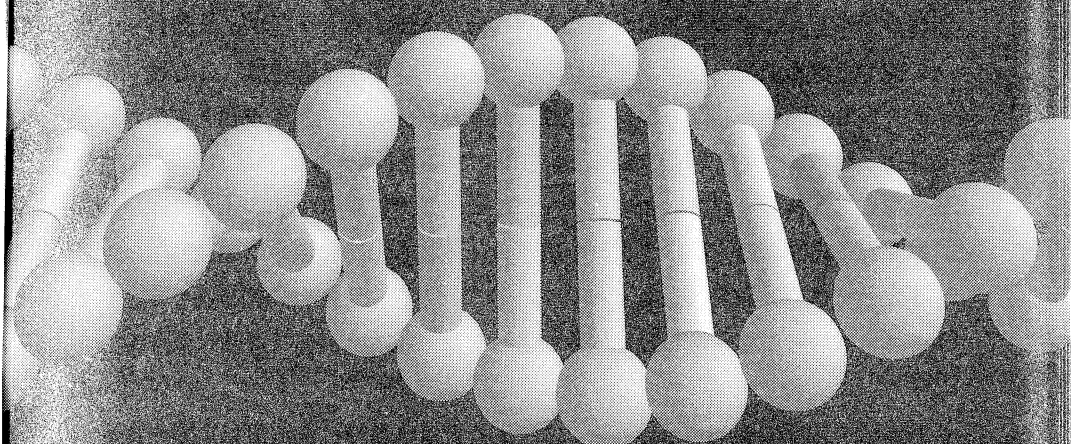
情★報★科★学

遺伝的アルゴリズムの方法

メラニー・ミッチェル 著／伊庭斉志 監訳
本堂直浩，伊藤拓也，丹羽竜哉，
高畠一哉，野添敏秀 訳



TDU 東京電機大学出版局

の戦略は，お互い同士うまくいき，初期の協調戦略に見られたように協調的でない戦略に完全に打ち負かされることはなかった．返礼戦略は，平均を上回る成績のため集団内に広まっていく．この結果，協調戦略の増加と適合度の増加をもたらした．

Axelrod の実験は，興味深い問題に対する解を進化させ，かつ理想的な方法で進化と共進化をモデル化するために GA を用いる方法を示した．この他にも多くの実験が考えられる．例えば，交叉の確率を 0 にしたような GA を実行する，すなわち選択と，突然変異のみを遺伝的操作として採用する（Axelrod 1987），あるいは戦略のメモリの使用が増減可能なより柔軟な進化を許すこと（Lindgren 1992）などである．

1.9.2 宿主と寄生虫：GA を用いたソーティングネットワークの進化

効率的に要素を順に並べること，すなわちソーティングアルゴリズムを設計することは，計算機科学において基本的な問題である．Donald Knuth (1973) は，彼の古典的シリーズである *The Art of Computer Programming* において，その半分以上の 700 ページを割いてこの問題を扱っている．ソーティングの目的は，データ構造（例えば，リスト構造や，木構造など）の要素をある規則的な順序（例えば，数字順やアルファベット順など）にしたがって，最小時間で並び替えることである．Knuth の著書において紹介されたソーティングのための 1 つのアルゴリズムとして，ソーティングネットワークがある．これは，ある有限の数 n 個の要素をもつリストのソーティングを並列デバイスを用いて行うものである．図 1.4 は，$n = 16$，すなわち $(e_0 - e_{15})$ の要素を持つリストのソートを行うためのネットワーク（「Batcher ソート」–Knuth 1973 を参照）を表している．それぞれの横線は，リスト中の要素を表している．また縦矢印は，2 つの間の要素間でなされる比較を表している．例えば，最も左の矢印の列は，e_0 と e_1 との間，e_2 と e_3 との間などでなされる比較を示している．もし比較した各要素が目的とする順序になっていない場合，それらは並び換えられる．

要素のリストをソートするためには，次々と列を移動して縦の列に記述された比較をしながら（必要であれば置き換えながら），ネットワーク中を左から右へと移動する．縦の列の比較は独立であり，並列に行うことが可能である．もしネットワークが正しければ，いかなるリストも正しくソートすることができる．ソーティングネットワークの最終的な目的は，ソートを正しく，効率的（すなわち最小の比較の回数）に行うことである．

ここでの興味深い理論的な問題は，与えられた n 個の要素を持つリストを正しくソートするための，必要最小限の比較回数を決定することである．60年代，$n = 16$ の場合をめぐって論争が行われた（Knuth 1973；Hillis 1990, 1992）．Hillis (1990) によると，1962年に Bose と Nelson は $n = 16$ のリストをソートするために65回の比較が必要なソーティングネットワークの開発を行った．そして彼らは，この値が $n = 16$ の場合に最小であると推測した．1964年，Batcher と，Floyd，Knuth は独自にわずか63回の比較で十分なネットワークを開発した．このネットワークは，図1.4に示したとおりである．これは最小であると再び考えられたが，1969年，Shapiro によって62回の比較によってソート可能なネットワークが開発された．さらに同年，Green により60回比較のソート可能ネットワークが開発されたため，この時点で，誰もがネットワークの最適化につ

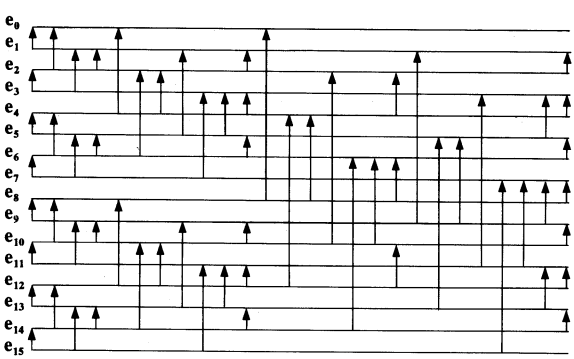


図 1.4 $n = 16$ であるソーティングネットにおける "Batcher Sort" (Knuth 1973)．各横線はリスト中の要素，縦矢印は，2つの要素間で行われる比較を表している．比較された要素の順序が違っているなら，それらは置換される．同じ列で行われた比較は並列に実行される．

いて推測は不可能であると考えた．n=16 のソーティングネットワークの設計という狭い領域においてこれが最も活発な時期であった．その最適性についての証明は与えられなかったが，Green による発見後，この論争は沈静化した．

1980 年代に **W. Daniel Hillis** は GA という助けを借りて，再びこの問題にチャレンジした（Hillis 1990, 1992）．特に Hillis は，$n = 16$ のソーティングネットワークの最適化設計問題を並列コネクションマシン（CM2）上で実行する GA に解かせてみた．囚人のジレンマと同様に，最初のステップはソーティングネットワークを文字列にコード化することから始まる．Hillis によるコード化はかなり複雑であり，多くの GA の応用で用いられているよりも，より生物的に現実的なものとなっている．ここでは，それのしくみについて述べる．

ソーティングネットワークは，次のような順序化されたリストのペアで記述される．

$$(2,5),(4,2),(7,14)....$$

これらのペアは，「まず 2 と 5 を比較し，必要ならば，置き換える．次に 4 と 2 を比較して，必要ならば置き換える」というような，一連の比較を表現している (Hillis によるコード化は，並列に比較するように記述されていない．なぜなら，最適な並列ソーティングネットワークの構築することというより，比較数の最小化を目的としているためである）．生物学的アナロジーに従うなら，Hillis はネットワークを表現する順序化つきリストを「表現型」とした．Hillis によるプログラムでは，それぞれの表現型は 60 から 200 のペアで構成され，それぞれはネットワークでの比較に対応している．現実の遺伝子と同様に，GA は表現型上に機能するのではなく，表現型をコード化した遺伝子型上に機能する．

GA における集団中の個体の遺伝子型は，表現型にデコード化される染色体の集合から構成される．Hillis は一般によく用いられている一倍体染色体（単一染色体）ではなく，二倍体染色体（ペアの染色体）を用いた．図 1.5 (a) に示したとおり，それぞれの個体は 32 ビットの染色体の 15 のペアから構成されている．また図 1.5 (b) では，ぞれぞれの染色体は 8 つの 4 ビットからなるコドンから構成され

