進化型ハードウェアに関する研究

－適応性をもつＬＳＩの実現－

村川正宏

学位論文

進化型ハードウェアに関する研究

– 適応性をもつ LSI の実現 –

村川 正宏

目次

第 1 章　序論 **1**

第 2 章　進化型ハードウェアの定義と意義 **5**

2.1 緒言 6

2.2 進化型ハードウェアの定義 6

2.3 従来の進化型ハードウェアの研究 8

2.3.1 樋口らの研究 8

2.3.2 辺見らの研究 9

2.3.3 Koza らの研究 10

2.3.4 Thompson の研究 10

2.4 適応的なハードウェアの実現 10

2.4.1 ソフトウェアとの比較 11

2.4.2 適応の対象 12

2.4.3 従来の適応的なハードウェア 13

2.5 進化型ハードウェアの分類 14

2.5.1 On-line デジタル EHW 16

2.5.2 Off-line アナログ EHW 16

2.5.3 本研究の位置付け 18

第 3 章　進化型ニューロチップ –GRD チップ– の開発 **19**

3.1 緒言 20
3.2 ニューラルネットワークの On-line 学習 21
3.2.1 ネットワークの構造決定 21
3.2.2 ニューラルネットワーク実行用ハードウェアの問題点 22
3.2.3 学習時間 22
3.3 Hybrid Neural Network の遺伝的学習手法 23
3.3.1 問題の定式化 23
3.3.2 遺伝的学習手法 25
3.3.3 コード化 27
3.3.4 適応度および淘汰方法 27
3.3.5 交叉 28
3.3.6 突然変異 29
3.3.7 個体の置き換え 30
3.3.8 最急降下法による学習 30
3.4 GRD チップのアーキテクチャ 31
3.4.1 概要 31
3.4.2 V830 プロセッサ 33
3.4.3 Programmable Function Unit (PFU) 35
3.5 提案手法の関数近似性能 36
3.5.1 RBF の近似 37
3.5.2 シグモイド関数の近似 40
3.5.3 RBF とシグモイド関数を重ね合わせた関数の近似 42
3.6 カオス時系列の予測問題 45
3.6.1 予測方法 45
3.6.2 学習結果の評価方法 47
3.6.3 シミュレーション結果 47
3.7 ベンチマーク問題による性能評価 49

3.7.1 心臓病診断問題 (hearta1) 49
3.7.2 エネルギ消費量予測問題 (building1) 51
3.7.3 太陽フレア数予測問題 (flare1) 52
3.8 本章のまとめと今後の課題 52

第 4 章 GRD チップの適応等化器への応用 54
4.1 緒言 55
4.2 適応等化器とは 55
4.3 従来の適応等化器 57
4.4 GRD チップを用いた適応等化 60
4.4.1 静的環境における非線形等化 61
4.4.2 時変環境における適応等化 61
4.5 本章のまとめと今後の課題 67

第 5 章 GRD チップの強化学習への応用 68
5.1 緒言 69
5.2 強化学習と Q-learning 70
5.3 GRD チップを用いた Q-learning 72
5.4 ロジスティック写像カオスの制御 73
5.4.1 実験方法 74
5.4.2 実験結果 75
5.5 本章のまとめと今後の課題 77

第 6 章 GRD チップを用いた ATM ネットワークの CAC 制御 80
6.1 緒言 81
6.2 ATM ネットワーク 81
6.2.1 ATM ネットワーク 81
6.2.2 ATM Adaptation Layer 83

6.3 ATM ネットワークの制御 83
6.3.1 ATM におけるトラフィック制御 83
6.3.2 Connection Admission Control 85
6.4 シミュレーション 86
6.4.1 ATM ネットワークのトラフィックシミュレータ 86
6.4.2 GRD チップを用いた CAC の枠組 86
6.4.3 実験 1：セル損失率の制御 89
6.4.4 実験 2：時間とともに呼の性質が変わる場合の制御 90
6.5 本章のまとめと今後の課題 94

第 7 章 進化型携帯電話用フィルタの開発 97

7.1 緒言 98
7.2 IF フィルタの構造 98
7.2.1 IF フィルタの要求仕様 98
7.2.2 IF フィルタの実装 100
7.3 IF フィルタ用アナログ EHW 102
7.3.1 コーディング方法および遺伝的操作 104
7.4 シミュレーション結果 105
7.5 実験結果 107
7.5.1 考察 109
7.6 本章のまとめと今後の課題 110

第 8 章 結論 111

8.1 結論 112
8.2 今後の予定および課題 113
8.3 進化するハードウェアへむけて 114

謝辞 116

参考文献 117

第 1 章

序論

近年、人間が機械に要求する機能は、複雑化の一途をたどっている。とくに、たえず予期せぬ新しいことが起こる環境、ノイズなど不確定な要素が多い環境、ゴールが漠然としか示せない環境などにおいては、機械が自律的に動作することが要求される。しかしこうした環境において、従来の工学の設計手法は対応することができない。つまり従来の工学では、環境を詳細に調査し、問題を明確に記述し、その中で目的を達成するための最適な機械を設計することがなされてきた。このような手法では、上記の環境において自律的に動作する機械を設計することは難しく、設計者が予測することができる、制限を設けた環境にのみ対応することができた。そこで本研究では、このような拘束を取り除き、設計者は目的と動作例題を与えるだけで、変化する環境に適応し、未知の環境での新しい動作を見つけだしてゆけるような機械の設計法を開発することを目的とする。

そのような適応的な機械を実現するために、従来の研究では、機械を制御する CPU (Central Processing Unit) 上で実行されるソフトウェアによって適応的な行動を実現しようとしてきた。しかし、ソフトウェアの実行時間の遅さから、実時間の高速性を要求される分野では用いることができなかった。よって、実時間の高速性が必要で、かつ環境に適応するためためには、ハードウェア自体に適応性をもたせなければならない。

近年生物の進化、適応の仕組みに発想を得た探索手法である遺伝的アルゴリズム (Genetic Algorithm: GA)[1, 2] が活発に研究されている。GA は、探索点を複数もつ並列探索手法であり、これまでに最適化問題や制御問題の解法として用いられ、いくつかの現実的な応用問題にも適用されはじめている [3, 4, 5]。本研究では、適応的なハードウェアを実現するための手段として、この遺伝的アルゴリズムと可変構造をもつハードウェア素子を組み合わせる。これを「進化型ハードウェア (Evolvable Hardware : EHW)」と呼ぶことにする。これまでにも、再構成が可能なゲート回路である FPGA (Field Programmable Gate Array) と遺伝的アルゴリズムを組み合わせた研究が EHW としておこなわれているが、それらは、あらかじめ定められた真理値表を満足するデジタル回路を自動合成することを目的とするもので適応能力を追求したものではなかった。一方本研究では、環境に応じて内部構造が適応的に変化する LSI を実現し、それを現実の問題に適用するためのアルゴリズムの提案および実装を行うことによって、このような方法の

有効性を示すことを目的としている。

具体的に本研究では、EHW の新たな手法として「On-line デジタル EHW」および「Off-line アナログ EHW」の二つを提案する。前者の「On-line デジタル EHW」は、LSI に入力されるデータの質の変化に適応することを目的とするもので、その実現例として、GRD (Genetic Reconfiguration of DSPs) チップと呼ぶ専用 LSI を開発した。この GRD チップは、自律再構成機能を持つニューロチップであり、非線形関数を学習する能力をもっているので、これまでのハードウェアでは困難であった、時間とともに問題の性質が変化し、かつ実時間の応答性が要求される様々な問題に適用することができる。また、同チップを用いた、カオス時系列予測、デジタル通信で用いられる適応等化器、時変環境における強化学習手法の提案もあわせて行い、動的に変動する環境を仮想的に生成し、その下での有効性を実験で示した。

一方、「Off-line アナログ EHW」の実現例としては携帯電話などで広く使用されている IF フィルタ (中間周波数フィルタ) 用 LSI を開発した。これは製造時の LSI 内部のアナログ素子の性能バラツキを遺伝的アルゴリズムを用いて吸収することを可能にしたものであり、歩留まりを向上しチップ面積を削減することができ、産業的にも画期的なチップである。提案した手法は、IF フィルタ以外の様々なアナログ LSI に適用可能であり、汎用的な手法である。

なお、本論文の構成は以下のようになっている。続く第 2 章では進化型ハードウェアの定義と意義について詳しく述べる。従来の関連研究について概観したのち、「On-line デジタル進化」および「Off-line アナログ進化」の二つの手法を提案する。第 3 章では、On-line デジタル進化の実現例である GRD チップの開発ついて述べる。ニューラルネットワークの中間層素子数と中間層素子の種類を動的に決定する学習手法を提案し、それを実現するチップのアーキテクチャについて説明する。また、提案手法をいくつかの関数近似問題に適用しその有効性を検証する。第 4 章では、GRD チップを適応等化器へ適用することを提案する。適用方法について説明した後、シミュレーションを通じて動的な環境の変化に GRD チップが適応可能であることを示す。第 5 章では、GRD チップを用いた時変環境における強化学習手法を提案する。提案する強化学習手法の枠組みに

ついて説明した後、カオス写像を一定値に制御する問題に同手法を適用し、その有効性を検証する。第6章では、前章で提案した強化学習手法を用いてATMネットワークの呼受付制御を行うことを提案する。制御の枠組みについて説明した後、通信環境が変化する場合においても、変化に適応して有効な制御方式をGRDチップが獲得できることをシミュレーションで示す。第7章では、Off-lineアナログ進化の実現例である進化型IFフィルタについて説明する。GAを用いたアナログ素子のバラツキ吸収の手法を提案したのち、開発したLSIがどの程度の歩留まりを実現できるかの実験結果について述べる。最後に、第8章で結論および今後の課題を述べる。

第 2 章

進化型ハードウェアの定義と意義

2.1 緒言

本章では、まず進化型ハードウェアの定義を述べた後、そのようなハードウェア構成手法をとることの利点について説明する。つぎに従来の EHW 研究について概観した後、EHW を二つの軸 - 適応の頻度および素子の種類で分類する。そして本研究が対象とする、その分類における二つの領域、「On-line デジタル EHW」および「Off-line アナログ EHW」について目的および利点を説明する。

2.2 進化型ハードウェアの定義

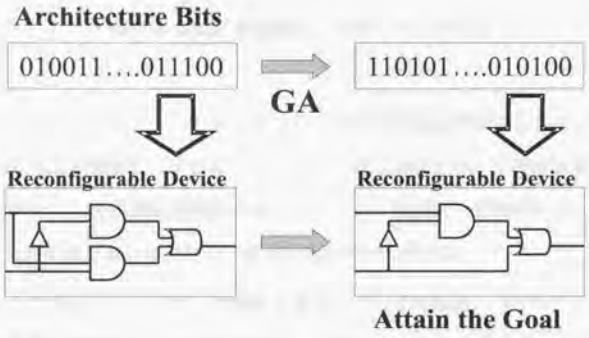


図 2.1: 進化型ハードウェアのアイデア

進化型ハードウェアの定義は、序章でも述べたように、可変構造をもつハードウェア素子と遺伝的アルゴリズムを組み合わせたものである。遺伝的アルゴリズムは、生物の進化に発想を得た探索手法であり、染色体とよばれる解候補を複数個もつ並列探索手法である。また、可変構造をもつハードウェア素子では、ハードウェア構成情報であるビット列を書き変えることで望ましいハードウェア素子を実現し、また容易に変更することができる。そこで進化型ハードウェアの基本的なアイデアは、図 2.1にあるように、それらビット列を遺伝的アルゴリズムにおける染色体と見なし、それに対して遺伝的操作を

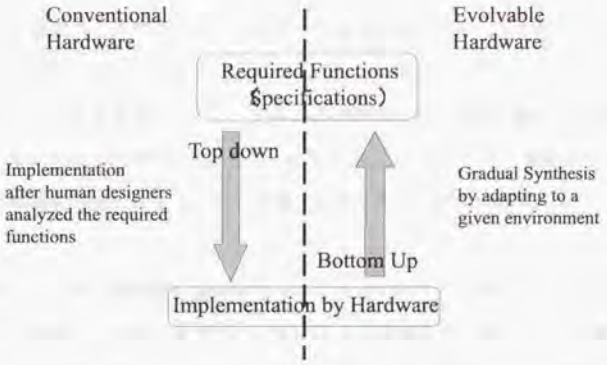


図 2.2: EHW の従来ハードウェアとの差異

行なうことによって、ビット列を書き換える。その結果構造が変化したハードウェア素子の入出力を GA が観察し、そのビット列の評価値を決定する。その評価値にもとづいて染色体を淘汰し、再び遺伝的操作を行う。このような評価 / 再構成の繰り返しによって、環境や目的に最適なハードウェア構成を適応的に決定する。

以上のような新しいハードウェア構成方式がもたらす利点は、(1) ハードウェアのもたらす実時間的な処理速度、(2) 適応的にハードウェア構成を変化させることができる、の 2 つに集約される。つまり、実時間処理が必要で、かつ実現すべきハードウェア構成を適応的に決定しなくてはいけないような応用が、進化型ハードウェアの適用領域であり、これは従来の一般的なハードウェアとは一線を画している。従来、このような応用に対しては、十分な実現手段がなかったともいえる。なぜなら、速度を追求すればハードウェアによる実装しかなく、しかし一度そうしてしまうと、もう環境の変化には対応できないからである。一方ソフトウェアによれば、適応という要件は満たされるが、実時間処理については当然限界がある。

また、進化型ハードウェアの設計手法を、従来のハードウェア設計手法と比較すると図 2.2に示すように極めて対照的である。すなわち従来のハードウェアでは、まずハード

ウェアが実現すべき要求仕様が最初に決定され、次にそれを実現するハードウェア仕様が決定し、最後にハードウェアによる設計／実装が行なわれる。与えられた仕様を元に最後に実装が行われるので、この一連の流れはトップダウンである。このアプローチでは、ハードウェアの実装の前に、ハードウェアが使用される環境や様々な物理的制約を考えた上で設計を行う必要がある。しかし現在では、ハードウェアの規模が大きくなり、用いられる環境が多様になるにつれ、設計にかかる人手と時間が多大なものとなっている。

これに対し、提案手法は全く逆のボトムアップアプローチである。つまり与えられた要求仕様を実現するために、まずビット列によって再構成が可能な素子を実装する。その素子に対して、遺伝的アルゴリズムを用いて調整 / 評価を繰り返し、仕様を満たすようなハードウェア構成を徐々に合成していく。このようなボトムアップ的なアプローチにより、人手によらない自律的なハードウェアの設計手法を実現することができる。またその結果、設計にかかる人手と時間を削減できるという大きな利点を生じる。

2.3 従来の進化型ハードウェアの研究

以上のべたように、適応的なハードウェアを実現する EHW の設計手法は、従来のハードウェアの設計手法とは対照的なものである。これまでにも、再構成が可能なゲート回路と遺伝的アルゴリズムを組み合わせた研究などが EHW としておこなわれているが、それらは、回路を自動合成することを目的とするもので適応能力を追求したものではなかった。以下ではそれらの研究について概観する。

2.3.1 樋口らの研究

樋口らは PLA（Programmable Logic Array）を用いて、その回路の物理的制約を満たすデジタル回路を遺伝的アルゴリズムをもちいて自動的に合成する実験を行った [6, 7]。

一般に PLA は、図 2.3に示すように AND アレイと OR アレイの二段構成からなる。すなわち、初段の AND アレイで入力信号の論理積を生成し、次に OR アレイにおいて

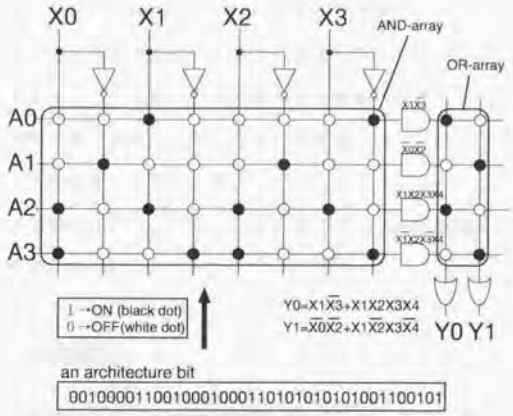


図 2.3: PLA の基本的な構造

それらの論理和を求め、それらを出力とする。その AND アレイと OR アレイの機能は、スイッチの ON、OFF を指定することで何度も変更することができる。このスイッチの設定はアーキテクチャビットと呼ばれ、PLA の外部から与える。

そこで樋口らは、アーキテクチャビットを GA の染色体と見なし、あらかじめ与えられた真理値表を満たすのに必要なスイッチの組合せを探索し、PLA 上の回路を自動合成した。実験の結果、PLA 上のマルチプレクサ、カウンタ、有限状態オートマトンを自動合成している。

2.3.2 辺見らの研究

樋口らのアプローチが、比較的小規模な回路を直接的に発見することを目指しているのに対し、辺見らの研究 [8] は、ハードウェア記述言語 (HDL) である SFL による記述を遺伝的プログラミングによって自動生成させることにより、回路の自動設計を目指すものである。遺伝的プログラミングとは、遺伝的アルゴリズムにおいて木構造を扱えるように染色体表現を拡張したものである [9]。

学習の手順としては、始めにHDLの文法を決めるプロダクションルールの集合を定義する。これらの文法ルールをどのような順番で適用していくかによって、異なるハードウェア記述ができあがる。そこでこの適用順序を木構造で表現してやり、これを遺伝的プログラミングの染色体とみなし、目的に沿う染色体、つまりはハードウェア記述を生成する。いったんこれが得られれば、あとはこれをコンパイルすることによって実際のFPGA上の回路に変換することができる。これまでに、2進数加算器のハードウェア記述言語の生成に成功している。

2.3.3 Kozaらの研究

上記ふたつの研究は、デジタル回路の自動合成の研究であったが、Kozaらは、遺伝的プログラミングを用いてアナログ回路の自動設計を行っている[10]。手法としては、アナログ回路の構成を木構造に表現し、それを遺伝的プログラミングをもちいて目的の回路を自動的に合成できるかの検証を行っている。実験の結果、ローパスフィルタ、増幅器などの様々なアナログ回路を人手を介さずに発見している。

2.3.4 Thompsonの研究

Thompsonは、Xilinx社のFPGAを用いて矩形波の周波数認識器を自動合成した[11]。この回路の目的は、ある特定の周波数のみに1を出力する回路をFPGA上に合成することである。実験ではデジタル回路であるFPGAを非同期モードで動作させたので、タイミングまでを考慮したFPGAの設定を遺伝的アルゴリズムが自動的に発見したことになる。

2.4 適応的なハードウェアの実現

以上従来のEHW研究を概観したが、いずれも人間があらかじめ定めた機能をみたす回路を遺伝的アルゴリズムが発見できるかの研究である。つまり遺伝的アルゴリズムの最適化手法の側面を積極的に用いている。これらの研究によって、回路設計における人

的コストが削減できるというメリットはあるが、提案手法をどの程度の規模の問題に適用できるかが問題である。大規模な回路を自動設計しようとする場合、染色体長が増大し、実時間では目的の回路が発見できない可能性が十分にある。たとえばThompsonの研究では、100程度の論理ゲートを用いて自動合成の実験を行っているが、染色体長はおよそ3,000ビットであり、回路の発見に何日も要している。彼らの手法では、染色体長は論理ゲート数nの二乗のオーダで増加するので、このままの手法では現実的な大規模な回路の自動設計は難しい。

そこで本研究では、従来のEHW研究とは一線を画し、適応的なハードウェアを実現するための手段として遺伝的アルゴリズムを用いる。

ここで、適応的なハードウェアを実現することの意義について考察するために、ソフトウェアとの違いを考えることにする。

2.4.1 ソフトウェアとの比較

ハードウェアとソフトウェアは、何らかの機能を実現する手段という観点からみれば、それほどの違いはない。何らかの情報処理を行なう場合、最低限何か物理的なハードウェアが必要なことはいうまでもない。このハードウェアがある程度以上の機能を持つ場合、様々な処理をその上で動作するソフトウェアを用いて行なうことができる。現在使われている汎用コンピュータはまさしくその例である。逆にソフトウェアを使わないですべての処理をハードウェアで行なうこともできるということは、初期のころのコンピュータがプログラムを物理的な結線で行なっていたことでもわかる。このようにソフトウェアとハードウェアは相互に置き換え可能なものである。

しかし、今述べたソフトウェアとハードウェアの交換可能性は、現実的問題としてソフトウェアとハードウェアとの間に違いがないことを意味するものではない。その大きな違いが、動作の並列度である。現在のRISCのCPUチップでは同時に4個程度の命令を実行するものがある。つまりソフトウェア的にみた並列度は4である。ところが、同じ一つのCPUチップでもハードウェア的にみれば回路を構成している数十万、数百万のゲートあるいはトランジスタがほとんど同時に動作している。原理的には同じ並列動作

であるが、この数の違いはほとんど質的な違いといってもよい。

このことは、現実的には桁違いな動作速度の差および実装上のサイズの差として現れる。動作速度に関していえば、例えばLSIの設計をするときに設計の検証をするために回路のゲートレベルの動作シミュレーションを行なうことがよくある。その際に、ハードウェアではわずか数十ミリ秒で起こる動作をソフトウェア的にシミュレーションするのに、数日を要することもまれではない。実装上のサイズに関していえば、専用ハードウェアを用いれば並列性をいかして非常にコンパクトに実装できるのに、CPUを用いた場合、外部にバス、メモリを配置するために規模が大きくなってしまうケースがよくある。

またソフトウェアによって適応性を実現する場合の問題点として、ソフトウェアを実行するハードウェア自体の不正確さには適応することができない。これはソフトウェアというものは、ハードウェアが完全に動作するものとして設計されているから原理的に当然のことである。しかし、ハードウェアの高密度化、高速化、大規模化に伴い、その完全性を確保するのが難しくなりつつある。とくに出力の「タイミング」の制約は重要な問題である。ソフトウェアの場合、答えが求められるかどうかが重要視され、出力のタイミングについては早いに越したことはないといった程度に、あまり厳密に要求されないことが多い。それに対し、ハードウェアの場合はある出力端子への出力のタイミングが、絶対的に要求され、また他の出力との相対的な関係においても相当な精密さをもって要求される。つまり、早くても遅くてもいけないのである。これは、ハードウェアの場合は、その出力が直接、電気的な信号として制御等に利用される傾向が強いということから要請される。

2.4.2 適応の対象

前節でのハードウェアとソフトウェアとの違いの考察の結果、ソフトウェアではなくハードウェアに適応性をもたせることの利点は、

- 高速性

- 実装の小型化
- ハードウェア自身の不正確さへの対応

の3点に集約される。また、ハードウェアが適応すべき対象の「環境」が以下の二つに大きく分類できることがわかる。

- ハードウェアがおかれる外的な環境
- ハードウェアの内的な環境

前者の外的な環境とは、ハードウェアに対する入力データの性質の変化を意味する。後者の内的な環境とは、ハードウェア素子内の故障やバラツキや温度変化による不正確さを意味する。これらの環境が動的に変化したり未知である場合に、人間が与えた目的を実現するためにハードウェアに適応性をもたせる必要がある。

これまでに本研究で提案するEHW以外に、適応的なハードウェアが存在しなかったかといえば、そうではない。いくつかの手法が存在するが、いずれも制限がおおい。以下では、従来の代表的な適応的なハードウェアおよびその問題について説明する。

2.4.3 従来の適応的なハードウェア

線形適応フィルタ

線形適応フィルタとは、フィルタ内の加重係数を、外的な環境の変化に応じて調整することが可能なハードウェアである。LMS (Least Mean Squares) アルゴリズムやRLS (Recursive Least Squares) アルゴリズムなどの適応アルゴリズム [12] を用いて係数を自動調整する。エコーキャンセラやノイズキャンセラとして工業的な実用化もされている [13]。しかし問題点として、線形性の仮定や、調整時のエネルギ関数の連続性の仮定から、問題に応じては十分な性能を得られないことがある。

人手による適応

内的な環境、とくにアナログ素子のバラツキに対応する手法として、トリミングと呼ばれる手法がある [14, 15]。これは、精度を必要とするアナログ LSI において、LSI 製造後に内部の抵抗値などを人手によって調整する手法である。レーザートリミングとよばれる手法では、R の素子をレーザーによって一部分焼き切り、その値を調整する。しかし、人手による調整には規模において限界があり、大量生産には向かない。またどのように調整するかは職人技に頼っているのが現状であり、いかに職人さんを確保するかが重要な課題となっている。

冗長な回路

内的な環境、とくに回路の故障に適応するために、内部に冗長な回路をもっている LSI がある。多重化回路 (多数決回路) などが代表的な回路である [16, 17]。この回路の問題点として、どのように冗長な回路を設計すればよいかの問題がある。状態空間の爆発の問題から、人手による設計には限界があり、自動化による大規模化が必要である。

問題点

以上従来の適応的なハードウェアを概観したが、性能が不十分である、人手を介す必要がある、小規模であるなどの制限が多いことがわかった。

2.5 進化型ハードウェアの分類

本研究では上記の問題点を解決するために、可変構造をもつハードウェアの適応手段として遺伝的アルゴリズムを用いる。遺伝的アルゴリズムの探索手法としての特徴は、

- 多点探索による局所解の回避 (準最適解にすみやかに到達する)
- 離散的な組み合わせ最適化問題に適用可能である

表 2.1: 進化型ハードウェアの分類

	デジタル素子	アナログ素子
Off-line	従来研究	本研究 内的な環境への適応
On-line	本研究 外的な環境への適応	

- 目的関数の性質が解らないような問題についても (とりあえず) 適用可能

がある。これらの性質を積極的に利用することによって、進化型ハードウェアでは前節で説明した従来の適応的なハードウェアの制限を解決する。その結果、

- 自律性
- 非線形性への対応
- 大規模な離散的な調整が可能

の特徴を持つ。

そこで、これらの特徴がもたらす利点について考察するために、表 2.1に示すように、進化型ハードウェアをふたつの軸で分類する。ひとつの軸は素子の種類、もうひとつの軸は適応の頻度である。素子の種類としては、デジタル素子とアナログ素子を考える。適応の頻度としては、素子の製造時に一回もしくはごくまれに行う場合 (これを Off-line 適応と呼ぶことにする) と、一定時間ごとに頻繁に行う場合 (これを On-line 適応と呼ぶことにする) を考える。

この分類においては、従来の進化型ハードウェアの研究は、いずれも Off-line 適応のデジタル素子に分類できる。一方本研究では、表 2.1に示すように、新たに二つの領域を対象とする。ひとつは、On-line 適応のデジタル素子 (On-line デジタル EHW と呼ぶ) であり、もうひとつは Off-line 適応のアナログ素子 (Off-line アナログ EHW と呼ぶ) である。

以下では、それぞれの EHW について利点を述べる。

2.5.1 On-line デジタル EHW

On-line デジタル EHW は、デジタル回路の外的な環境の変化への適応を目的とする。たとえばパタン識別器をこの EHW で実現しようとする場合、識別対象の性質が時間とともに変化する場合に有効である [18]。その場合、GA の適応度としてパタンの正解識別率を用いて、一定時間ごとに EHW の構造を変化させ、それに適応する。また、この EHW を画像データ圧縮器として用いる場合、画像データの性質に適応して EHW の構造を変化させる。画像データの性質は、事前に知ることができないので、処理する画像に応じた構造を EHW が獲得する [19, 20]。その場合、画像の圧縮率を GA の適応度として用いて、最高の圧縮率が得られるように適応する。

