Vol.2016-MPS-108 No.2
Vol.2016-BIO-46 No.2
2016/7/4

# ユニットの冗長化による耐故障性を考慮した進化型ニューラルネットワーク

工藤　理人[1,a)]　菅沼　雅徳[1]　長尾　智晴[1]

**概要**：本論文では，ニューラルネットワーク（NN）に耐故障性を獲得させる耐故障 NN の構築手法を提案する．従来研究では，耐故障性の獲得のためにユニットの冗長化などが行われてきたが，ユニットの重要度を考慮した冗長化はこれまでに行われていない．NN はユニット毎に出力の影響度が異なるため，適切な制御が必要になる．本手法では，NN が故障した際の出力への影響が最も大きい中間ユニットを複製することで，出力に重要なユニットを適切に冗長化する．NN にはネットワークの構造を進化計算法によって自動獲得する進化型 NN を用いる．これによって，任意の構造での冗長化が可能になる．提案手法を振り上げ倒立振子問題に適用した結果，従来研究と比較してより故障に強い NN を構築することができた．

**キーワード**：ニューラルネットワーク，進化型ニューラルネットワーク，耐故障性，冗長化

# Evolutionary Neural Network with Improving Fault Tolerance by Redundancy

Masato Kudo[1,a)]　Masanori Suganuma[1]　Tomoharu Nagao[1]

***Abstract:*** In this paper, we propose a new approach to improve fault tolerance of neural networks. There have been many stuides to obtain fault tolerance to add redundant units. However, no studies consider importance of untis. The degree of influence on the output between units is difference, therefore proper control is required. We propose a new technique that by duplicate the hidden unit that is the largest influence on the network, thereby we properly redundant an important unit to output. We use an evolutionary neural network which automatically acquires structure of the network by evolutionary computation. This allows redundancy in any structures. We apply the proposed method to the swing-up inverted pendulum problem. The results show that the proposed method has more fault tolerance than compared methods.

***Keywords:*** neural network, neuroevolution, fault tolerance, redundancy

## 1. はじめに

ニューラルネットワーク（NN）は生物脳の構造から着想を得て，その模倣から研究が始まり，現在様々な問題解決に用いられている．また，並列計算の有効性を活かして，Field Programmable Gate Array(FPGA) 等を用いることで，NN をハードウェアで実装する研究も多く行われている．NN が実際の産業や社会で利用されるようになった場合，ユニットの故障に対して頑健であることが求められる．そのため，NN の故障について考える必要がある．

NN の故障には，ユニットの値が異常になるものや，ユニット間の断線故障などがあり，従来研究では様々な NN の耐故障性を獲得する研究が行われてきた．Emmerson らは，中間ユニットを複製して冗長化することで，それらにかかる結合荷重を半分にし，NN の機能を分散させた [1]．冗長化とは，新しくユニットを追加して結合荷重を分散させることによって，ユニット故障や断線故障によるユニッ

[1] 横浜国立大学大学院環境情報学府
Graduate School of Environment and Information Sciences, Yokohama National University, Yokohama, Kanagawa 240-8501, Japan
[a)] kudo-masato-wh@ynu.jp

ト同士の影響を少なくすることである．Dias らは，中間ユニットだけでなく，入力ユニット，バイアス，断線故障も考慮した結合荷重の分散手法を提案した [2]．Bernier らは，NN の教師あり学習の手法として広く知られる Back Propagation の際に，結合荷重値の平均二乗誤差を最小化することで，安定して耐故障性を有する NN を獲得する手法を提案した [3]．また，ノイズを与えて学習し，NN のロバスト性を高めることで耐故障性の獲得を行う研究も行われてきた [4], [5]．更に，結合荷重を小さくする圧をかける，学習中にユニットを追加して冗長化する，ノイズを加えて学習するなど様々な手法を組み合わせた研究も行われてきた [6]．しかし，これらの手法は，NN の出力への影響は少なくなっているが，故障の影響を完全に抑えられていない点が問題として挙げられる．