10110101011110011110010010101001	01001100101101001111010001100011	01100111100000001101001101000111
10110101001001110011110010101001	01010101011101001100011111001100	01101011011110011000000110100100
11010100111101010011110111011110	00011101001011110000101010110111	01100111101100110111100111001111
10111011010001010000000100010010	11000110100001010010111111111100	11100000101101011000011100101111
01011000101000110001110110111001	11011011001110101001001101010110	11010011011101001011000000010010
11111000111001001110010101001001	00110000110011001010111110000110	00011111101111100100110110111001
11010100111101010011110111011110	10111001100010010010101011011000	01010111001011011001110101101000
10111011010001010000000100010010	00010000011101011101010101100011	01010001000001100001101011111111
10100010110111110001110100101001 1	01100111100000001101001101000111	01011101001000000001011111111111
00010111111110000110000000101001 0	01101011011110011000000110100100	00101100000110100000000110010110

(a)

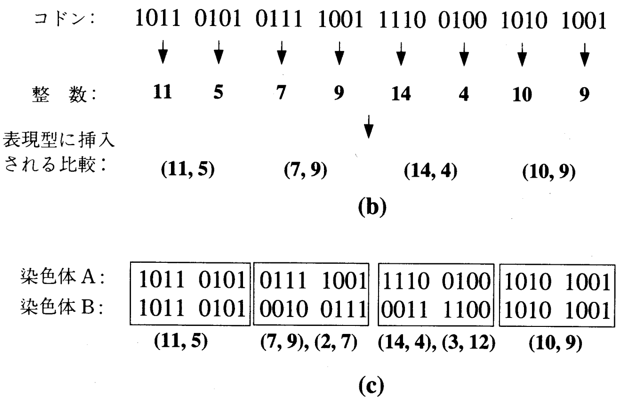


図 1.5　Hillis の実験で用いられたソーティングネットワークの遺伝子型表現の詳細. (a) 32 ビットの染色体の 15 のペアから構成されるソーティングネットワークのための遺伝子型の例. (b) 単一の染色体にコード化された整数の例. ここで与えられた染色体は, 11, 5, 7, 9, 14, 4, 10, 9 をコード化する. 隣接した整数の組は, 比較されるものとして解釈される. (c) 染色体の組によりコード化される比較の例. ここで与えられたペアは, 2 個所の同一の位置表現を持つため, 表現型に挿入されるのは合計 6 個の比較である. つまり, (15,5), (7,9), (2,7), (14,4), (3,12) および (10,9) となる.

ている. それぞれのコドンは, 16 の要素を持つリストの位置を示す整数 0 から 15 を表現する. 染色体中のぞれぞれの隣接したコドンのペアは, 2 つの要素間の比較を記述している. すなわち染色体は, 4 つの比較をコード化している. 図 1.5 (c) に示したとおり, それぞれの染色体のペアは 4 つから 8 つの間の比較をコード化する. 染色体のペアは整列され, 左から右へ読み出される. それぞれの位置において, 染色体 A 中のコドンのペアは, 染色体 B 中のコドンのペアと比較される. もし, それらが同じ数のペアをコード化するなら (すなわち, ホモならば), そのときは唯一つのペアだけが表現型に挿入される. もし, 異なる数のペアをコード

化しているなら（すなわちヘテロならば），その場合は両方のペアが表現型に挿入される．したがって染色体の15のペアは，60から120の比較を行う表現型を生成するために決まった順序で読み出される．それぞれの染色体のペアにホモな位置が多くなることは，結果的にソーティングネットワークに現れる比較の数が減少することを意味する．ここでの目標は，Greenによるネットワークと等しい最小ソーティングネットワークを発見することであり，GAは正しいソーティングを行う遺伝子型中においてすべてのホモの位置を持つ個体を発見する必要がある．Hillisのコード法では60回より少ない回数でのソーティングネットワークは発見できないことに注意しよう．

Hillisによる実験では，初期集団は遺伝子型からランダムに生成された個体から構成される．ただし以下の例外がある．知られている16の要素最小のソーティングネットワークの多くは，同じ32の比較パターンから開始されることにHillisは気づいた．したがって，最初の8つの染色体のペアをこれらの比較をコード化するためにセットした．これは，GAの探索を容易にするために問題領域（ここではソーティングネットワーク）の知識を使用する例である．

ランダムな初期集団から得られるネットワークの多くは，正しいネットワークではなく，すべての入力（16の要素を持つリスト）を正しくソートするものではない．Hillisによる適合度の基準は部分的な割り当てに基づく．すなわち，ネットワークに対する適合度は，リスト中の正しくソートした要素の割合と定義する．ネットワークには非常に多くの入力事例があるため，全例を試すことは実際的には困難である．したがって，各世代において，それぞれのネットワークはランダムに選択された入力事例によってテストされる．

HillisによるGAは，これまで述べた単純GAとちがって，かなり修正されている．初期集団における個体は2次元格子状に配置され，単純GAとは異なり文字列間の距離という空間的な概念が導入されている．空間的な格子に集団を配置することの目的は，集団内の「種形成」を助長させることである．すなわち，すべての個体が似たようなネットワークに収束するというより，むしろ，異なるタイプのネットワークが，別の空間的配置によって生成されることを目指した．

各個体の適合度は，ランダムに選択されたテスト事例によって計算される．このとき，適合度の低い半数の個体が淘汰され，生き残った高い適合度をもつ隣接する個体に置き換えられる．すなわち，高い適合度を持つ個体には複製の機会が与えられる．

次に，各個体は子孫を作るために空間的に隣接した他の個体とペアとなる．二倍体生物における組み替えは，一倍体による交叉とは異なる．図1.6に示すとおり，交叉が染色体のペア間で実行される．15個の染色体のペアで，交叉点は，ランダムに選択され，第1の染色体のペアの前半のコドンと，第2の染色体のペアの後半のコドンとを置き換えることにより，1つの「配偶子」が生成される．結果として，それぞれの親から，15個の一倍体配偶子を生じる．第1の親から生成された15個の配偶子は，第2の親から得られた15個の配偶子のペアになり，1つの二倍体の子孫を作り出す．この手続きは自然界に見られるような二倍体生物間での有性生殖におおよそ似通っている．

このような交配が，新しい集団が構成されるまで行われる．新たな集団の個体

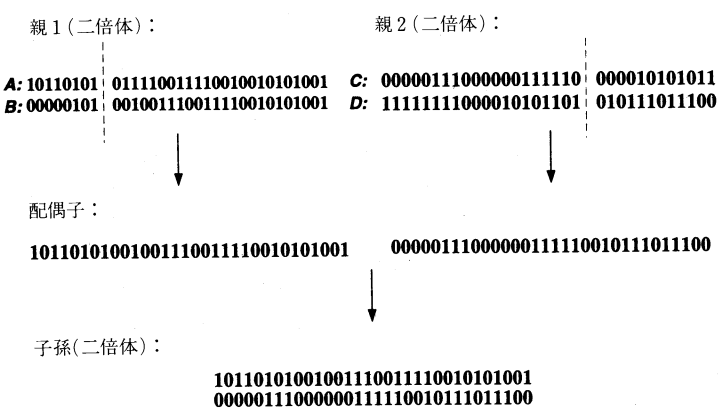


図 1.6 Hillisの実験で用いられた二倍体の繁殖．ここでは，個体の遺伝子型は15の染色体のペアを持つ（簡単のため，各親に対して1つのみを示した）．交叉点はそれぞれのペアに対してランダムに選択され，配偶子が第1の染色体の交叉点の前半部のコドンと，第2の染色体の交叉点の後半部のコドンを組み合わせることにより構成される．1つの親からの15の配偶子は，新たな個体を生成するために他の親からの配偶子とペアになる（簡単のため1つの配偶子のペアが示されている）．

は，確率 $p_m = 0.001$ によって突然変異が施される．以上のプロセスが，長い世代の間繰り返される．

ネットワークの大きさとは無関係に，ネットワークの正しさのみによって適合度が決定されるため，最小のネットワークを生成するのにどのような困難があるだろうか？ Hillis は，最小化に関して間接的な淘汰圧があると説明していた．それは自然におけるように，ホモ接合体が重要な比較を守ることができるからである．もし染色体中のヘテロな位置に重要な比較があれば，それは交叉によって失われるだろう．一方，ホモな位置において重要な比較があれば，交叉によって失われることはない．例えば，図 1.6 において染色体 B の左端の比較（すなわち，比較 (0, 5) をコード化した左端の 8 ビット）はヘテロな位置にあり，複製において失われている（その配偶子は，染色体 A の左端の比較を得る）．しかしながら，染色体 A の右端の比較は，(配偶子は染色体 B から右端の比較を得ているにもかかわらず）ホモな位置にあるため，その染色体は保存されている．一般的に，重要な比較が発見された場合，それをホモな位置に置くことは非常に有効である．そして，ホモの位置が増加するにつれ，得られるネットワークのサイズが減少する．

