# 遺伝的プログラミングを用いたニューロ進化に関する研究

海内 映吾[†], 伊庭 斉志[††]
東京大学工学部[†], 東京大学情報理工電子情報学専攻[††]

## 1. 序論

### 1.1 背景

現在機械学習の技法は多岐に渡る. その中でも人間の脳神経回路をコンピュータで実現しようとするニューラルネットワークにおいては, セントラルフロリダ大学の Stanley 准教授は構造や, ノード間の重みを決定する関数に進化計算を用いる[1]などして発展させている. その一つである NEAT(NeuroEvolution of Augmenting Topologies)[2]という手法を, 少し仕組みを変えて Modular Multiobjective NEAT[3]という枠組みを作っている研究もある. 本研究もその一つで NEAT を拡張するものである.

### 1.2 目的・意義

本研究の目標は, 機械学習の技法の一つであるニューラルネットワークにおいて, 出力の値を決めるのに必要なネットワークのノード間の重みという値を遺伝的プログラミングを用いた関数同定で決定し, ニューラルネットワークの構造を動的に変化させることである. そして最終的には, より複雑なシミュレーションにも高速で解を得られる有用性を検証することである.

ニューラルネットワークにノードは複数存在し, その分考えないといけない重みの数が増えていく. 従来手法ではその一つ一つを評価して, 最適となる値をそれぞれで探す必要がある. 本研究における関数同定の利点は一つの関数を定めることによって, それ一つですべてのノード間の重みを決定できるということである. つまりは 2 つのノードの座標を関数に代入するだけでよいということである. また, ノードの位置には座標を設定しているので, ニューラルネットワークの構造変化にも対応しやすくなる. HyperGP[4]という手法では, 重み決定に遺伝的プログラミングを用いているがノードの位置は固定させている. 本研究で構造変化を行う理由としては, ノードの位置, 特に入力ノードの位置が固定されていると関数を 1 つに定めていると難しい問題, 例えば入力値が非常に多い問題に対面したとき, 解探索に限界が出てくるからである. 構造変化には NEAT アルゴリズムを適用する. よって, NEAT アルゴリズムと遺伝的プログラミングを組み合わせた手法となる.

## 2. 関連研究

### 2.1 ニューロ進化

ニューロ進化とはニューラルネットワークに進化計算を用いる手法で, 正しい入力と出力の対の概要を作るのが困難なゲームやロボット制御に有効である. 強化学習の一種に位置づけられる. 近年流行しているディープラーニングはビッグデータを扱っていて, ニューロ進化とはタスクに違いがある. ネットワークを変化させて解くことができる問題を対象とする. 多くのアルゴリズムが存在しており, あらかじめ指定された構造のネットワークで結合重みの値を進化させる手法と重みに加えてネットワークの構造を進化させる手法の違いがある. さらに, パラメータと並行してニューラルネットワークの構造を進化させる手法とそれらを別々に開発する手法の違いがある.

### 2.2 NEAT

NEAT とは, 分かりやすく言えばニューラルネットワークを進化させるものである. 特徴としては, ニューラルネットワークの接続形態を交叉の原理に基づき, 種の形成をして構造をより良いものとし, 結果的に, 最小限の構造からノードの数やノード間を結ぶ接合部の数を増やしつつ成長させるというものである. 構造変化を示したものが Fig. 2.3 である. 進化には進化的アルゴリズムの 1 つである遺伝的アルゴリズムを用いており, その進化過程を表したものが Fig. 2.4 である. 対象となる問題に合わせてネットワークの構造が変化するので当然通常のニューラルネットワークより速く解の探索を行うことができる.