Zhou らは，中間ユニットが故障した場合を事前に学習することで，1 つの中間ユニットのみを対象とする故障において有効性を示した [7]．西垣らは，中間層と出力層の間の単一断線故障に対して，出力誤差が最大になる故障を考慮することで，従来より高速な耐故障学習アルゴリズムを提案した [8]．しかし，これらの手法は，複数箇所での故障を考慮できていない．

また，どの程度冗長化するかを，NN の故障率とそれによるエラー率のグラフを考慮して，効率的な冗長化を行う手法も提案されている [9]．しかし，一般的に NN はユニットごとに役割が異なり，特定のユニットが機能しなくなると出力に多大な影響が及んでしまうことが知られている [10]．

これらの従来研究に対して，本論文では複数の中間ユニットの故障を扱い，更にユニットの重要度を考慮した冗長化を行う手法を提案する．冗長化による耐故障性の獲得手法に関して，従来研究では NN 全体の耐故障性を考慮した研究はあるが，NN を構成するユニットごとの役割を考慮した研究は行われていない．ユニット故障を扱う理由としては，断線故障よりも NN への影響が大きいため，研究の意義が大きいと考えたためである．ユニット故障については，従来研究でも多く用いられてきた stuck-at-0(or 1) fault(0 または 1 に固定される故障) を扱う．本手法では，NN を学習する際に，各中間ユニットに故障を発生させ，出力に一番影響を与えた中間ユニットを複製して冗長化を行う．その際に，冗長化した中間ユニット同士をひとつのまとまりとして，ユニット群と呼ぶことにする．そして，ユニット群の出力をひとつにまとめることで，ユニット群中の各ユニットの出力を統合する．このとき，出力をまとめる機構をフィルタと呼ぶことにする．フィルタは，MAX フィルタと MIN フィルタの 2 種類を用意した．そのため，ユニット群中から 1 つでも正常なユニットがある場合ならフィルタによって故障を完全に吸収し，正常な NN として機能を果たすことができる．更に，本論文では問題に応じた任意のネットワーク構造を自動構築する進化型 NN を用いる．従来研究では階層型 NN を用いた研究がほとんどであり，耐故障性を考慮した進化型 NN の研究はない．進化型 NN は再帰結合や相互結合などの結合関係も構築可能であるため，従来の階層型 NN では解決できない問題にも対応できる．進化型 NN を用いることで，従来の階層型 NN よりも表現力が高い NN を獲得でき，様々な問題に対応することができる．

## 2. 進化型 NN

進化型 NN とは，問題に応じた任意のネットワーク構造を自動構築できる NN である．本章では代表的な進化型 NN のひとつである NeuroEvolution of Augmenting Topologies(NEAT)[11] と，筆者が所属する研究グループが提案した Real valued Flexibly Connected Neural Network(RFCN)[12] について説明する．

### 2.1 NEAT

NEAT とは，Stanley らによって提案された進化型 NN のひとつである．NEAT の大きな特徴として，(1) ユニット情報と結合情報のリストを用いた異なる構造間の交叉，(2) 種分化の概念を用いた個体の多様性の保護，(3) 中間層のない最小構造からの進化計算の開始，が挙げられる．一般的に進化型 NN は，異なる構造を持った個体同士の交叉や，進化途中の個体をどうやって保護するかが問題となっている．NEAT では，2 つのリスト構造を用いた交叉，種分化の概念による個体の保護によって，上記の問題を解決している．NEAT は XOR 回路の構築問題と，倒立振子問題へ適用し，その有用性が示されている．また，ほかの分野にも応用され，様々な改良手法が提案されている [13], [14], [15].