以上の例からわかるように、On-line デジタル EHW の利点は、以下の 2 点に集約される。

- 自律性
  外的な環境が動的に変化する場合、もしくは未知の場合において、人手を介さずに自律的に環境に適応することができる。

- 非線形性
  パタン識別器やデータ圧縮器のように、処理に非線形性が必要な場合においても有効な手法である。

2.5.2 Off-line アナログ EHW

Off-line アナログ EHW は、アナログ回路の内的な環境への適応を目的とする。アナログ技術は過去のものという認識が一部にあるが、実はハイエンドのデジタル情報処理ほど、アナログ素子の性能のバラツキや不完全さに影響され、それが処理上の隘路となっている。自然界とのインタフェイスは常にアナログ値であり、その入力が不完全であると、その後のデジタル処理での挽回にも限界がある。このアナログ回路の不完全さを解決するために、従来の設計手法では、設計者が回路の性質を詳細に把握し、バラツキや素子間の相互干渉を低減させるために、補正回路の挿入やダミー素子の配置などさまざ

まな工夫を行っていた。しかし、それらの理論的統一的な方法は確立されておらず、ノウハウ及び職人技に頼っているのが現状である。そのアナログ回路職人の確保は、デジタル回路の発展とともに急激に難しくなっている。

この問題点を解決するために、Off-line アナログ EHW では、アナログ回路に可変構造を導入し、その調整を遺伝的アルゴリズムによって行う。この調整は、回路の製造時に原則一回行うものとする。遺伝的アルゴリズムによって、人手を介さずに自動的に大規模な調整を行うことができる。

このような手法によって具体的にはどのような利点が生じるかといえば、以下の 3 点に集約される。

- 歩留まりの改善

  現在アナログ回路の製造時には、できあがった回路が仕様を満たすかどうか精査し、仕様を満たさないものは捨てられる。しかしこの EHW では、仕様を満たさない回路に対して GA が実行され、仕様を満たすように可変部分が調整される。

- チップサイズの削減

  アナログ LSI において素子のバラツキを低減する方法として、精度の良い製造プロセスを用いている。精度がよいということは、プロセスの幅が太いことを意味するので、その結果マスク面積が増大する。これは製造コストの増大および電力消費量の増大につながる。しかしこの EHW の手法によって、バラツキを吸収できるので、より精度の悪いプロセスを用いることができる。その結果、マスク面積を削減でき、製造コスト、電力消費量の削減につながる。

- 設計時の省力化

  この EHW では LSI 作成後に調整が可能であるので、従来の設計時のさまざまな工夫を低減することができる。その結果、開発期間の短縮、人的コストの削減が可能である。

これらの利点はさまざまなアナログ回路に適用することができるので、その産業的波及効果も大きい。

2.5.3 本研究の位置付け

本研究では、提案する On-line デジタル EHW および Off-line アナログ EHW の具体例である LSI を実現し、それを現実の問題に適用するためのアルゴリズムの提案および実装を行うことによって、このような方法の有効性を検証する。

On-line デジタル EHW の実現例として、GRD (Genetic Reconfiguration of DSPs) チップと呼ぶ専用 LSI を開発した。Off-line アナログ EHW の実現例としては、携帯電話などで広く使用されている IF フィルタ (中間周波数フィルタ) 用 LSI を開発した。以下の章では、それぞれの LSI について説明し、現実の問題にどのように適用するかの学習手法について述べる。

第 3 章

進化型ニューロチップ –GRD チップ–の開発

3.1 緒言

本章では、On-line デジタル EHW の実現例である進化型ニューロチップの開発について説明する。このチップの目的は、高速で自律的な非線形関数学習器を実現することにある。それによって、パタン分類、時系列予測、非線形適応制御などの問題において、従来手法では困難であった実時間性が要求されかつ環境が時間ともに変化するものへの適用が可能となる。

このチップでは、非線形関数の学習器としてニューラルネットワークを用いる。ニューラルネットワークの学習の問題点として、ネットワークの構造 (中間層素子数や中間層素子の種類) によって大きく性能が左右されるという問題がある。よって時変環境における On-line 学習においては、それらを問題に応じて変化させる必要がある。一方、ニューラルネットワークを実時間性が要求される問題に適用するには、高速に実行するためのハード (並列計算機もしくは専用チップ) が必要となる。そのため、実時間性が要求されかつ時変環境での問題にニューラルネットワークをもちいるためには、ハード上のネットワークの構造を問題に応じて柔軟に変化させる必要がある。しかし従来のハードではネットワークの構造を実行時に柔軟にかえることができないという問題点がある。

本研究では、これら二つの問題点を解決するために、

1. 3 層のニューラルネットワークの中間素子数と中間素子の種類を遺伝的アルゴリズムを用いて動的に決定する学習手法
2. 上記学習手法およびニューラルネットワークの実行を単一のチップで実現する専用 LSI のアーキテクチャ

の二つを提案する。本研究で実装した専用 LSI は、100MHz の RISC プロセッサと 33Mhz の DSP(Digital Signal Processor)15 個から構成される。RISC プロセッサ上で実行される GA が、ニューラルネットワークを実行する DSP の設定を一定時間おきに再構成し、常に環境に適応して最適な性能を発揮できるようにする。

以下ではまず、ニューラルネットワークの On-line 学習における問題点を概観したのち、提案する学習手法および開発した専用 LSI について説明する。つぎに、提案した学

習手法がどの程度の関数近似能力があるのかシミュレーションを行ったので、その結果についてのべる。

3.2 ニューラルネットワークのOn-line学習

階層型ニューラルネットワークは任意の入出力関数の近似が可能で、誤差逆伝搬法(BP法)[21]と呼ばれる教師あり学習を用いて、パタン分類装置や制御装置として広く研究されている。しかし現在ニューラルネットワークの産業応用においては、off-line学習を行うものがほとんどである。つまり、ネットワークの学習をあらかじめ行っておいて、実行時にはネットワークの構造および重み係数はまったく変化させない。そのため、ネットワークがおかれる環境が変化する場合には用いることができない。よって、ニューラルネットワークを幅広い分野で応用するためには、実行時にネットワークを変化させるon-line学習が必要である。以下ではニューラルネットワークのon-line学習を行う場合の問題点について述べる。

3.2.1 ネットワークの構造決定

ニューラルネットワークのon-line学習を行うには、ネットワーク内の結合重みを環境に応じて変化させてやればよい。しかし、ネットワークの性能はネットワークの構造によって大きく左右されることが知られている[22]。もし中間層素子の数がその問題を近似するのに十分な数存在しない場合は、いくら結合重みを変化させても学習はうまくいかない。また、逆に中間層素子の数が、必要以上におおければ過学習[23]が生じやすくなり、学習の汎化能力が低下する。しかし現状では、ネットワークの最適な構造を決定するための理論的な指針はなく、人間が試行錯誤を通じて決定していることが多い。

この問題点を解決するために、ネットワークの構造を問題に応じて自動的に決定するさまざまな研究が行なわれており、それらは次に示す三つの方式に分類できる。

1. AIC (Akaike's Information Criterion) [24]等の情報量基準量を用いて、中間層素子数を決定する[22]。

2. 不要な中間層素子を削除したり、必要な中間層素子を追加する [25]。

3. GA を用いて最適なネットワーク構成を決定する [26, 27]。

On-line 学習においては、これらの手法による、問題に応じた最適な構造決定が必須である。本研究のアプローチは3番目に分類される。

3.2.2 ニューラルネットワーク実行用ハードウェアの問題点

しかし、前節のアルゴリズムをもちいてネットワークの構造を決定できるとしても、それを実行するニューラルネットワーク実行用ハードウェアは、現在開発されているものでは実行中に構造を柔軟に変化させることができない [28, 29]。

よって、On-line 学習をハードウェア上で行なうためには、柔軟にネットワーク構造を変化させられるハードウェアが必要となる。さらに、そのハードウェアを産業応用でもちいる場合には、実装がコンパクトにならなくてはならない。そのため、ホストマシンを必要とせずに単独で実行できるハードウェアが望まれる。

本研究で開発した専用 LSI では、上記二つの要請を満たすものである。つまり、GA を実行する RISC プロセッサが、ネットワーク演算のための DSP を問題に応じて再構成することができる。またその RISC が LSI 内部に内蔵されているために、ホストマシンを必要としない。

3.2.3 学習時間

On-line 学習においては、学習時間も実用上重要な要素となる。一般的な多層パーセプトロン (Multi-Layer Perceptron : MLP) を BP 法で学習する場合は、学習が遅くなる場合が多い。これは、MLP はシグモイド素子を中間層素子として用いているために、入力に対する大域的な反応性から学習時にエネルギ関数の局所解に捕らわれやすくなるためである。

この問題を解決する方法として、RBF (Radial Basis Function) ネットワークという手法が提案されている [30, 31]。RBF ネットワークにおいては、中間層の素子にガウシ

アンなどの、入力に対して局所的な反応をする素子が用いられている。このため、問題によっては学習速度が最大で問題によっては数百倍以上の高速な学習が可能となっている [32]。

しかしながら高次元の入出力関数を近似する場合、RBF ネットワークは MLP と比較して必要な中間層素子数が非常に増大するという問題点がある。このことは、計算時間の増大につながり、ハードウェアで実行する場合にはその規模が大きくなってしまう。つまり、MLP と比較すると学習時間とハードウェア量の間でトレードオフの関係が成立することになる。したがって、シグモイド関数あるいは RBF の単種の関数を用いるよりは、その両者を混在させることによって、学習時間およびハードウェア量の両者において妥当なネットワークを構成できると考える。本研究では、このシグモイド関数と RBF が混在したネットワークを Hybrid Neural Network (HNN) とよぶことにする。

この HNN においては、どのようにシグモイド素子と RBF の素子を混在させるかが重要な問題である。本研究では、この混在方法の決定に遺伝的アルゴリズムを用いることを提案する。遺伝的アルゴリズムによって、与えられた問題に対して適応的にネットワーク構成を決定する。以下では提案する遺伝的アルゴリズムについてくわしく説明する。

3.3 Hybrid Neural Network の遺伝的学習手法

3.3.1 問題の定式化

まずはじめに HNN が対象とする問題を定義する。ネットワークへの入力変数を $\boldsymbol{x} = (x_1, \cdots, x_r) \in \boldsymbol{R}^r$、出力を $y \in \boldsymbol{R}$ とすると、HNN は N 個の入出力データの組

$$(\boldsymbol{x}^1, y^1), \cdots, (\boldsymbol{x}^N, y^N) \tag{3.1}$$

が与えられたもとで、関数 $y = f(\boldsymbol{x}) : \boldsymbol{R}^r \to \boldsymbol{R}$ を近似する。なお、出力 y は簡単のためにスカラであるとしたが、容易にベクトルに拡張することができる。

図 3.1(b) に HNN の構成を示す。HNN は、入力層、中間層、出力層の 3 層からなる。入力層の各素子は、外部からの入力を中間層の各素子に配分し、中間層の各素子はその

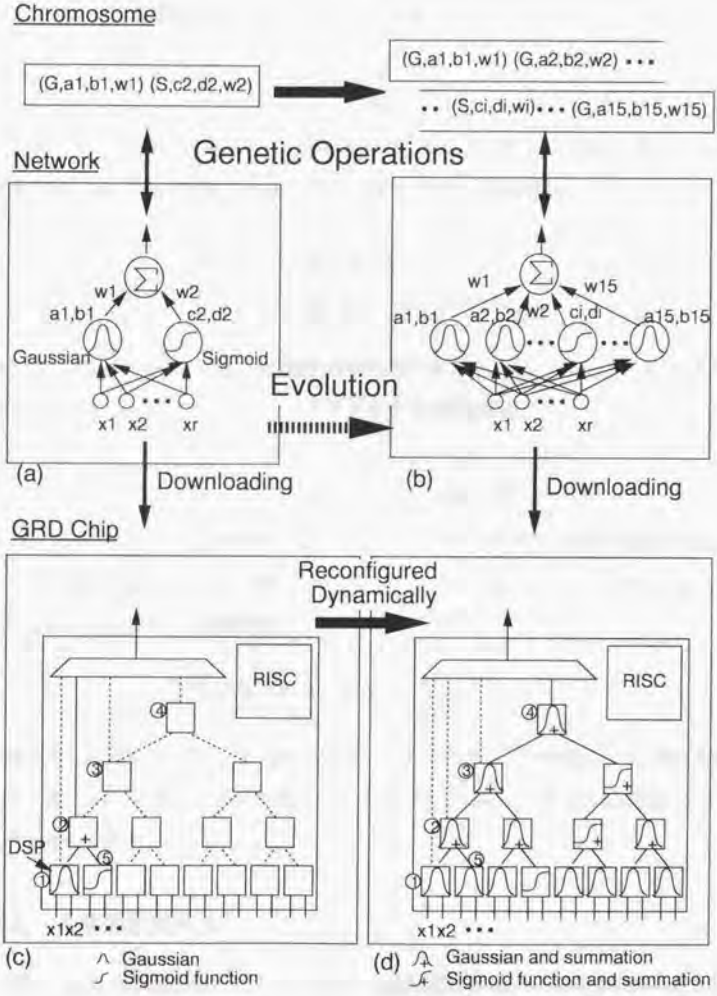


図 3.1: GRD チップ概念図

入力から RBF もしくはシグモイド関数を用いて出力を計算する。出力層の素子は、中間層各素子の出力の加重和をとり、ネットワークの出力とする。つまり、HNN

$$y^* = f^*(\boldsymbol{x}) = \sum_{k=1}^{n} w_k \mu_k(\boldsymbol{x}) \tag{3.2}$$

という関数 f^* を与える。ここに、$\mu_k(\boldsymbol{x})$ は RBF またはシグモイド関数であり、n は中間層素子数、w_k は重み係数である。また、RBF の具体的な関数形としては、

$$\mu_k(\boldsymbol{x}) = \phi(\boldsymbol{x}, \boldsymbol{a}_k, \boldsymbol{b}_k) \tag{3.3}$$

$$= \prod_{i=1}^{i=r} exp(-(x_i - a_{ik})^2 / b_{ik}^2) \tag{3.4}$$

を用いる。$\boldsymbol{a}_k = (a_{1k}, \cdots, a_{rk})$ は RBF の中心値ベクトル、$\boldsymbol{b}_k = (b_{1k}, \cdots, b_{rk})$ は RBF の幅を示すベクトルである。一方、シグモイド素子の関数形は、

$$\mu_k(\boldsymbol{x}) = \frac{1}{1 + exp(\sum_{i=1}^{r} c_{ik} x_i - d_k)} \tag{3.5}$$

である。$\boldsymbol{c}_k = (c_{1k}, \cdots, c_{rk})$ は重み係数ベクトル、d_k はシグモイド関数の閾値である。

ここで、$\boldsymbol{a}^n = (\boldsymbol{a}_1, \cdots, \boldsymbol{a}_n)$、$\boldsymbol{b}^n = (\boldsymbol{b}_1, \cdots, \boldsymbol{b}_n)$、$\boldsymbol{c}^n = (\boldsymbol{c}_1, \cdots, \boldsymbol{c}_n)$、$\boldsymbol{d}^n = (d_1, \cdots, d_n)$、$\boldsymbol{w}^n = (w_1, \cdots, w_n)$ とし、評価関数

$$J^n(\boldsymbol{a}^n, \boldsymbol{b}^n, \boldsymbol{c}^n, \boldsymbol{d}^n, \boldsymbol{w}^n) = \frac{1}{2} \sum_{p=1}^{N} (y^{p*} - y^p)^2 \tag{3.6}$$

を定義する。ただし、$y^{p*} = f^*(\boldsymbol{x}^p)$ である。このとき解くべき問題は、N 個の入出力データの組 (3.1) に対し、$J < \varepsilon$ をみたす $n, \boldsymbol{a}^n, \boldsymbol{b}^n, \boldsymbol{c}^n, \boldsymbol{d}^n, \boldsymbol{w}^n$ を求める問題になる。ε は必要な近似の精度である。

3.3.2 遺伝的学習手法

本研究で提案する遺伝的アルゴリズムは、上記問題において中間層素子数 n および中間層素子の種類 (シグモイド素子または RBF) を、与えられた問題に対して適応的に決定する。また、パラメータ $\boldsymbol{a}^n, \boldsymbol{b}^n, \boldsymbol{c}^n, \boldsymbol{d}^n, \boldsymbol{w}^n$ の初期値も遺伝的アルゴリズムによって決定し、最急降下法をもちいて調整する。

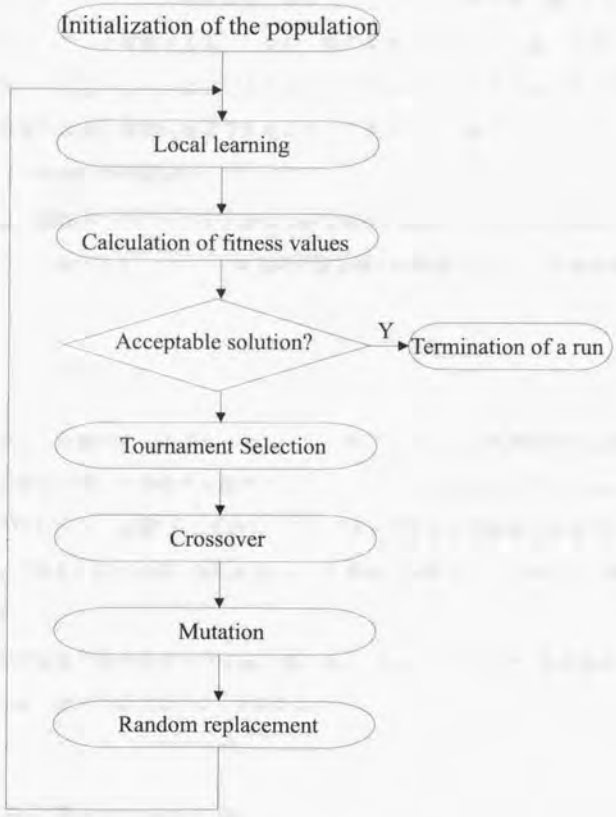


図 3.2: HNN の遺伝的学習手法のフローチャート

図 3.2に本論文で提案する遺伝的学習方法のフローチャートを示す。まず初期集団を用意した後、毎世代ごとに、適応度に応じて個体を淘汰し、生き残った個体に対し、交叉、突然変異の遺伝的操作を行う。そして、遺伝的操作を行った全個体に対して最急降下法を用いた学習を行う。以上の操作を繰り返すことによって、局所解に捕らわれずに適切な解を発見することを期待できる。しかし、時変環境においては、遺伝子型の多様性が消失した場合、環境の変化に追従できないという問題が知られている [33]。そこで本研究では、環境の変動に頑健に対応できるように、集団中の一部分をランダムな個体と置き換える (Random Immigrant)。

以下では、染色体へのコード化方法について述べ、淘汰、交叉、突然変異および置換の方法について説明する。つぎに、最急降下法を用いた学習方法について説明する。

3.3.3 コード化

各染色体は、n 個の遺伝子からなり、1 つの遺伝子で 1 つの中間層素子を表現する。つまり、k 番目の素子を表現する遺伝子には、$(G, \boldsymbol{a}_k, \boldsymbol{b}_k, w_k)$ もしくは $(S, \boldsymbol{c}_k, d_k, w_k)$ がコード化されており、RBF もしくはシグモイド素子であるかの情報と各素子のパラメータが記述してある。たとえば、$(G, \boldsymbol{a}_1, \boldsymbol{b}_1, w_1)$ の遺伝子は図 3.1(b) において一番左側の RBF 素子を示す。

また、初期個体の染色体はすべて n_{ini} 個の遺伝子をもっているが、交叉および突然変異によって n の値は各染色体によって異なるようになる。

3.3.4 適応度および淘汰方法

適応度としては、各染色体が表現する HNN に対する評価を用いる。評価関数としてすでに式 (3.6) で J^n を定義したが、教師データにノイズが含まれている場合には、RBF 素子数 n の増加に伴って過学習を起こす可能性がある。そこで提案手法では、n のいたずらな増加を防ぐために RBF ネットワークに対する評価関数として、赤池の情報量基準

（AIC）の一種である次式の C を用いる。

$$C = Nlog(E^n) + 2S^n \tag{3.7}$$

$$E^n = \frac{1}{N}\sum_{p=1}^{N}(y^{p*} - y^p)^2 \tag{3.8}$$

$$S^n = 2n \tag{3.9}$$

E^n は、与えられた教師データに対する近似誤差の平均であり、この値が小さい程精度の高い学習が行なわれていることを意味する。また S^n は、ネットワークの複雑度を表す項であり、この値が小さい程、教師データに含まれるノイズ成分が学習結果に反映されにくくなっている。式（3.7）の C は E^n と S^n のバランスをとるものであり、この値が小さいものほどすぐれたネットワーク構成であると考える。

淘汰方法としては、サイズが2のトーナメント方式を用いる [34]。この方法は、まず集団中からランダムにふたつの個体を選び、そのふたつのうち優性染色体の評価値 C がより小さいほうの個体を次世代に生き残らせる個体とする。そして生き残った個体の数が集団の個体数 Pop に達するまで、その操作をくり返すものである。また、集団中で最も優秀な個体を無条件で次世代に残すエリート保存戦略も併用する。

3.3.5 交叉

交叉は、淘汰によって選ばれた親個体 A,B から子供個体 A',B' を作る方法である。GA では染色体の交叉方法の設計が重要な問題であり、染色体において適応度の上昇に貢献している部分的な構造（Building Block）を破壊しないようにする必要がある [35]。

そこで、本研究では図3.3に示すような交叉方法を用いる。まず、親個体 A のネットワークからシグモイド素子のひとつをランダムに選び、それが p 番目の素子であったとする。そのとき、交叉の結果子供 A' の染色体に含まれる遺伝子は、親 A において

$$\forall j,\ \boldsymbol{c}_j \cdot \boldsymbol{c}_p + d_j \cdot d_p > 0 \quad (\text{シグモイド素子の場合}) \tag{3.10}$$

$$\forall j,\ \boldsymbol{a}_j \cdot \boldsymbol{c}_p - d_p < 0 \quad (RBF\text{素子の場合}) \tag{3.11}$$

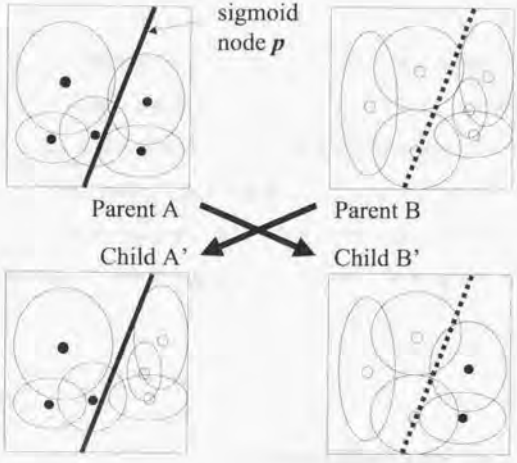


図 3.3: 交叉方法

をみたすすべての遺伝子と、親 B においてこの条件をみたさないすべての遺伝子とする。子供 B' の染色体には、子供 A' の染色体に含まれなかった親 A、親 B のすべての遺伝子が含まれる。この操作の意味することは、シグモイド素子に関しては、p 番目の素子の重み係数と自分の重み係数の相関が 0 以上のものを選び、RBF 素子に関しては、p 番目の素子の判別面の一方の側に含まれるものをすべて選んでいる。このような操作によって、ネットワークの部分構造が保たれ、探索が有効に進むと考える。なお交叉確率は P_c とする。

3.3.6 突然変異

突然変異は、交叉後の各染色体に発生する。突然変異の操作としては、遺伝子の挿入、遺伝子の削除の 2 種類を用いる。

遺伝子の挿入

この操作は、新たな中間層素子一つを、ネットワークに加える操作である。シグモイド素子をネットワークに加える場合は、そのパラメータはランダムに決定する。また、RBF 素子を新たに加える場合の RBF の中心位置および重み係数もまたランダムに決定する。新たに加える RBF の幅は、各入力変数ごとに $[0, b_0 \times (MAX_i - MIN_i)]$ の範囲でランダムに定める。なお、b_0 はあらかじめ与えられた定数である。また、遺伝子を挿入するか否かの確率は、シグモイド素子については P_{si}、RBF 素子については P_{ri} とする。

遺伝子の削除

この操作は、中間層素子の一つをネットワークから削除する操作である。重み係数 $w_1, \cdots, w_n$ の中で絶対値が最も大きいものを w_{max}、最も小さいものを w_{min} とするとき、$|w_{min}/w_{max}|$ が一定の値 D よりも小さい場合、その素子を削除する。もしそのような素子がない場合は、ランダムに素子をひとつ選んで削除する。なお、遺伝子を削除するか否かの確率は P_d とする。

3.3.7 個体の置き換え

この操作は、集団中のある個体を新規な個体に置換する操作である。これは環境の変化に追従するために、集団中の多様性を維持するために行なう。新規な個体は、遺伝子を n_{ini} 個もつ染色体をランダムに生成する。個体を置き換えるか否かの確率は、P_{im} とする。

3.3.8 最急降下法による学習

遺伝的操作によって作られたすべての染色体について、推定の精度をよりよくするために次式で与えられる最急降下法によって学習を行なう。

$$a_{ik}(h+1) = a_{ik}(h) - \alpha \frac{\partial J^n}{\partial a_{ik}} \tag{3.12}$$

$$b_{ik}(h+1) = b_{ik}(h) - \alpha\frac{\partial J^n}{\partial b_{ik}} \tag{3.13}$$

$$c_{ik}(h+1) = c_{ik}(h) - \alpha\frac{\partial J^n}{\partial c_{ik}} \tag{3.14}$$

$$d_k(h+1) = d_k(h) - \alpha\frac{\partial J^n}{\partial d_k} \tag{3.15}$$

$$w_k(h+1) = w_k(h) - \alpha\frac{\partial J^n}{\partial w_k} \tag{3.16}$$

$$i = 1, \cdots, r \quad k = 1, \cdots, n$$

ここに、h は最急降下法のくり返し回数、α は学習係数である。

最急降下法による学習は、式 (3.12) – (3.16) を h_{max} 回くり返した後に終了し、$a_{ik}(h_{max})$, $b_{ik}(h_{max})$, $c_{ik}(h_{max})$, $d_k(h_{max})$,$w_k(h_{max})$ $(i = 1, \cdots, r\ k = 1, \cdots, n)$ の値を新たな遺伝子の値にする。つまり最急降下法による学習結果は次世代に引き継がれることになる。

ところで、最急降下法を用いた学習では最適な学習係数を決定するのが一般的に難しい。値が大きすぎる場合には学習が発散し、値が小さすぎる場合には学習に時間がかかる。そこで本手法では、学習係数 α を可変とし、各染色体ごとに異なる値をもたせる。各染色体での学習係数の初期値はすべて α_{ini} とし、以下の方法で値を変更する。最急降下法によって h_{max} 回学習を繰り返した後に、J^n の値が学習前の状態より小さくなっている場合には、α の値を 2 倍にする。逆に、学習前の状態より大きくなっている場合には、α の値を 1/2 にする。なお、一度 α の値を 1/2 にする操作を行なった染色体では、それ以降に交叉、突然変異が発生しないかぎり 2 倍にする操作は行なわない。