並列コネクションマシンを有効に活用するために，Hillis は 512 から 100 万の個体を有する非常に巨大な集団を使用した．実行は 500 世代まで行った．GA によって得られた最小のネットワークは，Bose や Nelson のネットワークと同じ 65 回の比較を行うものであり，Green のネットワークより 5 回多いネットワークを発見した．

Hillis はこの結果に失望した．「なぜ GA を用いて良い結果が出ないのか？」 GA は適合度ランドスケープにおける最適解に収束するのではなく，局所解，すなわち，適合度ランドスケープの局所的頂上に収束しているためであることが判明した．GA は数多くのかなり良い解（65 回の比較）を発見したが，それ以上に良い解は発見できなかった．理由の 1 つは，適合度を計算するために使用されるテスト事例が十分に変化しないことにある．ネットワークはテスト事例に機能する戦略を発見するが，世代が進んでもテスト事例の困難さは変化しないからである．すなわち，初期世代後テスト事例が変化しないため，局所解である戦略を変化させ

るための圧力が存在しないからである．

この問題を解決するために，Hillisは生物学から別のアナロジーを採用した．すなわち，「宿主–寄生虫」の関係にみられる**共進化**現象である．生物界には次に示すような例を数多く見ることができる．宿主である生物は寄生虫から身を守るための防御を進化させ，また寄生虫はその防御を打ち破るために進化する．結果として，宿主は新たな進化を行い，こうして無限に続く「生物的軍拡競争」となる．Hillisのアナロジーによれば，ソーティングネットワークを宿主，テスト事例を寄生虫とみなすことができる．Hillisは，ネットワークの集団が寄生虫の集団と同じグリット上で共進化を行うように修正を加えた．ここで，寄生虫は10から20のテスト事例の集合からなる．また，どちらの集団もGAのもとで進化を行う．ネットワークの適合度は，同じグリッド位置に配置された寄生虫での事例中で正しくソートした割合である．寄生虫の適合度は，ネットワークを困惑させた（ネットワークが正しくソートできなかった）事例の比率として与えられる．

テスト事例の集団の進化は，ネットワークの集団の進化を促進させる．ネットワークが正しくソートすればするほどテスト事例は困難になり，ネットワークの弱点を突くように特殊化した進化を行う．この作用はネットワーク集団の変化を促す．すなわち，同じ局所解に落ち着くのではなく，常に新たな戦略を求めて変化を続けることになる．この結果共進化を用いることにより，61回の比較でソートが可能なネットワークを発見した．しかしながら，Greenのネットワークを上回ることには失敗した．

生物学における共進化によって触発された有効なGAのテクニックを導き出したという点で，Hillisの研究は非常に重要である．そして彼の研究結果は，生物学的ひらめきが有効であることを確信させた．しかしながら，宿主–寄生虫の考え方は非常に魅力的ではあるが，その有効性はHillisの研究以外では確立されていない．そして一般にどのような対象に適用可能か，あるいはより困難な問題（例えば，より大きなソーティングネットワーク）にどの程度応用可能かという点に関しては，明らかになっていない．多くの研究がこの興味深い領域において行われるべきである．

2.3 ニューラルネットワークの進化

ニューラルネットワークは生物学的に由来する機械学習の方法で，神経科学からアイディアを得たものである．最近，GA を使って，ニューラルネットワークの形態を進化させる研究が行われた．

最も単純な**「前向き」方式**（図 2.16）では，ニューラルネットワークは，実数値で重みづけられた結合で連結される活性値ユニット（**ニューロン**）の集まりである．ネットワークは識別すべき画像の特徴集合のような（例えば，アルファベットの手書き文字の画像の画素），入力ユニットの活性化したパターンで与えられる．活性値は重みづけられた結合を介して，入力ユニットから中間（隠れ）ユニットの層を通って出力ユニットへ前向きに広がる．一般的に，あるユニットから別のユニットへ入る活性値は連結の重みで乗ぜられ，そして他から入ってくる活性値が加えられる．出力結果は一般的に閾値で制限される（もし結果の活性値がユニットの閾値よりも大きいならば，そのユニットは「活性化する」）．このプロセスは，脳でニューロンのネットワークを介して活性化が広がる方法を大まかに真似たものである．前向きネットワークでは，活性値は入力層から中間層を経て出力層への前方向にしか広がらない．また多くの研究者は，層間の前向き結合と同じように，後向き結合のある**「リカレント」ネットワーク**で実験を行った．

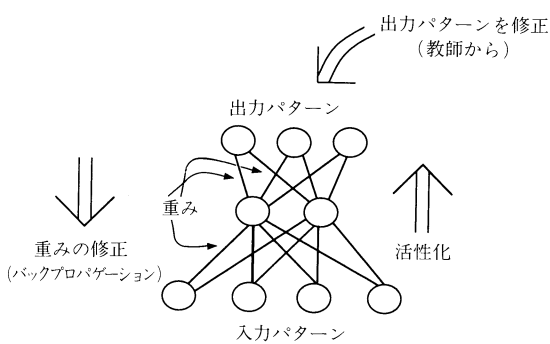


図 2.16 単純な前向きニューラルネットワークと重み値が調整されることによるバックプロパゲーション処理の概略図．

前向きネットワークで活性値が広がった後，出力ユニットでの活性値パターンは入力に対するネットワークの「答え」（例えば，文字Aの入力パターンの識別）をコード化する．ほとんどの応用では，学習アルゴリズムによって，ネットワークは入力と出力パターンの間の正しいマッピングを学習する．一般的に，重みは最初は小さな乱数値としてセットされる．それから，訓練入力データの集合がネットワークに逐次的に与えられる．**バックプロパゲーション学習手続き**（Rumelhart, Hinton, and Williams 1986）では，各入力がネットワークを経て伝えられ，出力が生成され，「教師」が各出力ユニットの活性値と正しい値を比較する．そして，ネットワークの重みはネットワークの出力と正しい出力との差を減らすために調整される．この手続きの各繰り返しは**訓練サイクル**と呼ばれ，訓練入力データの集合による訓練サイクルの段階は**訓練期間**と呼ばれる（一般的に，多くの訓練期間が与えられた訓練入力の集合をうまく識別して学習するために必要とされる）．学習プロセスを制御するために，正しい出力値を与えることで教師は学習を監督するので，この手続きは**教師あり学習**として知られている．**教師なし学習**では教師がなく，学習システムは自らの性能に対する大まかな（そして時には信頼性のない）環境のフィードバックによって，自分自身で学習しなければならない（ニューラルネットワークとその応用の概要については，Rumelhart et al. 1986, McClelland et al. 1986 およびHertz, Krogh, and Palmer 1991 を参照）．

ニューラルネットワークにGAを適用する方法は数多くある．進化するのは固定したネットワークでの重みや，ネットワーク構造（すなわち，ユニットの数とそれらの内部結合）とネットワークによって使われた学習ルールである．以下では，GAを使った4つの異なるプロジェクトを紹介する（ネットワーク構造を進化させる2つの方法が紹介されている．GAとニューラルネットワークの様々な組合せに関する論文の収集については，Whitley and Schaffer 1992 を参照）．

2.3.1 固定したネットワークでのニューラルネットワークの重みの進化

David Montana and Lawrence Davis (1989) は固定した結合ネットワークで重みを進化させる最初のアプローチを採用した．すなわち，Montana と Davis は結合の固定集合に対する良い重みの集合を見つける方法として，バックプロパゲーションのかわりに GA を使った．バックプロパゲーションアルゴリズムに関係した問題（例えば，重み空間で局所最適なところに陥る傾向や，いくつかのタスクで学習を監督するための「教師」が利用できないこと）が指摘されているので，かわりの重み学習手法を見つけることは望ましいことである．

水面下の音波の “lofargram”（スペクトログラムに似たようなもの）を，「興味ある」と「興味ない」の 2 つのクラスに識別するために，ニューラルネットワークを使うことに Montana と Davis は関心を持っていた．目的は「様々な種類の音響的なノイズと海での干渉の中から興味ある信号を検出して判断する」ことである．lofargram を含んだデータベースとそれの専門家による「興味ある」かどうかの識別によってネットワークが訓練された．各ネットワークは，同様の識別を実行するエキスパートシステムによって使われた，4 つのパラメータを表現する 4 つの入力ユニットがあった．各ネットワークには 1 つの出力ユニットがあり，2 つの隠れユニットの層（1 つ目は 7 ユニットで，2 つ目は 10 ユニット）があった．ネットワークは完全に前向きネットワーク結合されていた．すなわち，各ユニットは次のより高い層のすべてのユニットに連結されていた．したがって，ユニット間で合計 108 個の重みづけられた結合があった．さらに，各非入力ユニットに対して閾を実装するための「しきいユニット」への結合として，18 個の重みづけられた結合があり，したがって合計 126 個の重みがあることになる．