### 2.2 RFCN

RFCN とは，遺伝的アルゴリズム（GA）により NN の結合荷重と構造を最適化することで，問題に応じた任意のネットワーク構造を自動構築する手法である．RFCN は実数値環境下でのエージェント制御問題に適用し，高い性能が示されている．各ユニットには応答速度という特性があり，応答速度の早いユニットから順番に処理することで，全ユニットが同期的に処理するよりも複雑な状態遷移が可能になっている．

**図 1** に，RFCN の構造を示す．RFCN の各ユニットは，再帰結合と相互結合が可能である．ただし各ユニットから入力ユニットへの信号出力はできない．RFCN の染色体は 2 次元のビット配列によって構成されており，ユニット間の結合荷重，出力関数の種類等を表している．最適化に用いる遺伝操作には，ブロック交換交叉，中間ユニット数の突然変異，遺伝子の突然変異の 3 種類がある．ブロック交換交叉は，染色体の行と列それぞれをランダムに決めて染

Vol.2016-MPS-108 No.2
Vol.2016-BIO-46 No.2
2016/7/4

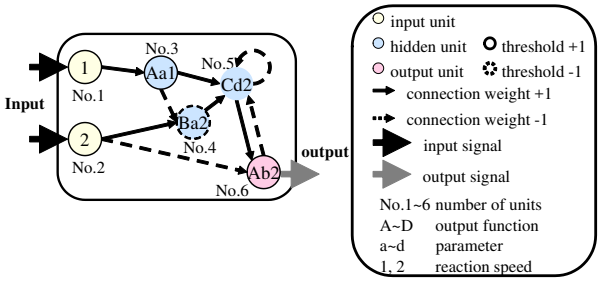


図 1　RFCN の構造例

**Fig. 1**　An example of RFCN structure.

色体を 4 分割し，そこから生成されるブロックから交叉する．中間ユニット数の突然変異は，突然変異率 $M_h$ によって中間ユニット数を 1 つ増加または減少させる．遺伝子の突然変異は，突然変異率 $M_g$ によって遺伝子単位で突然変異を発生させる．

NEAT は最小構造から最適化を進めるため，中間ユニット数が少ないネットワークを獲得すると考えられる．本論文は中間ユニットの冗長化による耐故障性を検証するため，ある程度の中間ユニット数がなければ有効性の確認ができない．そのため，本手法では RFCN を用いることでネットワーク構造を獲得する．

## 3. 耐故障性を考慮した進化型 NN の設計

本手法では，2 つのフェーズから冗長化による耐故障性を考慮した NN を獲得する．第 1 フェーズでは，問題に対して適切な入力と出力の関係を学習したネットワーク構造を獲得する．第 2 フェーズでは，中間ユニットを複製して冗長化することにより耐故障性を獲得する．その際に，進化計算を用いることで逐次的に中間ユニットを追加して冗長化を行う．これによって，通常学習によるネットワークの表現力を維持したまま，任意のネットワーク規模かつ任意のユニット数での冗長化が可能となり，より柔軟に耐故障性の獲得を行うことができる．以降，第 1 フェーズの学習を通常学習 (normal learning)，第 2 フェーズの学習を耐故障学習 (fault tolerant learning) と呼ぶ．

### 3.1 ネットワーク構造

**図 2** に中間ユニットの複製の流れを示す．最初に通常学習によって問題を解決するネットワーク構造を獲得する．その後ユニットを複製して，耐故障性を有するネットワーク構造を新たに獲得する．

**図 3** と**表 1** および**表 2** に，図 2 によって獲得されたネットワークの構造とそのパラメータを示す．複製して冗長化したユニット同士はひとつのまとまりとしてユニット群を形成している．このユニット群は，どのユニットに接続しているかという接続関係は維持したまま，結合荷重値やしきい値がそれぞれ異なっている．そして，ユニット群の出