3.4 GRD チップのアーキテクチャ

3.4.1 概要

図 3.4に GRD チップのマスク写真、図 3.5 に GRD チップの全体の構造を示す。GRD チップは、RISC プロセッサと 15 個の専用 DSP から構成される。RISC プロセッサは、NEC で開発された V830 RISC コア (32bit、100Mhz) である。DSP は、33Mhz 16bit 固定小数点 で動作し、RBF やシグモイド関数の演算を高速に行なうための関数近似ユ

図 3.4: GRD Chip

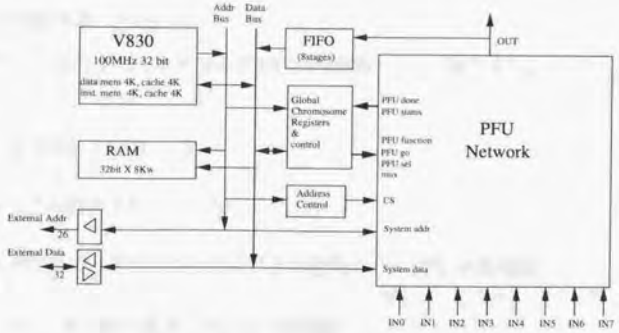


図 3.5: Overview of the GRD Chip

ニットをもっている。この DSP を以下では PFU (Programmable Function Unit) とよぶ。15 個の PFU は 2 進木状に接続されており、その木の高さを再構成することができる。2 進木状に接続した理由は、ネットワークの計算を行なう際に出力層における加算を高速に行なうためである。また、中間素子数が大きなネットワークを実行する際に複数個チップを接続する場合にも、2 進木状に接続したために並列度を損なうことなく拡張が容易に行なえる。

RISC プロセッサは、前節で提案した遺伝的アルゴリズムを実行し、PFU の再構成を行なう。つまり GRD チップは、外部ホストを必要とせず自律的に自分自身を再構成することができる。これは産業応用を考えた場合、コンパクトな実装が可能であり、大きな利点となる。

PFU は、1PFU につきひとつの中間層素子を同時に計算できる。よって、1GRD チップでは 15 素子の計算を並列に実行可能である。実行速度は、MLP の計算においては 319 MCPS (Mega Connection Per Second) である。また最急降下法による学習も、各 PFU において並列に実行される。もし 15 ノード以上のネットワークを計算したい場合は、複数個チップを接続するか、1PFU を時分割で使用して複数の中間層素子を計算するかのいずれかを選ぶ。もし 1 枚の VME triple-height ボードに 9GRD チップを搭載し、19 インチラックをもちいて 16 ボードを接続した場合は、46 GCPS (Giga Connection Per Second) を達成可能である。

以下では、V830 プロセッサおよび PFU の機能について説明する。

3.4.2 V830 プロセッサ

V830 は以下の機能を担っている:

1. 遺伝的アルゴリズムの実行およびその結果による PFU の再構成
2. 各 PFU の最急降下法実行時における制御
3. 複数個 GRD チップが接続される場合の全体の制御

以下ではそれぞれの機能について説明する。

遺伝的再構成

V830 上では遺伝的アルゴリズムが実行され、ネットワークの素子数および各素子の種類を決定する。また、最急降下法のための各素子のパラメータの初期値も遺伝的アルゴリズムが決定する。V830 は染色体情報のうち素子数と素子の種類に関するデータを、図 3.5にある global 染色体レジスタに書き込む。その結果、PFU の 2 進木の高さが再構成され、各 PFU が実行する関数が選択される。また、その他の各素子のパラメータの初期値は、各 PFU が持っている local 染色体レジスタに V830 が直接書き込む。

この再構成にかかる時間は、各染色体レジスタにアドレスが割り付けられているために非常に短い。その時間は再構成すべきネットワークの大きさに依存するが、455 nsec から 461 μsec の間である。(参考：Xilinx 社や Altera 社の FPGA チップの再構成にかかる時間はおよそ 3msec から 19msec の間である。)

次に、遺伝的アルゴリズムの結果得られたネットワークがどのように PFU 上に写像されるかについて図 3.1を例に用いて説明する。まず図 3.1(a) の HNN では、シグモイド素子ひとつと RBF 素子ひとつが存在する。このネットワークは、図 3.1(c) に示すように左下の三つの PFU を用いて実行される。この場合、2 進木の高さは 2 である。

その後、遺伝的アルゴリズムの結果、ネットワークが図 3.1(b) に示すように 15 個の素子をもつように学習された。このネットワークは、図 3.1(d) に示すように 15 個すべての PFU を使って実行される。この場合、2 進木の高さは 4 に再構成される。また各 PFU で実行する素子の種類も図 3.1(b) に対応して再構成される。各 PFU は RBF 素子もしくはシグモイド素子の出力値に重みをかけた後、2 進木の下層の PFU からの出力と自分の計算した値を足しあわせて、上層の PFU に出力する。この結果、ネットワークの並列度を最大限に生かすことができる。

PFUの最急降下法時の制御

遺伝的アルゴリズムによってネットワーク構成が決定し、染色体情報が各PFUに書き込まれた後、各素子のもつパラメータは最急降下法によって調整される。この最急降下法の計算は、各PFUにおいて並列に実行される。このためにブロードキャストレジスタ(BC)がPFUには装備されている。V830は、教師データとPFUで実行されたHNNの計算結果との誤差を計算して、その結果をBCに書き込む。その後、V830は各PFUで同時に最急降下法を開始させる。V830は各PFUで最急降下法による学習が終了したことを確認し、その学習結果を各PFUの持つ染色体レジスタからよみとり遺伝的アルゴリズムの実行を継続する。

GRDチップを複数接続時の制御

V830が複数個接続される場合には、GRDチップをツリー状に接続する。つまり、GRDチップの出力はツリーの上段のGRDチップの入力にFIFOを通じて直接接続される。この場合、ツリーの最上段のチップのV830がツリー全体の制御およびGAのプログラムを実行する。他のチップのV830は、ノンマスカブル割込によって、最上段のチップのV830によって制御され、最急降下法の制御等を行なう。最上段チップのV830とその他のV830間のデータのやりとりは、チップ外部に設けられた2ポートFIFOバッファを通じて行なわれる。

3.4.3 Programmable Function Unit (PFU)

PFUは16ビット入力を二つもつDSPであり、16ビットの出力を生成する。PFUの入力は、他のPFUの出力もしくは外部機器からの入力を8段階のFIFOを通じて供給される。PFUはALU (Arithmetic Logic Unit) およびMultiplierをもっており、それらは並列に動作する。また、PFUを制御するために、4種類のファームウェアのインストラクション(32bit)がある。詳細は付録に記述してある。

各PFUは、16bitx252個のlocal染色体レジスタをもっており、そこに中間層素子の

表 3.1: 基本的な演算の計算時間

	GRD	SUN Ultra2 200MHz
execution of 15 RBF node	0.88 μsec	5.05 μsec
execution of 15 sigmoid node	0.64 μsec	4.95 μsec
execution and learning of 15 RBF node	3.49 μsec	138 μsec
execution and learning of 15 sigmoid node	2.12 μsec	132 μsec

パラメータを保持する。1 つの RBF 素子には、$2 \times r + 1$ のレジスタが必要であるので、1PFU は最大で 84RBF 素子を時分割で計算することができる。結局、GRD1 チップでは最大 $84 \times 15 = 1260$ 素子の実行が可能である。

また各 PFU は、ガウシアンやシグモイド関数を高速に計算するための関数近似ユニット (CAM) をもっている。このユニットは、非線形関数を折れ線近似で計算するユニットであり、3 サイクルで近似計算を終了する (1 サイクルは約 30nsec)。その結果、r 入力の RBF 素子の計算には $11+9r/2$ サイクル、シグモイド素子の計算には $15+3r/2$ サイクルかかる。また最急降下法の計算には、RBF 素子で $14+18r$ サイクル、シグモイド素子で $21+7d$ サイクルかかる。

表 3.1に 4 入力 15 素子の場合の、SUN Ultra2 200MHz 上でのシミュレーション時間との比較を示す。およそ 40 倍はやいことがわかる。

3.5 提案手法の関数近似性能

以下では、提案手法の性能を、静的な関数近似問題において評価する。動的な問題における性能は、次章以降で検証する。

以下の節では、人工的な関数近似問題、カオス時系列の予測問題、現実問題を扱うベンチマーク集に対して、提案した学習手法を適用しその有効性を検証する。

まずはじめに、人工的な関数近似問題に提案手法を適用する。この問題は、(1) RBF

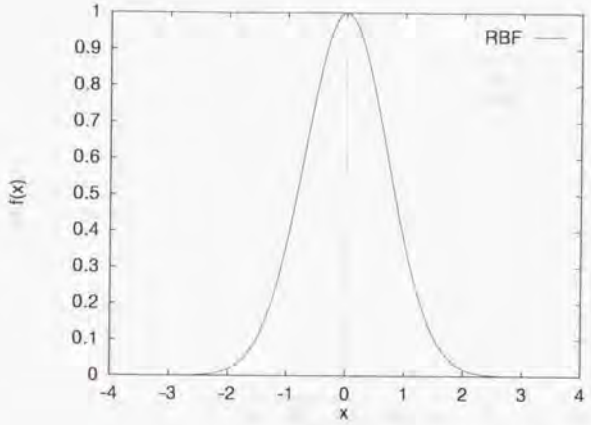


図 3.6: 問題 1: RBF

(2) シグモイド関数 (3) RBF とシグモイド関数を重ね合わせた関数 を提案手法により HNN が正しく近似できるかの実験を行なう。とくに、RBF 素子とシグモイド素子の混在するネットワークが学習の結果どうような混在比になるのかを観察する。実験は、学習手法自体の有効性を示すのが目的であるので、GRD チップのシミュレーションを行なうのではなく、浮動小数点をもちいて提案手法のシミュレーションを行なった。

3.5.1 RBF の近似

近似対象の関数を図 3.6に示す。トレーニングデータとしては、x を $[-4, 4]$ の範囲で均等に選んだ 41 点を用い、提案手法で HNN を学習する。また、GA の探索は G_{max} 世代で打ち切る。なお実験で用いたパラメータは、$G_{max} = 50$、$Pop = 49$、$\alpha_{ini} = 0.1$、$n_{ini} = 4$、$P_c = 0.6$、$P_{ri} = 0.1$、$P_{si} = 0.1$、$P_{im} = 0.1$、$P_d = 0.2$、$b_0 = 0.2$、$D = 0.01$、$h_{max} = 10$ である。また、比較対象として、GA において RBF 素子のみを用いる場合、シグモイド素子のみを用いる場合の実験も行なった。RBF 素子のみを用いる場合は、$P_{ri} = 0.2$、$P_{si} = 0.0$、シグモイド素子のみを用いる場合は $P_{ri} = 0.0$、

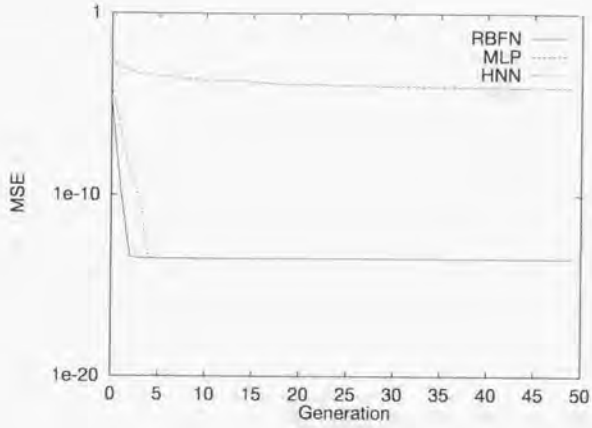


図 3.7: 問題 1: 関数近似誤差

$P_{st} = 0.2$ とし、その他のパラメータは同じ値を用いる。

それぞれの学習方式による各世代における最優秀個体の関数近似誤差および中間層素子数を図 3.7、図 3.8に示す。なお、各結果は 10 試行の平均値が示してある。図より RBF のみを用いる場合と HNN はほぼ完全に RBF を近似できていることがわかる。また、中間層素子数も、初期値の 4 から徐々に減少し、理想的な値 1 に収束している。一方、シグモイド素子のみを用いる場合では、誤差は徐々に減少しているものの完全には近似できていない。またその近似するために素子を 5 つ程度使用していることがわかり、この関数形にはシグモイド素子が不向きであることがわかる。

図 3.9に各世代における HNN の RBF 素子、シグモイド素子それぞれの数を示す。図より、20 世代以降ではシグモイド素子が削除されて RBF 素子のみのネットワークになっていることがわかる。

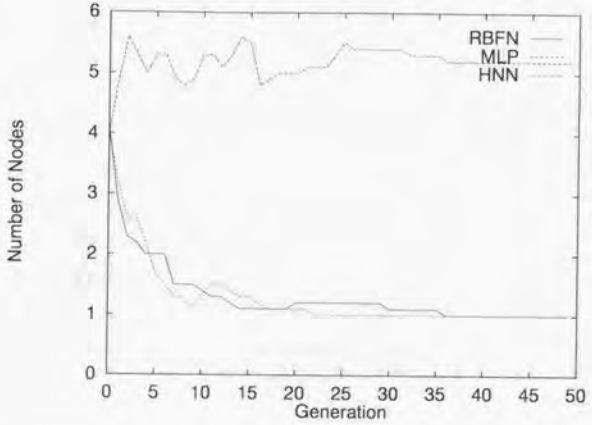


図 3.8: 問題 1: 中間層素子数

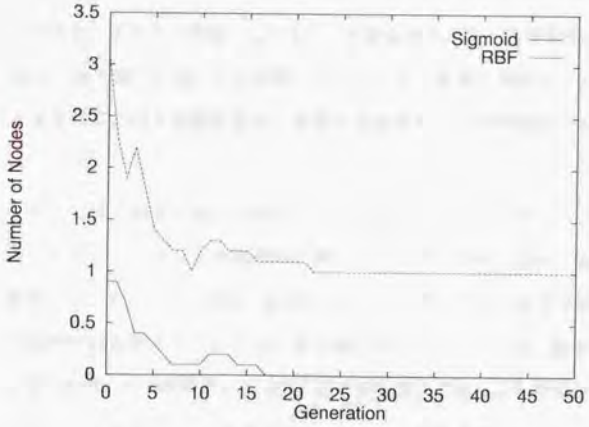


図 3.9: 問題 1: HNN の各素子数

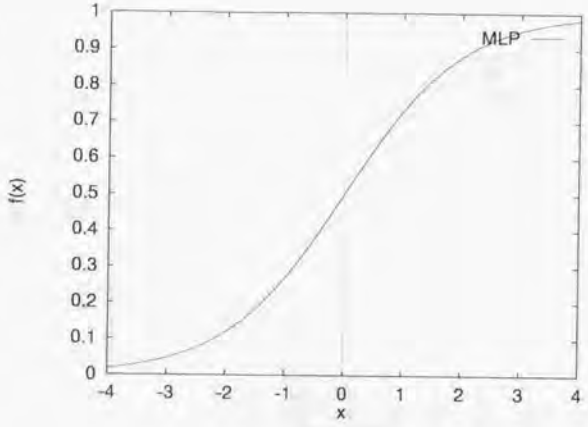


図 3.10: 問題 2: シグモイド関数

3.5.2 シグモイド関数の近似

つぎに近似対象をシグモイド関数にした場合の実験結果を示す。実験条件は前節とまったく同様である。図 3.10に対象とする関数形を示す。また前節と同様に、それぞれの学習方式による各世代における最優秀個体の関数近似誤差および中間層素子数を図 3.11、図 3.12に示す。

図よりシグモイド素子のみを用いる場合と HNN はほぼ完全にシグモイド関数を近似できていることがわかる。また、中間層素子数も、初期値の 4 から徐々に減少し、理想的な値 1 に収束している。一方、RBF 素子のみを用いる場合では、誤差は徐々に減少しているものの完全には近似できていない。また世代がたつにつれ、素子数が増加しつづけていることがわかり、この関数形には RBF 素子が不向きであることがわかる。

図 3.13に各世代における HNN の RBF 素子、シグモイド素子それぞれの数を示す。図より、20 世代以降ではシグモイド素子数がほぼ 1 に収束しておりそれ以降は RBF 素子が削除され最終的に素子数が 0 になったことがわかる。

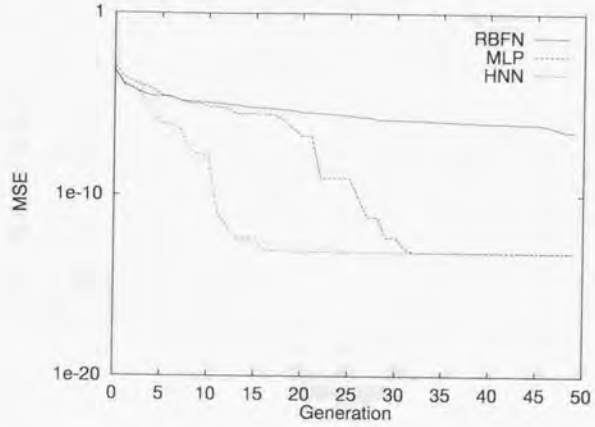


図 3.11: 問題 2: 関数近似誤差

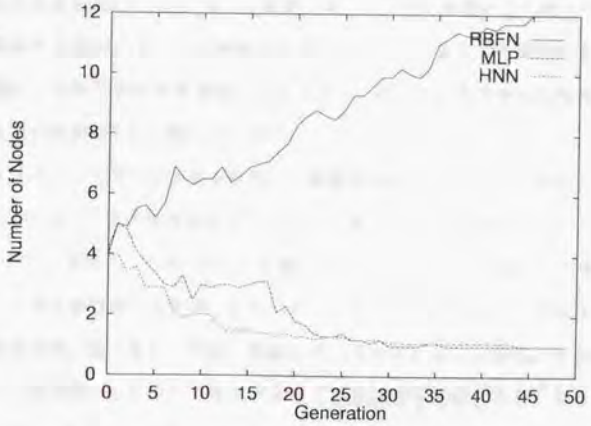


図 3.12: 問題 2: 中間層素子数

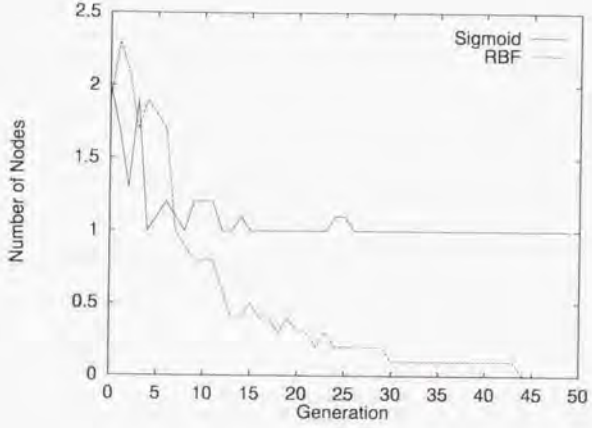


図 3.13: 問題 2: HNN の各素子数

3.5.3 RBF とシグモイド関数を重ね合わせた関数の近似

つぎに近似対象を RBF とシグモイド関数を重ね合わせた関数にした場合の実験結果を示す。実験条件は前節とまったく同様である。図 3.14に対象とする関数形を示す。また前節と同様に、それぞれの学習方式による各世代における最優秀個体の関数近似誤差および中間層素子数を図 3.15、図 3.16に示す。

図より 3 手法のうち HNN が最も精度よく関数を近似していることがわかる。これはこの関数形が、RBF 素子のみおよびシグモイド素子のみでは近似しにくいことが原因である。ただし、理想的な RBF 素子が 1 個、シグモイド素子が 1 個という形には収束しておらず、エネルギ関数の局所解にとらわれていることがわかる。シグモイド素子のみを用いた場合では、素子数が 5 程度に収束してしまっている。これは、それ以上に素子をふやしても局所解にとらわれる確率がふえて近似精度を改善できなくなったためである。一方、RBF 素子のみを用いた場合では、世代とともに素子数が増加して近似精度を改善していることがわかる。これは RBF が入力に対して局所的な応答を示すために、他の学習結果に影響を及ぼすことなく素子を追加できるためである。

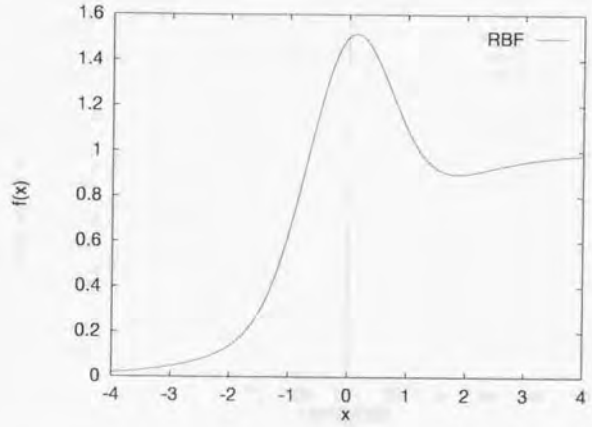


図 3.14: 問題 3: RBF とシグモイド関数を重ね合わせた関数

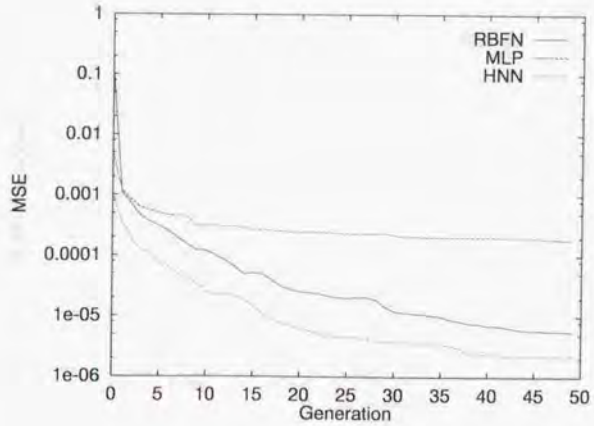


図 3.15: 問題 3: 関数近似誤差

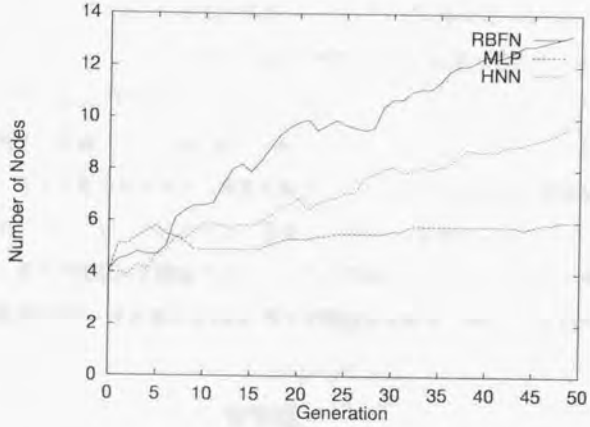


図 3.16: 問題 3: 中間層素子数

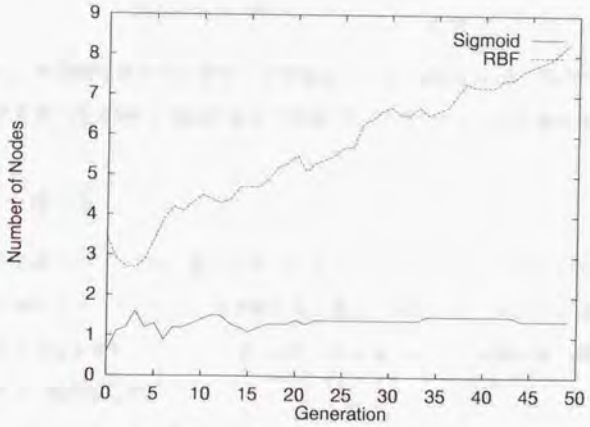


図 3.17: 問題 3: HNN の各素子数

図 3.17に各世代における HNN の RBF 素子、シグモイド素子それぞれの数を示す。図より、シグモイド素子は 1.4 程度に収束しているが、局所解に捕らわれてしまい理想的な 1 にはなっていない。このため与えられた関数を完全に近似できずに、RBF 素子が増加しつづけていることがわかる。

以上問題 1-3 を通じて、RBF 素子のみもしくはシグモイド素子のみを用いて GA でネットワークを学習するよりも、両者を混在させたほうが与えられた関数を高い精度で近似できることがわかった。それは、提案した HNN の学習手法によって、問題の関数形に応じて素子の混在比を調整できているためである。ただし、エネルギ関数の局所解にはまる場合があり、その場合は RBF 素子が増加する傾向にあることがわかった。

3.6 カオス時系列の予測問題

次にカオス時系列の予測問題を通じて、提案した学習手法の有効性を示す。対象とする時系列は、以下の式で与えられる Mackey-Glass の時系列 [36] である。

$$x(t+1) = -bx(t) + a\frac{x(t-T)}{1+x(t-T)^{10}}. \tag{3.17}$$

この問題は、時系列予測のベンチマーク問題としてよく用いられる。以下では、提案手法によって学習した HNN と MLP および RBF ネットワークの性能比較を行なう。

3.6.1 予測方法

時系列の予測方法としては、過去の 4 つのデータ $\{x(t), x(t-D), x(t-2D), x(t-3D)\}$ を用いて未来の 1 データ $x(t+I)$ を予測する。過去の研究に従い [37, 32]、本研究ではパラメータとして $a = 0.2$、$b = 0.1$、$T = 17$、$D = 6$、$I = 85$ を用いる。図 3.18に発生した時系列の一部を示した。

ニューラルネットワークに与えるトレーニングデータとしては、t=500–4000 の間でランダムに選んだ 500 点を用いる。そのデータを用いて、以下のように各ネットワークを学習させる。

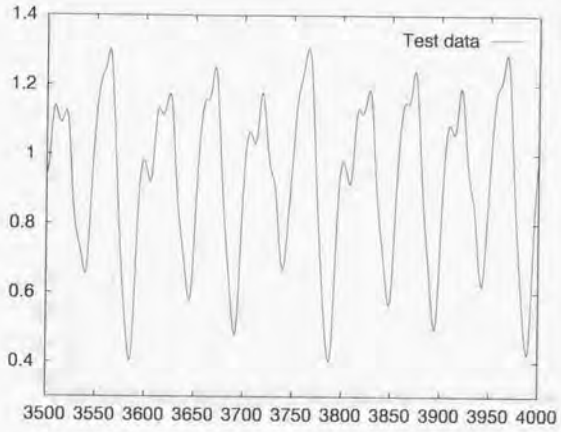


図 3.18: Mackey-Glass の時系列

Hybrid Neural Network

遺伝的アルゴリズムにおいては、中間層素子数の上限を 15、30、45、60、75、135 と設定してそれぞれ実験を行なった。集団の個体数は 80 とし、打ち切り世代 (G_{max})=150 において探索をうちきった。実験で用いた GA のパラメータは、$\alpha_{ini} = 0.01$、$n_{ini} = 8$、$P_c = 0.6$、$P_{ri} = 0.1$、$P_{hi} = 0.1$、$P_{im} = 0.1$、$P_d = 0.2$、$b_0 = 0.05$、$D = 0.01$、$h_{max} = 10$ である。