### 2.3 HyperNEAT

次に NEAT とは異なる HyperNEAT[5]について説明する. NEAT がニューラルネットワークを進化させる一方, HyperNEAT はノード間の重みを決定する CPPNs(Compositional Pattern Producing Networks)を進化させる. CPPNs とは関数の組み合わせによるネットワークのことで Fig. 2.5 にあるように, 入力値から複数の関数を経て出力値を決定するものである. 例として, Fig. 2.6 ではあるノード座標 $(x_1, y_1)$ から別のノード座標 $(x_2, y_2)$ までの重みを, 4 つの入力値を CPPNs に入れて出力する. NEAT では幾何学的規則

を明確に学習することができないが HyperNEAT ではそれができるというのが利点である. 本研究は, ニューラルネットワークに座標を設定するという点で同じである. 相違点は, CPPNs の代わりに遺伝的プログラミングを用いることであり, その手法に関しては次章で説明する.

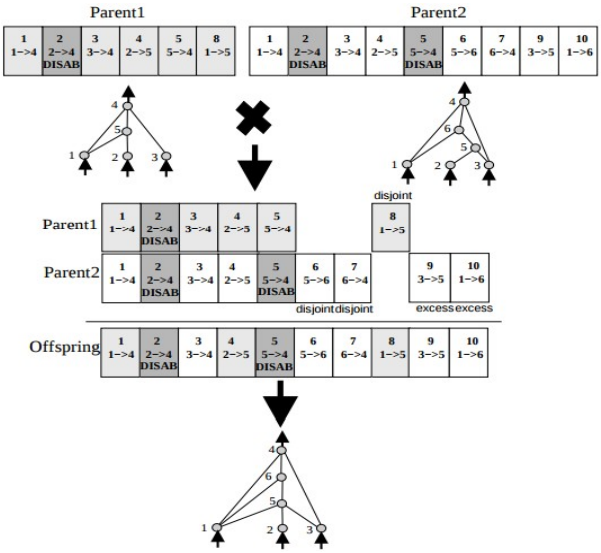


Fig. 2.3 NEAT の構造変化

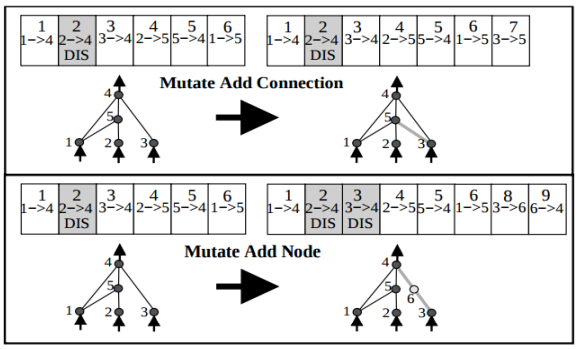


Fig. 2.4 NEAT の進化過程

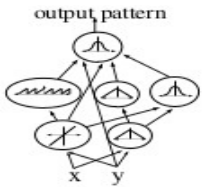


Fig. 2.5 CPPNs の構造

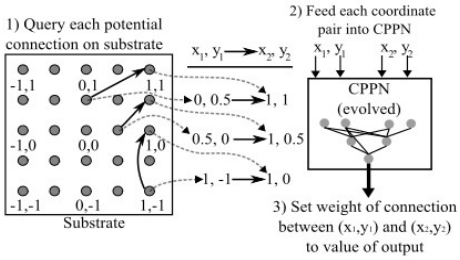


Fig. 2.6 HyperNEAT の構造

## 3. 提案手法

### 3.1 関数同定

関数同定とは, 数式空間を探索する回帰分析の一つで与えられたデータセットに対して正確かつ単純な最もふさわしいモデルを見つけようとするものである. NEAT や HyperNEAT でもこの作業は行わない. 遺伝的プログラミングを用いている点でそれらとは異なり, 一つ一つの結合重みを評価することを変えている. ニューラルネットワークの N 個のノードを Fig. 3.1 のように$xy$平面にまずは固定し, 座標をそれぞれ$(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$とする. 座標が$(x_i, y_i)$で与えられるノードの番号を$i$と決め, ノード$i$とノード$j$の間の重み$w_{ij}$は式(3.1)で表される.

$$w_{ij} = f(x_i, y_i, x_j, y_j) \tag{3.1}$$

この 4 変数の関数を関数同定問題とみなして解けばニューラルネットワークが完成することになる.