GA は以下のように用いられた．各染色体は 126 個の重みのリスト（もしくは「ベクトル」）であった．図 2.17 はどのように染色体にエンコードされたのかを（より小さなネットワークに対して）示している．重みは固定された順番（左から右かつ上から下）でネットワークを読み上げられ，リストに置かれる．ここで注

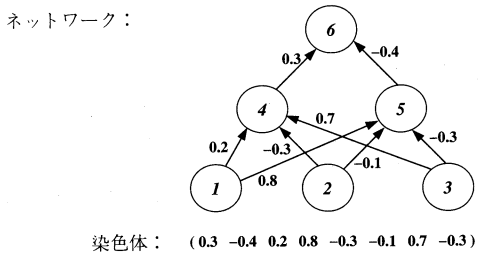


図 2.17 Montana と Davis の GA の染色体として使われるリストに対するネットワークの重みのエンコード．ネットワークでのユニットは後で参照するために番号が付けられている．リンク上の実数値が重みである．

意しなければならないことは，染色体の各「遺伝子」はビットではなく実数であることである．与えられた染色体の適合度を計算するために，染色体での重みは相当するネットワークの結合に割り当てられた．そして，そのネットワークは訓練集合（ここでは，lofargram のデータベースから 236 例）で実行され，(すべての訓練サイクルにわたって集められた）誤差の 2 乗和が返された．ここで，「誤差」は望ましい出力活性値と実際の出力活性値との差であった．したがって，少ない誤差は高い適合を意味する．

50 の重みベクトルの初期集団がランダムに，各重みが −1.0 から +1.0 の間になるように選ばれた．Montana と Davis は様々な実験で多くの異なる遺伝操作を試みた．GA とバックプロパゲーションの比較のために用いた突然変異と交叉が図 2.18 と 2.19 に示してある．突然変異操作は n の非入力ユニットを選び，これらのユニットへの各入力結合に対して，結合の重みに −1.0 から +1.0 の間の乱数値を加える．交叉操作は 2 つの親重みベクトルを取り，各非入力ユニットに対してランダムに親の 1 つを選び，親から子へ入力結合の重みをコピーする．ここで注意しなければならないことは，1 つの子のみ生成されることである．

これらの操作を使った GA の性能はバックプロパゲーションアルゴリズムの性能と比較された．GA は 50 の重みベクトルの集団からなり，ランク方式選択が使われた．また 200 世代を実行するように設定されていた（すなわち，10,000 ネットワークの評価）．バックプロパゲーションアルゴリズムは 5000 回の反復を実行

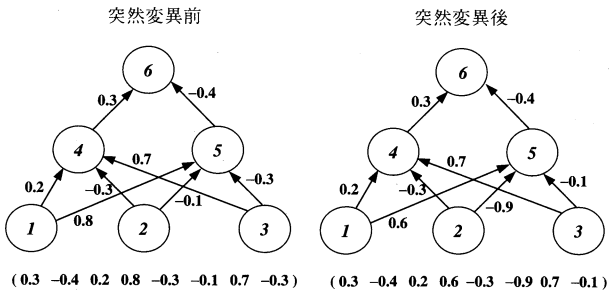


図 2.18 Montana と Davis の突然変異手法．ここでユニット 5 に入る結合での重みが突然変異を受けている．

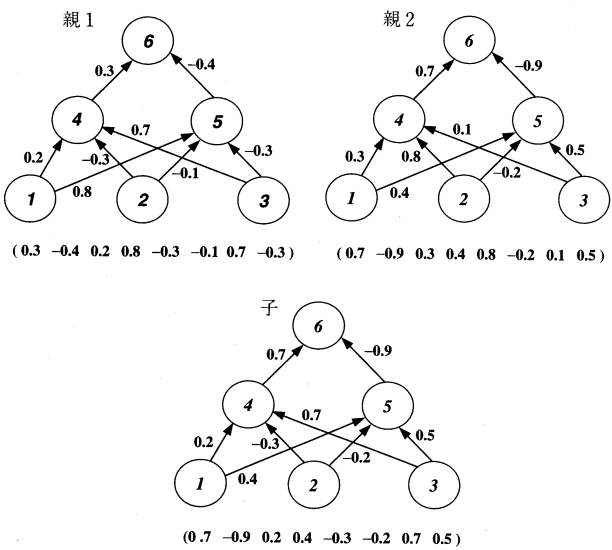


図 2.19 Montana と Davis の交叉方法．子は次のようにして作られる．各非入力ユニットに対して，親はランダムに選ばれ，ユニットへの入力結合での重みは選ばれた親からコピーされる．ここで示されている子のネットワークでは，ユニット 4 への入力結合は親 1 から来て，ユニット 5 と 6 への入力結合は親 2 から来ている．

するように設定されていた．ここで，1回の反復は1つの完全な期間（訓練データでの1つの完全な段階）のことである．与えられた訓練例でのバックプロパゲーションは，活性値の前向き伝搬（そして，出力ユニットでの誤差の計算）と後向きの誤差の伝搬（そして，重みの調整）の2つの部分から構成される．そのため，MontanaとDavisはGAのもとでの2回のネットワークの評価が1回のバックプロパゲーションの反復と等価であると推論した．GAは1番目の部分のみ実行する．2番目の部分はより計算量を必要とするため，2回のGAの評価は1回のバックプロパゲーションの反復の半分以下の計算量となる．

比較の結果は図2.20に示してある．ここでは，1回のバックプロパゲーションの反復が2回ごとのGAの評価に対してプロットされている．x軸は反復回数を示し，y軸はその時点で見つけられた最良の評価値（誤差の2乗和の最小値）を示している．GAはバックプロパゲーションに対して，より早く良い重みベクトルを得ているため，かなり優秀であることがわかる．

この実験は，ある状況で単純なバックプロパゲーションよりGAは良い訓練手法であることを示している．これはすべての場合においてGAがバックプロパゲー

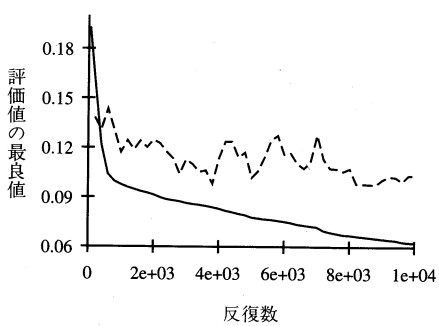


図 2.20 バックプロパゲーションとGAの性能を比較したMontanaとDavisの結果．図は見つかった最良の評価値（低いほど良い）をプロットしてある．実線：GA，破線：バックプロパゲーション．(*Proceedings of the International Joint Conference on Artificial Intelligence;* ©1989 Morgan Kaufmann Publishers, Inc. Reprinted by permission of the publisher).

ションよりも優れていることを意味しているのではない．バックプロパゲーションの能力を高めることで，GA と同じ位良い成績を与える可能性もある．Schaffer, Whitley, and Eshelman (1992) は，GA は教師あり学習のタスクで最良重み調整手法（例えば，**クイック・プロップ**）に対して優れていなかったことを指摘した．しかし，学習システムに対して各出力ユニットでの誤差が利用できない教師なし学習のタスク，すなわち，強化があまり利用できない状況で，バックプロパゲーションなどの手法が使えないタスクで重みを見つけるのに GA は最も有用になるだろうと予測している．これは，例えばニューラルネットワークが不案内な環境でロボットを操縦するような，複雑なシステムを制御するために使われる，**ニューロコントロール**などである．

2.3.2 ニューラルネットワークの構造の進化

Montana と Davis の GA は固定したネットワークで重みを進化した．ほとんどの応用では，ネットワークの構造—ユニットとそれらの間の内部結合の数—はプログラマーによって適当に，また，しばしばいくつかのヒューリスティクス（例えば,「より多くの隠れユニットはより難しい問題に対して必要とされる」）と試行錯誤によって前もって決められる．ニューラルネットワークの研究者は，特定の構造の選び方が成功もしくは失敗を決定するのをよく知っているので，特定の応用に対して構造の決定の手続きを自動的に最適化することを強く望んでいるだろう．多くの研究者は GA がこのタスクによく適していると考えている．これらの手法は大まかに言って**直接コード化**と**文法コード化**の 2 つのカテゴリに分類される．これらの方針に沿ったいくつかの研究があった．直接コード化では，ネットワーク構造は直接的に GA の染色体にコード化される．文法コード化では，GA はネットワーク構造を進化させないが，ネットワーク構造を発展させるために使われる文法を進化させる．