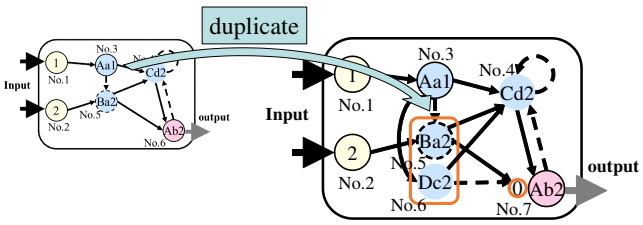


図 2　中間ユニットの複製の流れ

**Fig. 2**　Duplicating a hidden unit sequence.

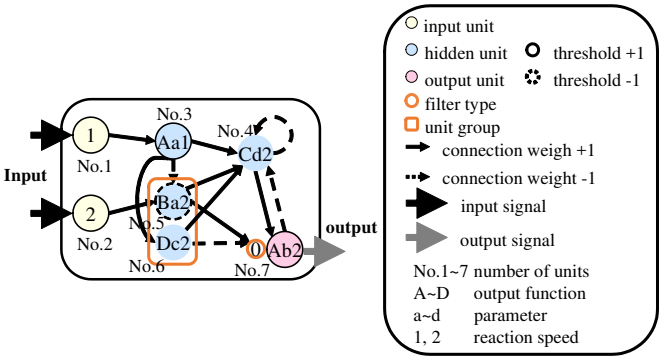


図 3　提案手法の構造例

**Fig. 3**　An example of the proposed method structure.

力をひとつにまとめるフィルタを通すことにより各ユニットの出力を統合する．

フィルタの種類としては，MAX フィルタと MIN フィルタの 2 種類を用意した．本手法では故障値が固定値であるため，MAX フィルタと MIN フィルタによってユニット群中から正常なユニットを選択することができる．これによって，ユニット群中に正常なユニットが存在すれば正常なネットワークとしての機能を果たすことができる．フィルタの種類についても進化計算によって獲得する．

### 3.2 学習アルゴリズム

提案手法の学習アルゴリズムは以下の 5 ステップからなる．

(1) GA によって制御ネットワークを獲得する（通常学習）．
(2) 各中間ユニットを故障させて，適応度が最も下がったユニットを複製し，ユニット群を生成する．
(3) すべてのユニットが正常の場合と，ユニット群の各ユニットがひとつ故障した場合を学習する．すべての学習パターンにおいて，制御成功率がしきい値 $S_{th}$ 以上になるまで最適化する（耐故障学習）．
(4) 中間ユニットが指定数 $H_{max}$ に達したら学習を終了する．
(5) ステップ（2）-（4）を繰り返す．

ステップ（2）において，適応度が最も下がったユニットを複製対象とする目的は，適応度の下がり度合いがそのネットワークにおけるユニットの重要度の尺度であると考えたためである．そのため，適応度の低下が一番大きい，つま

Vol.2016-MPS-108 No.2
Vol.2016-BIO-46 No.2
2016/7/4

表 1　ネットワーク構造パラメータ
**Table 1**　Network parameters.

| To / From | | input | | hidden | | | | output |
|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| output function | | ∗ | ∗ | A | C | B | D | A |
| parameter | | ∗ | ∗ | a | d | a | c | b |
| reaction speed | | ∗ | ∗ | 1 | 2 | 2 | 2 | 2 |
| threshold | | ∗ | ∗ | +1 | 0 | -1 | 0 | +1 |
| input | 1 | ∗ | ∗ | +1 | 0 | 0 | 0 | 0 |
| | 2 | ∗ | ∗ | 0 | 0 | +1 | 0 | 0 |
| hidden | 3 | ∗ | ∗ | 0 | +1 | -1 | +1 | 0 |
| | 4 | ∗ | ∗ | 0 | -1 | 0 | 0 | +1 |
| | 5 | ∗ | ∗ | 0 | +1 | 0 | 0 | +1 |
| | 6 | ∗ | ∗ | 0 | +1 | 0 | 0 | -1 |
| output | 7 | ∗ | ∗ | 0 | -1 | 0 | 0 | 0 |

∗ : forbidden

表 2　フィルタパラメータ
**Table 2**　Filter parameters.

| hidden | 3 | 4 | 5 | 6 |
|---|---|---|---|---|
| filter type | MAX | MIN | MIN | MAX |

り出力に最も影響を与える重要なユニットを冗長化することで，ネットワークの主要な部分を守り，故障に強いネットワークを構築する．

### 3.3　染色体コーディング

提案手法では，従来手法の RFCN の染色体配列に加えて，フィルタ用の染色体配列を用意する．耐故障学習の最適化においては，遺伝操作として交叉は行わず，突然変異のみによって最適化を行う．耐故障学習では部分最適化を行うため，ブロック交換交叉は向いていない．ブロック交換交叉では遺伝子全体を変更してしまうため，維持するべき構造を破壊してしまうおそれがある．そのため，突然変異のみを用いることで最適化を行う．