RBF Network

RBF ネットワークの学習方法としては、k-means クラスタリング法を用いた [32]。この方法は、RBF ネットワークの代表的な学習方法である。素子数は GA と同様の設定にする。

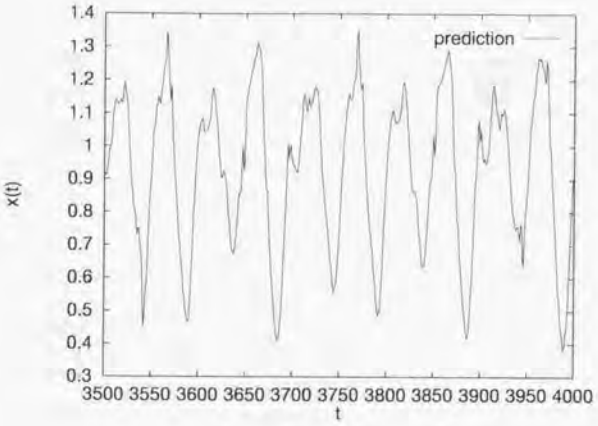


図 3.19: HNN による予測

Multi Layer Perceptron

MLP の学習には、最急降下法を用いる。学習係数は $\alpha_{ini} = 0.01$ とし、素子数は GA と同様の設定にする。計算量を提案手法と同等にするために、トレーニングデータに対して最急降下法の繰り返し回数は、$G_{max} \times h_{max}$ とし、初期ネットワークを毎回変えて GA の個体数と同一の回数だけ試行する。全試行中最も良い結果を最急降下法による学習結果とする。

3.6.2 学習結果の評価方法

各ネットワークの学習終了後に、t=4000–4500 のテストデータに対する正規化予測誤差で結果を評価する。正規化予測誤差とは、平均自乗誤差の平方根をテストデータの標準偏差で割ったものである。

3.6.3 シミュレーション結果

各手法の実験は乱数種を変えてそれぞれ 10 回行なった。得られた正規化誤差の平均を

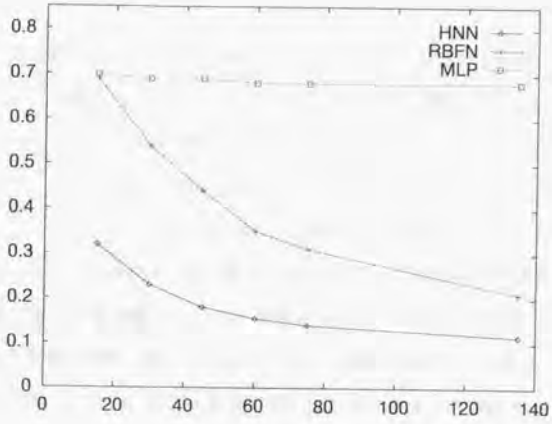


図 3.20: 正規化誤差と中間層素子数の関係

図3.20に示す。また、$n_{max} = 135$ の場合の提案手法によるテストデータの予測結果を、図3.19に示す。図3.18と比べると、おおよその傾向は予測できていることがわかる。ただし、データが極値をとるあたりでは、うまく予測はできていない。

図3.20をみると、提案手法は その他の手法に比べて同一の中間素子数でより良い予測精度を実現していることがわかる。とくに、k-means クラスタリング法に比べると予測の正規化誤差がおよそ 1/2 となっている。これは、GA による多点探索によって局所解をさけながら、適切なパラメータを最急降下法により学習した結果といえる。

実行時間に関しては、9GRD チップ (135 中間層素子) を用いて学習するのにおよそ Sun Ultra2 200MHz の 160 倍である (シミュレーション結果)。9 GRD チップでは、262 秒かかるのに対して、SUN では 41797 秒かかった。

3.7 ベンチマーク問題による性能評価

最後に現実問題を扱うベンチマーク問題を通じて、関数近似性能を評価する。問題としては、ベンチマーク集 Proben1 [38] から、三つの関数近似問題 hearta1、building1 および flare1 を選んだ。k-means 法を用いた RBF ネットワーク、BP 法を用いた MLP および提案手法をもちいた HNN において性能の比較をする。

k-means クラスタリング法の RBF 素子数は、提案手法で得られたネットワークの素子数と同一にする。MLP の成績は文献 [38] に記述されている最良の学習結果を用いる。また、提案手法において最急降下法による学習を行わない場合の実験も行う。これは、最急降下法による学習が有効に働いているかどうかを確認するためである。

なお文献 [38] の指示に従い、結果はテストデータに対する以下の指数 E によって評価する。

$$E = 100 \cdot \frac{y_{max} - y_{min}}{N \cdot P} \sum_{j=1}^{P} \sum_{p=1}^{N} (y_j^{p*} - y_j^p)^2 \tag{3.18}$$

ここで、P はトレーニングデータの出力次元数、y_{max}, y_{min} はそれぞれトレーニングデータ中の最大の出力値、最小の出力値を表す。

3.7.1 心臓病診断問題 (hearta1)

この問題は 35 個の特徴値を用いて、ある心臓病が進行しているかどうかを 5 段階で診断する問題である。トレーニングデータ、テストデータはそれぞれ 690 個、230 個ある。

提案手法を用いた実験では、トレーニングデータに対する評価値 C が低くなるように HNN を学習する。探索は G_{max} 世代で打ち切り、最終世代における最も優秀な個体を、提案手法の学習結果とする。なお実験で用いたパラメータは、$\alpha_{ini} = 0.1$、$n_{ini} = 1$、$Pop = 25$、$P_c = 0.5$、$P_{si} = 0.1$、$P_{ri} = 0.1$、$P_d = 0.2$、$P_{im} = 0.1$、$b_0 = 10.0$、$D = 0.01$ 、$h_{max} = 10$、$G_{max} = 40$ である。

それぞれの学習手法によって得られた評価値 C およびテスト結果 E を表 3.2に示す。MLP を除いた各手法は、それぞれ 10 試行し、その結果の平均が示してある。

実験の結果、提案手法は他のいずれの手法に比べても優れた学習結果を得ている。図

表 3.2: hearta1 の学習結果

手法	評価値 C	テスト結果 E
HNN(提案手法)	-2.22×10^3	3.82
HNN(GA: 最急降下法なし)	-1.94×10^3	5.67
RBFN(k-means 法)	-1.72×10^3	7.68
MLP(BP 法)	-1.85×10^3	4.55

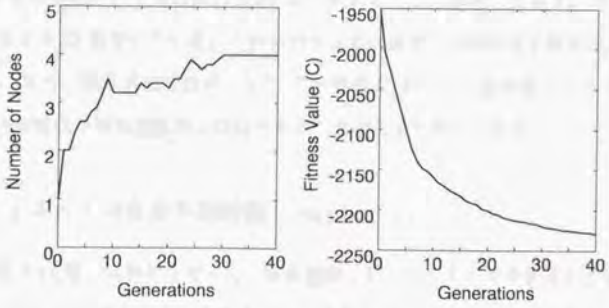


図 3.21: 中間層素子数および評価値 C

表 3.3: building1 の学習結果

手法	評価値 C	テスト結果 E
HNN(提案手法)	-1.46×10^4	0.23
HNN(GA: 最急降下法なし)	-0.99×10^4	0.62
RBFN(k-means 法)	-0.94×10^4	0.86
MLP(BP 法)	-0.86×10^4	1.36

3.21には、各世代において最も優秀な個体の中間層素子数および評価値 C の変化を示した。世代が経つにつれて素子数が増加し、第 30 世代以降では一定の値をとっている。その結果、過学習をおこすことなくテストデータに対しても良好な性能を得ており、提案手法で採用した適応度の有効性を確認できた。また、最急降下法を用いなかった場合と比較すると、E の値が2/3 程度になっており、最急降下法を用いた学習が有効に働いていることがわかる。

提案手法の結果得られたネットワークの RBF 素子数の平均は 4.0 であり、k-means 法では 比較条件を同一にするために RBF 素子数を 4 として固定してある。MLP ではシグモイド素子を 32 個用いている。これらの手法では適切な中間層素子数を決定するのが難しいのに比べ、提案手法では式 (3.7) の評価値 C を GA の適応度として用いることにより、中間層素子数を適応的に決定できることが大きな利点である。

3.7.2 エネルギ消費量予測問題 (building1)

この問題は 14 個の特徴量を用いて、ある建物におけるエネルギ消費量を予測する問題である。トレーニングデータ、テストデータはそれぞれ 3156 個、1052 個ある。提案手法を用いた実験では、探索は第 20 世代で打ち切り、その他のパラメータは hearta1 の実験と同一の値を用いる。

それぞれの学習手法によって得られた評価値 C およびテスト結果 E を表3.3に示す。MLP を除いた各手法は、それぞれ 10 試行し、その結果の平均が示してある。hearta1

表 3.4: flare1 の学習結果

手法	評価値 C	テスト結果 E
HNN(提案手法)	-3.46×10^3	0.56
HNN(GA: 最急降下法なし)	-3.45×10^3	0.65
RBFN(k-means 法)	-3.51×10^3	0.58
MLP(BP 法)	-3.03×10^3	0.74

の実験と同様に、提案手法は他のいずれの手法に比べても優れた学習結果を得ている。

3.7.3 太陽フレア数予測問題 (flare1)

この問題は 24 個の特徴量を用いて、太陽フレア数の時系列を予測する問題である。トレーニングデータ、テストデータはそれぞれ 800 個、266 個ある。提案手法を用いた実験では、探索は第 10 世代で打ち切り、その他のパラメータは heart1 の実験と同一の値を用いる。

それぞれの学習手法によって得られた評価値 C およびテスト結果 E を表 3.4に示す。MLP を除いた各手法は、それぞれ 10 試行し、その結果の平均が示してある。提案手法は他のいずれの手法に比べても優れた学習結果を得ているが、その差はそれほど大きくない。これは、この問題自体の学習が難しく、トレーニングデータが十分な情報を持っていないためである。

3.8 本章のまとめと今後の課題

本章では、On-line デジタル EHW の実現例である GRD チップの開発について説明した。GRD チップは、100MHz の RISC プロセッサと 33Mhz の DSP15 個から構成される。RISC プロセッサ上で実行される GA が、ニューラルネットワークを実行する他の DSP の設定を一定時間おきに再構成し、常に問題に対応して最適な性能を発揮できるよ

うにする。その結果、従来手法では困難であった実時間の応答性が必要でかつ環境が時間ともに変化する問題に適用することが可能である。またネットワークの再構成、学習、実行にホストマシンを必要とせずコンパクトな実装が可能であり、制御機器への埋め込み用途に適している。

また上記RISCで実行されるGAは、本研究で提案した手法であり、ニューラルネットワークの中間素子数と中間素子の種類を問題に応じて動的に決定する。その学習性能を、本章ではまず静的な関数近似問題で評価した。人工的な問題、カオス時系列予測、現実問題のベンチマークにおいてシミュレーションを行なった結果、いずれの問題においても従来手法にくらべて高い近似精度を得ており、その有効性を確認できた。

以下の章では、GRDチップを動的な環境における問題 - 適応等化器、強化学習、ATMネットワークの制御に応用することによってその有効性を示すことにする。

第 4 章

GRD チップの適応等化器への応用

4.1 緒言

本章では、ディジタル通信における適応等化器に、GRD チップを用いることを提案する。適応等化器とは、通信の受信器側でノイズや波形歪みの影響を除去し送信符合を推定するものである。適応等化器は時変環境で用いられるため、推定には On-line 学習が必要となる。また高速通信のためには、ハードウェアによる高速化小型化が必須である。従来の線形フィルタを用いた等化器では、伝送路の非線形歪みが無視できない場合著しい性能の低下が生じる。提案する等化器では、GRD チップが実現する非線形関数によって、非線形歪みが存在する場合にも送信符合を推定することができる。また、計算量の多い指数関数を表引きによる近似計算で求めることができるので、高速な等化器を構成することができる。シミュレーションの結果、従来の線形等化器に比較して、優れた等化性能を実現できた。

以下ではまず、従来の適応等化器について概観し、その問題点を述べる。次に提案する GRD チップを用いた適応等化器について説明する。学習時に教師データが与えられる場合について、非線形歪みが存在する伝送路の適応等化のシミュレーションを行ない、その結果について説明する。

4.2 適応等化器とは

近年、携帯電話の爆発的な普及など、ディジタル通信は急速に発展してきた。さらに今後は、画像などの大容量データ転送などのため、高速ディジタル通信の実現が必要不可欠とされている。高速な通信を実現するためには、熱雑音、インパルス雑音、変復調処理等による非線形歪みの影響を除去することが重要な問題である。また、移動体通信においては、送信電波は直接受信器に到達する他に、自然の地形や周囲の建造物などで反射、散乱し多数の経路を経て受信器に到達する。このような異なる伝送路 (マルチパス伝送路) を経た電波の干渉は、マルチパスフェージングと呼ばれ、遅延時間の異なる信号が干渉を起こすため、受信信号に波形歪が生じる [39] (図 4.1)。ケーブル TV で用いる伝送路においても、インピーダンス不整合から生じる反射波によって同様の現象が生じる

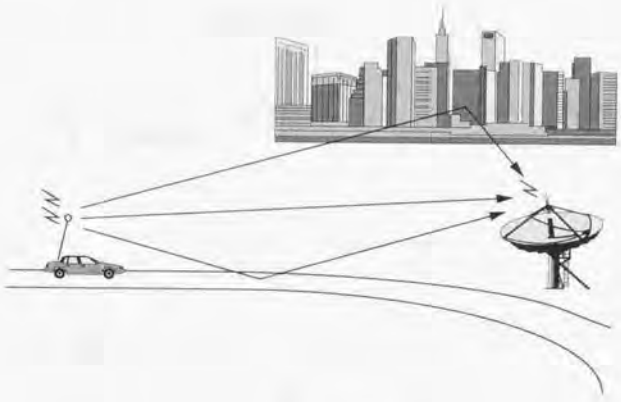


図 4.1: マルチパスフェージング

[39]。このような波形歪は、伝送符号誤りの原因となるため、これを克服する技術が必要となる。これまでに、耐マルチパス変復調方式、スペクトル拡散通信などとともに、適応等化器が研究されてきた。

適応等化器とは、通信の受信器側でフィルタを使用し、正しい送信符合を推定するものである [40][41]。現在一般に用いられている適応等化器では、トランスバーサル型の線形フィルタが使用され、このフィルタの係数を伝送路の状況に応じて学習する。学習時に既知の信号列を送信し、その受信信号を教師データとして使える場合には、LMS (Least Mean Squares) アルゴリズムや RLS (Recursive Least Squares) アルゴリズムなどの適応アルゴリズム [12] を用いて係数を調整する。教師データを使えない場合は blind 等化と呼ばれ、CMA (Constant Modulus Algorithm) [42] [43] が主に使われている。しかしいずれの場合にも、伝送路の非線形歪みが無視できない場合、線形フィルタでは著しい性能の低下が生じる。

そこで本論文では、GRD チップを用いて適応等化器を構成することを提案する。HNN のもつ非線形性によって、伝送路に非線形の歪みが存在する場合にも、状況に応じて送信信号を推定することができる。また、計算量の多い指数関数を表引きによる近似計算

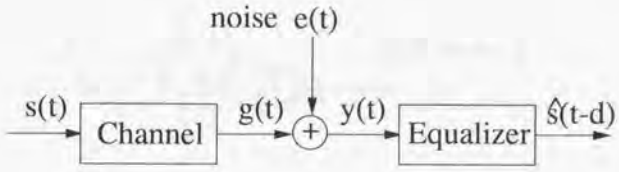


図 4.2: データ伝送システム

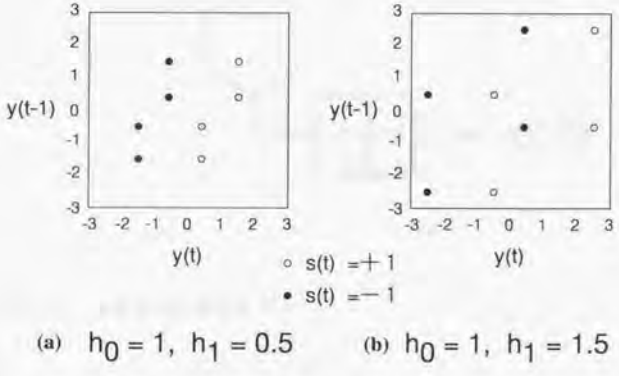


図 4.3: 等化器の入出力

で求めることができるので、高速な等化器を構成することができる。

4.3 従来の適応等化器

想定したデータ伝送システムを図 4.2に示す。送信符合 $s(t)$ は、1 または -1 の 2 値をとる符号系列を考える。この送信信号は、伝送路によって波形歪みを受け、ノイズ $e(t)$ が加算されて、信号 $y(t)$ として受信される。この受信信号は、m 個の受信信号データの組として等化器に入力される。等化器は、このデータの組を用い送信信号 $s(t-d)$ を推定することによって、波形歪みを補償する。なお d は、等化器における遅延の値である。

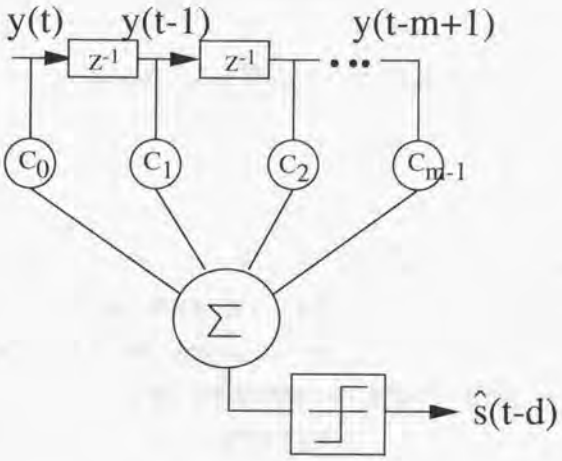


図 4.4: トランスバーサルフィルタ

ここで例として、伝送路の伝達関数 $G(z)$ に、

$$G(z) = h_0 + h_1 z^{-1} \tag{4.1}$$

を考える。これはマルチパス伝送路として、直接波と遅延波 1 つの 2 波が存在することを意味する。また受信信号系列 $y(t)$ を、2 個を組として等化器に入力する $(m = 2)$。この場合等化器に求められる機能は、受信信号 $y(t)$、$y(t-1)$ から送信符号を推定することになる。例として、$h_0 = 1, h_1 = 0.5$ および $h_0 = 1, h_1 = 1.5$ の場合について、送信符号 $s(t)$ と、受信信号 $y(t)$、$y(t-1)$ の関係をそれぞれ図 4.3 (a),(b) に示す。図中の白丸および黒丸は、それぞれ送信符号が 1 および -1 の場合に対応する。ノイズが存在する場合、受信信号は、図 4.3においては白丸および黒丸を中心にノイズの大きさに応じてばらつくことになる。よって求められる等化器の機能は、できるだけ正しく送信符合を推定するように、$y(t)$、$y(t-1)$ 平面を $s(t-d)$ が 1 または -1 の領域に分割することになる。

現在一般的に用いられている等化器は、図 4.4に示すような、受信信号の線形加重和を

とるトランスバーサルフィルタによって構成されている [40][41]。つまり、推定送信信号列 $\hat{s}(t-d)$ は、

$$\hat{s}(t-d) = sgn\left(\sum_{i=0}^{m-1} c_i y(t-i)\right) = sgn(c^T y) \tag{4.2}$$

$$c = [c_0\ c_1 \cdots c_{m-1}]^T \tag{4.3}$$

$$y = [y(t)\ y(t-1) \cdots y(t-m+1)]^T \tag{4.4}$$

によって与えられる。ここで、c は、フィルタの係数であり、このフィルタ係数を伝送路の状況に応じて学習する。学習時に既知の信号列 (トレーニング系列) を送信しその受信信号を教師データとして使える場合には、LMS アルゴリズムや RLS アルゴリズムなどの適応アルゴリズム [12] を用いて係数を調整する。教師データを使えない場合は blind 等化と呼ばれ、CMA[42] [43] が主に使われている。

この線形等化器を図 4.3の経路に用いた場合、式 (4.2) より、$y(t)$、$y(t-1)$ 平面を一本の直線で分割することになる。図 4.3(a) のように、遅延波の振幅が直接波の振幅より小さい場合は、$s(t)$ が 1 の領域と -1 の領域を 1 本の直線で分割できるため、トランスバーサルフィルタを用いても、等化は可能である。しかし、図 4.3(b) のように、遅延波の振幅が直接波の振幅より大きい場合 (非最小位相系の伝送路という)、原点を通る一本の直線では $s(t)$ が 1 の領域と -1 の領域とを完全には分割できず、誤り率を 25% 以下にすることはできない。また、伝送路が式 (4.1) のように線形経路でなく、非線形の性質をもつ場合には、トランスバーサルフィルタでは性能が悪化することが知られている [44]。

これらの問題点を解決するために、これまでに DFE (Decision Feedback Equalizer) [40]、MLSE (Maximum Likelihood Sequence Estimation) [45][46]、RBFN (Radial Basis Function Network) を用いた等化器 [47][48] などが提案されている。DFE では、図 4.4の判定出力をフィルタへの入力にフィードバックさせることで非線形性をもたせている。しかし、基本的に線形構造をもつことから、経路の非線形性が大きくなった場合には、等化性能が極端に悪化する。MLSE では、精度良く伝送路応答を推定できるが、非常に複雑な演算を行なわなければならないという欠点がある。RBFN を用いた等化器では、図 4.4のトランスバーサルフィルタのかわりに RBF ネットワークによってフィルタ

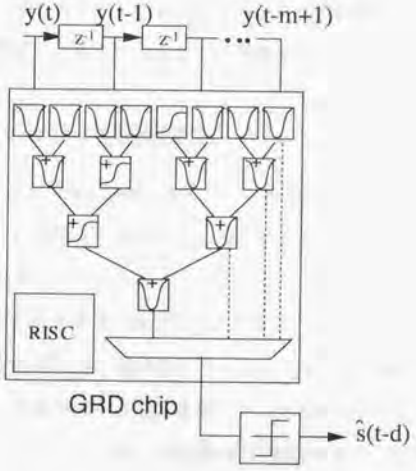


図 4.5: GRD チップを用いた適応等化

を構成する。この等化器では、RBF の個数および各 RBF の中心値、幅が適切に選ばれた場合、精度の良い非線形の等化器を構成することができる。しかし、それらの値が適切に選ばれなかった場合には、性能を十分に発揮できないことが指摘されている [49]。また、RBF の計算のために指数関数の演算を行なわねばならず、実行時に非常に多くの演算量を必要とすることが問題である。

そこで本論文では、GRD チップを用いて等化器を構成することを提案する。HNN が実現する非線形関数によって、非線形歪みが存在する場合にも状況に応じて送信符合を推定することができる。また、指数関数を表引きによる近似計算で高速に計算することができる。

4.4 GRD チップを用いた適応等化

トレーニング系列が教師データとして送信される場合における、GRD チップを用いた適応等化器 (GRD 等化器とよぶ) について説明する。図 4.5に示すように、GRD 等化

器ではトランスバーサルフィルタのかわりにGRDチップを用いる。GRDチップは、与えられた教師データを正しく推定できるように学習する。

4.4.1 静的環境における非線形等化

GRD等化器が、非最小位相伝送路においてどの程度ビット誤り率を低減できるかシミュレーションを行った。伝送路モデルとして、図4.3(b)のように$h_0 = 1, h_1 = 1.5$となる場合を考える。GRDチップへの入力としては、$m = 2$でサンプリングされた受信データの組を与え、等化における遅延の値dは0とする。

GAの集団の個体数は81とし、図4.3(b)の$y(t)$、$y(t-1)$平面上の点8個を教師データとして与えている。教師データの受信信号には、信号対ノイズ比 (SNR) 15dBの白色ガウスノイズをのせ、1世代毎に新しい教師データを与える。この教師データを用いて、世代毎に各個体が10回最急降下法を用いた学習を繰り返す。

まず収束性を調べるために、グラフ横軸に示される世代だけGRDチップを進化させ、その後集団中の最良個体の構造を固定して、それに10^5個のランダム符号を与えることによりビット誤り率を求めた。なお、試行は100回行われており、その結果の平均が、図4.6に示してある。図中には、トランスバーサルフィルタをGRDチップと学習符号数を同一の条件で動作させたときのビット誤り率も示してある。結果は、トランスバーサルフィルタがおよそ25%の高い率で誤っているのに対し、GRD等化器では数%の誤り率に収束している。これは、GRD等化器のもつ非線形関数の合成能力によるものである。

次に、SNRの影響について示す。伝送路などの条件は前述のものと同じである。GAの実行世代は100世代で、試行は100回行った。図4.7にその結果を示す。SNRが大きくなるにつれ、ビット誤り率が小さくなることがわかる。

4.4.2 時変環境における適応等化

前節では、伝送路の特性が時間的に変化しない場合を考えたが、実際の伝送路、特に移動体通信においては受信信号の状態は環境に応じて時間的に大きく変動する。そのた

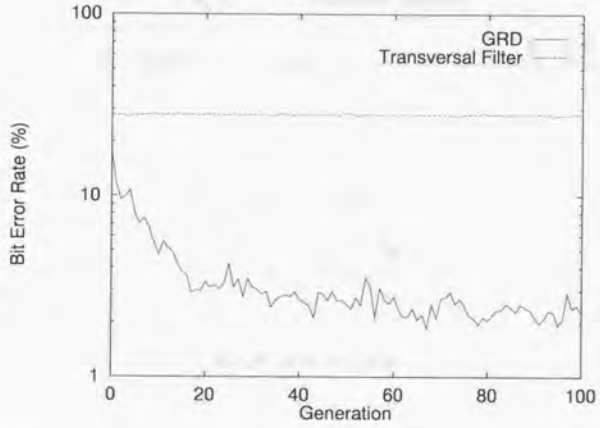


図 4.6: GRD 等化器の収束性 (SNR 15dB)

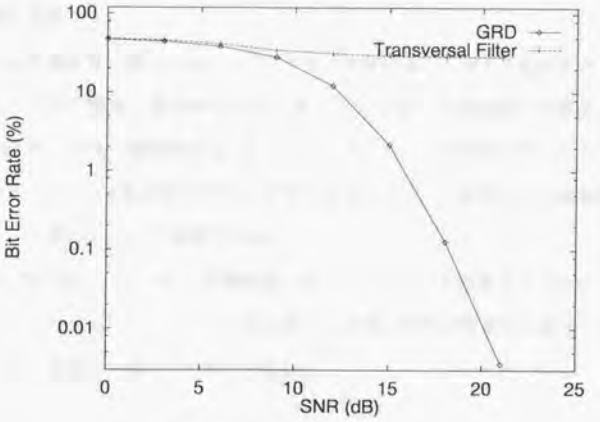


図 4.7: GRD 等化器の SNR 依存性

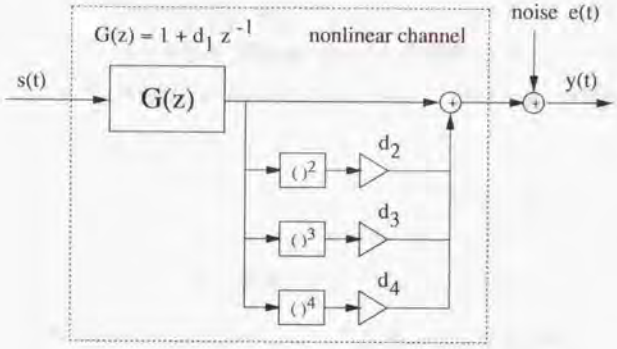


図 4.8: 非線形伝送路

め、伝送路特性の時間変動に適応的に追従することができる適応等化器が必要となる。

適応等化器を用いる場合、環境の変化に追従するために送信信号列の中に既知の信号系列が一定周期で挿入されている。GRD 適応等化器では、その信号の歪み方から伝送路の特性を推定し、等化器の各係数を調整する。このことによって、伝送路の変動に対して適応的に追従する。