2.3.3 ニューラルネットワークの直接コード化

直接コード化の方法は Geoffrey，Miller，Peter Todd および Shailesh Hegde

(1989) によって行われた．彼らは初期の研究においてGAが結合トポロジーを進化させる対象を一定数のユニットからなる前向きネットワークに制限した．図2.21に示すように，結合トポロジーは各要素が「ユニットから」から「ユニットへ」へ結合のタイプをコード化する，$N \times N$ の行列（図2.21では 5×5）で表現された．結合行列での各要素は「0」（結合なしを意味する）か「L」（「学習可能な」結合，すなわち，学習を通して重みが変化可能なものを意味する）かのどちらかである．さらに図2.21はどのように結合行列がGAに対する染色体に変換されたか（「0」は0に,「L」は1に一致する）およびどのようにビット列がネットワークにデコードされたかを示している．学習可能であることを明記された結合は小さな乱数の重みで初期化された．Miller, Todd と Hedge はこれらのネットワークを前向きに制限したので，染色体で明記されるいかなる入力ユニットへの結合あるいはフィードバック結合も無視される．

Miller, Todd と Hegde は簡単な適合度比例選択と突然変異（列中でのビットは低い確率で反転された）を用いた．交叉操作では，ランダムに列インデックスを選び，2つの親の間でその列を交換して2つの子を生成した．このオペレータの背景にある直観は Montana と Davis のものと似ている．つまり，各列はあるユニットへのすべての入力結合を表現しており，この集合がネットワークの機能的積み木と考えられた．染色体の適合度は Montana と Davis のプロジェクトと

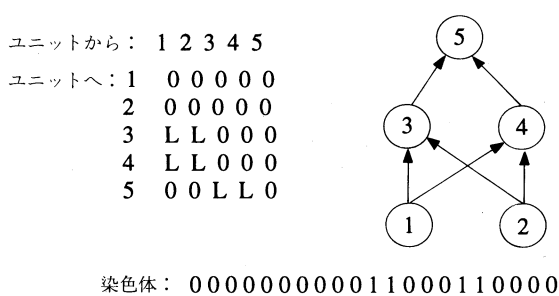


図 2.21 Miller, Todd と Hegde の表現手法の図．行列での各要素は「ユニットから」（列）と「ユニットへ」（行）の間の接続での結合の種類を表現している．行列の列は図の下に示してあるビット列のコード化を作るために並べられる．結果のネットワークは右側に示してある（Miller, Todd, and Hegde 1989 からの改編）．

同じ方法で計算された．すなわち，ネットワークは重みを修正するためにバックプロパゲーションを使って，訓練集合で訓練された．染色体の適合度は最終期間での訓練集合に対する誤差の自乗和であった．少ない誤差は高い適合度に変換される．

Miller，Todd と Hegde は 3 つのタスクで彼らの GA を試みた．

XOR　2 つの入力ユニットの初期値（1 はオン，0 はオフ）の排他的論理和が 1 ならば，1 つの出力ユニットはオンになる（すなわち，活性値は閾値より高くなる）．

Four Quadrant　2 つの入力ユニットの実数値での活性値（0.0 から 1.0 の間）は単位正方形での点の座標を表現する．正方形の左下と右上の四分円での点を表現する入力に対して出力ユニットは 0.0，他の点に対しては 1.0 の活性値となる．

エンコーダー/デコーダー（パターンコピー）　出力ユニット（入力ユニットと数が同じ）は入力ユニットでのパターンをコピーしなければならない．これは簡単に思えるが，隠れユニットの数が入力ユニットの数よりも小さな場合は，コード化とデコード化を行わなければならない．

これらをバックプロパゲーションで解くことは多層ニューラルネットワークにとって，比較的簡単である．ネットワークは異なるタスクに対して異なるユニット数であった（XOR のタスクでの 5 ユニットからエンコーダー/デコーダーのタスクでの 20 ユニットまでの範囲）．目的はそれぞれのタスクに対して，GA が良い結合トポロジーを発見できるかどうかを確かめることである．各実験において，集団のサイズは 50，交叉率は 0.6，そして突然変異率は 0.005 である．すべてのタスクで，GA はほとんど誤差がなく入力から出力へのマッピングを簡単に学習するネットワークを容易に見つけることができた．しかし，3 つのタスクはこの方法を正確にテストするにはあまりにも簡単すぎた．より多くの内部結合で，かつより大きなネットワークが必要とされる複雑なタスクに，この方法がスケール

アップできるかどうか確認すべきである．簡単のため，Miller，Todd と Hegde の研究を選んでこの方法を説明した．直接コード化を用いたネットワーク構造を進化させるためのより洗練された方法のいくつかの例を知るには，Whitley and Schaffer 1992 を参照していただきたい．

2.3.4 ニューラルネットワークの文法コード化

文法コード化の方法は Hiroaki Kitano (1990) の研究によって実証された．彼は直接コード化法は望ましいネットワークのサイズが増加したとき，使うことが困難になると指摘した．ネットワークのサイズが大きくなると，必要とされる染色体のサイズが急速に増加し，性能（どのように高い適合度が得られるか）と効率（どれ位の期間で高い適合度を得ることができるか）の両方が問題となる．さらに，直接コード化法ではネットワークの各結合を明確に表現するので，繰り返した，あるいは入れ子になった構造は，効率的に表現できない．

Kitano および他の研究者によって追求された解法は，文法としてネットワークをコード化することである．GA は文法を進化させるが，適合度は文法からネットワークが発達した「発達」ステップの後にテストされる．すなわち，「遺伝型」は文法で，「表現型」は文法から引き出されたネットワークである．

文法は構造の集合を生成するために適用されるルールの集合である（例えば，自然言語の文，コンピュータ言語でのプログラム，ニューラルネットワークの構造）．簡単な例は以下の文法である．

$$\begin{aligned} S &\rightarrow aSb, \\ S &\rightarrow \epsilon \end{aligned}$$

ここで，S はスタートシンボルで非終端記号であり，a および b は終端記号，そして ϵ は空の文字列の終端記号である（$S \rightarrow \epsilon$ は S が空の文字列に置き換わることを意味する）．この文法から構造を作るには，S でスタートし，右側で与えられた許可された変換の 1 つでそれを置き換える（例えば，$S \rightarrow aSb$）．そして，結果の構造を取り，許可された変換の 1 つで任意の非終端記号（ここでは S）を

置き換える（例えば，$aSb \rightarrow aaSbb$）．左側に非終端記号がなくなるまで，このことを続ける（例えば，$S \rightarrow \epsilon$ を使って $aaSbb \rightarrow aabb$）．この文法によって作られた構造の集合は，左側に a が右側に b が同数ある文字列から構成される $a^n b^n$ であることが簡単にわかる．

Kitano はこの一般的なアイディアを**グラフ生成文法**と呼ばれる文法を使って，ニューラルネットワークの生成に適用した．簡単な例は図 2.22 (a) に示されている．ここで，各ルールの右側は 1 次元の列ではなく，2×2 の行列である．大文字は非終端記号で，小文字は終端記号である．a から p の各小文字は 16 通りの 1 と 0 の 2×2 の配列の 1 つとして表現される．上の $a^n b^n$ に対する文法と違って，この文法での各非終端記号はひと通りの書き換え法しかない．したがって図 2.22 (b) に示した 8×8 の行列の文法から形成される構造は 1 つしかない．この行列がニューラルネットワークに対する結合行列として解釈される．すなわち，列 i と行 i での 1 はユニット i はネットワークで存在することを意味し，列 i と行 j，$i \neq j$ での 1 はユニット i から j への結合があることを意味する（Kitano の実験では，存在していないユニットへの，あるいは存在していないユニットからの結合，およびリカレント結合は無視された）．結果は図 2.22 (c) で示したネットワークであるが，それは適切な重みでブール関数 XOR を計算するものである．