## 4.　ユニットの冗長化による耐故障性を考慮した実験

### 4.1　問題設定

本論文では，提案手法を振り上げ倒立振子問題に適用する．**図 4** に振り上げ倒立振子問題のモデルを示す．振り上げ倒立振子問題とは，台車に接続している棒を倒立させ安定させる問題である．初期状態では棒は真下（$\theta = \pi$）を向いており，振り上げて倒立させることで真上に棒を維持することを目的とする．この問題を選択した理由として，(1) ネットワークの学習が容易である，(2) 故障の発生により制御不能判断がしやすい，という点が挙げられる．本実験では，ハードウェア上で実装する前段階として，図 4 の倒立振子モデルを**図 5** のシミュレートプログラムで再現し，実験を行った．扱う故障については，ユニットの故障値が

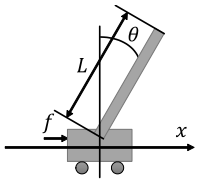


図 4　振り上げ倒立振子モデル
**Fig. 4**　A swing-up inverted pendulum model.

図 5　シミュレートプログラム
**Fig. 5**　The simulation program.

表 3　シミュレートプログラムの環境設定
**Table 3**　Environment settings of the simulation program.

| parameter | value |
|---|---|
| environment width(m) | 3.0 |
| gravity(m/s$^2$) | 9.8 |
| cart mass(kg) | 1.0 |
| cart width(m) | 0.25 |
| pole mass(kg) | 0.2 |
| pole length(m) | 0.5 |
| friction | 0.05 |
| max force(N) | 40 |
| max speed(m/s) | 16 |
| max angular velocity(m/s$^2$) | 10 |

表 4　GA のパラメータ
**Table 4**　GA parameters.

| parameter | normal learning | fault tolerant learning |
|---|---|---|
| num. of generation | 10000 | 15000 |
| max step | 1000 | 1000 |
| population size | 150 | 150 |
| children size | 30 | 30 |
| generation alternation model | MGG* | MGG |
| crossover rate | 0.9 | – |
| mutation rate $M_h$ | 0.0 | 0.0 |
| mutation rate $M_g$ | 0.01 | 0.05 |

* Minimal Generation Gap[16]

0 または 1 に固定される stuck-at-0(or 1) fault を扱う．

### 4.2　実験設定

#### 4.2.1　環境設定と実験パラメータ

**表 3** にシミュレートプログラムの環境設定，**表 4** に GA のパラメータ，**表 5** および**表 6** に RFCN のパラメータを示す．提案手法では通常学習によって獲得された中間ユニット数に応じて，耐故障学習のユニット群の数が変わってしまう．本実験ではユニット群が有効に機能していることを検証するため，通常学習時から中間ユニット数を固定する．そのため，従来研究の RFCN における中間ユニット数の突然変異は行わない．

#### 4.2.2　ネットワークの制御成功判定

獲得した RFCN が制御に成功したかどうかの判定条件

表 5　RFCN のパラメータ
**Table 5**　RFCN parameters.

| parameter | value |
|---|---|
| output function | See Table 6. |
| parameter | 0.25, 0.5, 1.0, 2.0 |
| reaction speed | fast, slow |
| threshold | $\pm 1.0, \pm 0.5, \pm 0.25, 0.0$ |
| connection weight | $\pm 1.0, \pm 0.5, \pm 0.25, 0.0$ |

表 6　RFCN の出力関数
**Table 6**　Output functions of RFCN.

| function | equation |
|---|---|
| threshold | $f(x) = \begin{cases} \alpha & (x > 0) \\ 0 & (x \leq 0) \end{cases}$ |
| sigmoid | $f(x) = \dfrac{1}{\exp(-\alpha x)}$ |
| linear | $f(x) = \alpha x$ |
| piecewise linear | $f(x) = \begin{cases} 0 & (x \leq 0) \\ \alpha x & (0 < x < \frac{1}{\alpha}) \\ 1 & (\frac{1}{\alpha} \leq x) \end{cases}$ |

∗ output unit function is sigmoid or piecewise linear.

