SCI'23 論文 特集号—V

論　文

Surrogate-Assisted Self-Adaptive MBEANN の提案とその評価*

小村　真央† ・宮本　明治† ・平賀　元彰‡ ・森本　大智§ ・大倉　和博†

Proposal and Evaluation of Surrogate-Assisted Self-Adaptive MBEANN*

Masahiro Komura†, Akiharu Miyamoto†, Motoaki Hiraga‡,
Daichi Morimoto§ and Kazuhiro Ohkura†

This paper proposes an extension for Mutation-Based Evolving Artificial Neural Networks (MBEANN) algorithm. The proposed method consists of two parts: a surrogate model and a self-adaptive mutation. Firstly, the surrogate-assisted mechanism is introduced to MBEANN for reducing the cost of fitness evaluations. This mechanism employs approximated fitness values predicted by a surrogate model instead of true fitness functions. Secondly, the self-adaptive mutation is applied to MBEANN for adjusting the exploring area in parameter space. The performance of the proposed method is compared with the normal MBEANN and NEAT algorithms by using the three benchmarks of OpenAI Gym. The experimental results showed that the proposed method outperformed other algorithms in all benchmarks.

1. はじめに

人工神経回路網 [1] には進化計算を用いて設計を試みる研究領域があり，進化型人工神経回路網 (Evolving Artificial Neural Networks: EANN)[2] とよばれている．勾配降下法と比較した利点として，ネットワーク構造のみならず，学習率などの学習の進行速度を決定するハイパーパラメータ，学習ルールそのものを遺伝子型に組み込んでの進化が可能であることが挙げられる [3]．一般的に EANN では，シナプス結合荷重値のみを進化の対象としている．しかし，ネットワーク構造を固定した場合，設計者が問題ごとに適切なネットワーク構造を決定する必要がある．そこで，結合荷重値だけでなく，ネットワーク構造も進化計算により獲得する研究が行われており，この分野は Topology and Weight Evolving Artificial Neural Networks (TWEANN)[4] とよばれている．TWEANN のアルゴリズムを設計するには，(i) 交叉によって遺伝情報が欠損する，(ii) 構造の変化により，適応度が低下する，(iii) 適切な初期構造を設定することが困難である，などの課題が知られている [5]．

TWEANN の代表的手法として，NeuroEvolution of Augmenting Topologies (NEAT)[4] がある．この手法は，初期構造を入力層と出力層のみとし，最小構造から探索を開始することにより，適切な初期構造を設定することが困難であるという問題に対処している．また，(i) 刷新番号を利用して交叉を行うことにより，親個体の持つ遺伝子情報を重複，欠損なしに受け継ぐ，(ii) 種分化により刷新構造を保護する，といった操作により交叉によって引き起こされる様々な問題に対処している．しかし NEAT には，交叉による構造の肥大化や，種分化による性能の低い個体の過剰な保護などの問題点がある．

TWEANN の手法として NEAT のほかに，単為生殖をモチーフとし，遺伝的操作に交叉を用いない Mutation-Based Evolving Artificial Neural Networks (MBEANN)[6] がある．MBEANN は，自然淘汰と中立

* 原稿受付　2023 年 8 月 17 日
* 第 67 回システム制御情報学会研究発表講演会にて発表（2023 年 5 月）
† 広島大学 大学院 先進理工系科学研究科　Graduate School of Advanced Science and Engineering, Hiroshima University; 1-4-1, Kagamiyama, Higashi Hiroshima, Hiroshima 739-8524, JAPAN
‡ 京都工芸繊維大学 機械工学系　Faculty of Mechanical Engineering, Kyoto Institute of Technology; Goshokaido-cho, Matsugasaki, Sakyo-ku, Kyoto 606-8585, JAPAN
§ 九州工業大学 大学院 工学研究院 機械知能工学研究系 Department of Mechanical and Control Engineering, Kyushu Institute of Technology, 1-1 Sensui, Tobata, Kitakyushu, Fukuoka 804-8550, JAPAN
Key Words: surrogate modeling, surrogate-assisted evolutionary algorithms, topology and weight evolving artificial neural networks, self-adaptive mutation.