Kitano の目的は GA に上のような文法を進化させることであった．図 2.23 は図 2.22 (a) の文法をコード化した遺伝子を示したものである．染色体はそれぞれ，5 つの遺伝子座からなる分離したルールに分割される．1 番目の遺伝子座はルールの左側であり，2 番目から 5 番目の**遺伝子座**はルールの右側での行列での 4 つのシンボルである．それぞれの遺伝子座での可能な**対立遺伝子は** $A-Z$ と $a-p$ のシンボルである．染色体の最初の遺伝子座はスタートシンボル S で固定されている．S を 2×2 の行列にする少なくとも 1 つのルールがネットワークの構築を開始するためには必要である．すべての他のシンボルはランダムに選ばれる．前もって決められた反復の回数に対して，染色体でコード化された文法ルールが適用されネットワークが構築される（$a-p$ を 0 と 1 からなる 16 通りの 2×2 の行列に変換するルールは固定されているので，ルールは染色体では表現されない）．

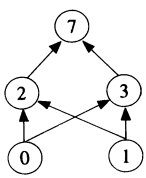
$$S \rightarrow \begin{matrix} A & B \\ C & D \end{matrix} \quad A \rightarrow \begin{matrix} c & p \\ a & c \end{matrix} \quad B \rightarrow \begin{matrix} a & a \\ a & e \end{matrix} \quad C \rightarrow \begin{matrix} a & a \\ a & a \end{matrix} \quad D \rightarrow \begin{matrix} a & a \\ a & b \end{matrix}$$

$$a \rightarrow \begin{matrix} 0 & 0 \\ 0 & 0 \end{matrix} \quad b \rightarrow \begin{matrix} 0 & 0 \\ 0 & 1 \end{matrix} \quad c \rightarrow \begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix} \quad e \rightarrow \begin{matrix} 0 & 1 \\ 0 & 1 \end{matrix} \quad p \rightarrow \begin{matrix} 1 & 1 \\ 1 & 1 \end{matrix}$$

(a)

$$S \Rightarrow \begin{matrix} A & B \\ C & D \end{matrix} \Rightarrow \begin{matrix} c\,p\,a\,a \\ a\,c\,a\,e \\ a\,a\,a\,a \\ a\,a\,a\,b \end{matrix} \Rightarrow \begin{matrix} 1\,0\,1\,1\,0\,0\,0\,0 \\ 0\,1\,1\,1\,0\,0\,0\,0 \\ 0\,0\,1\,0\,0\,0\,0\,1 \\ 0\,0\,0\,1\,0\,0\,0\,1 \\ 0\,0\,0\,0\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0\,0\,1 \end{matrix}$$

(b)



(c)

図 2.22 XOR 問題を解くネットワークを生成するための「グラフ生成文法」の利用．(a) 文法ルール．(b) 結合行列は文法から生成される．(c) 結果のネットワーク．(Kitano 1990 からの改編)．

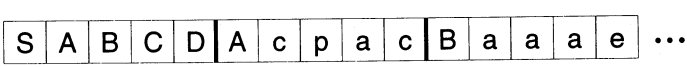


図 2.23 染色体にコード化された文法の図

Kitano によって使われた簡単なものでは，非終端記号（例えば，A）が 2 つ以上のルールで左側に現れた場合，最初のルールのみが文法に含められる（Hiroaki Kitano，私信）．

文法の適合度は以下のように計算される．まず文法からネットワークを構成す

る．次にバックプロパゲーションを用いてネットワークを訓練してから，訓練集合か別のテスト集合で出力の自乗和誤差を計る（これは Montana と Davis，および Miller，Todd と Hegde によって使われた適合度尺度に似ている）．GA は適合度比例選択，多点交叉（交叉は染色体に沿って，1 あるいはそれ以上の点で実行された），そして突然変異を用いた．突然変異においては，$A-Z$ および $a-p$ のアルファベットからランダムに選ばれたシンボルで染色体中での 1 つのシンボルを置換する．Kitano は**適応変異**と呼ぶものを用いた．それは子の突然変異の確率は 2 つの親の間の**ハミング距離**（一致しない数）に依存するというものである．遠い距離は低い突然変異を生じる．この方法により，GA は選択的に突然変異率を上昇することで集団における多様性の損失を補うと思われた．

Kitano (1990) は文法コード化法と直接コード化法を比較するために簡単な「エンコーダー/デコーダー」問題に対するネットワークの進化の実験を行った．これらの比較的簡単な問題では，結果のニューラルネットワークの正しさおよび GA がそれらを見つけるまでのスピードの両方において，文法コード化法の性能は首尾一貫して直接コード化法を上回ることがわかった．Kitano の結果の例は図 2.24 に示してある．この図には世代に対する集団での最良のネットワークの誤差率（20

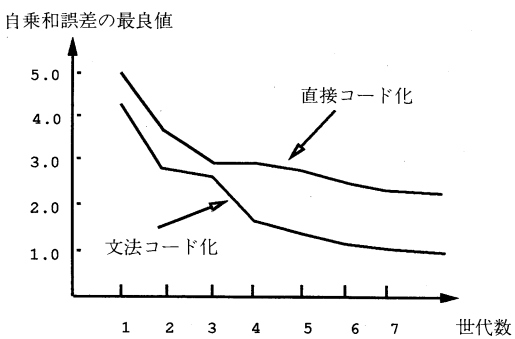


図 2.24 直接コード化法と文法コード化法を比較した Kitano の実験の結果．最良個体に対する自乗和誤差（TSS）が世代数に対してプロットしてある（20 回の平均．低い TSS が望ましい．Reprinted from Kitano 1990 permission of the publisher. ©1990 *Complex Systems.*）

回の平均）をプロットしてある．文法コード化法では，直接コード化法よりも低い誤差率のネットワークを見つけ，より早く最良のネットワークを見い出した．Kitanoはまた GA の性能は文法コード化法が使われた場合に，ネットワークサイズに対しより良くスケールアップしたことを見い出した．つまり，直接コード化を用いると，性能はネットワークのサイズ増加に伴いかなり急速に減少するが，文法コード化法では定常値に留まっていた．

文法コード化法が優れた点はどれくらいあるだろうか．Kitanoは，文法コード化法は結合の繰り返しパターンである「正則性」を簡単に作ることができ，これは文法ルールを繰り返し適用したことから生じる繰り返しパターンの結果であると論じた．この種の正則性を必要とする問題で文法コード化法がうまく動作することが期待される．ネットワーク構造自身よりも，ネットワークの構築に対する命令（文法）に対して GA は動作するため，文法コード化法はまたより短い染色体を獲得するという有利な利点がある．複雑なネットワークの場合，どんな探索アルゴリズムに対してもネットワーク構造自身は巨大で扱いにくくなる．

これらの属性は一般的に文法コード化法に有利かも知れないが，Kitano (1990) によって報告された実験で文法コード化法の優勢を説明できたかどうかははっきりしない．エンコーダー/デコーダー問題はニューラルネットワークにとって，最も簡単な問題の１つである．その上，隠れユニットの数が入力ユニットに対して小さいときにのみ意味がある．この条件での実験は文法コード化ではなく，直接コード化でのみ実施された．これらの実験での文法コード化の利点は，単に GA が問題を容易にするようなネットワーク・トポロジーを見つけることによるのかもしれない．この方式は比較されている特定の直接コード化法では利用できないので，比較は公平であるとは言えない．

Kitanoの文法進化のアイディアは興味深く，彼の非公式的な主張は文法コード化法（あるいはその拡張）が，複雑なニューラルネットワークが必要とされる問題に対して，うまくいくと信じる根拠となりうる．しかし問題があまりにも簡単過ぎたため，その主張を指示するために使われた実験には説得力がない．ネットワークの構造の進化と重みの設定が統合されたKitanoの初期の研究の拡張は

Kitano (1994) に報告されている．文法コード化のより意欲的なアプローチは Gruau (1992) と Belew (1993) によって行われた．

2.3.5 ニューラルネットワークの学習規則の進化

David Chalmers (1990) は違ったアプローチで GA をニューラルネットワークへ適用することを考えた．彼はニューラルネットワークに対する学習ルールを進化させるために GA を使った．Chalmers は，初期の研究を入力層と出力層のみで隠れ層のない全結合前向きネットワークに制限した．一般的に，学習ルールは訓練データでのネットワークの性能に応じて，ネットワークの重みを修正するために訓練手続きの間に使用される．各訓練サイクルでは，1 つの訓練ペアがネットワークに与えられ，出力を生み出す．この時点で，学習ルールは重みを修正するために呼び出される．1 層，全結合前向きネットワークに対する学習ルールを考えよう．このとき入力ユニット i から出力ユニット j へのリンク上の重みを修正するために，以下の局所情報が使われる．

a_i： 入力ユニット i の活性値
o_j： 出力ユニット j の活性値
t_j： 出力ユニット j での訓練信号
（すなわち，教師によって提供された正しい活性値）
ω_{ij}： i から j へのリンク上の現在の重み