は，「棒が上方 ±5° で連続 600 ステップ制御できたこと」とする．棒の制御範囲としてこのような条件とした理由は，ほぼまっすぐ上を向いた状態を維持すれば，安定した制御ができているという判断ができるためである．

#### 4.2.3　RFCN の入出力

入力は，台車の位置，台車の速度，棒の角度（cosine, sine），棒の角速度の 5 つである．出力は，台車にかける外力である．入力について，棒の角度に三角関数を用いた理由は，頂点では角度が 0 と $2\pi$ の値をとり，そのままでは異なる値で同じ制御をしなくてはならないためである．そのため，三角関数を用いることで，連続的な値の変化で入力を制御できるようにした．出力について，出力 $o$ の範囲は [0, 1] である．この範囲を式 (1) によって $[-F_{max}, F_{max}]$ の範囲に変換し，台車にかける外力とする．$F_{max}$ は表 3 の最大外力と同一である．

$$F = 2 \cdot F_{max} \cdot (o - 0.5) \tag{1}$$

#### 4.2.4　適応度関数

通常学習時の適応度関数 $eval_1$ は式 (2) で計算する．

$$eval_1 = 0.2 \cdot (1 - x) + 0.8 \cdot (1 + \cos\theta) \tag{2}$$

$x$ は中心を 0 とする台車の位置，$\theta$ は上方を 0° とした棒の角度である．つまり，台車が中心に近く，棒が上方にあるほど適応度が高くなる．

耐故障学習時の適応度関数 $eval_2$ は式 (3) で計算する．

$$eval_2 = \frac{eval_1 + \sum_{i=1}^{H} eval_1^i}{1 + H} \tag{3}$$

$H$ はユニット群中にあるユニットの数を表す．$eval_1^i$ はユニット群の $i$ 番目のユニットが故障したときの適応度関数である．つまり，すべてのユニットが正常な場合と，対象とするユニット群のユニットがそれぞれ 1 つ故障した場合の平均値を最終的な適応度として評価する．

#### 4.2.5　比較手法

本実験では，耐故障性を獲得する比較手法として Emmerson らの手法 [1] を 3 層の階層型 NN（FFNN）に適用した手法を用いた．始めに，耐故障性を有していないネットワークを，中間ユニットの数を 5 個で作成した（RFCN5, FFNN5）．その後，RFCN と FFNN の耐故障性を学習させたネットワークを，中間ユニット数をそれぞれ 10，20，40 の組み合わせで行った (RFCN10, RFCN20, RFCN40, FFNN10, FFNN20, FFNN40)．

### 4.3　実験結果と考察

**図 6** および**図 7** に全ユニットに占める故障ユニットの割合（故障率）と，それによる成功率の推移グラフを示す．成功率は 5%刻みで各 250 試行分の平均値とする．故障させるユニットについては，毎試行ランダムに選定した．これらの結果より，提案手法が比較手法より高く成功率を維持できていることが分かった．出力故障値が 0 のとき，故障率が 20%の場合では，提案手法では冗長化によって成功率が最大 65%向上した．しかし，比較手法ではほとんど向上が見られていない．これによって，提案手法による冗長化の手法の有効性を示すことができた．また，出力故障値が 1 のとき，比較手法では故障率が 20%ですべての場合において成功率が 30%を下回っているが，提案手法では 70%以上の成功率を維持することができた．比較手法では，ユニット数が増えるにつれて成功率が悪化しているが，提案手法では同等もしくはそれ以上の成功率を達成できた．提案手法では，ユニットの重要度を考慮して冗長化を行うことができたため，故障の割合が増えたとしても，出力に重要なユニットを多く冗長化することで，成功率を維持できたためと考えられる．