また、GRD 適応等化器は 2 組の GRD チップを用いる。一方の GRD チップは、実行用 GRD チップで、実際に受信信号を入力することによって送信信号を推定する。もう一方の GRD チップは、学習用 GRD チップで、トレーニング系列を用いて GRD チップを進化させる。実行用 GRD チップは、学習用 GRD チップで獲得された最も優秀な染色体によって、1 世代ごとに再構成される。

GRD 適応等化器が、伝送路の時間変動に追従できることを確認するために、以下のふたつのシミュレーションを行った。ひとつは、伝送路の特性が断続的に変化する場合で、もうひとつは、連続的に変化する場合である。

断続的に伝送路特性が変化する場合

シミュレーションで想定した伝送路は、図 4.8の非線形伝送路である。この伝送路は、

伝達関数が$G(z)$である線形経路を通過した後、その出力に、非線形の高調波が加算される経路である。このような経路は、現実的にもみられる経路で、ディジタル衛星通信の際などに、高いゲインで増幅するばあいなどにみられる。このような、非線形性が強い経路では、従来のトランスバーサルフィルタを用いるのは難しい。

実験では、$d_2 = 0.6, d_3 = 0.5, d_4 = 0.4$とし、$d_1$が時間変動するものとした。ノイズとしては、SNR 15dBの白色ランダムノイズを加えている。GRDチップへの入力としては、$m = 2$でサンプリングされた受信データを与え、$d = 0$とする。

GAの集団の個体数は81とし、$y(t)$、$y(t-1)$平面上の点8個を教師データとして与えている。d_1の時間変動を反映した教師データが1世代毎に与えられる。d_1は、図4.9に示すように第50世代と、第100世代において急激に変化する。

実行用GRDチップは、学習用GRDチップにおいて学習が1世代おこなわれる間に10^5符号分のランダム符号を処理するものとして、その世代におけるビット誤り率を求めた。その100試行の結果の平均が、図4.10に示してある。d_1が急激に変化しても、一時的にビット誤り率は高くなるものの、すぐにその状況に追従して、誤り率が低くなっているのがわかる。これは、
GRDチップが遺伝的アルゴリズムを用いており、集団中に個体の多様性を維持していることから、対処することができる。

連続的に伝送路特性が変化する場合

前節と同様の条件で、d_1が連続的に変化する場合のシミュレーションも行った。

実験では、はじめに$d_1 = 0.3$の伝送路でGRDチップを進化させ、第100世代から図4.11に示すように環境を変化させる。図4.12には、ビット誤り率の変化(100試行の平均)が示してある。第100世代からの変化をみると、環境の変化に追従して、ビット誤り率を低く保っていることがわかる。

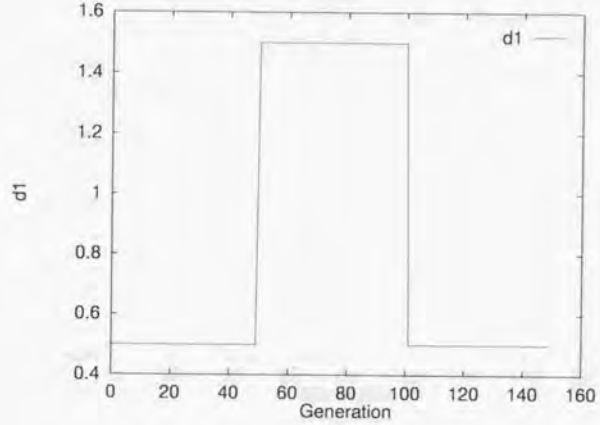


図 4.9: d1 が断続的に変化する環境

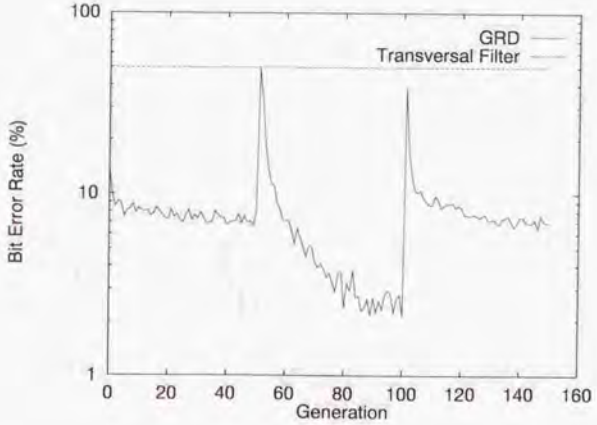


図 4.10: GRD 等化器の適応追従特性 (SNR 15dB)

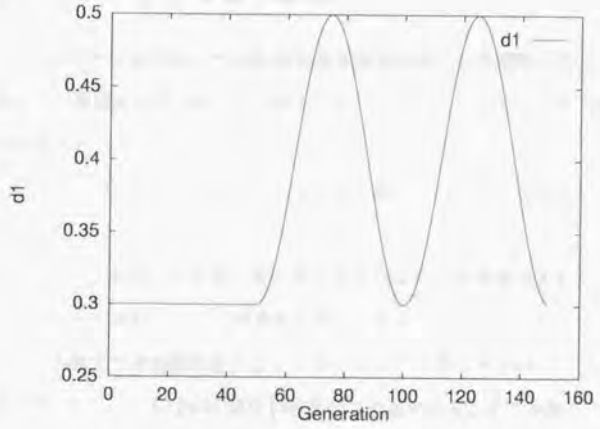


図 4.11: d1 が連続的に変化する環境

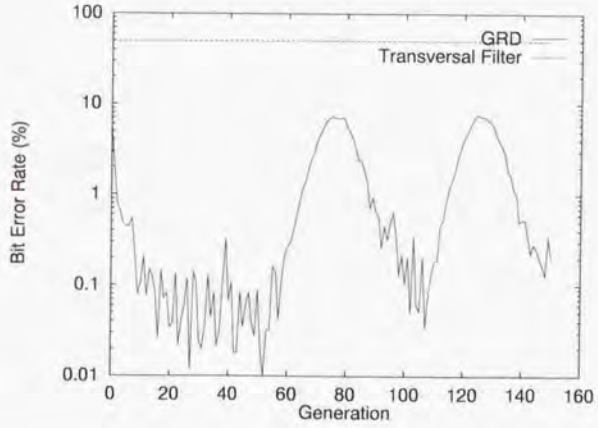


図 4.12: GRD 等化器の適応追従特性 (SNR 15dB)

4.5 本章のまとめと今後の課題

本章では、GRD チップを用いて適応等化器を構成することを提案した。シミュレーション結果から、提案する等化器は伝送路が非線形の歪みをもつ場合でも、高速な等化器として用いることができる。また、進化型ハードウェアが遺伝的アルゴリズムを用いていることから、環境が断続的・連続的に変化する場合にもそれに追従することができる。

今後、GRD チップを用いて実際に適応等化器を作成し、実環境で実験を行なう予定である。また、それと並行してより現実的なモデルを用いたシミュレーションが必要となる。とくに、本論文では送信符合として 1 または -1 の 2 値系列を扱ったが、実際のデジタル通信で用いられている QAM 信号 [39] を扱う必要がある。また本論文の実験では、環境の変化の速度を進化型ハードウェアが対応できる程度のものを選んでおこなったが、どの程度の速度まで追従できるのかなど今後細かいシミュレーションを行なう必要がある。

第 5 章

GRD チップの強化学習への応用

5.1 緒言

本章では、強化学習の一手法である Q-learning に GRD チップを用いることを提案する。誤差逆伝搬法に代表されるようなニューラルネットワークの教師付き学習は、ネットワークの目標とすべき入出力パターンが与えられることを前提としている。しかし、現実の環境においては、目標とすべき入出力パターンが与えられることはまれで、出力の善し悪しを示すスカラー値の強化信号だけが与えられる場合がほとんどである。このような場合に、様々な出力パターンを On-line で試してみて、よりよい結果に結び付く出力パターンを選択する戦略が強化学習（Reinforcement Learning）である [50, 51]。

強化学習には様々な手法があるが、強化信号が一連の出力系列と環境のダイナミクスの結果として未知の遅れの後に与えられる場合に用いられる Q-learning に近年注目が集まっている [52, 53]。Q-learning では、最終的に強化信号が得られるまでに経た各状態に対する評価（有用度）を経験をもとに学習しておき、その有用度が高くなるように制御を行なう。この Q-learning を実環境での問題に適用する場合、有用度関数の状態空間が大きくなり、学習に非常に時間がかかるという問題点がある。この問題点を解決するために、これまでにニューラルネットワークなどの関数近似器を用いる手法が提案されている [54, 55]。しかし、シグモイド素子を用いたニューラルネットワークでは学習が発散することが多い上という問題点がある [55]。また Q-learning を現実問題に対して適用するには、環境の変動に対して高速な学習を行える関数近似器が必要となる。

そこで本論文では、有用度関数の学習に GRD チップを用いることを提案する。これにより、事前の知識を必要とせずに遺伝的アルゴリズムによって適切なネットワーク構成を動的に決定することできる。ロジスティック写像を一定値に制御する問題に提案手法を用いた結果、従来手法に比べてより少ない誤差を実現することができた。

以下ではまず、強化学習の一手法である Q-learning について簡単に説明する。次に提案する強化学習の枠組について説明し、時間とともに目的の変化するロジスティック写像の制御問題に適用した例について報告する。

5.2 強化学習と Q-learning

強化学習の基本的な原理は、さまざまな出力パターンを確率的に試してみて、より良い結果に結びつく出力パターンを選択する、というごく自然な「試行錯誤」によるものである。この時、出力の善し悪しを示す信号は強化信号と呼ばれ、実際の目標に近い出力には正の強化信号を、目標に遠い出力には負の強化信号が出力される。

現実の環境においては、強化信号が一連の出力系列と環境のダイナミクスの結果として未知の遅れの後に与えられる場合が多い。そこで、ある結果が得られるまでに行ったさまざまな出力や通過した状態のうち、どれがその結果に影響を及ぼしたのかを推定することが強化学習での大きな問題となる。この場合、最終的に強化信号が得られるまでに経た各状態に対する評価を経験をもとに学習しておき、各時点においては、次の状態での評価がより大きくなるような出力を選ぶ、という戦略が一般的にとられている。

ここで問題を定式化する。システムの状態変数 $x(t)$ は、制御量 $u(t)$ により、

$$x(t+1) = f(x(t), u(t)) \tag{5.1}$$

に従って変化し、強化信号は状態変数と出力変数の関数として、

$$r(t) = r(x(t), u(t)) \tag{5.2}$$

で与えられるとする。このとき、状態の価値関数（value function）を、その状態から始まって今後得られる強化信号の重みつき総和

$$V(x(t), p) = r(t) + \gamma r(t+1) + \gamma^2 r(t+2) + \cdots \tag{5.3}$$

として定義する。ここで、p はどのような $u(t)$ を出力していくかを決定する制御ポリシーである。また、係数 $0 \leq \gamma \leq 1$ を、遠い将来に得られる強化信号ほど割り引いて評価するための割り引き率と呼ぶ。確率的なシステムでは、$V(x(t), p)$ は式（5.3）の右辺の期待値として定義する。なお、単位時刻後の状態 $x(t+1)$ の価値関数は、

$$V(x(t+1), p) = r(t+1) + \gamma r(t+2) + \gamma^2 r(t+3) + \cdots \tag{5.4}$$

で与えられるため、この 2 つの状態の価値関数の間には、

$$V(x(t),p) = r(t) + \gamma V(x(t+1),p) \tag{5.5}$$

という関係が成り立つ。

このとき、$V(x(t),p)$ を最大にするような $u(t)$ の系列を出力する制御ポリシー p を獲得することが学習の目的となる。システムの状態遷移則および与えられる強化信号の法則性がわかっている場合、とるべき制御ポリシーは、可能な出力 u のうち、その時点での強化信号と次の状態の評価の和が最大になる

$$u(t) = arg \max_{u} [r(x(t),u) + \gamma V(f(x(t),u))] \tag{5.6}$$

を選ぶことである。

しかし、現実の問題ではシステムの状態遷移則および各状態の価値関数は事前にはわからない。そのような場合にシステムの状態遷移則の推定と価値関数の学習を同時に行なう枠組が Q-learning である。具体的には、任意の状態における出力の有用度（utility function）

$$Q(x(t),u(t)) = r(t) + \gamma V(x(t+1)) \tag{5.7}$$

を試行錯誤に基づいて学習し、各時刻の $u(t)$ には、状態 $x(t)$ のもとで最大の有用度を与える出力 $u(t) = arg \max_{u} Q(x(t),u)$ を選ぶ。

有用度の境界条件は、一般に制御系列の終点で与えられるため、有用度 $Q(x(t),u(t))$ の学習は、$r(t)$ と $Q(x(t+1),u(t+1))$ をもとに $Q(x(t),u(t))$ を修正する形で行う。すなわち、

$$Q(x(t),u(t)) := r(t) + \gamma \max_{u}[Q(x(t+1),u)] \tag{5.8}$$

という形で行う。これにより $Q(x)$ の値は、強化信号が実際に与えられる時点から、状態の系列をさかのぼって徐々に $(x(t),u(t))$ の状態空間全体に拡散していく。

これまでの強化学習の研究では、環境から観測される連続量を離散化し、離散的な状態空間において表引き方式で学習を行なっていた [50]。しかしこの方式では、環境に対する事前知識がない場合に機械的に状態空間の離散化を行なうので、状態変数の増加に

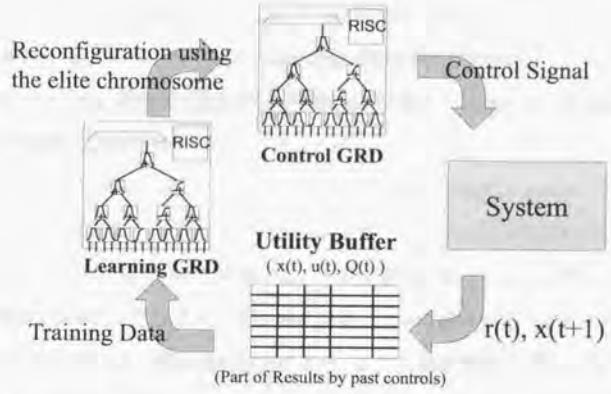


図 5.1: GRD チップを用いた Q-learning の枠組

伴って状態数が指数関数的に増大するという問題が生じる。また、環境に対する事前知識がある場合においても人間が状態空間を設計しなければならず、試行錯誤と経験による部分が大きくなる。

これらの問題点を解決するために、ニューラルネットワークなどの関数近似器を学習の際に用いる手法がこれまでに提案されている [54, 55]。しかし、シグモイド素子を中間層に使ったネットワークでは、最悪の場合学習が不安定化し、重み付けの値が発散するといった例が知られている [55]。また、時間とともにシステムの状態遷移則や強化信号の法則性が変わる時変環境においては、環境が変わるたびに有用度関数を学習しなおさなければならないため、広い状態空間に対して高速な学習をおこなえることが必要となる。

そこで本論文では、関数近似器として GRD チップをもちいる手法を提案する。以下では、その学習の枠組について説明する。

5.3 GRD チップを用いた Q-learning

提案するQ-learningの枠組を図5.1に示す。制御には二組のGRDチップを用いる。ひとつは制御用GRDチップ、もうひとつは学習用GRDチップである。

まず環境に対して、制御用GRDチップに実際の制御を行なわせる。その制御の結果得られた各時刻における$(x(t), u(t))$と$Q(t)$の組を大きさT_{size}の有用度バッファに保存していく。つぎに学習用GRDチップは、バッファに蓄えられたトレーニングデータをもとに、HNNを構成し、有用度を学習する。なお、バッファ内の各教師データは忘却率βで古いデータほど割り引いて評価する。その学習の結果、GAの集団中で最も優秀な個体の情報で制御用GRDチップを再構成する。この再構成を繰り返すことにより、3章の結果示されたHNNの関数近似能力により、広大な状態空間を効率よく学習できる。またGAのもつ個体多様性によって、環境が変動した場合でもすばやく有用度関数を学習しなおすことが期待できる。

以下では、提案手法の有効性を示すために、ロジスティック写像の返す値を一定値に制御する数値実験を行なった。環境の変動として、その目標値を時間とともに変化させる。

5.4 ロジスティック写像カオスの制御

ロジスティック写像とは以下の数式で与えられる写像で、単純な式の形にもかかわらず、aの値が$1+\sqrt{5} < a < 4$のときに$F(t)$の値はカオス的な振舞をとる。

$$F(t+1) = a \times F(t) \times (1 - F(t)) \tag{5.9}$$

本実験では、時系列に沿って生成される状態量$F(t)$に絶対値が$|u(t)| < u_{max}$となる補正を毎回加え、あらかじめ与えられた定数Fに近づける事を目標とする。ここでu_{max}は補正量の最大値である。具体的には、

$$F'(t) = F(t) + u(t) \tag{5.10}$$

$$F(t+1) = a \times F'(t) \times (1 - F'(t)) \tag{5.11}$$

において、$F(t+1)$ を F になるべく近づける制御量 $u(t)$ を探索する。この問題は、現在の制御量 $u(t)$ が遠い将来に渡って影響を及ぼすので、限られた制御量のもとで状態量を一定の値に近づける制御則を発見するのは非常に難しい。

なお、Q-learning における強化信号 $r(t)$ は以下の式で与える。

$$r(t) = 1 - |F - F(t+1)| \tag{5.12}$$

この $r(t)$ を用いて、式 (5.8) にしたがって有用度 $Q(t)$ を計算する。制御の結果、$|F'(t)|$ の値が 1 を超えた場合、制御は失敗したものとみなし $r(t) = -1$ を与える。この場合には、$F(t+1)$ に 0 から 1 の間で任意に発生させた値を設定して制御を再開させる。

5.4.1 実験方法

時間とともに、F を変化させて実験を行う。実験で用いた各パラメータは、$u_{max} = 0.2$、$a = 3.9$、$T_{size} = 200$、$\beta = 0.99$、$\gamma = 0.8$、$\alpha_{ini} = 0.01$、$n_{ini} = 8$、$Pop = 25$、$P_c = 0.6$、$P_{ri} = 0.1$、$P_{si} = 0.1$、$P_{im} = 0.1$、$P_d = 0.2$、$b_0 = 1.0$、$D = 0.01$、$h_{max} = 10$ である。なお、$u(t) = arg \max_u Q(F(t), u)$ を求める際には、u を $-u_{max}$ から u_{max} の間で $u_{max}/100$ きざみで動かすことによって最大値を求める。

また、ROLS 法によって構成された RBF ネットワークおよびシグモイド素子を用いたニューラルネットワークを有用度関数に用いて同様の実験を行ない、提案手法と性能を比較する。ROLS 法を用いた学習方法は、文献 [56] の方法を用い、RBF 素子数は 21 で実験を行なった。この RBF 素子数は、GA の結果得られたネットワークの RBF 素子数の時間平均値と同一にするものである。シグモイド素子を用いたニューラルネットワークの学習方法は、最急降下法を用いる。計算量を提案手法と同等にするために、1 時刻毎にトレーニングデータに対して $h_{max} \times Pop$ 回の学習を行ない、学習係数を調整する。中間層のシグモイド素子数は 21 とし、出力層の素子は、重みつき加算和のみを行なう。

さらに、提案手法において最急降下法による学習を行わない場合の実験も行う。これは、最急降下法による学習が有効に働いているかどうかを確認するためである。

5.4.2 実験結果

連続的な環境変化

まず、F を連続的に変化させる実験を行った。$t < 500$ では、$F = 0.8$ とし、それ以降は $F = 0.6 + 0.2\cos((t - 500)\pi/1000)$ に従い変化させた。なお、ロジスティック写像は、この実験の設定では不動点が 0.74 に存在する。F がこの不動点からはなれるにしたがって、制御を行うのが難しくなる。

実験の結果、各学習手法で得られた式 (5.3) の価値関数の変化の様子を図 5.2に示す。各学習手法は 10 試行し、その結果の平均が示してある。比較のために、なにも制御しない場合の価値関数の値を同時に破線でプロットした。

図 5.2 (1) に示すように、提案手法は従来手法と比べて価値関数の値がより大きくなるような制御を行なっている。また従来手法では、制御が難しくなる時間帯 ($t = 1000-2000, 3000 - 4000$) において、制御を行わない場合に比べかえって価値関数の値が悪くなっている。それに対し提案手法では、制御を行わない場合に比べより大きな価値関数の値をとっている。これは GA による多点探索によって局所解を避けながら、有用度関数を精度よく学習した結果といえる。

一方、ROLS 法においては、図 5.2 (3) に示すように提案手法に比べると全体的に価値関数の値が低い。これは、ROLS 法では、与えられた RBF 素子数で有用度関数を十分に近似できないためである。同様の結果は、シグモイド素子を用いたニューラルネットワークの場合にもいえる。図 5.2 (2) に示すように、有用度関数を十分に近似できないために価値関数の値が全体的に低い。いずれの場合も、中間層素子数を増やすことによって、近似精度を高くすることができるが、時変環境においては事前にその値を決定するのは難しい。

図 5.2 (4) には、提案手法において最急降下法による学習を行わない場合の制御結果を示した。有用度関数を十分に近似できていないため、全体的に価値関数の値が低いことがわかる。このことから、提案手法では最急降下法による学習が有効に働いていることが確認できた。

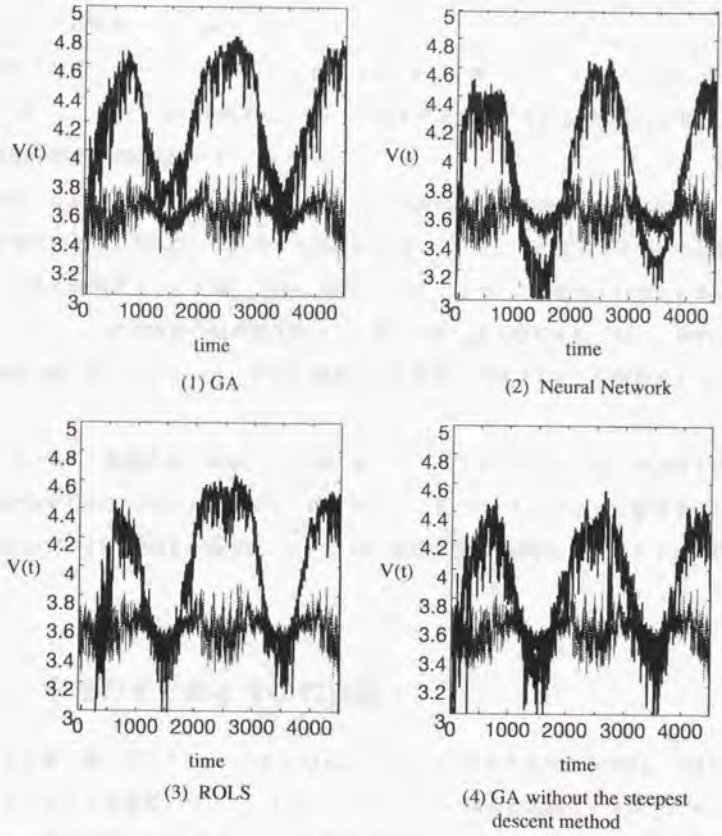


図 5.2: 各学習手法で得られた価値関数

断続的な環境変化

次に、F を断続的に変化させる実験を行った。$t = 500$ ごとに、F を、0.4、0.6、0.8 の順番で周期的に変化させる。$F = 0.4$ の区間は、不動点から離れたところに目標値があるために制御するのが難しい。

実験の結果、各学習手法で得られた価値関数の変化の様子を図 5.3に示す。各学習手法は 10 試行し、その結果の平均が示してある。比較のために、なにも制御しない場合の価値関数の値を同時に破線でプロットした。

前節の実験結果と同様に、図 5.3 (1) に示した提案手法の結果は、従来手法と比べて価値関数の値がより大きくなるような制御を行なっている。また従来手法では、制御が難しくなる時間帯 ($t = 1500 - 2000, 3000 - 3500$) において、制御を行わない場合に比べてかえって価値関数の値が悪くなっている。それに対し提案手法では、一時的に価値関数の値は低下するものの、変化に追従し、目標値に近づけるような制御を行っている。

一方、前節の実験結果と同様に、ROLS 法、ニューラルネットワーク、最急降下法による学習を行わない GA においては、図 5.3 (2) (3) (4) に示すように提案手法に比べると全体的に価値関数の値が低い。これは、有用度関数を十分に近似できないためである。

5.5 本章のまとめと今後の課題

本章では、強化学習のひとつである Q-learning における有用度関数の学習に GRD チップを用いることを提案した。この提案手法により、状態量が連続値をとる場合においても事前知識を必要とせずに動的に有用度関数を構成することができた。

本手法をロジスティック写像の制御問題に適用した結果、従来の ROLS 法を用いる場合とニューラルネットワークを用いる場合に比べ、より少ない誤差を実現することができた。

今後の課題としては、Q-learning は一般に学習が遅く、適切な制御則を獲得するまで

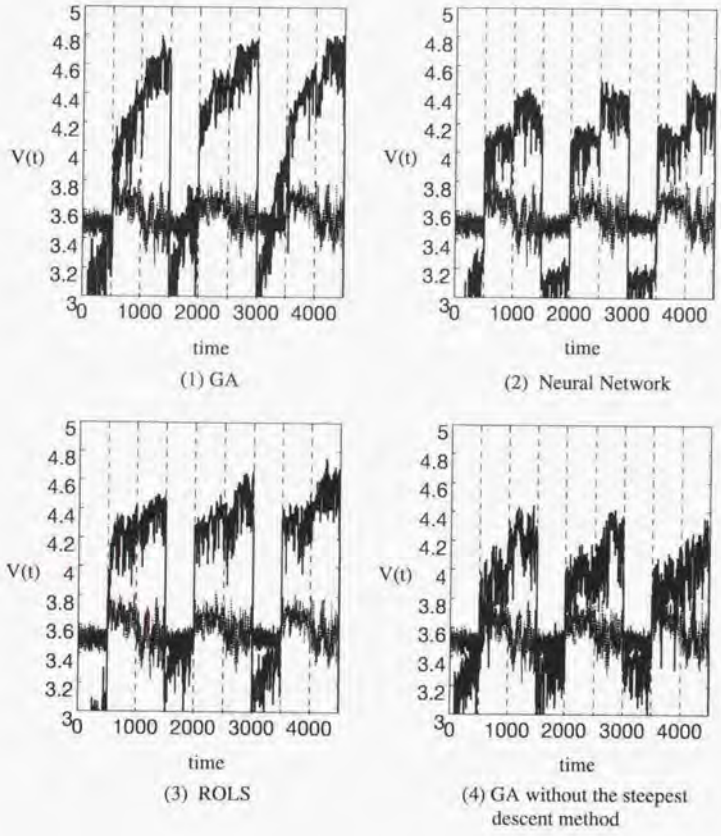


図 5.3: 各学習手法で得られた価値関数

に多くの時間を必要とするため、初めて遭遇した状況をより強く学習するような新奇性のパラメータ導入を考える必要がある。

第 6 章

GRD チップを用いた ATM ネットワークの CAC 制御

6.1 緒言

本章では、次世代の通信モードとして期待されている ATM (Asynchronous Transfer Mode: 非同期転送モード) ネットワーク [57] [58][59][60] のトラフィック制御の一つである CAC (Connection Admission Control) に、前章で提案した GRD チップを用いた強化学習手法を適用する。シミュレーションの結果、通信環境が変化する場合においてもセルの損失率を低く保ったまま与えられた帯域を有効に使用する制御方式を学習できた。