重み ω_{ij} の変更値，$\Delta\omega_{ij}$ はこれらの値の関数である．

$\Delta\omega_{ij} = f(a_i, o_j, t_j, w_{ij})$

GA の染色体はこのような関数をコード化したものである．

Chalmers は，学習ルールはこれらの変数とすべての 2 変数の組合せの線形関数でなければならない，という仮説をたてた．すなわち，学習ルールの一般型は，

$$\Delta\omega_{ij} = k_0(k_1\omega_{ij} + k_2a_i + k_3o_j + k_4t_j + k_5\omega_{ij}a_i + k_6\omega_{ij}o_j + k_7\omega_{ij}t_j + k_8a_io_j + k_9a_it_j + k_{10}o_jt_j)$$

であった．$k_m (1 \leq m \leq 10)$ は定数の係数であり，k_0 はどの 1 サイクルにおいて

どれくらい重みが変更できるかに影響を与える尺度パラメータである（k_0 は**学習率**と呼ばれている）．Chalmers の学習ルールについての仮説はネットワークなどに対するよく知られた学習ルール—**Widrow-Hoff やデルタ規則**—が持つ形である，

$$\Delta\omega_{ij} = \eta(t_i o_j - a_i o_j)$$

の事実から部分的に生じたものである（Rumelhart et al. 1986）．ここで，η は学習率を表現した定数である．Chalmers の研究の目的の 1 つは，GA がデルタ規則と同じ位良い性能を持った規則を進化させられるかどうかを確認することであった．

GA のタスクは k_m の値を進化させることであった．k_m の集合に対する染色体のコード化は図 2.25 に示してある．尺度パラメータ k_0 は 5 ビットでコード化される．0 番目のビットは符合をコード化し（1 は + をコード化し，0 は − をコード化する），1 から 4 番目のビットは整数 n をコード化する．もし，$n = 0$ ならば，$k_0 = 0$ となり，その他の場合は $|k_0| = 2^{n-9}$ となる．よって，k_0 は 0, $\pm 1/256, \pm 1/128, \ldots, \pm 32, \pm 64$ を取る．他の係数 k_m はそれぞれ 3 ビットでコー

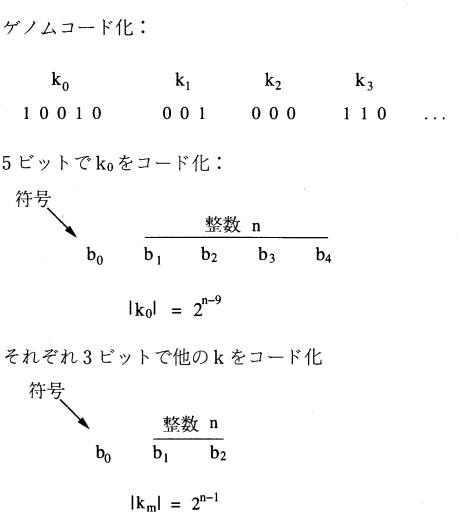


図 2.25 Chalmers のシステムで k_m のコード化のための方法．

ド化され，0番目のビットは符合をコード化し，1番目と2番目のビットは整数 n をコード化する．$m = 1 \ldots, 10$ について，$n = 0$ ならば $k_m = 0$，その他の場合は $|k_m| = 2^{n-1}$ となる．

1層のネットワークは**線形分離可能**の入出力マッピングしか学習できないことは知られている（Rumelhart et al. 1986）．進化する学習ルールの「環境」として，Chalmers は30の異なる線形分離可能なマッピングを使った．マッピングは常に1つの出力ユニットと，2から7までの入力ユニットであった．

各染色体（学習ルール）の適合度は以下のように決定された．20のマッピングの部分集合が30のマッピングから選ばれた．各マッピングに対して12の訓練例が選ばれた．これらの各マッピングに対して，適切な入力ユニットの数を持つネットワークが生成された（各ネットワークには1つの出力ユニットがあった）．ネットワークの重みはランダムに初期化された．染色体によって示された学習ルールを使って，ネットワークはある期間（一般的には10）訓練された．あるマッピングでの学習ルールの性能はより低い誤差が高い性能となるような訓練集合での平均誤差の関数であった．ルールの適合度は選ばれた20のマッピングの部分集合について20のネットワークの平均誤差の関数であり，低い誤差は高い適合度になるようにしてあった．そして，この適合度は百分率に変換され，高い百分率は高い適合度を意味していた．

この適合度関数を使って，二点交叉と標準的な突然変異を用いた GA が40の学習ルールの集団に対して実行された．交叉率は0.8で突然変異率は0.01であった．一般的に，集団中の最良の学習ルールの適合度は初期世代で40%から60%の間（全く学習能力がないことを示している）であるが，1000世代以上では80%から90%の間へと増加して平均すると約92%となった．デルタ規則の適合度は約98%であり，10回の実験の内の1回で GA はこのルールを発見した．10回中3回で GA はより低い適合度のこのルールのわずかな変形を発見した．

これらの結果は，幾分制限された表現が与えられれば，単純な1層ネットワークに対して，GA は学習規則をうまく進化させられることを示した．この方法がより複雑なネットワーク（隠れ層のあるネットワークなど）に対する学習ルール

を見つけることができるかは，いまだに未解決であるが，以上の結果は解決への最初のステップである．Chalmers は，進化的手法がバックプロパゲーションより強力な学習手法を発見することはありそうにないと述べているが，教師なし学習パラダイム（例えば，**強化学習法**）あるいはネットワーク構造の新しいクラス(例えば，リカレント・ニューラルネットワーク）に対する学習ルールを発見するための強力な方法に GA はなりうるとも推測している．

Chalmers は，進化した学習ルールの汎化能力の研究も行った．彼は適合度計算で使われなかった 10 個のマッピング（「テスト集合」）で，進化した最良のルールをテストした．もとのマッピングでの最良ルールの平均適合度は 92%であったが，テスト集合でのこれらのルールの平均適合度は 91.9%であることがわかった．要するに，進化したルールは極めて一般的であった．

さらに，Chalmers は一般的なルールを生成するには環境がどのくらい多様であるべきかという問題を考察した．彼はマッピングの数を 1 と 20 の間で変えた環境で実験を繰り返した．ルールの進化的な適合度とはルールをもとの環境でテ

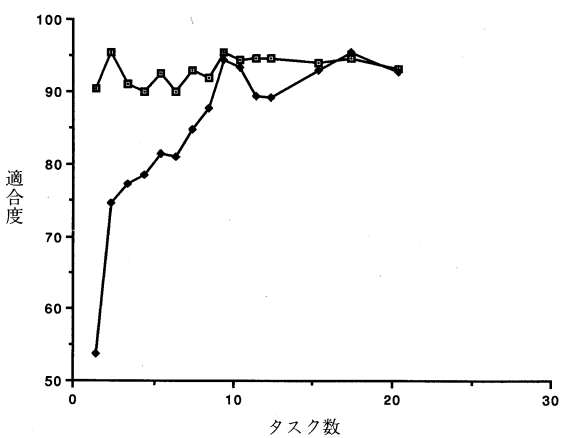


図 2.26 一般化の能力についての環境の多様性の効果をテストするための Chalmers の実験結果．図は環境でのタスクの数の関数として，進化的な適合度（四角）とテスト適合度（ひし形）を示している（D.S. Touretzky et al. (eds.), *Proceedings of the 1990 Connectionist Models Summer School.* ©1990 Morgan Kaufmann. Reprinted by permission of the publisher.）．

ストしたことによって得られた適合度である．ルールのテスト適合度とは最初の環境にはない 10 個の付加的なタスクでルールをテストすることによって得られる適合度である．Chalmers は，最初の環境でタスクの数の関数としてこれらの 2 つの適合度を計った．結果は図 2.26 に示してある．2 つのカーブは与えられたタスクの数の環境でテストされたルールに対する，進化的な適合度の平均とテスト適合度の平均である．このプロットは異なる数のタスクに対して進化的な適合度が大体一定に留まっているのに対して，テスト適合度はタスクの数に従って急激に増加し，10 から 20 のタスク数のあたりで水平になっていることを示している．結論として言えることは，一般的な学習ルールの進化は多様なタスクの環境を必要とするということである（この簡単な単層のネットワークの例では，必要な多様性の程度は極めて小さい）．