**表 7** に提案手法における，耐故障学習において獲得したフィルタの割合を出力故障値に分けて示す．出力故障値 0 の場合は，通常のユニット出力は正と負の範囲にあり，故障値がその中間にあるため，どちらの値も考慮できるようにフィルタの割合がほぼ 1 対 1 となり，出力の表現をバランスよく獲得できていることがわかる．出力故障値 1 の場合は，出力が 1 を超えることがほぼないため，故障出力を抑制するためにほぼすべて MIN フィルタが獲得された．この結果から，故障値に応じて適切なフィルタの種類を適切に獲得できたことがわかる．

## 5.　まとめと今後の課題

本論文では，複数の中間ユニットが 0 または 1 に固定

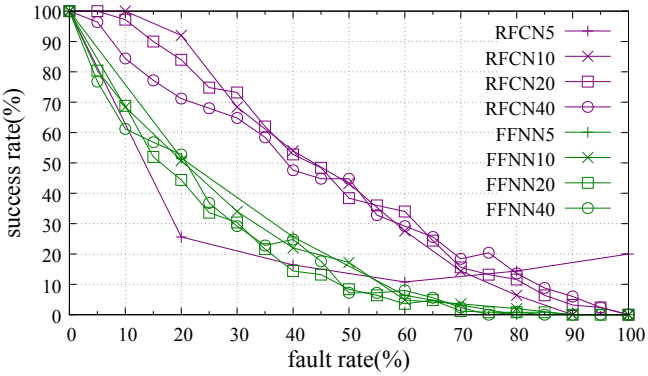


**図 6** 出力故障値が 0 の場合
**Fig. 6** Stuck-at-0 fault.

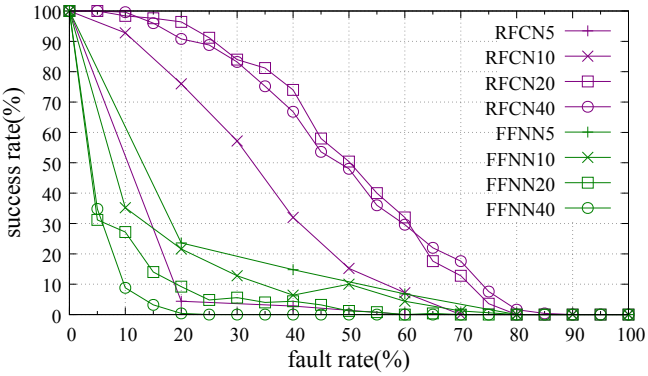


**図 7** 出力故障値が 1 の場合
**Fig. 7** Stuck-at-1 fault.

**表 7** 獲得したフィルタの割合
**Table 7** Rates of acquired filters.

| | stuck-at-0 fault | stuck-at-1 fault |
|---|---|---|
| MAX filter | 45% | 4% |
| MIN filter | 54% | 96% |

される故障について，ユニットの冗長化により耐故障性を獲得する手法を提案した．冗長化に際して，ユニットの重要度を考慮することで，ネットワークで重要な役割を果たすユニットを優先的に複製した．また，複製したユニットをユニット群としてひとつにまとめ，ユニット群の出力をフィルタを用いて統合することで，ユニットの故障を完全に吸収した．提案手法を振り上げ倒立振子問題に適用し，従来手法と比較して提案手法の有効性を検証した．実験によって，重要なユニットであるほど冗長化できているため，故障率が増えても成功率を維持できることが分かった．さらに，故障値に応じて適切なフィルタの種類を獲得することができた．

今後の課題としては，本研究では NN の故障をかなり限定したものにしている．具体的には，中間ユニットのみを対象とし，故障値は 0 と 1 の 2 種類のみを扱っている．NN の故障は様々な種類が考えられるため，今後はより多くの NN の故障を考慮し，それに対する耐故障性をどのように獲得するか検討する必要がある．また，提案手法はユニット群にフィルタを通すことによって冗長性の獲得を行ったが，フィルタ自体の故障も検証する必要がある．またフィルタについて，提案手法では MAX フィルタと MIN フィルタの 2 種類を用いたが，より柔軟な表現を可能にするためには，様々な種類のフィルタを考える必要がある．本論文では単純な問題として振り上げ倒立振子問題に適用し，性能評価を行った．そのため，より複雑な問題に適用することによって，提案手法の有効性を示せるか確認する必要がある．また，本実験ではシミュレータ上での実験であったため，実際にハードウェア上で実装し，耐故障性が正しく確保されているかを確認する必要がある．