以下では、まず ATM ネットワークおよび ATM ネットワークにおける柔軟な制御手法の必要性について説明する。そして、ATM ネットワークにおける CAC を GRD チップで行う枠組について説明し、シミュレーション結果について述べる。

6.2 ATM ネットワーク

6.2.1 ATM ネットワーク

ATM とは、音声・画像・データなどを統合した柔軟な通信サービスを実現するために考案された通信方式である。具体的には、セルと呼ばれる固定長の短パケットの非同期高速転送を行う方式である。

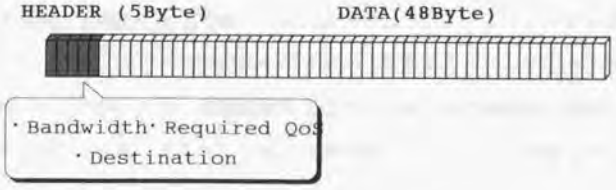


図 6.1: ATM におけるセル

ATM のセルはパケット交換方式におけるパケットと異なり、図 6.1に示すようにその長さはヘッダ情報 5 バイト、データ 48 バイトの合計 53 バイトで固定となっている。ATM では回線上のセルの送出タイミングは任意 (非同期) で、高速な通信では一定時間に多数

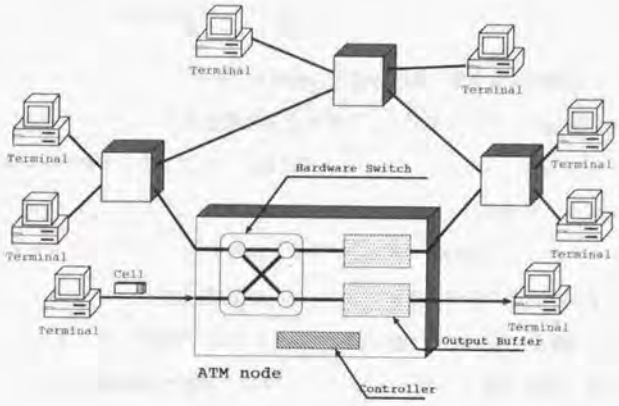


図 6.2: ATM 網の構成例

のセルを、低速な通信では少数のセルを送ることで、任意の通信速度を実現することができる。これがATM の名前の由来でもあり最大の特徴ともなっている。

ATM においては回線交換方式と同様に、通信要求の発生時にあらかじめ通信する経路を決める。また、回線交換方式で用いるセルのフロー制御やエラー時の再送制御等の多くの制御を省略したため、各 ATM ノードにおいてはハードウェアによる高速な処理が可能である。さらに、送信する情報量が時々刻々と変動するメディアの通信チャンネルを複数重ね合わせて送るため、回線交換方式よりも柔軟でかつ効率的な通信が可能である。すなわち ATM 交換方式とは、いわば回線交換方式とパケット交換方式の中間的方式である。

実際の ATM ノードは、図 6.2のようにハードウェアスイッチ、出力バッファ、制御部から構成されている。入力回線から交換ノードに送られたセルは, セル内部のヘッダ情報に基づいて、ハードウェアスイッチにより目標の出力回線に対応した出力バッファに送られる。

6.2.2 ATM Adaptation Layer

ATM においては、データの種類にかかわらず全てのデータをセルに変換して送るため、ユーザはデータタイプの違いを意識することはない。実際にメディアの違いを意識し、サービス特性の差異を吸収しているのは各端末の AAL(ATM Adaptation Layer) とよばれる部分である。これは OSI(Open Systems Interconnection: 開放型システム間相互接続) 7 階層モデルでいえばレイヤ 3 の機能として規定されている。

AAL はマルチメディア情報を取り扱う上で非常に重要な役割を担っている。ユーザが送るデータを ATM セルに分解・組立する基本機能の他に、例えば、従来の 64Kbps 音声通信のような固定速度通信の場合にはネットワーク内で生じる遅延変動を吸収する。また、データ通信の場合には転送誤りの検出や相手端末へ正しく情報が到達したかの確認を行い、必要に応じてデータを再送するなど、個々のサービス特性に応じた処理を行う。このように、再送制御や遅延制御などの複雑な処理を端末内の AAL の機能を用いて行ない、各ノードの処理を簡略化することにより、ATM は高速かつ柔軟な通信を実現している。

6.3 ATM ネットワークの制御

6.3.1 ATM におけるトラフィック制御

これまで述べてきたように、ATM の特徴のひとつは、通信帯域を必要に応じて自由に設定できることにある。これによりユーザの利便性やネットワークの柔軟性を増すことができる。しかし、この柔軟性ゆえにネットワークのトラフィック管理は複雑になる。例えば、接続中の通信が一斉に使用帯域を増加させると、ネットワーク内に逼迫が生じてセル損失が生じる。セル損失とは、回線容量を超える量のセルが同時に到着し、あふれたセルが廃棄される現象である。ATM ネットワーク内で要求される許容セル損失率は、メディアによって異なり、$10^{-3} - 10^{-10}$ とされる。

セル損失を防ぎ効率良くネットワークを使用するためにはノード内に逼迫を吸収する

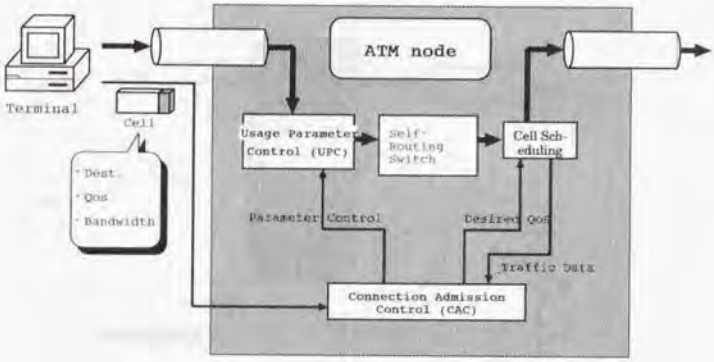


図 6.3: ATM 交換機に必要な制御

バッファをおけばよいが、その分ネットワーク内の転送遅延時間が大きくなるという問題が生じる。転送遅延とは、セルがノード内のバッファに一時的に蓄えられることにより、目的の相手に届くまでに遅れが生じる現象である。メディアによって異なるが、動画など連続的なマルチメディア情報を送る際には転送遅延時間が厳しく制限される。

そのため各種サービスに対応した通信品質を確保しつつ、効率的にネットワークを運営するには適切なトラフィック制御を行うことが必要である。ATM 交換機に必要な制御を示したものが図 6.3である。ユーザの申告した使用帯域、要求品質を考慮して呼の受付を制御する CAC(Connection Admission Control)、通信中にユーザ端末から送られてくるセル・トラフィックをモニタし、発呼時に申告した帯域を超過していないかどうかを監視する UPC(Usage Parameter Control)[61]、ネットワークが逼迫した時に重要でないセルを選択的に廃棄して全体の通信品質を確保する優先制御 [62, 63]、などがある。これまでに、ニューラルネットワークをこれらの制御に応用する研究がいくつか提案されている [64] が、ハードウェア化が困難であったり扱う対象が単純であるなど十分な成果は得られていない。本研究では制御の対象として CAC を扱う。次節で CAC の詳細を述べる。

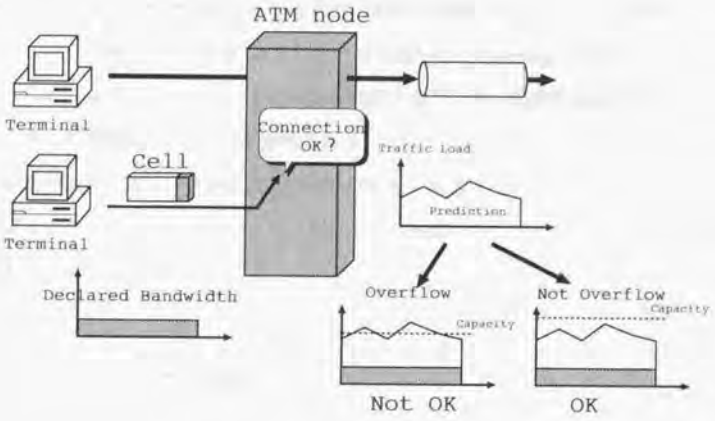


図 6.4: Connection Admission Control

6.3.2 Connection Admission Control

ATM では発呼要求時に、宛先や使用帯域、要求品質などの接続情報を含んだ信号をセルの形で交換ノードに送る。交換ノードでは、現在使用中のトラフィックに申告されたトラフィックを加算して、この呼を受け付けた場合に要求品質が満足できるかを推定し、満足できると判断した場合に受付を行う。この制御を CAC と呼ぶ (図 6.4)。ATM ネットワークでは交換ノード間でのフロー制御を行なわないため、ネットワーク内のセル転送量を操作して通信品質を制御しなければならない。そこで現在、ATM 網に加わる多種多様な呼源全てに対して, 要求される通信品質を呼ごとに満足させることが可能な CAC 制御方式の開発が重要な課題となっている。

従来の方式では、制御目標ごとにネットワークに加わる呼源の特性や負荷の変化の状況の網羅的な解析やシミュレーションを行ない、観測されるデータから抽出すべき特徴をパラメータ化し、そのパラメータから操作量を導く関数の決定を行なっていた。しかし、ATM 網に加わるトラフィック特性は変化が激しい上に、新しいサービスの追加等も容易に実現できなければならない。この手法では、扱うべき特徴パラメータの数や操作すべき値の種類が非常に多くなってしまい、トラフィック特性の変化や、制御目標の

変更などに柔軟に対応できる通信品質制御の実現は困難である。そこで本研究では、このようなATM網のCAC制御にGRDチップを用いたQ-learningを適用することを提案する。これにより、ネットワークにかかる負荷と通信品質の関係を動的に学習し、常に環境に応じた制御を行なうことが期待される。

本研究の提案手法が、ATMの特性の時間変動に適応的に追従することができることを確認するために、シミュレーションを行った。次節では、シミュレーションの枠組および結果について述べる。

6.4 シミュレーション

6.4.1 ATMネットワークのトラフィックシミュレータ

シミュレーションにあたっては、以下の仮定を導入する[65]。呼の到着間隔および保留時間は指数分布に従う。また、図6.5のように1つの呼からセルはバースト時間中に送出され、アイドル時間中はセルを送出しない。この送出タイミングすなわち通信速度は一定とする。また、時刻tの呼数をあらわす確率変数を$N(t)$とする。CACがない場合は$N(t)$は図6.6の出生死滅過程となる。実際には呼数が増える場合にCACを行い、呼が受け付けられた場合のみ、ひとつ右へ状態遷移が可能となる。

呼源モデルのパラメータを表6.1に示す。このパラメータを適切に変えることによって、ネットワーク内を通過する様々な性質のデータをシミュレートすることができる。なお、ATMネットワークの回線容量Cは160.0Mbpsとする。

6.4.2 GRDチップを用いたCACの枠組

GRDチップを用いたQ-learningをCACに適用する目的は、セル損失を許容セル損失率内に押さえつつ、受付時の報酬の和を最大にする制御方式を探索することである。ここで受付時の報酬とは、ユーザの回線使用料に対応するものである。

制御は図5.1に示した枠組において、有用度バッファを呼の受付／拒否用に2つ用意す

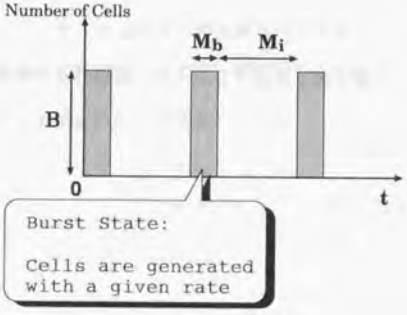


図 6.5: セルの送出タイミング

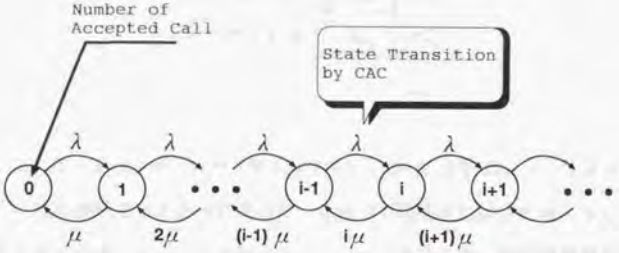


図 6.6: 接続数 $N(t)$ の状態遷移図

表 6.1: 呼源モデルのパラメータ

呼の到着間隔	$\lambda^{-1}[sec]$
呼の保留時間	$\mu^{-1}[sec]$
バースト時間	$M_b[sec]$
アイドル時間	$M_i[sec]$
セルバースト量	$B[Mbps]$

表 6.2: HNN への入出力パラメタ

入力	セル到着量観測値ヒストリの平均値と最大値 過去 n[msec] のセル到着量の平均値 過去 n[msec] のセル到着量の最大値
出力	受付／拒否それぞれの制御行動に対する状況の有用度

表 6.3: 与える強化信号

	受付	拒否
あふれが生じた	p_{ao}	p_{ro}
あふれが生じない	p_{au}	p_{ru}

$(p_{au} > 0 > p_{ru} > p_{ro} > p_{ao})$

る。それに応じて、GRD チップで学習する HNN も受付／拒否用に 2 つ用意する。各時刻において、呼の接続要求があった場合に、集団中で関数近似誤差の最も少ない個体を受付／拒否それぞれ選び出し、その時点でのノードにおけるセル到着量観測値のヒストリ $x(t), x(t-1), \cdots x(t-n)$ の平均値と最大値を入力する。これによって得られた受付／拒否の有用度を比較して、高い値を出力した制御を実際の制御とする (表 6.2参照)。以下の実験では $n = 100$[msec] で固定とする。

学習のための強化信号 p には、行った制御 (受付 / 拒否) と、あふれが生じたかどうかの 4 通りの組合せで、表6.3のような単位セルあたりの値があらかじめ設定してある。ある通信クラスの呼発生時にその呼を受け付け、次の呼発生までの間にセル損失が生じなければ、その制御は成功として

$$r = \frac{1}{T - T'} \int_{T'}^{T} x(t)dt \times p_{au} \tag{6.1}$$

の強化信号を与える。ここで $x(t)$ は各時刻における通信回線使用量であり、T、T' は

それぞれある呼が発生した時刻、その一つ前の呼が発生した時刻である。また、呼の受付を拒否し、次の呼発生までの間にセル損失が生じなかった場合には、通信回線の空き容量に比例した負の強化信号

$$r = \frac{1}{T - T'} \int_{T'}^{T} (1 - x(t))dt \times p_{ru} \tag{6.2}$$

を与える。

一方、セル損失が生じていればその制御は失敗として、

$$r = \frac{1}{T - T'} \int_{T'}^{T} o(t)dt \times \begin{cases} p_{ao} & \text{(accept)} \\ p_{ro} & \text{(reject)} \end{cases} \tag{6.3}$$

をその制御に対する強化信号とする。ここで $o(t)$ は、各時刻のあふれ量である。これら p_{ao}、p_{ro}、p_{au}、p_{ru} の値は通信クラスによって固定であり、その値はユーザが要求する通信品質を反映して決まる。ここで、$p_{au} > 0 > p_{ru} > p_{ro} > p_{ao}$ の関係を満たすものとする。

6.4.3 実験1：セル損失率の制御

まずはじめに、強化信号を調整することで、通信回線使用率およびセル損失率を制御することができることを示す。強化信号のパラメータの組は表6.4の通りである。このパラメータは予備実験によって良い結果を出したパラメータをもとに定めた。

表 6.4: 強化信号のパラメータ

	p_{au}	p_{ru}	p_{ao}	p_{ro}
ケース1	0.02	-0.01	-2.0	-0.5
ケース2	0.002	-0.001	-2.0	-0.5
ケース3	0.002	-0.001	-20.0	-5.0

このパラメータを用いて、有用度関数を学習させる。また呼の性質については、表6.5の通りとする。結果については、最初の10秒を予備学習時間とし、それ以降20秒の平均

表 6.5: 実験 1 における呼源モデルのパラメータ

呼の到着間隔 λ^{-1}	16.7[msec]
呼の保留時間 μ^{-1}	5000[msec]
バースト時間 M_b	12.0[msec]
アイドル時間 M_i	8.0[msec]
バースト量 B	160[Kbps]

通信回線使用率と平均セル損失率を用いて評価する。同時に、現在 ATM ネットワークの CAC に主に用いられている固定帯域割り当て法との比較を行なう。固定帯域割り当て法とは、呼ごとの最大通信量の申告にしたがって通信帯域を固定的に割り当てる、いわば回線交換方式を ATM で実現した手法である。今回のシミュレーションでは、その申告値は理論上の通信量の最大値とした。

結果を表 6.6および図 6.7に示す。グラフは x 軸が通信回線使用率を表し、y 軸はセル損失率の対数軸となっている。また、図 6.8にケース 2 における時間と通信量の関係を示す。4000msec 周辺で一度大きく回線容量を超えてしまっているが、その後適切に制御していることが分かる。これは、セルがあふれてしまっている状況から効率的に学習し、GA による多点探索によって局所解を避けながら、適切な有用度関数を精度良く構成した結果である。図 6.9にケース 2 と固定帯域割り当て法における時間と通信量の関係を示す。固定帯域割り当て法はセル損失率は理論上 0 に押えられるが、回線使用効率に大きな差があることがわかる。

6.4.4 実験 2：時間とともに呼の性質が変わる場合の制御

次に、時間とともに呼の性質が変わる場合に、環境の変化に追従し適切な制御を行うことができることを示す。強化信号のパラメータの組は表 6.7の通りである。

このパラメータを用いて有用度関数を学習させる。また、呼源の性質は表 6.8の通りと

表 6.6: 実験 1 の結果

	平均回線使用率	平均セル損失率
ケース 1	98.71%	2.72×10^{-3}
ケース 2	95.59%	6.52×10^{-7}
ケース 3	91.57%	$< 1.0 \times 10^{-10}$
固定帯域	61.30%	0.0

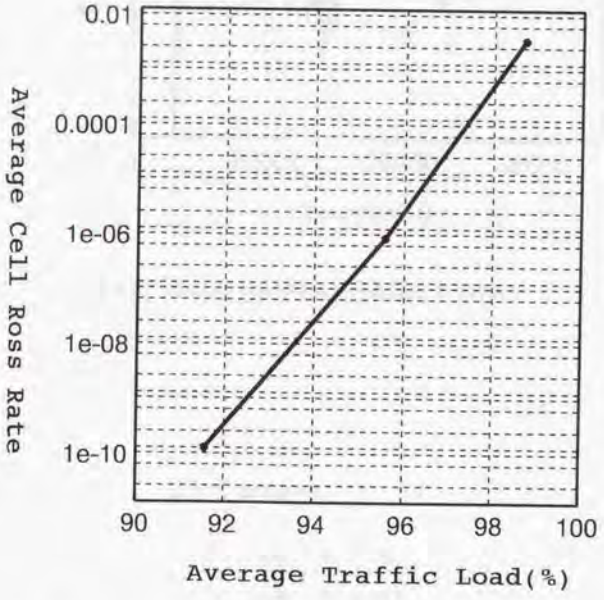


図 6.7: 実験 1 の結果 (セル損失率と回線使用率の関係)

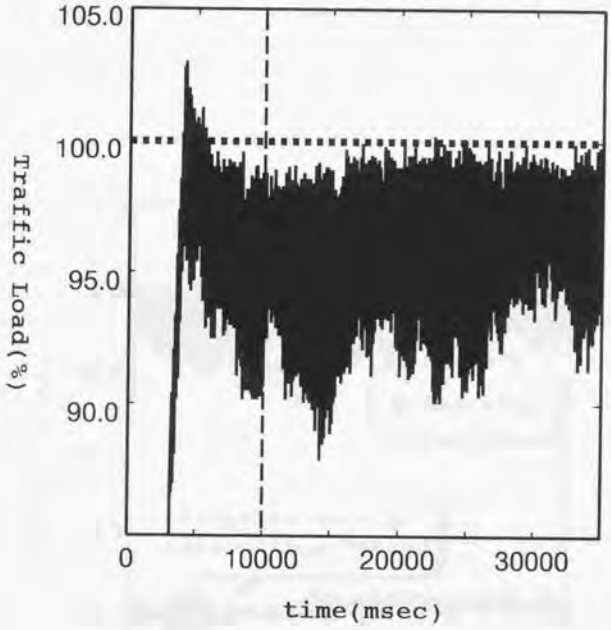


図 6.8: 実験 1 の結果 (時間と回線使用率の関係)

表 6.7: 強化信号のパラメータ

p_{au}	p_{ru}	p_{ao}	p_{ro}
0.002	-0.01	-2.0	-0.5

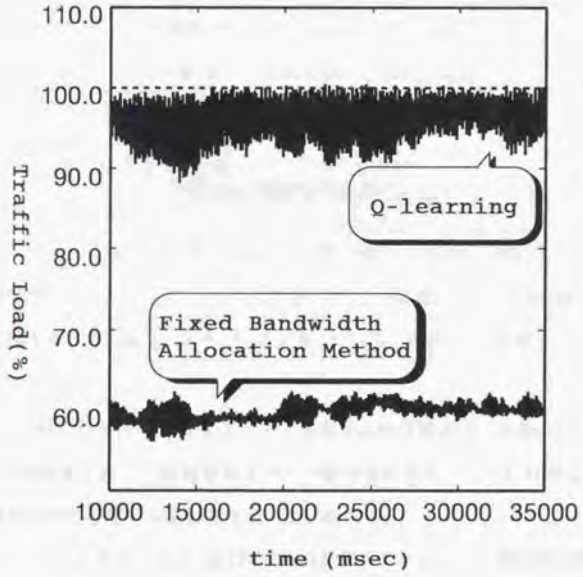


図 6.9: 実験 1 の結果 (固定帯域割り当て法との比較)

表 6.8: 実験 2 における呼源モデルのパラメータ

	クラス 1	クラス 2
呼の到着間隔 λ^{-1}	16.7[msec]	25[msec]
呼の保留時間 μ^{-1}	5000[msec]	2500[msec]
バースト時間 M_b	8.0[msec]	12.0[msec]
アイドル時間 M_i	12.0[msec]	8.0[msec]
バースト量 B	320[Kbps]	800[Kbps]

表 6.9: 実験 2 の結果

	40 ~ 60 秒	60 ~ 80 秒	80 ~ 100 秒	100 ~ 120 秒
回線使用率 (%)	93.92	92.38	95.03	92.48
セル損失率	5.62×10^{-3}	1.39×10^{-3}	4.52×10^{-4}	4.67×10^{-4}

し、20 秒ごとに呼の性質クラス 1 とクラス 2 を交互に切り替える。結果については、最初の 40 秒、すなわち 2 度目の環境変動までを予備学習時間とし、それ以降 20 秒ごとの平均通信回線使用率と平均セル損失率を用いて評価する。

結果を表 6.9に示す。また、図 6.10に時間と通信量の関係を示す。環境が変動し、呼の性質が変わった場合もそれに追従し、適切な制御を行っていることが分かる。

6.5 本章のまとめと今後の課題

本章では、ATM ネットワークの CAC に、GRD チップを用いた強化学習手法を適用することを提案した。実験 1 では、強化信号を変化させることにより、通信回線使用率およびセル損失率を制御することができることを示した。実験 2 では、呼の性質が時間とともに変わる場合においても、環境に応じて適切な制御を行なえることを示した。

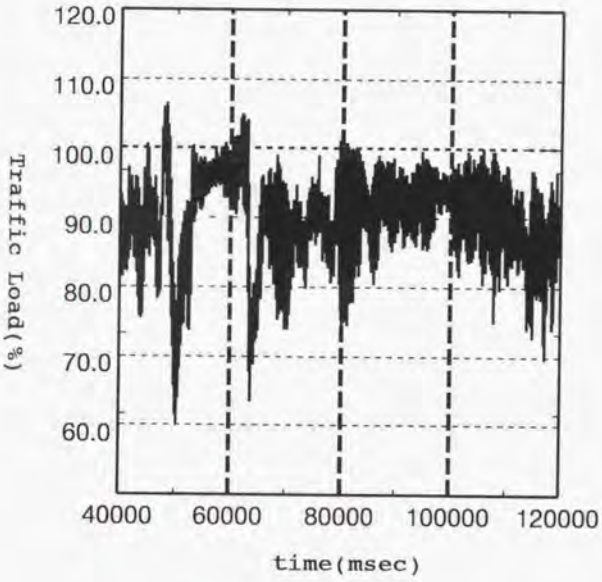


図 6.10: 実験 2 の結果 (時間と回線使用率の関係)

今後の課題として、複数通信クラスがある場合においても、適切な制御をおこなえるかどうかを示す必要がある。また、現在強化信号のパラメータ設定には理論的な目安は存在せず、設計者の職人芸的試行錯誤によって決定している段階である。このことは本手法を実用化する際には大きな問題となる。今後は強化信号パラメータを適応的に変化させることのできる上位の枠組を導入することが必要である。

第 7 章

進化型携帯電話用フィルタの開発

7.1 緒言

本章では、Off-line アナログ EHW の実現例である進化型携帯電話用フィルタについて説明する。このフィルタは、携帯電話で用いられるために、ハードウェアによる高速化、小型化が必須である。実現するフィルタは中間周波数 (Intermeditate Frequency: IF) フィルタとよばれ、携帯電話において広く用いられている。IF フィルタの仕様は非常に高い精度が要求され、中心周波数の 1% の誤差も許されない。よって、アナログ素子の製造時のわずかな誤差が性能に大きな影響を与えるために，歩留まりの悪さは深刻である。そこで提案する EHW では、遺伝的アルゴリズムを用いてこの誤差を吸収することを目的とする。実験の結果、歩留まり率が 0% から 90% に劇的に改善された。

以下では IF フィルタの構造を説明した後、提案する EHW の構造についてのべる。そして実験方法および実験結果を説明する。

7.2 IF フィルタの構造

7.2.1 IF フィルタの要求仕様

まずはじめに、IF フィルタの要求仕様および構造について説明する。IF フィルタの要求仕様は製品によって異なるが、典型的なものを図 7.1および図 7.2に示す [41]。このフィルタは、中心周波数 455kHz、バンド幅 21kHz のバンドパスフィルタである。中心周波数から 30kHz および 50kHz 離れた周波数でそれぞれ 48dB、72dB 減衰していることが要求仕様である。また、$455 - 10.5 \pm 1$kHz および $455 + 10.5 \pm 1$kHz において減衰量が -3dB になることが重要な仕様である。この要求仕様は、非常に厳しいものであり、製造時のアナログ素子のバラツキによって中心周波数がわずか 1% ずれただけでも満たすことができなくなる。

