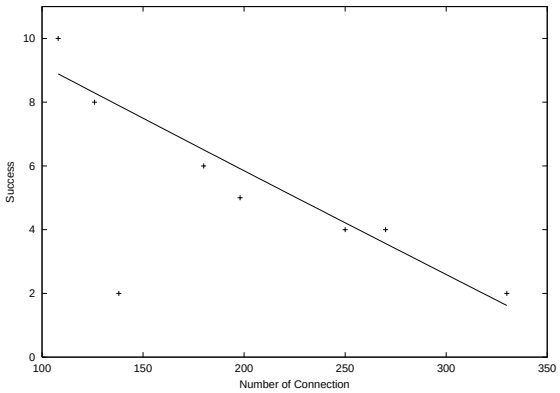


図 5.1: 結合数と学習成功率の推移

Fitness Function	Success	Average(50[s] Over)
No.1	4 / 10	36.8 / 512
No.2	6 / 10	60.8 / 512
No.3	6 / 10	47.7 / 512

(Feedforward , Hidden layer=10, Loop=1)

表 5.2: 適応度関数の評価

5.2 適応度関数の評価

表 5.2 は、Feedforward 型のネットワーク形状 (隠れ層素子数=10) についての学習の成功数と 50 秒以上フォーメーションを形成できた初期状態の数をまとめたものである。

適応度関数 1 はもっとも成功数が少なく、初期状態に対する適応度も低くなっている。適応度関数 2 と適応度関数 3 を比較すると、成功数は同じであるがフォーメーションを 50 秒以上形成できた初期状態の数は、適応度関数 2 のほうが多い。適応度関数 3 ではフォーメーションを作るまでの段階にそって得点を与えているため、フォーメーションを形成するまでの道のりが制限されているために初期状態の適応が低くなったと考えられる。この結果より、column formation の形成においては適応度関数 2 のように設計したほうがよい。

5.3 利き腕の発現

すべての学習に成功した事例に共通して、ロボットは右または左のどちらか一方行に徘徊する行動 (利き腕) が見られる。右回りにフォーメーションを形成する個体は、右に他の個体が見えた場合は、その個体を追いかけていく。逆に左に他の個体を発見したときに、ソナーの値が遠距離ならばその個体を追いかけるが、ソナーの値が近距離の場合にはその個体を視界から外すように行動をする。また、各ロボットは同じネットワーク構成をもっているため、右方向に巡回する個体が、左に見える個体を視界から外す行動は、相手の先回りにもなる。

5.4 初期状態の依存

本研究では適応度の計算時に初期状態として 1 または 5 つの状態を用いているが、学習結果の評価時のように、1 つの個体の初期状態を単純に 8 つに分類した場合でも、3 台では 512 状態になる。そのため、適応度の計算時に評価している初期状態の数が少なく、多くの状態で適応する個体の適応度が高くなるようになっていないため、すべてに適応する個体は現れなかったと考えられる。しかし、学習の際に 1 つの個体の適応度を 512 状態について計算を行うと、非常に計算回数が増加し、非常に時間がかかる可能性がある。

一部の初期状態についてフォーメーションを形成できる個体において、フォーメーションの形成に失敗する事例として良くあるのは、2 台または 3 台のロボット同士でぶつかり合い、停止してしまうことがほとんどである。そのため、ロボット同士による衝突回避と相手を追いかけるとの間のトレードオフがうまく働いていない可能性がある。フォーメーションがなかなか形成されない個体と、フォーメーションをよく形成する個体を比較すると、衝突回避能力が強いとフォーメーションを作ろうとするが、近づきすぎると衝突すると誤認して回避してしまい、フォーメーションが崩れてしまう。逆に、よくフォーメーションを形成する個体は、他のロボットに対してよく追いかけるが、正面衝突のときになかなか回避しない。

第6章　謝辞

本研究を行うにあたり、多大なるご指導を頂きました櫻井彰人教授に感謝致します。研究に関する様々なご助言を頂きました、藤波 努 助教授, 荒木 修 助手に感謝致します。また、知識システム構築論講座のみなさまにこの場をお借りしてお礼を申し上げます。

参 考 文 献

[1] T. Balch and R. Arkin. Behavior-based Formation Control for Multi-robot Teams. IEEE Transactions on Robotics and Automation,Vol. 14,No. 6,pp. 1–15, Dec 1998.

[2] David Baraff. An Introduction to Physically Based Modeling: Rigid Body Simulation I/II,SIGGRAPH COURSE NOTES,1997.

[3] J. P. Desai, J. Ostrowski, and V. Kumar. Controlling formations of multiple mobile robots. pp. 2864–2869, May 1998.

[4] Jakob Fredslund and Maja J Mataric. A General, Local Algorithm for Robot Formations,Institute for Robotics and Intelligent Systems Technical Report IRIS-01-396,2001.

[5] M. Mataric. Designing and understanding adaptive group behavior. Technical report, CS Dept, Brandeis Univ., 1995.