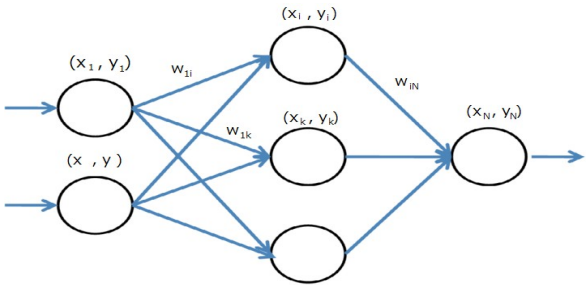


Fig. 3.1 座標設定されたニューラルネットワーク

### 3.2 遺伝的プログラミング

非線形関数の関数同定問題を解く場合には遺伝的プログラミング(GP)を用いる. GP では遺伝子型の表現に木構造を用いる. 木構造とは木の形をしたデータ構造である[7]. これは, ノードとノード間を結ぶ枝で構成されている. 下位の末端にある葉ノード以外は、最上位にある根ノードから下方向に子ノードを持つ. 数式として成り立つために葉ノードは必ず文字, それ以外は必ず演算子が入る. 演算子が sin または cos の場合は子ノードは一つとなり, $+ - \times \div$の場合は子ノードは 2 つとなる.

Fig. 3.2 は木構造を用いた数式表現の一例で, $(a - b) * (a - c)$となる.

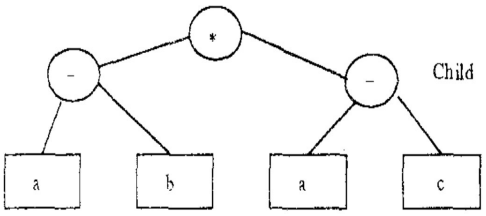


Fig. 3.2 $(a - b) * (a - c)$の木構造

### 3.3 アルゴリズム

ここでは非線形関数の関数同定問題を解く流れを説明する.

ステップ 1)初期化:初期集団のそれぞれの個体にランダムな深さ, ランダムな演算子で木構造を生成する. ここで集団の要素数を N, それぞれの個体を $tree_i(i = 0, 1, ..., N-1)$ とおく.

ステップ 2)評価:個体の出力値とデータの真値との誤差を表す評価関数を$fit()$とし, $fit(tree_i)$を$tree_i$の評価値とする. $fit(tree_i) < value$ を満たすとき操作を終了する. 但し$value$は自分で決められる定数である.
ステップ 3)選択:評価の高い, すなわち$fit(tree_i)$の小さいものを選択し, それを親としてコピーすることで子供を作る.

ステップ 4)変異:ある確率で交叉, 突然変異を起こす. 変異が起こるノードの位置はランダムである. Fig. 3.3 は交叉と突然変異を表していて, 前者では, 2 つの個体 $tree_i, tree_j$の, あるノード以下の木構造がすべて入れ替わる. 後者では, 1 つの個体$x_i$のあるノード以下の木構造が破棄され, ランダムで生成される新たな木構造に生まれ変わる.

ステップ 5)　操作が終了するまでステップ 2～4 を繰り返す.

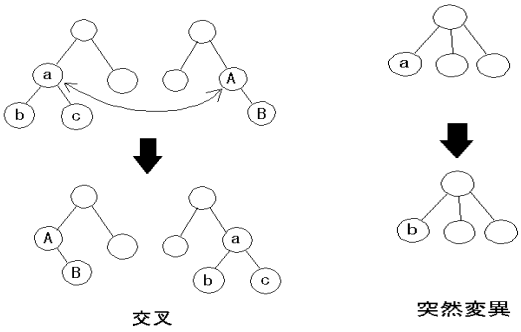


Fig. 3.3　木構造における交叉と突然変異

## 4. 実験

### 4.1 倒立振子問題

本研究の手法で正しく問題を解くことができるかを試すために倒立振子[8]という代表的な制御問題でシミュレーション実験を行った. Fig. 4.1 のように, 棒を乗せた台を$x$軸方向に動かすことで棒が倒れずに立ったままの状態を何秒保てるかを測る. $m, l$をそれぞれ棒の質量, 長さとして, $x$軸方向に力 F を与えたときの運動方程式を考えると式(4.2)と式(4.3)が得られる.