本研究が提案する手法は、この素子のバラツキを遺伝的アルゴリズムによって吸収することで要求仕様を満たすものである。

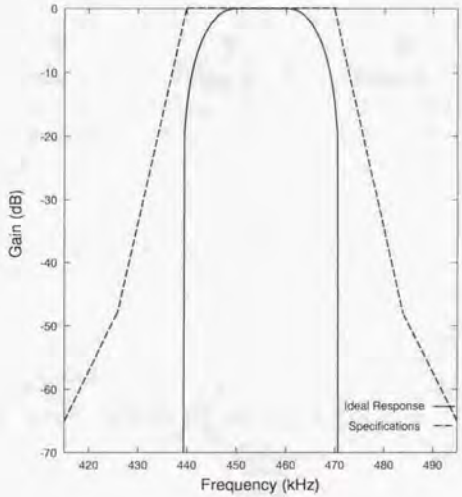


図 7.1: IF フィルタの理想的な周波数特性

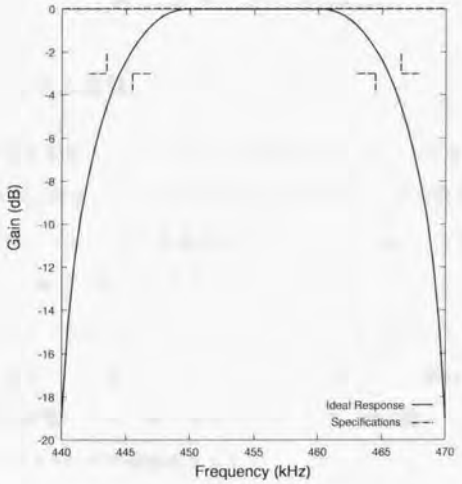


図 7.2: IF フィルタの理想的な周波数特性 (拡大図)

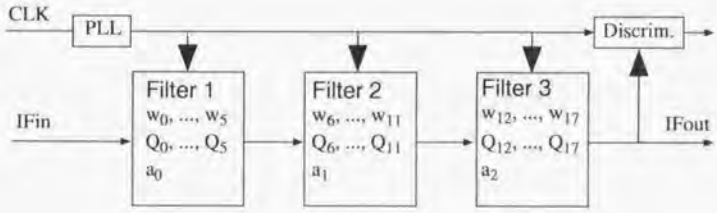


図 7.3: IF フィルタのアーキテクチャ

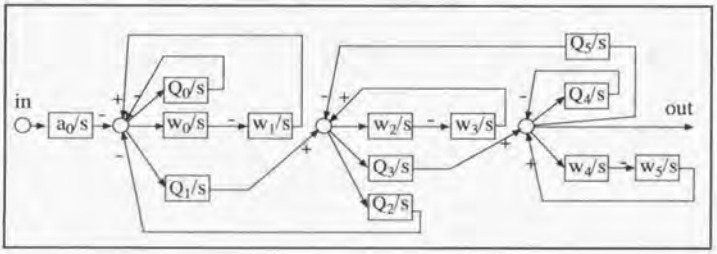


図 7.4: 6 次 G_m-C フィルタ

7.2.2 IF フィルタの実装

IF フィルタの要求仕様をみたすフィルタにはさまざまなものがあるが、本研究で採用したフィルタは 18 次の線形フィルタである [66]。そのフィルタの構造は、図 7.3に示すように 3 つの 6 次の G_m-C フィルタを直列につないだものである。その 6 次の 6 次の G_m-C フィルタのブロック線図を図 7.4に示す。

各 6 次の G_m-C フィルタは、13 個のパラメータをもっており、IF フィルタ全体では 39 個のパラメータをもっている。それらのうち $w_0, \cdots, w_{15}$ の 16 個は中心周波数に関連し、$Q_0, \cdots, Q_{15}$ の 16 個はバンド幅に関連し、$a_0, \cdots, a_2$ の 3 個はゲインに関連する。表 7.1は、それらのパラメータの設計値を示す。

この IF フィルタを LSI で実装する際に、これら 39 個のパラメータの値は G_m 素子とよばれる増幅素子の値 (トランスコンダクタンス) に対応する。G_m 素子の回路図を図 7.5に

表 7.1: 各パラメータの設計値

$w_0, \cdots, w_{17}$	$2\pi f_0$
$Q_0, \cdots, Q_{17}$	$2\pi f_0/22$
$a_0, \cdots, a_2$	$2\pi f_0/11$
	($f_0 = 455 \times 10^3$)

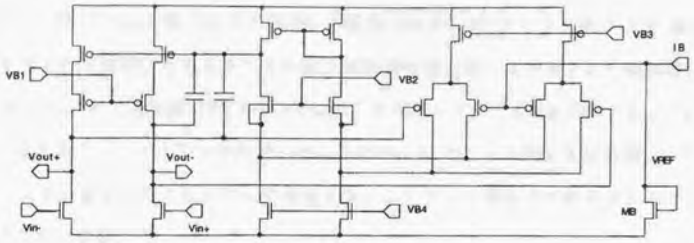


図 7.5: G_m 素子

示す。このトランスコンダクタンス値のバラツキは、半導体プロセスの精度に依存する。本研究において実装に用いるプロセスでは、トランスコンダクタンスの値は最大で 20% 設計値からずれてしまう。その 20% のうち 5-10% は温度の変動によるものである。残りの 10–15% は製造誤差によって生じる。それらバラツキのため、無調整では要求仕様をみたす LSI はひとつもない。

そこでこのバラツキを吸収するために、PLL (Phase-Locked Loop) [67] を用いた補正回路を用いる。この補正回路は、すべての G_m 素子の値が同一に変動すればその変動を吸収することができる。つまり温度による変動は、素子の値の変動が同一に近いのでこの回路によって吸収可能である。しかし、製造誤差による変動は、そのような傾向をもっていないので、この回路によっては吸収することができない。

この問題点を解決するために本研究では、Off-line アナログ EHW の考え方を用い、各 G_m 素子の値を微調整可能にし、その調整を遺伝的アルゴリズムを用いて行なうことを提案する。以下では、提案する EHW について詳しく述べる。

7.3 IF フィルタ用アナログ EHW

図 7.6に IF フィルタ用アナログ EHW の概要を示す。IF フィルタ部には 39 個の G_m 素子が含まれており、それらすべての値が微調整可能になっている。この微調整は、図 7.5におけるバイアス電流 (IB) を図 7.6に示すようにスイッチを On-Off することによっておこなわれる。このスイッチの設定は、外部から 01 のビット列を LSI 内部のコントロールレジスタに書き込むことによって決定する。このビット列をアーキテクチャビット列とよぶことにする。

遺伝的アルゴリズムによる調整では、アーキテクチャビット列を GA の染色体とみなして最適化を行なう。具体的にはフィルタに入力を与え、その出力を GA 操作を実行する外部の CPU が観察する。その観察の結果、要求仕様からどれくらい離れているかを示す GA の適応度を計算し、遺伝的操作を行なう。遺伝的操作の結果得られた染色体をふたたび LSI に書き込み、スイッチを再構成する。以上のような、観察、遺伝的操作、再

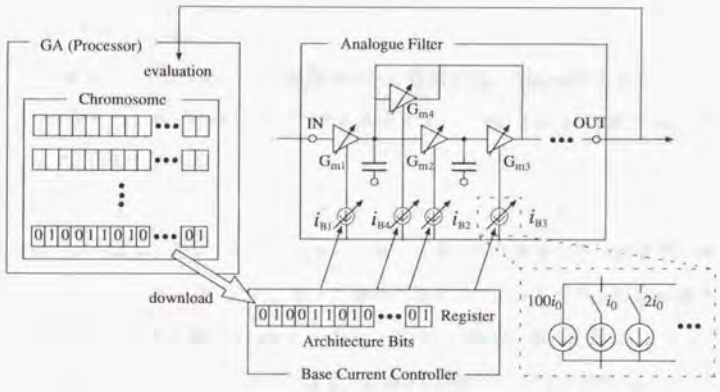


図 7.6: IF フィルタ用アナログ EHW

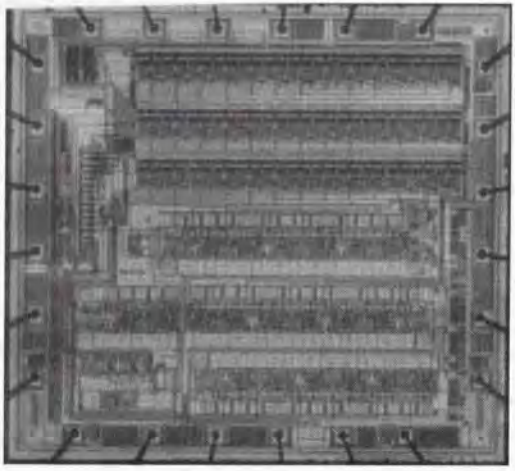


図 7.7: IF フィルタ用アナログ EHW のマスク写真

構成をくりかえすことにより、最終的に要求仕様をみたすアーキテクチャビットが発見されることが期待される。

図 7.7が開発したチップのマスク写真である。製造には、0.8μm のダブルポリダブルメタルプロセスの CMOS を用いた。マスクのサイズは、縦が 3.5mm 横が 4.5mm である。このチップは、現在旭化成マイクロシステム (株) において製品化されている IF フィルタ用 LSI の設計を基本として、それに調整部分を追加したものである。この LSI に用いたプロセスは、製品で用いられているものと同一であり、無調整でもほぼ要求仕様をみたす。そこで、このチップには G_m 素子の値が人為的にバラックような回路が挿入されており、そのバラツキの幅は 0.5μm のプロセスを用いた場合に相当する。

以下では染色体へのコーディング方法および遺伝的操作について説明する。

7.3.1 コーディング方法および遺伝的操作

GA の染色体は、39 個の遺伝子から構成される。ひとつの遺伝子は、ひとつのフィルタのパラメタの値に対応している $(a_0, \cdots, a_2, w_0, \cdots, w_{15}, Q_0 \cdots, Q_{15})$。それぞれの遺伝子は、$N$ ビットで表現され、トランスコンダクタンスの値を決定する。たとえば $N = 2$ 場合、トランスコンダクタンスの値は 4 通りの選択の幅がある。遺伝子 00、01、10、11 は トランスコンダクタンスの値がそれぞれ $1.0 - 2 \times D$,$1.0 - D$,$1.0 + D$,$1.0 + 2 \times D$ 倍になることを意味する。ここで D はあらかじめ与えた定数である。

この染色体を複数個用意し、GA の集団を形成する。GA の淘汰においては、以下で定義される適応度を用いる。

$$\text{fitness} = \sum_{i=1}^{n} w_i |S(f_i) - O(f_i)| \tag{7.1}$$

この適応度は、周波数 f_i における理想的なゲイン $S(f_i)$ と、EHW で計測されたゲイン $O(f_i)$ との差の加重和になっている。

淘汰手法は、サイズが 2 のトーナメント戦略を用いる。また、エリート保存戦略も併用する。交叉には一点交叉を用い、その確率は P_c とする。また、突然変異は一般的なビット反転方法とし、その確率は 1 ビットあたり P_m とする。

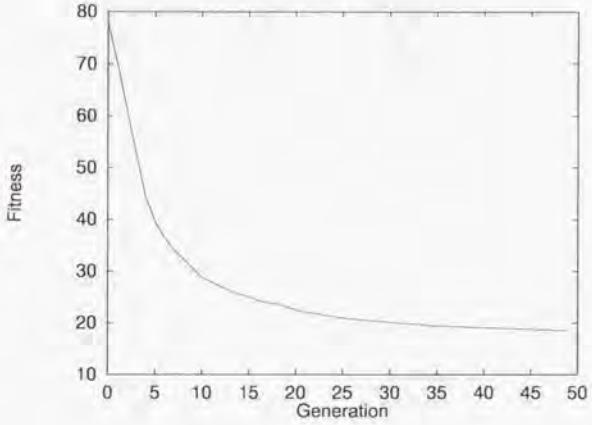


図 7.8: 世代と適応度の推移

7.4 シミュレーション結果

実 LSI を用いて性能を評価する前に、シミュレーションによる性能評価を行ったので、その結果について説明する。

シミュレーションにおいては、各 G_m 素子の値が設計値から $\sigma = 5\%$ のガウス分布に従い独立に変動すると仮定した。パラメータ N, D はそれぞれ $2, 0.025$ とした。つまり、トランスコンダクタンス値は、実装された値より -5%、 -2.5%、 +2.5% 、 +5.0% の調整が可能である。

適応度は、式 (7.1) において $n = 7$ とし、 440.0, 444.5, 449.75, 455.0, 460.25, 465.5 and 470.0 (kHz) において計測される EHW のゲインを用いた。-3dB 点に対する重み係数を 5.0 とし ($w_2 = 5.0, w_5 = 5.0$)、その他の重み係数は 1.0 とした。GA の個体数は 50、染色体長は 78 である。交叉率 P_c は 0.5、突然変異率 P_m は 0.013 とした。GA の探索は 40 世代でうちきった。

図 7.8はある仮想チップに対しての調整結果である。横軸に世代数、縦軸に集団中の最

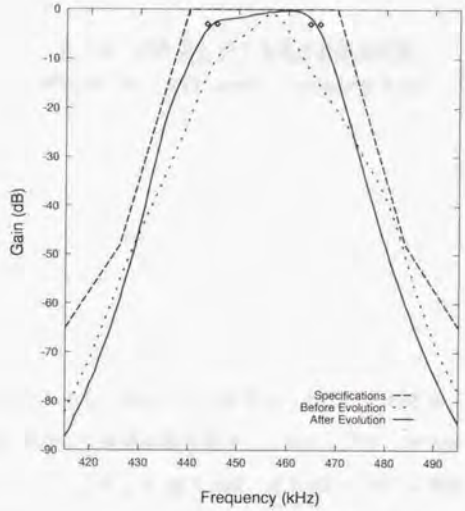


図 7.9: EHW の周波数特性

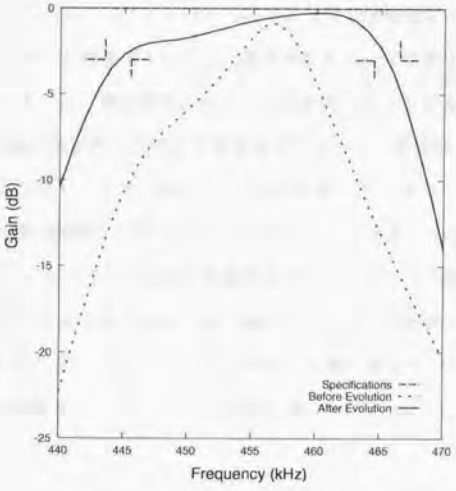


図 7.10: EHW の周波数特性 (拡大図)

表 7.2: -3dB 点に対する重み係数の影響

Weights for -3dB points	Successful Runs
1.0	26
3.0	79
5.0	95
7.0	86
9.0	4

良個体の適応度がプロットしてある。なお結果は、独立した 100 試行の平均値である。図をみると、世代が進むにつれ適応度が減少しており、徐々に理想的なゲインに近づいていることがわかる。図 7.9 および 図 7.10は、ある試行における調整前と調整後の周波数応答である。調整なしでは満たすことのできなかた要求仕様を、GA による調整後はみたしていることがわかる。

シミュレーションの結果、100 チップ中、95 チップで要求仕様をみたすことができた。比較のために、GA のよる調整のかわりに山登り法をもちいて調整するシミュレーションを行った。山登り法では、染色体中の 1 ビットを変化させることを状態遷移として用い、適応度関数を 2000 回評価した時点で探索を打ち切った。その結果、75% のチップにおいて要求仕様をみたすことができた。GA の結果が 95% であったことから、GA の多点探索による局所解の回避が効率的におこなわれていることがわかる。

表 7.4に、-3dB 点に対する重み係数の影響を示す。表より 5 つの設定値中 5.0 の値がもっとも成績がよく、探索は重み係数の値に敏感であることがわかる。値が 5.0 より小さい場合は、-3dB 点の要求仕様をみたせずに調整が失敗に終っている場合が多かった。逆に 5.0 より大きい場合は、-3dB 点以外の要求仕様をみたせないことが多かった。

7.5 実験結果

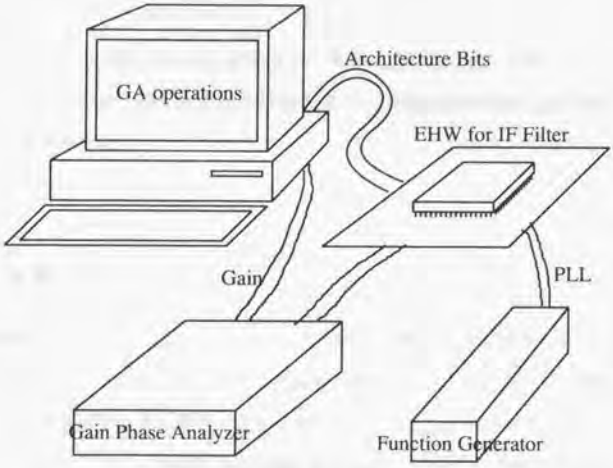


図 7.11: 実験構成図

前節でのシミュレーション結果をうけて、実チップを用いて歩留まりがどの程度になるのか調整実験を行った。

実験の全体構成を図7.11に示す。実験には、ホストコンピュータ (DOS/V 機)、EHW チップを含んだ評価用ボード、ゲインフェイズアナライザ (HP4194A)、正弦波発生器 (HP3324A-1) を用いた。ホストコンピュータは、GA の実行、I/O ポートを通じて EHW チップの制御、GP-IB ボードを通じてゲインフェイズアナライザの制御を行う。ゲインフェイズアナライザは、EHW チップの周波数応答を計測する。正弦波発生器は、PLL に与える正弦波を発生させる。

これら実験装置を用いて、周波数応答計測を含んだ一回の適応度関数の計算におよそ 1 秒ほどかかる。よって、GA の 1 試行 (2000 回の評価) にはおよそ 30 分程度かかることになる。調整における諸パラメータは、$N = 0.0247$ であること以外は前節のシミュレーションと同一の値を用いた。

20 チップにおいて調整実験を行った結果、無調整ではひとつも要求仕様を満たすもの

はなかった。GA による調整の結果、18 チップで要求仕様をみたすことができた (歩留まり率 90%)。また山登り法を用いた場合は、要求仕様を満たしたものは 13 チップであった。シミュレーションで得た歩留まり率 95% より、実験結果の 90% は若干低い値となっているが、その原因として、サンプル数が少ないこともしくは設計予想値よりもバラツキが大きいということが考えられる。

7.5.1 考察

この実験結果によって、0.5μm のプロセスを用いた場合でも歩留まりを 90% 程度にできることがわかった。今回作成したチップは 0.8μm のプロセスを用いて、疑似的に 0.5μm のプロセスによるバラツキを実現しているので、もし 0.5μm のプロセスをもちいてチップをつくった場合、マスク面積を $(0.5/0.8)^2 \approx 0.39$ 倍にすることが可能である。

ただし図 7.7のマスク写真において、およそ 1/3 の面積はフィルタ部分、のこり 1/3 は調整のための回路、最後の 1/3 はその他の回路となっている。よって、もし調整のための回路をもちいなければ面積は 2/3 程度ですむ。

以上の結果から、0.5μm のプロセスを用いて提案手法で調整した場合、0.8μm のプロセスをもちいて調整回路を用いない場合に比べて、マスク面積を $0.39/0.67 = 0.58$ 倍にすることができる。

アナログ LSI の生産コストは、1. マスク面積による項 、2. パッケージ代、3. LSI のテスト の大きく三つの項からなる。提案手法によって、これらのうち第 1 の項をおおきく削減することができる。しかし、LSI のテスト時間が調整によって増加してしまうために、第 3 の項がおおきくなってしまう。よって、提案手法の商用化のためには、調整時間をなるべく減らすことが重要となる。

7.6 本章のまとめと今後の課題

本章では、IF フィルタ用 EHW チップについて説明した。このチップは、フィルタ内の G_m 素子のバラツキを遺伝的アルゴリズムを用いて吸収することができる。実験の結

果、無調整ではひとつも要求仕様をみたさなかったが、調整後には90%のチップでみたすことができた。また同手法をもちいて、チップ面積を40%程度削減可能であることがわかった。このことは、チップの生産効率を著しく向上し、産業的にも画期的なチップである。

今後は、実チップを用いた実験のサンプル数を増やす予定である。また、重み係数の値をいろいろとかえて実験し、最適な値を発見する必要がある。

また商用化にむけては、1チップの調整時間が1秒以内という要請がある。現在の実験装置では、GP-IBの制御やゲインフェイズアナライザの計測に時間がかかるため、1回の適応度計算におよそ1秒かかり、1チップの調整に30分程度かかっている。しかし、LSIの生産工場における汎用LSIテスタをもちいた場合、1回の適応度計算におよそ3msecですむという見積もりがあるので、およそ300-400回程度に評価回数をへらすことができれば調整時間の要請をみたすことができる。シミュレーションでは、2000回の評価回数を必要としたが、GAのチューンアップ、仕様をみたせば探索を途中でも打ち切る、などの工夫により評価回数を400回程度に減らす目処がついている。

第 8 章

結論

本研究の全体を通したまとめと今後の課題および展望を述べる。個々の問題に対するまとめ等は、各章の最後に記述してある。

8.1 結論

本研究は、進化型ハードウェアと称し、環境に応じて内部構造が適応的に変化するLSIを実現した。進化型ハードウェアの基本的な考え方は、ビット列の書き変えによってハードウェア構成を変えることができる電子回路をもとに、そのハードウェア構成を遺伝的アルゴリズムによって適応的に変化させるものである。本研究では、EHWの新たな手法として「On-lineデジタルEHW」及び「Off-lineアナログEHW」の二つを提案し、その有用性を専用LSIで設計/実装することによって検証した。

前者はLSIに入力されるデータの質の変化に適応することを目的とするもので、その実現例として、GRDチップと呼ぶ専用LSIを開発した。GRDチップは、非線形関数の学習を目的とした、自律再構成機能を持つニューロチップである。動的に変動する環境下において、ニューラルネットワークの最適な性能をひきだすために、GAを用いてチップが自律的に再構成される。同チップを用いた、カオス時系列予測、デジタル通信で用いられる適応等化器、時変環境における強化学習手法の提案もあわせて行い、動的に変動する環境を仮想的に生成し、その下での有効性を実験で示した。

後者はLSI内部の故障や素子のバラツキに適応することを目的とし、その実現例としてIFフィルタ用LSIを開発した。同LSIは、製造時のLSI内部のG_m素子の性能バラツキを遺伝的アルゴリズムを用いて吸収することが可能である。調整実験の結果、歩留まりが0%から90%へ劇的に改善された。またその結果、チップサイズを4割程度削減できる見込みを得た。提案した手法は、IFフィルタ以外の様々なアナログLSIに適用可能な汎用的な手法であり、産業的にも波及効果が大きい。

8.2 今後の予定および課題

今後の課題であるが、まずGRDチップに関しては、開発したチップおよび提案した学習手法の有効性を実システムを完成することによって検証する必要がある。適応等化器については、CATVの回線を利用したケーブルモデム [68] に同チップを適用する予定である。また、ATMネットワークのCACについては、より現実に近い回線シミュレータを用いて、提案手法の有効性を調べる予定である。また、同チップを多自由度ロボットハンド [69] の埋め込みコントローラとして使用可能であるかを現在検討中である [70]。GRDチップ以外のOn-lineデジタルEHWに関しては、電総研で画像データ圧縮用のチップを開発中であり [71]、電子印刷機への応用を検討している。

次に、IFフィルタに関しては、商品化における問題点をクリアして、実際に携帯電話に組み込まれることを目標としている。Off-lineアナログEHWに関しては、同手法の超高精度 (20bit以上)D/Aコンバータ、CPU内の超高速 (数百MHz–1GHz) クロック分配回路への適用を検討中である。また今後は、「On-lineアナログEHW」の開発を行ないたい。特にアナログLSIにおける温度変化や経年変化への対応においてEHWの手法が有効であると考えている。その場合には、GA操作部分をLSI内部に組み込むことが必要で、専用ハードウェアとしてコンパクトな実装が必須である [72]。

現在考えうる進化型ハードウェアの問題点として、

1. 評価関数の定義の難しさ
2. 他の探索手法との比較
3. 安全性の確保

の三つが考えられる。まず第一の評価関数の問題であるが、人間が機械に求める要求は多岐にわたっているのに、現状では評価関数は単一のスカラ関数を定義している。それによって得られる解は、単一目的化のためのパラメータに大きく依存するので、そのパラメータを決めること自体が難しい問題となる。よって、パラメータ決定のために、設計者にはあらかじめ問題に対する詳細な知識が要求されることになる。この問題を解決

する手法として、多目的最適化手法 [73] の導入が考えられる。遺伝的アルゴリズムの集団による多点探索は、多目的最適化手法と相性がよいので [74, 75]、今後多目的最適化を用いた EHW が研究課題である。

次に第二の他の探索手法との比較であるが、EHW は評価関数の最適化に GA を用いている。GA には、準最適解の発見が速い、離散的最適化問題に適用可能である、多様性の維持により時変環境へ追従しやすい、などの利点があるが、GA でなければできない最適化問題というのは現状では存在しない。今後は、Simulated Annealing[76]、TABU サーチ [77]、山登り法などの他の探索手法との比較 (性能、収束までの計算時間、ハードウェア化が容易であるか) が必要である。

第三の安全性の確保であるが、GA には理論的には収束性の保証がない。そのため、とくに On-lineEHW では、EHW がどのような動作をするかの保証がない。よって、安全性を保証するためには、絶対におきてはいけないことを評価関数に組み込むことや、安全性が保証されている既存手法が存在する場合にはそれと組み合わせるなどの手法が必要である。

8.3 進化するハードウェアへむけて

最後に、本研究で提案した進化型ハードウェアは、「進化型」ではあっても「進化する」ではない。つまり本研究では、「進化型」ということばを、環境への適応の手段として遺伝的アルゴリズムを用いるという意味で用いてきた。しかし遺伝的アルゴリズムは、進化のもつ特徴をごく一部分アルゴリズム化したものであって、これを用いたからといって、ハードウェアが生物のように「進化する」ことにはならない。

これまで「進化」という言葉の意味をはっきり定義しないで用いてきた。しかし、生物の進化という現象の本質も分かっていない現在、それを正確に定義することは難しい。ただし、進化という現象を工学的に利用することを考えた場合は、進化の「環境に適応する個体をつくり出す手法」の側面を重要視すべきである。その場合、進化の特徴としては、まず第一に集団全体で適応能力の向上を目指す点があげられる。第二に、集団中

の遺伝子の多様性が維持されている場合、大きな環境の変化にも対応することができるという特徴がある。第三は、進化の過程で、遺伝子によって定まる各個体の構造は概して複雑化し、さまざまな機能ブロックを組み合わせて構成される点である。

この第三の特徴が、現在の遺伝的アルゴリズムには明示的には組み込まれていない。つまり、部分的な機能ブロックの獲得や、それら機能ブロックを統合する一段上の機構の発生などが、現在の遺伝的アルゴリズムでは扱うことができない。これは、現在の遺伝的アルゴリズムでは、遺伝子型から表現型への変換が非常に単純な規則でおこなわれることに原因がある。そのために、進化の過程で獲得した機能ブロックをまとめて利用する手段がなく、より高度な機能を獲得しようとしても低レベルの機能からつくりあげていくしかない。

そこで、今後大規模な「進化する」ハードウェアの実現には、上記の第三の特徴を組み込むことが鍵になる。そのためには、生物の発生の仕組みを研究し、それを工学的に利用できるようにモデル化することが重要である。とくに、自己増殖機械 [78] や L-system[79, 80] のようなオートマトンを利用したモデル化を、遺伝子型から表現型への変換にもちいることが解決手段のひとつとして考えられる。また、遺伝的プログラミングにおける、ADF (automatically defined function)[81] の考え方も、解決手段として有望である。

また、進化するハードウェアの実現のためには、ハードウェア側にも改善が必要である。可変構造をもつ素子として、最小機能単位としてなにをもちいればいいのか、また機能ブロックをより獲得しやすいような素子間の接続構造はどのようなものか、などをアルゴリズムと平行して研究する必要がある。さらに、大規模なハードウェアの実現には、電気的な論理素子ではなく、分子的な機能素子の利用も視野にいれる必要がある。

以上、進化するハードウェアの実現にむけての課題をのべたが、アルゴリズム的にも、ハードウェア的にも課題は多い。しかしこれは決して、SF の世界ではない。情報工学、電子工学、機械工学、生物学、人工知能など関連する研究分野は幅広い。今後より多くの研究者がこの魅力的な分野に参入して、活発な議論が行なわれることを期待する。

謝辞

本研究を行うに当たり多くの方々にお世話になったので、ここに自分の感謝の意を表し謝辞を述べさせていただきます。

卒業論文、修士論文、博士論文と6年間にわたり貴重な御助言と親切な御指導を頂いた吉澤教授に心から感謝申し上げます。先生には具体的な研究内容のアドバイスだけでなく、研究者としての物事の見方など、さまざまなことを御教授頂き感謝の意にたえません。

電子技術総合研究所の樋口ラボリーダには、討論や研究する場を与えて頂いただけでなく、論文の書き方や発表方法など、さまざまな御指導をいただき、本当にありがとうございます。

電子技術総合研究所の岩田さん、坂無さん、筑波大の梶谷君、また吉澤研OBの平岡さん、森永さんには、本研究について討論して頂き、問題点が明らかになりました。感謝申し上げます。

OA研究所の深谷さん、岩崎さん、旭化成マイクロシステムの安達さん、内藤さんには、チップ作成に関してさまざまなアドバイスや作業をして頂き、LSI作成のまったく経験のない私をたすけていただきました。非常に感謝しております。

最後に大学、大学院を通じて9年間も私の学生生活を温かく見守ってくれた両親に感謝します。本当にどうもありがとうございました。

参考文献

[1] J. H. Holland. *Adaptation in Natural and Artificial Systems.* The University of Michigan Press, 1975.