突然変異の間にほぼ中立突然変異が存在する分子進化のほぼ中立説 [7] に着想を得ている．そのため，中立突然変異であるシナプス結合付加構造突然変異とほぼ中立突然変異であるノード付加突然変異を採用し，TWEANN の問題点である構造変化による適応度低下に対処している．また，NEAT 同様に最小構造から探索を開始することにより，適切な初期構造を設定することが困難であるという問題に対処している．TWEANN は，進化計算手法の一種であるため，多数の解候補の評価を想定している．そのため，1 回の解評価に高い計算コストがかかる問題 (Expensive Optimization Problem: EOP)[8] への適用には，多くの計算リソースが要求される．EOP への一般的なアプローチとして，問題の近似モデル，すなわちサロゲートモデルを用いることが挙げられる．現在まで，サロゲートモデルを進化計算に適用した，サロゲート進化アルゴリズム (Surrogate-Assisted Evolutionary Algorithm: SAEA)[9] の研究が盛んに行われている．2018 年，A. Gaier らは SAEA を NEAT に適用した Surrogate-Assisted NEAT (SA-NEAT)[10] を提案した．

本研究では，SA-NEAT におけるサロゲート評価のアプローチを MBEANN にも適用することを試みる．これに加えて，シナプス結合荷重突然変異およびバイアス値突然変異におけるステップサイズの設定法として，進化戦略で用いられている自己適応型突然変異法 [11] を導入した Surrogate-Assisted Self-Adaptive MBEANN (SA-MBEANN) を提案する．また，SA-MBEANN の有効性を検証するために，TWEANN の代表的手法である NEAT, SA-NEAT および従来の MBEANN と比較する．SA-MBEANN により，NEAT や MBEANN と比べ，少ない解評価回数で同等以上の適応度を獲得することを目指す．

本論文は以下のように構成される．第 **2** 章では，TWEANN について説明する．第 **3** 章では，サロゲートモデルの構築方法について説明する．第 **4** 章は，提案手法である SA-MBEANN について説明する．第 **5** 章では，実験結果について議論する．第 **6** 章では，実験結果に対する考察を行う．第 **7** 章では，本論文のまとめと今後の展望を示す．

2. TWEANN

一般的に，EANN ではシナプス結合荷重値のみを進化対象としているため，設計者がタスクごとに適切なネットワーク構造を決定する必要がある．そこで，結合荷重値とともに構造自体を進化計算によって獲得する分野があり，TWEANN[4] とよばれている．

2.1 NEAT

NEAT[4] は，2002 年に発表されて以来，TWEANN の最も標準的なアルゴリズムとなっている．このアルゴリズムは，まず最小限の位相構造を持つ個体の集団から始まり，突然変異によって構造が複雑になっていく．新しいノードと結合が追加されるたび，それらに刷新番号 (innovation number) という全探索集団を通して固有の番号が付与される．この番号により，異なる個体における共通ネットワーク要素が特定され，交叉や種分化のための指標となる．

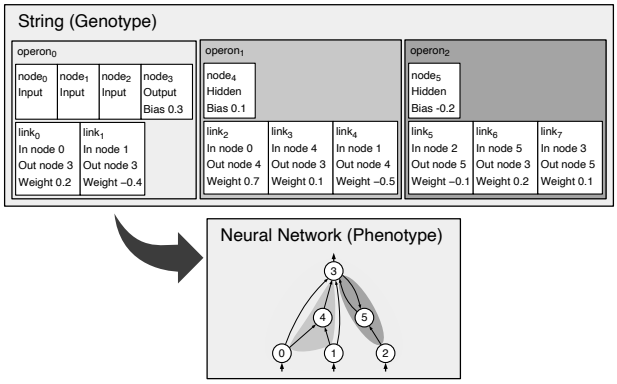


Fig. 1 Example of a genotype to phenotype mapping in MBEANN

NEAT の主要な特徴は，種分化によって構造差異の大きい個体を保護する点である．新しい個体が生成されると，既存の個体との類似度を測る適合性距離 (compatibility distance) が計算される．この距離があるしきい値を超えると新しい種に分類される．NEAT の個体 $\boldsymbol{x}$, $\boldsymbol{x'}$ の適合性距離は，不一致遺伝子数 G と，一致遺伝子の結合荷重値の差の平均 $\overline{W}$ を用いて，次のように計算される．

$$d(\boldsymbol{x},\boldsymbol{x'}) = c_1 G(\boldsymbol{x},\boldsymbol{x'}) + c_2 \overline{W}(\boldsymbol{x},\boldsymbol{x'}) \tag{1}$$

ここで，c_1, c_2 は定数である．NEAT では，異なる種に属する個体同士の交叉を制限するとともに，淘汰を各種内に限定することにより，異なる位相構造を持つ個体を保護している．