$$\ddot{x} = \frac{F}{m} + \dot{\theta}^2 \sin\theta - \ddot{\theta}\cos\theta \tag{4.2}$$

$$\ddot{\theta} = \frac{g\sin\theta - \cos\theta(\frac{F}{m} + l\dot{\theta}\sin\theta)}{l(\frac{4}{3} - \cos^2\theta)} \tag{4.3}$$

これら 2 つの式から $x, \dot{x}, \theta, \dot{\theta}$ が分かるので, そのうちの $\dot{x}, \theta, \dot{\theta}$ を見て力 F を決定する. 力 F を 0 より大きい定数と見なして, $x$軸正方向か$x$軸負方向のどちらかに力 F を与えることとする.

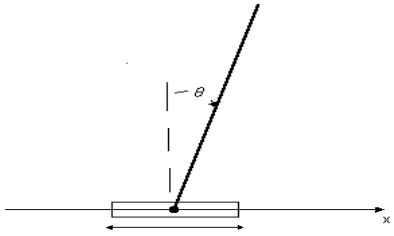


Fig. 4.1　倒立振子

ニューラルネットワークで考えるとき, $\dot{x}, \theta, \dot{\theta}$ が入力で F が出力ゆえ 3 入力 1 出力となる. ノードの数は全部で 10 個とし, Fig. 4.2 のように, それぞれのノードに座標を定めた. また, 第 3 章 3.3 で説明したアルゴリズムの評価方法を変える必要がある. 関数 $f(x_i, y_i, x_j, y_j)$ によって定まる重みで倒立振子が倒れるまでの時間を測定し, その時間を評価値とした. つまり評価値が大きいほど評価が高いということになる.

また, 本実験では GP のアルゴリズムにおけるステップ 3 の「選択」の方法としてトーナメント方式選択法[9]を採用した. これは, 個体群から一定数 $n$ の個体をランダムに取り, その中で最も評価が高いものを次世代に残す手法である. トーナメントという名であるが, 実際には最大評価度を持つ個体を残しているだけである. $n$ の数は小さいほうが低い評価を持つ個体を残しやすく, ランダムに選ぶ中の順位なので偶然性は高い.

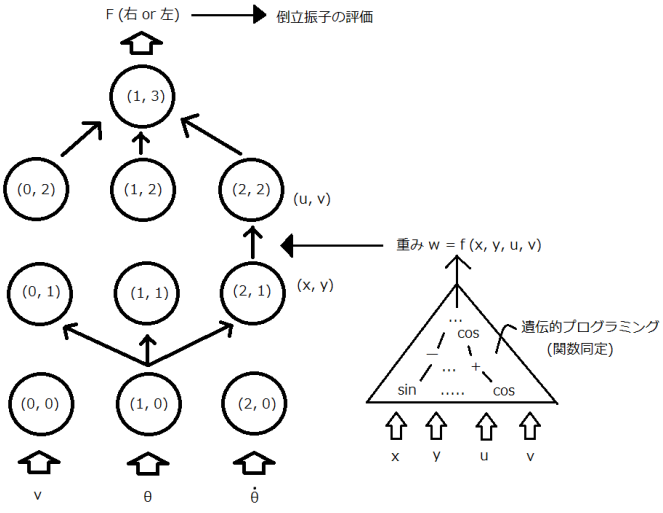


Fig. 4.2　本実験の概観図

## 4.2 倒立振子問題の結果

GP の各パラメータ, およびシミュレーション結果は以下に示す通りである.