## 参考文献

[1] Emmerson, M. D. and Damper, R. I.: Determining and improving the fault tolerance of multilayer perceptrons in a pattern-recognition application, *IEEE Transactions on Neural Networks*, Vol. 4, No. 5, pp. 788–793 (1993).
[2] Dias, F. M. and Antunes, A.: Fault Tolerance improvement through architecture change in Artificial Neural Networks, *In Advances in Computation and Intelligence, Springer Berlin Heidelberg*, pp. 248–257 (2008).
[3] Bernier, J. L., Ortega, J., Rojas, I., Ros, E. and Prieto, A.: Obtaining fault tolerant multilayer perceptrons using an explicit regularization, *Neural Processing Letters*, Vol. 12, No. 3, pp. 107–113 (2000).
[4] 佐藤勝，高浪五男，堀田忠義：故障値注入による階層型ニューラルネットワークの重み故障に対する耐故障化，電子情報通信学会技術研究報告. FTS, フォールトトレラントシステム， Vol. 100, No. 30, pp. 49–56 (2000).
[5] Zarafshan, F., Latif-Shabgahi, G. R. and Karimi, A.: A novel weighted voting algorithm based on neural networks for fault-tolerant systems, *3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT)*, Vol. 9, pp. 135–139 (2010).
[6] Chin, C. T., Mehrotra, K., Mohan, C. K. and Rankat, S.: Training techniques to obtain fault-tolerant neural networks, *24th International Symposium on Fault-Tolerant Computing (FTCS-24)*, Vol. 9, pp. 360–369 (1994).
[7] Zhou, Z. H., Chen, S. F. and Chen, Z. Q.: Evolving Fault-Tolerant Neural Networks, *Neural Computing & Applications*, Vol. 11, No. 3-4 (2003).
[8] 西垣正勝，都筑輝泰，曽我正和：ニューラルネットワークの耐最悪故障化学習，電子情報通信学会論文誌 D，Vol. 83, No. 1, pp. 203–214 (2000).
[9] Zhou, Z. H., Chen, S. F. and Chen, Z. Q.: Improving tolerance of neural networks against multi-node open fault, *International Joint Conference on Neural Networks*, Vol. 3, pp. 1687–1692 (2001).
[10] Singh, A. P., Chandra, P. and Rai, C. S.: Empirical study of FFANNs tolerance to weight stuck at zero fault, *International Journal of Engineering and Technology*, Vol. 2, No. 2, pp. 65–70 (2010).
[11] Stanley, K. O. and Miikkulainen, R.: Evolving neural networks through augmenting topologies, *Evolutionary computation*, Vol. 10, No. 2, pp. 99–127 (2002).
[12] Shirakawa, S. and Nagao, T.: Action control of autonomous agents in continuous valued space using

RFCN, *IEEJ Transactions on Electronics, Information and Systems*, Vol. 127, pp. 762–769 (2007).

[13] Stanley, K. O., D'Ambrosio, D. B. and Gauci, J.: A hypercube-based encoding for evolving large-scale neural networks, *Artificial Life*, Vol. 15, No. 2, pp. 185–212 (2009).

[14] Risi, S. and Stanley, K. O.: Evolving the placement and density of neurons in the hyperneat substrate, *In Proceedings of the 12th annual conference on Genetic and evolutionary computation. ACM*, pp. 563–570 (2010).

[15] Risi, S. and Stanley, K. O.: Enhancing es-hyperneat to evolve more complex regular neural networks, *In Proceedings of the 12th annual conference on Genetic and evolutionary computation. ACM*, pp. 1539–1546 (2011).

[16] 佐藤浩，小野功，小林重信：遺伝的アルゴリズムにおける世代交代モデルの提案と評価，人工知能学会誌，Vol. 12, No. 5, pp. 734–744 (1997).