[2] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison Wesley, 1989.

[3] 北野宏明（編）. 遺伝的アルゴリズム. 産業図書, 1993.

[4] 北野宏明（編）. 遺伝的アルゴリズム 2. 産業図書, 1995.

[5] 北野宏明（編）. 遺伝的アルゴリズム 3. 産業図書, 1997.

[6] T. Higuchi, T. Niwa, T. Tanaka, H. Iba, H. Garis, and T. Furuya. Evolvable hardware with genetic learning. In *Proceedings of Simulation of Adaptive Behavior*, pp. 417–424. MIT Press, 1992.

[7] T. Higuchi, H. Iba, and B. Manderick. Evolvable hardware with genetic learning. In H. Kitano, editor, *Massively Parallel Artificial Intelligence*, pp. 398–421. MIT Press, 1994.

[8] H. Hemmi, J. Mizoguchi, and K. Shimohara. Development and evolution of hardware behaviors. In R. Brooks and P. Maes, editors, *Proceedings of the Artificial Life IV*, pp. 371–376. MIT Press, 1994.

[9] J. Koza, editor. *Genetic Programming.* MIT Press, 1994.

[10] J. R. Koza, F. H. Bennett III, D. Andre, M. A. Keane, and F. Dunlap. Automated synthesis of analog electrical circuits by means of genetic programming. *IEEE Transactions on Evolutionary Computation*, Vol. 1, No. 2, pp. 109–128, 1997.

[11] A. Thompson. An evolved circuit, intrinsic in silicon, entwined with physics. In *Proceedings of the First International Conference on Evolvable Systems*, pp. 390–405. Springer Verlag, 1996.

[12] B. Widrow. *Adaptive Signal Processing*. Prentice Hall Inc., 1985.

[13] 辻元重男. 適応信号処理. 昭晃堂, 1995.

[14] 河合一, 上ヶ平裕彦. マルチビット型 dac ic の徹底解明. MJ 無線と実験 1 月号, pp. 133–139, 1995.

[15] P.R. グレイ, R.G. メイヤー. アナログ集積回路設計技術 上, pp. 332–388. 培風館, 1991.

[16] 当麻善弘, 南谷崇, 藤原秀雄. フォールトトレラントシステムの構成と設計. 槙書店, 1991.

[17] 南谷崇. フォールトトレラントコンピュータ. オーム社, 1991.

[18] 村川正宏. 進化するハードウェアによる関数合成の研究, 1996.

[19] M. Salami, M. Murakawa, and T. Higuchi. Data compression based on evolvable hardware. In *Proceedings of the First International Conference on Evolvable Systems*, pp. 169–179. Springer Verlag, 1996.

[20] M. Salami, M. Murakawa, and T. Higuchi. Lossy image compression by evolvable hardware. In *Proceedings of the Workshop on Evolvable Systems in IJCAI-97*, pp. 53–59, 1997.

[21] D. E. Rumelhart, J. L. McClelland, and the PDP Research Group. *Parallel Distributed Processing*. MIT Press, 1986.

[22] 栗田. 情報量基準による3層ニューラルネットの隠れ層ユニット数の法. 電子情報通信学会論文誌, Vol. J73-D-II, No. 11, pp. 1872–1878, 1990.

[23] 喜多. ニューラルネットワークの汎化能力. システム / 制御 / 情報, Vol. 36, No. 10, pp. 625–633, 1992.

[24] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, Vol. AC-19, No. 6, pp. 716–723, 1974.

[25] E. Fiesler. Comparative bibliography of ontogenic neural networks. In *Proceedings of the International Conference on Artificial Neural Networks*, pp. 793–796. Springer Verlag, 1994.

[26] X. Yao. A review of evolutionary artificial neural networks. *International Journal of Intelligent Systems*, Vol. 8, No. 4, pp. 539–567, 1993.

[27] J. D. Schaffer, D. Whitley, and L. J. Eshelman. Combinations of genetic algorithms and neural networks: A survey of the state of the art. In L. D. Whitley and J. D. Schaffer, editors, *Combinations of Genetic Algorithms and Neural Networks*, pp. 1–37. IEEE Computer Socciety Press, 1992.

[28] 岩田, 雨宮. ニューラルネットワーク LSI. 電子情報通信学会, 1995.

[29] 久間, 中山. ニューロコンピュータ工学. 工業調査会, 1992.

[30] T. Poggio and F. Girosi. Networks for approximation and learning. *Proceedings of the IEEE*. Vol. 78, pp. 1481–1497, 1990.

[31] M. J. D. Powell. Radial basis functions for multivariable interpolation: A review. In M. G. Cox, editor, *Algorithms for Approximation*, pp. 143–167. Clarendon Press, 1987.

[32] J. Moody and C. J. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, No. 1, pp. 281–294, 1989.

[33] D. E. Goldberg and R. E. Smith. Nonstationary function optimization using genetic algorithms with dominance and diploidy. In *Proceedings of the Second International Conference on Genetic Algorithms*, pp. 59–68. Morgan Kaufmann, 1987.

[34] B. L. Miller and D. E. Goldberg. Genetic algorithms, tournament selection, and the effects of noise. *Complex Systems*, No. 9, pp. 193–212, 1996.

[35] L. J. Eshelman, R. A. Caruana, and J. D. Schaffer. Biases in the crossover landscape. In *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 10–19. Morgan Kaufmann, 1989.

[36] M. C. Mackey and L. Glass. Oscillation and chaos in phsiological control systems. *Science*, Vol. 197, pp. 287–289, 1977.

[37] J. D. Farmer and J. J. Sidorowich. Prediction chaotic time series. *Physical Review Letters*, Vol. 59, No. 8, pp. 845–848, 1987.

[38] L. Prechelt. Proben1 – a set of neural network benchmark problems and benchmarking rules. Technical Report 94–21, University Karlsruhe, 1994.

[39] J. R. Treichler, I. Fijalkow, and C. R. Johnson Jr. Fractionally spaced equalizers. *IEEE Signal Processing Magazine*, Vol. 13, No. 3, pp. 65–81, 1996.

[40] S. U. H. Qureshi. Adaptive equalization. *Proceedings of the IEEE*, Vol. 73, pp. 1349–1387, 1985.

[41] J. Proakis. *Digital Communications*. Prentice Hall Inc., 1988.

[42] D. N. Godard. Self-recovering equalization and carrier tracking in two dimensional data communication systems. *IEEE Transactions on Communications*, Vol. 28, pp. 1867–1875, 1980.

[43] J. R. Treichler and B. G. Agee. A new approach to multipath correction of constant modulus signals. *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. 31, No. 2, pp. 459–472, 1983.

[44] S. Benedetto and E. Biglieri. Nonlinear equalization of digital satellite channels. *IEEE Journal on Selected Areas in Communications*, Vol. 1, pp. 57–62, 1983.

[45] D. G. Forney Jr. Maximum-likelihood sequence estimation of digital sequences in the presence of intersymbol interference. *IEEE Transactions on Information Theory*, Vol. 18, No. 3, pp. 363–378, 1972.

[46] Y. Sato. A method of self-recovering equalization for multilevel amplitude modulation. *IEEE Transactions on Communications*, Vol. 23, pp. 679–682, 1975.

[47] S. Chen, G. J. Gibson, F. N. Cowan, and P. M. Grant. Reconstruction of binary signals using an adaptive radial basis function equalizer. *Signal Processing*, Vol. 22, No. 1, pp. 77–93, 1991.

[48] B. Mulgrew. Applying radial basis functions. *IEEE Signal Processing Magazine*, Vol. 13, No. 2, pp. 50–65, 1996.

[49] S. Chen, F. N. Cowan, and P. M. Grant. Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Transactions on Nerural Networks*, Vol. 2, No. 2, pp. 302–309, 1991.

[50] A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man and Cybernetics*, Vol. SMC-13, No. 5, pp. 834–846, 1983.

[51] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, Vol. 4, No. 5, pp. 237–285, 1996.

[52] C. J. C. H. Watkins. *Learning From Delayed Rewards*. PhD thesis, Cambridge University, 1989.

[53] Long-Ji Lin. Programming robots using reinfocement learning and teaching. In *Proceedings of the 1991 National Conference on Artificial Intelligence (AAAI-91)*, pp. 781–786, 1991.

[54] Long-Ji Lin. Scaling up reinforcement learning for robot control. In *Proceedings of the Tenth International Conference on Machine Learning*, pp. 182–189, 1993.

[55] J. A. Boyan and A. W. Moore. Generalization in reinforcement learning: Safely approximating the value function. In *Proceedings of Neural Information Processings Systems 7*, pp. 369–376, 1994.

[56] S. Chen, E. S. Chng, and K. Alkadhimi. Regularized orthogonal least squares algorithm for radial basis function networks. *International Journal of Control*, Vol. 64, No. 5, pp. 829–837, 1996.

[57] マルチメディア通信研究会. 標準 ATM 教科書. アスキー出版局, 1995.

[58] 石川, 三宅. 絵とき ATM ネットワークバイブル. オーム社, 1995.

[59] R. Handel, M. Huber, and S. Schroder. *ATM Networks – Concepts, Protocols, Applications*. Addison–Wesley, 1994.

[60] S. E. Minzer. Bisdn and asynchronous transfer mode. *IEEE Communications Magazine*, No. 9, pp. 17–24, 1989.

[61] ITU-T: Recommendation I.371. Traffic control and congestion control in b–isdn, 1995.

[62] W.Liu, M. Murakawa, and T. Higuchi. Atm cell scheduling by function level evolvable hardware. In *Proceedings of the First International Conference on Evolvable Systems*, pp. 180–192. Springer Verlag, 1996.

[63] W. Liu, M. Murakawa, and T. Higuchi. Evolvable hardware for on-line adaptive traffic control in atm networks. In *Proceedings of the Genetic Programming 1997*, pp. 504–509. Morgan Kaufmann, 1997.

[64] I. W. Habib. Applications of neurocomputing in traffic management of atm networks. *Proceedings of the IEEE*, Vol. 84, No. 19, pp. 1430–1440, 1996.

[65] 塩本, 山中. 瞬時使用率の測定に基づくアドミッション制御. 電子情報通信学会論文誌, Vol. J80–B–1, No. 12, pp. 950–960, 1997.

[66] T. Adachi, A. Ishikawa, K. Tomioka, S. Hara, K. Takasuka, H. Hisajima, and A. Barlow. A low noise integrated amps if filter. In *Proceedings of the IEEE 1994 Custom Integrated Circuits Conference*, pp. 159–162, 1994.

[67] 畑, 古川. PLL–IC の使い方. 秋葉出版, 1991.

[68] 宮地孝. Catv の現状と動向. 電子情報通信学会誌, Vol. 81, No. 2, pp. 183–190, 1998.

[69] 原田, 西川, 愈, 樋口, 横井, 嘉数. 軽量多自由度ロボットハンドの開発. 第 16 回日本ロボット学会学術講演会予稿集, pp. 267–268, 1998.

[70] 西川, 愈, 樋口, 横井, 嘉数. Ehw 筋電義手の構築に関する基礎研究. 第 19 回バイオメカニズム学術講演会講演予稿集, pp. 17–22, 1998.

[71] H. Sakanashi, M. Salami, M.Iwata, S. Nakaya, T. Yamauchi, T. Inuo, N. Kajihara, and T.Higuchi. Evolvable hardware chip for high precision printer image compression. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI98)*, 1998.

[72] 梶谷勇, 星野力, 村川正宏, 樋口哲也. Ga によるニューラルネットワークの構造学習用回路の実現. 日本神経回路学会誌, Vol. 5, No. 4, pp. 145–153, 1998.

[73] 西川, 三宮, 茨木. 最適化, pp. 135–173. 岩波書店, 1982.

[74] C. M. Fonseca and P. J. Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, Vol. 3, No. 1, pp. 1–16, 1995.

[75] 村川正宏, 吉澤修治. 近傍モデル遺伝的アルゴリズムによる多目的最適化. 電子情報通信学会技術研究報告 NC97–6, pp. 41–48, 1997.

[76] E. Aarts and J. Korst. *Simulated Annealing and Boltzmann Machines*. JOHN WILEY & SONS, 1989.

[77] F. Glover and M. Laguna. *TABU SEARCH*. Kluwer Academic Publishers, 1997.

[78] J. Neumann. *Theory of Self-reproducing Automata*. University of Illinois Press, 1966.

[79] A. Lindenmayer. *Journal of theoretical biology*, Vol. 18, No. 455, 1971.

[80] 土居洋文. 生物のかたちづくり. サイエンス社, 1988.

[81] J. Koza, editor. *Genetic Programming II*. MIT Press, 1994.

発表論文一覧

第2章

- M. Murakawa, S. Yoshizawa, I. Kajitani, T. Furuya, M. Iwata, and T. Higuchi. Hardware evolution at function level. In *Proceedings of the Parallel Problem Solving from Nature IV*, pp. 62–71. Springer Verlag, 1996.
- 樋口哲也, 村川正宏. 関数レベルにおけるハードウェア進化, pp. 271–298. 遺伝的アルゴリズム 3, 産業図書, 1997.
- M. Salami, M. Murakawa, and T. Higuchi. Data compression based on evolvable hardware. In *Proceedings of the First International Conference on Evolvable Systems*, pp. 169–179. Springer Verlag, 1996.
- M. Salami, M. Murakawa, and T. Higuchi. Lossy image compression by evolvable hardware. In *Proceedings of the Workshop on Evolvable Systems in IJCAI-97*, pp. 53–59, 1997.

第3章

- M. Murakawa, S. Yoshizawa, I. Kajitani, and T. Higuchi. On-line adaptation of neural networks with evolvable hardware. In *Proceedings of the Seventh International Conference on Genetic Algorithms*, pp. 792–799. Morgan Kaufmann, 1997.
- Masahiro Murakawa, Shuji Yoshizawa, Isamu Kajitani, Xin Yao, Nobuki Kajihara, Masaya Iwata, and Tetsuya Higuchi. The GRD chip: Genetic reconfiguration of DSPs for neural network processing. *IEEE Transactions on Computers*, accepted.

第4章

- M. Murakawa, S. Yoshizawa, and T. Higuchi. Adaptive equalization of digital communication channels using evolvable hardware. In *Proceedings of the First International Conference on Evolvable Systems*, pp. 379–389. Springer Verlag, 1996.

- M. Murakawa, S. Yoshizawa, I. Kajitani, and T. Higuchi. Evolvable hardware for generalized neural networks. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI–97)*, pp. 1146–1151. Morgan Kaufmann, 1997.

- M. Murakawa, K. Hiraoka, T. Higuchi, T. Furuya, and S. Yoshizawa. Adaptive blind equalization using bottleneck networks implemented by evolvable hardware. In *Proceedings of the Fifth International Conference on Neural Information Processing*, pp. 79–83, 1998.

- 村川正宏, 中根牧子, 樋口哲也, 古谷立美, 吉澤修治. RBF を用いた進化型ハードウェアによる適応等化器. 電子情報通信学会技術研究報告 NC97–95, pp. 51–58, 1998.

第 5 章

- 米井友浩, 村川正宏, 吉澤修治. 遺伝的アルゴリズムを用いた時変環境における Q-learning. 電子情報通信学会技術研究報告 NC97–64, pp. 71–78, 1997.

- 村川正宏, 米井友浩, 樋口哲也, 吉澤修治. RBF ネットワークを用いた時変環境における Q-learning – 遺伝的アルゴリズムによる有用度関数の構成法 –. 電子情報通信学会論文誌, Vol. J81–D–II, No. 12, pp. 2828–2839, 1998.

第 6 章

- なし

第 7 章

- M. Murakawa, S. Yoshizawa, Toshio Adachi, Shiro Suzuki, Kaoru Takasuka, and T. Higuchi. Analogue ehw chip for intermediate frequency filter. In *Proceedings of the Second International Conference on Evolvable Systems*, pp. 134–143. Springer Verlag, 1998.

第 8 章

- 村川正宏, 吉澤修治. 遺伝的アルゴリズムを用いた多目的最適化. 日本神経回路学会第 6 回全国大会論文集, pp. 251–252, 1995.
- 村川正宏, 吉澤修治. 近傍モデル遺伝的アルゴリズムによる多目的最適化. 電子情報通信学会技術研究報告 NC97-6, pp. 41–48, 1997.
- 村川正宏, 吉澤修治. 近傍モデル遺伝的アルゴリズムによる多目的最適化. 情報処理学会論文誌, accepted.
- 梶梶谷勇, 星野力, 村川正宏, 樋口哲也. GA によるニューラルネットワークの構造学習用回路の実現. 日本神経回路学会論文誌, Vol. 5, No. 4, pp. 145–153, 1998.

その他

- 岩田昌也, 梶谷勇, 村川正宏, 平尾友二, 伊庭斉志, 樋口哲也. 進化するハードウェアを用いたパターン認識システム. 電子情報通信学会論文誌, Vol. J81-D-II, No. 10, pp. 2411–2420, No.10, 1998.
- W.Liu, M. Murakawa, and T. Higuchi. ATM cell scheduling by function level evolvable hardware. In *Proceedings of the First International Conference on Evolvable Systems*, pp. 180–192. Springer Verlag, 1996.
- W. Liu, M. Murakawa, and T. Higuchi. Evolvable hardware for on-line adaptive traffic control in ATM networks. In *Proceedings of the Genetic Programming 1997*, pp. 504–509. Morgan Kaufmann, 1997.

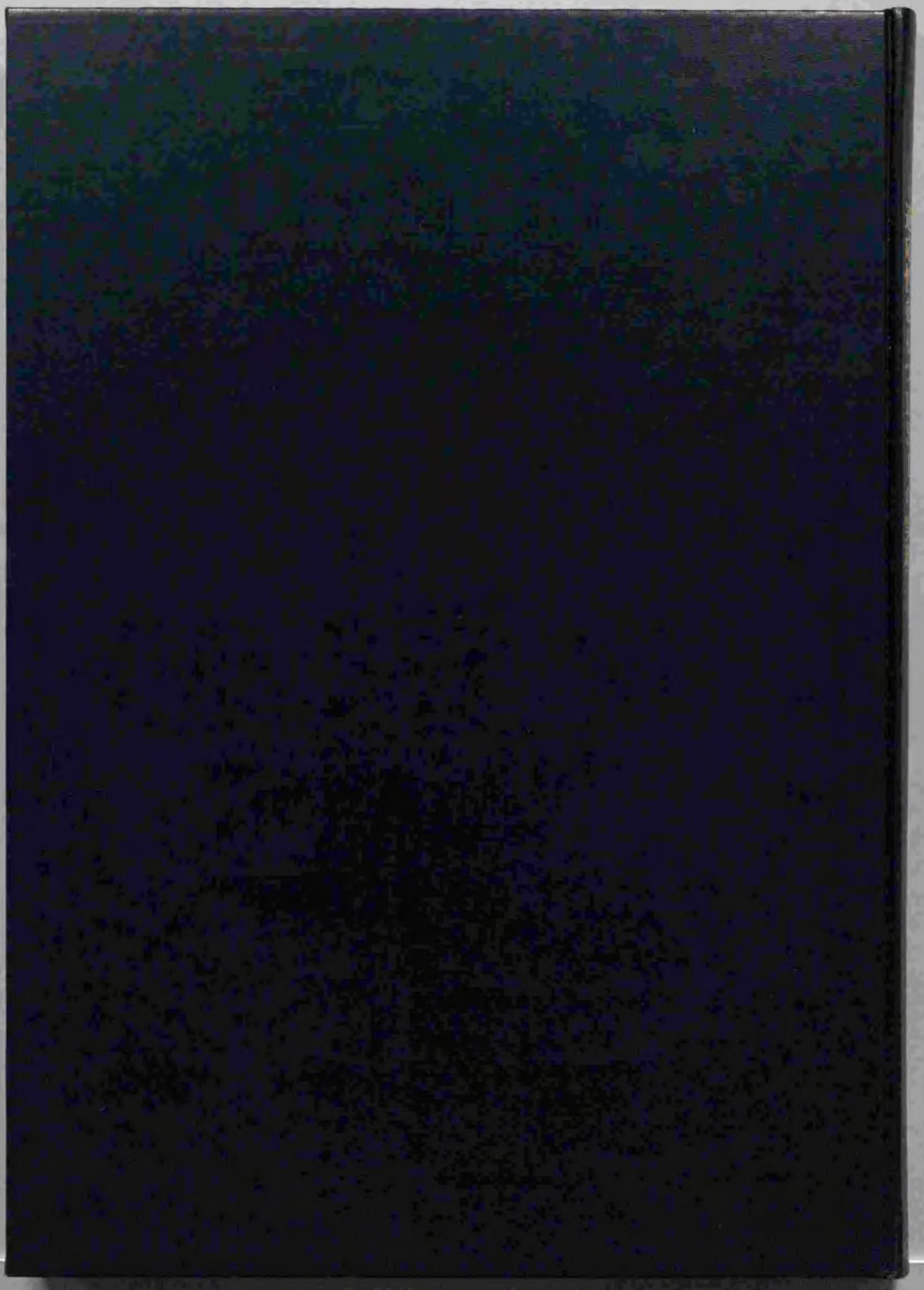
- 村川正宏, 吉澤修治. 遺伝的アルゴリズムを用いたステレオ画像からの曲面再構成. 日本神経回路学会第 5 回全国大会論文集, pp. 181–182, 1994.
- 米井友浩, 村川正宏, 吉澤修治. 時変環境に対する 2 倍体染色体を用いた遺伝的アルゴリズム. 日本神経回路学会第 8 回全国大会論文集, pp. 58–59, 1997.

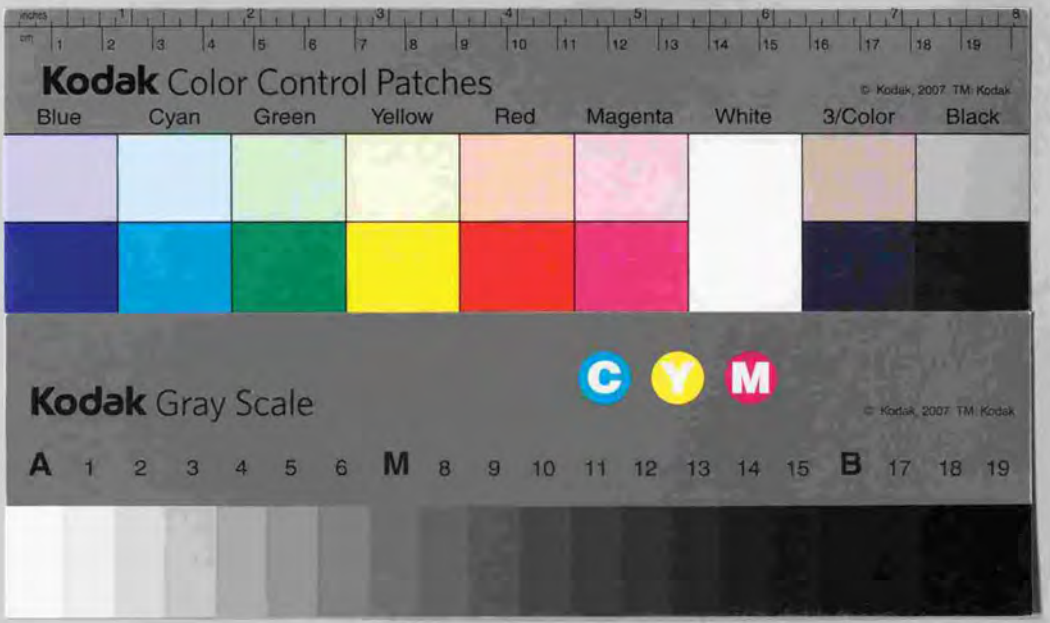
以上

学位論文

平成 10 年　12 月 18 日　提出

村川 正宏




Kodak Color Control Patches
© Kodak, 2007 TM: Kodak
Blue
Cyan
Green
Yellow
Red
Magenta
White
3/Color
Black
Kodak Gray Scale
C
Y
M
© Kodak, 2007 TM: Kodak
A 1 2 3 4 5 6 M 8 9 10 11 12 13 14 15 B 17 18 19