**Table 4.1: GP の各パラメータ**

| | |
|---|---|
| 個体数 | 100 |
| 選択割合 | 0.06 |
| 木構造の最大の深さ | 15 |
| 突然変異率 | 0.1 |
| 交叉率 | 0.9 |
| トーナメント選択の閾値 | 0.7 |

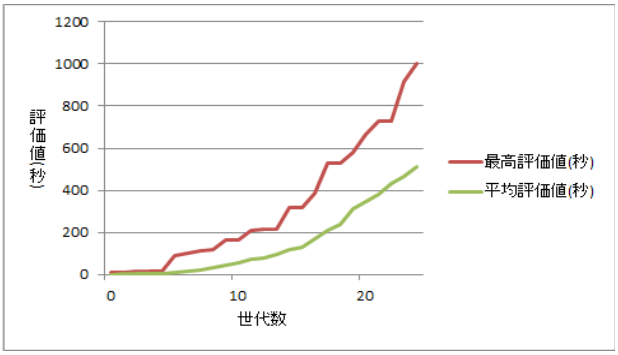


Fig. 4.3 世代ごとの評価値

用いた倒立振子のプログラム[10]は, 元々遺伝的アルゴリズムを用いて問題を解こうとしたものであり, 第 300 世代の段階で評価値は 36 秒であった. 遺伝的アルゴリズムのパラメータ設定が粗いといえ, 本手法で十分良い結果が得られたと言える.

次に, 得られた関数 $f(x, y, u, v)$ の有用性を確認するために Fig. 4.4 のように, 番号を事前に割り振られたノードを 1 つ減らして, それぞれで最終世代である第 24 世代の関数を使って, 評価値を求める実験を行った. これはネットワークが壊れてもある程度成績を保証するという意味で眼鏡性を持っているとともに, ネットワークの構造を進化させる際に, 1 から結合重みを決める関数を作る必要がないことを示すための実験である. Table 4.2 はその結果である. ノード番号 1, 2, 3 は入力値, ノード番号 10 は出力値を表すノードゆえ減らすことはしなかった.

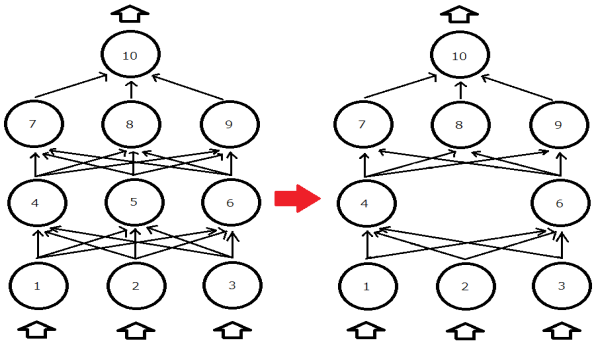


Fig. 4.4 ノードを 1 つ減らしたときの概観図

**Table 4.2: ノードを 1 つ減らしたときの評価値**

| 減らしたノードの番号 | 評価値(秒) |
|---|---|
| none(元の状態) | 1859.24 |
| 4 | 469.40 |
| 5 | 1693.40 |
| 6 | 28.16 |
| 7 | 1859.34 |
| 8 | 0.18 |
| 9 | 1859.34 |

## 4.3 Ms.Pacman

Ms.Pacman は, クリアするのに複数の要素を必要とするビデオゲームである. ピルと呼ばれるアイテムを集めるのに加え, ランダムで動く 4 体のゴーストから逃げ, パワーピルを得て, 食用となったときのゴーストは捕まえなければならない. ゴーストに 3 回捕まるまでに獲得した得点を競い合う. Fig. 4.5 はプレイ中の様子である.

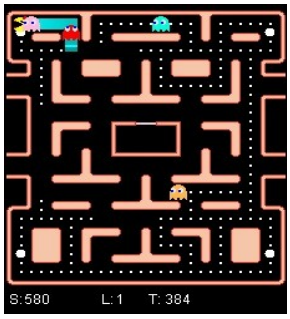


Fig. 4.5 Ms.Pacman

自分が次にどの方向に動けばいいのかが問題となる. これをニューラルネットワークで決定することにする. まずは今の位置から移動しうる方向を割り出し, その方向に移動したあとの位置における 10 個の情報を入力とし, 評価値を出力とする. 出力値が最も大きい方向に移動する. つまりは 10 入力, 1 出力のニューラルネットを移動しうる方向の数だけ考えることとなる. これを表したのが Fig. 4.6 と Fig. 4.7 である. 具体的に何を入力とするかは Table 4.3 に示す通りである.