[6] M. Mataric, M. Nilsson, and K. Simsarian. Cooperative multi-robot box pushing. In Proceedings of IEEE International Confereence on Intelligent Robots and Systems (IROS), pages 556-561, 1995.

[7] J. Meyer. Evolutionary approaches to neural control in mobile robots,in Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, San Diego, (October 1998) 15.

[8] Stefano Nolfi and Dario Floreano. *Evolutionary Robotics*. MIT Press, 2000.

[9] R. Pfeifer and C. Scheier. 知の創成. 共立出版株式会社, 2001. 石黒章夫, 小林宏, 細田耕 監訳.

[10] C. Reynolds. Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics*, Vol. 21, No. 4, pp. 25–34, 1987.

[11] Y. Saito and Y. Saito. Adaptive Action Selection without Explicit Communication for Multi-robot Box-pushing. To appear in 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems.

[12] 小林佐藤. 遺伝的アルゴリズムにおける世代交代モデルの提案と評価. 人工知能学会誌, Vol. 12, No. 5, pp. 734–744, 19997.

[13] 喜多一小野功. 実数値 ga とその応用. 人工知能学会誌, Vol. 15, No. 2, pp. 259–266, 3 2000.

[14] 北野宏明（編）. 遺伝的アルゴリズム 4. 産業図書, 2000.

付 録A　UNDXによる交叉の実装

本研究で用いた交叉方法は、実数値 GA においては適切な交叉ではない [13]。そのため、GA の探索能力不足により、初期値に依存しない最適な解が得られなかったと考えることができる。

そこで付録として、実数値 GA で有効であると考えられる単峰性正規分布交叉 (UNDX)[14] を本研究の問題に摘要した結果を示す。

A.1　単峰性正規分布交叉 (UNDX)

単峰性正規分布交叉 (UNDX) では、両親を結ぶ直線上およびその近傍に、両親と第 3 の親によって決まる正規分布にしたがって子を生成する A.1。

UNDX のアルゴリズムは以下の通りである:

1. 3 個の親を $\boldsymbol{x}^1, \boldsymbol{x}^2, \boldsymbol{x}^3$ とする。
2. 親 $\boldsymbol{x}^1, \boldsymbol{x}^2$ の中点を $\boldsymbol{x}^p = (\boldsymbol{x}^1 + \boldsymbol{x}^2)/2$ とする。
3. 親 $\boldsymbol{x}^1, \boldsymbol{x}^2$ の差のベクトルを $\boldsymbol{d} = \boldsymbol{x}^1 - \boldsymbol{x}^2$ とする。
4. 親 $\boldsymbol{x}^1, \boldsymbol{x}^2$ を結ぶ直線を主探索直線と呼び、親 $\boldsymbol{x}^3$ から主探索直線までの距離を D とする。
5. 子 $\boldsymbol{x}^c$ を以下の式に従って生成する:

$$\boldsymbol{x}^c = \boldsymbol{x}^p + \xi d + \sum_{i=1}^{n-1} \eta_i D \boldsymbol{e}_i, \xi \sim N(0, \sigma_\xi^2), \eta_i \sim N(0, \sigma_\eta^2) \tag{A.1}$$

ここで n は探索空間の次元を、$N(0, \sigma^2)$ は平均 0, 分散 σ^2 の正規分布を、$\boldsymbol{e}_i$ は主探索直線に直交する部分空間の正規直交基底ベクトルをそれぞれ表す。

このアルゴリズムのパラメータは経験的に $\sigma_\xi = 1/2$, $\sigma_\eta = 0.35/\sqrt{n}$ が推奨されている。

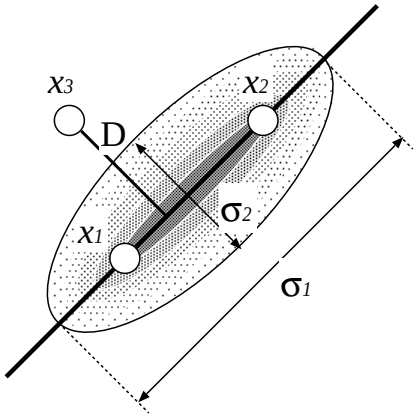


図 A.1: UNDX による交叉

Crossover	Success	Average(50[s] Over)	Max Time[s]
UNDX	8 / 10	45.3 / 512	87.3

(Fitness Function No.2, Hudden layer=6, Loop=1)

表 A.1: UNDX を用いた学習結果

A.2 UNDX を用いた学習結果と検討

表 A.1 に UNDX を用いた学習結果を示す。交叉率 0.5, 突然変異率 0.1, 個体数 100, 最大世代数 300 で、適応度関数は No.2 を用いて Feedforward 型 (Hidden layer=6) で学習を行った。

学習結果を見ると、本研究の手法とくらべて変化は見当たらない。しかし、最大世代を 300 までと限定しているため、さらに世代を重ねた場合についても検討する必要がある。また、UNDX と相性のよいとされる世代交代モデルとして MGG モデル [12] の使用した場合についても検討する必要がある。