演習問題

A–思考演習

1. 関数集合 {AND, OR, NOT} と，終端集合 $\{ s^{-1}, s^{0}, s^{+1} \}$ を使って，$r = 1$ で，CA の大域的なルールをコード化する解析木（もしくは Lisp の式）を作ってみなさい．ここで，s^{i} は中心のセルから i サイト離れた近隣のサイトを示している（− は左の方に，+ は右の方に離れた距離を示す）．AND と OR はそれぞれ 2 つの引数を取り，NOT は 1 つの引数を取る．
2. “MUTATE-TREE (TREE)” はランダムに生成した部分木で TREE の部分木を置換する関数であると仮定する．この関数を使って，ランダムに選ばれた解析木からスタートして，GP 解析木の空間を探索する最急勾配による山登り法に対する疑似コードを書いてみなさい．同様にランダムな突然変異の山登り法の疑似コードを書いてみなさい．
3. 半径 r の CA ルールの数に対する公式を書いてみなさい．
4. 図 2.27 に示す文法によって与えられるネットワークを作るために，図 2.23 と同じ手続きを行ってみなさい．

S → A B / C D　A → c p / a c　B → c a / g e　C → a a / b a　D → f e / a b

a → 0 0 / 0 0　b → 0 0 / 0 1　c → 1 0 / 0 1　e → 0 1 / 0 1　f → 1 1 / 0 1

g → 0 0 / 1 0　p → 1 1 / 1 1

図 2.27　問題 4 に対する文法

5. 図 2.28 で与えられるネットワーク構造を作る文法を設計しなさい.

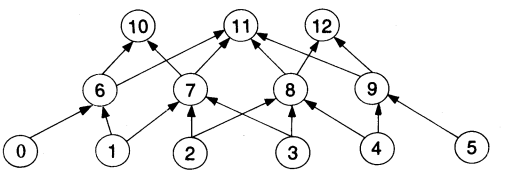


図 2.28　問題 5 に対するネットワーク

B–コンピュータ演習

1. GP のアルゴリズムを実装し，それを使って「**6 - マルチプレクサ**」問題を解いてみなさい（Koza 1991）. この問題では，$\{ a_0, a_1, d_0, d_1, d_2, d_3 \}$ の 6 つのブール値の終端記号と，{ AND, OR, NOT, IF } の 4 つの関数記号がある. 最初の 3 つの関数記号は普通の論理オペレータで，それぞれ 2，2，1 の引数を取り，IF 関数は 3 つの引数を取る.（IF X Y Z）は 1 番目の引数 X を評価する. もし，X が真であれば，2 番目の引数 Y が評価され，その他の場合は 3 番目の引数の Z が評価される. この問題は 2 つの終端記号 a_0 と a_1 によってアドレスされた終端記号 d の値を返すようなプログラムを見つけることである. 例えば，もし $a_0 = 0$ で $a_1 = 1$ ならばアドレスは 01 であり，答えは d_1 の値である. 同様に，もし $a_0 = 1$ で $a_1 = 1$ ならば，アドレスは 11 で答えは d_3 の値となる. 異なる初期状態，交叉率，集

団サイズ（300 の集団サイズでスタート）で実験しなさい．プログラムの適合度はすべての 2^6 の可能な適合度事例に対する正解率としなさい．

2. 演習 1 と同様の実験を行ってみなさい．ただし，関数集合と終端集合にいくつかの「余計なもの」—問題を解くのに必要でない余分な関数記号と終端記号—を加えてみなさい．これがこの問題において GP の性能にどのように影響するだろうか．
3. 演習 1 と同様の実験を行ってみなさい．ただし，各適合度の計算は全体の集合ではなく可能な適合度事例 2^6 のうち 10 のランダムサンプルを使ってみなさい（各適合度計算に対して新しいランダムサンプルを使うこと）．これがこの問題において GP の性能にどのように影響するだろうか．
4. 6 マルチプレクサ問題に対して分析木を探索するランダム探索法を実装してみなさい．すなわち，各タイムステップで，新しいランダム解析木を生成し（木の最大長は前もって固定されている），その適合度を計算する．最良個体の適合度の増加率を GP の場合と比較してみなさい（300 タイムステップごとにプロットする．そうすれば実験 1 における GP の 1 世代と等価になる）．
5. 6-マルチプレクサ問題（思考演習 2 を参照）に対する分析木を探索するランダムな突然変異による山登り法を実装してみなさい．その性能を GP の性能およびコンピュータ演習 4 のランダムサーチ法の性能と比較してみなさい．
6. 演習 1 で使われた適合度を，正しい性能と同様に小さなサイズのプログラムにも報酬を与えるように修正してみなさい．GP の手続きを使ってこの新しい適合度関数をテストしてみなさい．GP はこの方法によって正しくてより小さいプログラムを見つけることができるだろうか．

*7. $\rho_c = \frac{1}{2}$ 問題を解く $r = 3$ の CA を進化させるために，Crutchfield, Mitchell, Das と Hraber の実験を追試してみなさい（この問題にはセルラ・オートマトンをシミュレートするプログラムを書くことも必要となる）．

*8. 演習 7 での実験の結果を，ランダムな突然変異の山登り法の結果と比較し

てみなさい（比較のためにMitchell, Crutchfield, and Hraber 1994aを参照してみなさい）.

*9. 演習7と同様の実験を行ってみなさい．ただし，CAの解析木の表現としてGPを使ってみなさい（思考演習1を参照．これには解析木表現とCAシミュレータに与えるCAの参照表を変換するためのプログラムを書くことが必要になる）．実験結果を参照表のコード化を使って演習7で得た結果と比較してみなさい．

*10. 図2.29は「エンコーダー/デコーダー」問題に対する19ユニットのニューラルネットワーク構造である．問題は表2.2で与えられるマッピングを実行するネットワークの重みを見つけることである．すなわち，与えられた入力パターンを，ネットワークは出力ユニットにコピーしなければならない．

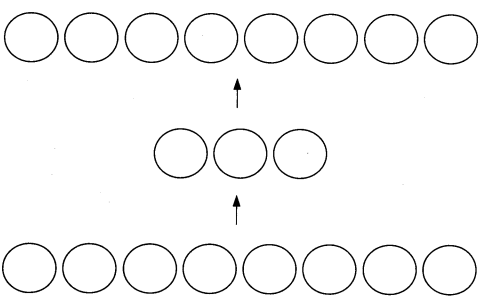


図 2.29 コンピュータ演習10に対するネットワーク．矢印は各入力ノードが各隠れノードに連結し，各隠れノードが各出力ノードに連結していることを示している．

表 2.2 コンピュータ演習10に対する表

入力パターン	出力パターン
10000000	10000000
01000000	01000000
00100000	00100000
00010000	00010000
00001000	00001000
00000100	00000100
00000010	00000010
00000001	00000001

入力ユニットと出力ユニットよりも隠れユニットの数が少ないので，隠れユニットによる入力のコード化とデコード化の方法を学習しなければならない．各隠れユニット j と各出力ユニット j は閾値 σ_j を持っている．もし入ってくる活性値が σ_j よりも大きいか等しいならば，そのユニットの活性値は 1 にセットされる．その他の場合は 0 にセットされる．最初のタイムステップでは，入力ユニットは入力パターンに従って活性化される（例えば，10000000）．その後，活性値は入力ユニットから隠れユニットへ広がる．各隠れユニット j の入力活性値は $\Sigma_i a_i \omega_{i,j}$ で与えられる．ここで，a_i は入力ユニット i での活性値で，$\omega_{i,j}$ はユニット i から j への結合での重みである．隠れユニットが活性化された後，同様の手続きによって出力ユニットが活性化する．この問題を解くために，重み $\omega_{i,j}(0 \leq \omega_{i,j} \leq 1)$ と閾値 $\sigma_j(0 \leq \sigma_j \leq 1)$ を進化させる Montana と Davis の方法を使ってみなさい．$\omega_{i,j}$ の値と σ_j の値を同じ染色体に置くとよい（σ_j の値は入力ユニットでは無視され，常に 0 か 1 にセットされる）．染色体の適合度は，全体の訓練集合における自乗誤差の平均和とする（各位置での出力と入力との間の差）．どの位 GA は成功するだろうか．さらに熱心な読者諸氏に対しては，Montana と Davis が行ったのと同様の方法で GA の性能とバックプロパゲーション（Rumelhart, Hinton, and Williams 1986a）の性能を比較してほしい（この問題はニューラルネットワークでよく知られているものである）．