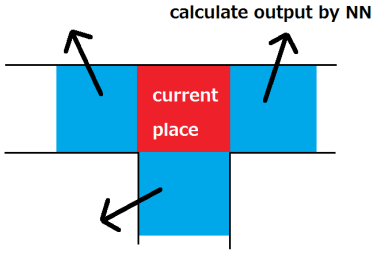


Fig. 4.6 現在の位置と移動しうる位置

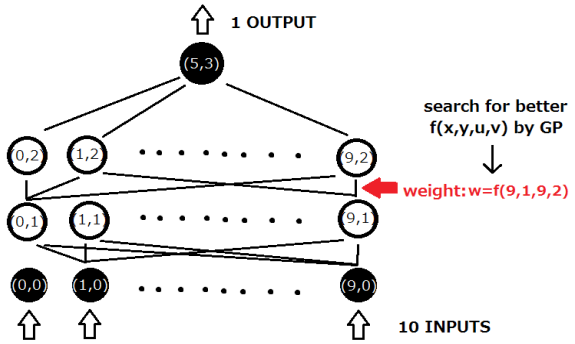


Fig. 4.7 ニューラルネットワークの概観図

**Table 4.3: ニューラルネットワークの入力**

| 入力の番号 | 情報 |
|---|---|
| 1 | 1 番目に近いゴーストとの距離 |
| 2 | 1 番目に近いゴーストの状態<br>(-1:ノーマル, 1:食用) |
| 3 | 2 番目に近いゴーストとの距離 |
| 4 | 2 番目に近いゴーストの状態 |
| 5 | 3 番目に近いゴーストとの距離 |
| 6 | 3 番目に近いゴーストの状態 |
| 7 | 4 番目に近いゴーストとの距離 |
| 8 | 4 番目に近いゴーストの状態 |
| 9 | 最も近いピルまでの距離 |
| 10 | 最も近いパワーピルまでの距離 |

### 4.4 Ms.Pacman の結果

予備実験として, ランダムで動くようにコントローラを設定すると結果のスコアは100点にも満たなかった. また, 常に最も近いピルの方向に動くようにしたときにはスコアは 4500 点程度となった. 今回実験で得られた結果は Fig. 4.8 である. ゴーストはランダムで動くため同じ木構造からなる関数で再度実験しても同じスコアは得られないことに注意しないといけない.

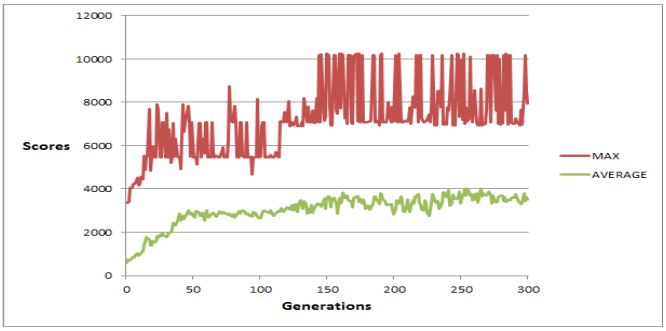


Fig. 4.8 Ms.Pacman の実験結果

**Table 4.4: 他手法との比較**

| 手法 | 平均得点 | 最高得点 |
|---|---|---|
| ランダム(後退あり) | 88 | 170 |
| ランダム(後退なし) | 1513 | 5010 |
| 最短距離法 | 4316 | 6860 |
| GP(BA) | 19198 | 33420 |
| ACO | 36031 | 43467 |
| MM-NEAT | 65447 | 100070 |
| 本手法 | 4014 | 10240 |

## 5. 考察

倒立振子問題について, 初期世代では平均で 1 秒ももたなかったものの最終世代である第 24 世代では, 平均で 500 秒を超え, 最良個体は, 上限に設定していた 1000 秒に到達することができた. 重みを決定する関数 $f(x, y, u, v)$ は構造がより複雑になっていってはいるものの, 類似性がある程度見られる. 木構造の深さの上限を 15 に決めているので関数の式の項数も限られてくる. 結果を見る限りでは, 良い結果を出しているものは項数も多い. よって、解く問題が難しくなればなるほど木構造の深さの上限を大きくする必要がある.

1 つのノードを減らして, GP で得られた最終世代の関数の有用性を確かめる実験では, 減らすノードの番号によって結果が大きく異なることとなった. 例えば, 図 4.5 のノード番号 7, 9 では, 減らす前と結果が全く変わらない 1859.34(秒)であったのに対し, ノード番号 8 のとき評価値が 1(秒)を切る結果となった. また, それ以外では, GP の第 0 世代より良い結果となった. これより, 今後ノードも変化させていくときの初期関数として, この最終世代の関数は活用できると言える. ノードを減らしても結果が変わらないなら, その分結果出力が速くなる.

Ms.Pacman について, 予備実験で得られたスコアより高いが, 10,000 点近くが上限となっており途中で何かしらつまづいているのがわかる. 第 150 世代あたりから, GP だけではスコアを伸ばすことができずにいる. これを解決するために入力情報を見直したり, 倒立振子問題とは違って, ニューラルネットワークの構造を動的に変化させられるようにする必要がある.

## 6. 結論

適正度が順調に上がっていることから, 解探索の速度に従来の手法と優劣があるといえども, 単純な制御問題における解探索としての正しさは証明された. ニューラル

ネットワークが様々な形で性能を向上させている中で, これも一つの進歩と言える. GP における, 個体数, 木構造の深さ, などのパラメータも必要に応じて変化させられるので, より良くなる可能性を秘めている.

今後の課題について, 今回倒立振子問題で, 自分のノードの位置を変化させたが, それを動的に行えるようにする. そうすることでスコアが伸び悩む Ms.Pacman のような問題に対してもさらに良い結果が得られるようにする.

## 参考文献


[1] Jason Gauci, Kenneth O. Stanley (2008) A Case Study on the Critical Role of Geometric Regularity in Machine Learning. Twenty-Third AAAI Conference on Artificial Intelligence: 628-633

[2]Kenneth O. Stanley (2002) Evolving Neural Networks through Augmenting Topologies. Evolutionary Computation Volume 10: 99-127

[3]Jacob Schrum, Risto Miikkulainen (2014) Evolving multimodal behavior with modular neural networks in Ms. Pac-Man. 2014 Annual Conference on Genetic and Evolutionary Computation: 325-332

[4]Zdenek Buk, Jan Koutnik, Miroslav Snorek(2009) NEAT in HyperNEAT Substituted with Genetic Programmming. Adaptive and Natural Computing Algorithms: 243-252

[5]Jason Gauci, Kenneth O. Stanley (2008) A Case Study on the Critical Role of Geometric Regularity in Machine Learning. Twenty-Third AAAI Conference on Artificial Intelligence: 628-633

[6]Douglas C. Montgomery, Elizabeth A. Peck, Geoffrey Vining (2012) Introduction to Linear Regression Analysis. John Wiley & Sons

[7]Ben McKay, Mark J. Willis, Geoffrey W. Barton (1995) Using A Tree Structured Genetric Algorithm To Perform Symbolic Regression. Genetic Algorithms in Engineering Systems: Innovations and Applications: 487-492

[8]Stefan Schaal (1997) Learning From Demonstration. Advances in Neural Information Processing Systems 9: 1041-1046

[9]Brad L. Miller, David E. GoldBerg (1995) Genetic Algorithms, Tournament Selection, and the Effects of Noise. Complex Systems 9:193-212

[10]最適解を模索する遺伝的アルゴリズム
www6.plala.or.jp/mnagaku/cmag/ac19999/ga2.html