2.2 MBEANN

EANN で交叉を用いた場合，構造の競合現象 (competing conventions)[12] が発生し，タスクに重要な遺伝子情報が失われる場合がある．これに対し，MBEANN は単為生殖をモチーフとし，突然変異のみによって子個体が生成される．以下に MBEANN について詳細を述べる．

2.2.1 遺伝子型へのエンコーディング法

MBEANN の個体の構造の例を Fig. 1 に示す．string は (2) 式に示すように，遺伝子座部分集合である operon で構成されている．operon を構成する遺伝子座集合は，(3) 式に示すように，ノードとそれに接続するシナプス結合から構成されている．

$$\text{string} = \{\text{operon}_0, \text{operon}_1, \ldots, \text{operon}_m\} \tag{2}$$

$$\text{operon}_i = \{\text{link}_j | j \in O_{li}\} \cup \{\text{node}_k | k \in O_{ni}\} \tag{3}$$

ここで，m は string に含まれる operon の最大添字，O_{li} は operon_i に属するシナプス結合番号の集合，O_{ni} は operon_i に属するノード番号の集合である．各ノードはノード番号とノード形式（入力層ノード/隠れ層ノード/出力層ノードのいずれか）の値を持つ．隠れ層・出力層ノードの場合は，上記の値に加えてバイアスの値を持つ．また，link はシナプス結合を表しており，シナプス結合番号，入力（接続元）ノードの識別番号，出力（接続先）ノードの識別番号，結合荷重値の値を持つ．結合荷重値 w は実数値をとる．各 operon は表現型において部分ネットワークを構成し，異なる operon に属するノード間でのシナプス結合は生成されない．したがって，operon ごとに独立した部分ネットワークを形成することが期待される．初期個体が持つネットワーク構造を operon_0 とし，operon_0 は，すべての入力層・出力層ノードとそれらを結ぶシナプス結合によって構成される．

2.2.2 突然変異法

TWEANN では，構造自体を変化させると表現型の特徴が大幅に変化し，適応度が急激に低下するおそれがある．そこで，MBEANN は参考文献 [13] と同様に分子進化のほぼ中立説 [7] に則り，構造の突然変異が発生したとき，適応度がほとんど低下しないように設計されている．本項では，従来の MBEANN と比較した際の変更点であるシナプス結合荷重突然変異法およびバイアス値突然変異について説明する．

従来の MBEANN では，シナプス結合荷重突然変異およびバイアス値突然変異は，進化計算にて広く用いられている突然変異法を採用していた [6,11]．すなわち，シナプス結合荷重値およびバイアス値 $w_i(i=0,1,\ldots,n)$ は各個体が持つ固定値であるステップサイズ ς を用いて，一定の確率で (4) 式に示すように w_i' に変異する．

$$w_i' = w_i + \varsigma \cdot \mathcal{N}_i(0,1) \tag{4}$$

ここで，$\mathcal{N}_i(0,1)$ は i ごとに独立な平均 0，分散 1 のガウス分布から得られる乱数値を表す．

提案手法では，自己適応的なステップサイズ [11] をシナプス結合荷重突然変異およびバイアス値突然変異に導入している．これにより，ANN のパラメータだけでなく，ステップサイズにも選択圧が作用し，探索の進行に応じて突然変異幅を適切に調節することが期待される．この手法では，個体ごとのステップサイズ ς を使用し，突然変異操作を行う．ステップサイズ ς は，突然変異操作を行う前に更新される．ステップサイズの更新および突然変異操作は以下の式で示される．

$$\varsigma' = \varsigma \cdot e^{\tau \cdot \mathcal{N}(0,1)} \tag{5}$$

$$w_i' = w_i + \varsigma' \cdot \mathcal{N}_i(0,1) \tag{6}$$

ここで，ς' は更新後のステップサイズ，$\mathcal{N}(0,1)$ は平均 0，分散 1 のガウス分布から得られる乱数値を表す．パラメータ τ は，比例定数 c を用いて以下のように設定する．

$$\tau = c/\sqrt{n} \tag{7}$$

本研究では $c=1.0$ を用いる．

3. サロゲートモデルの構築

3.1 ガウス過程回帰

サロゲートモデルは，機械学習によって評価済みの解データから生成される．このモデルを用いて未評価解の評価値を予測する．様々な機械学習でサロゲートモデルを構成することができる [14] が，近年ではガウス過程回帰モデル [15] が最もよく用いられている．ここでは，参考手法である SA-NEAT[10] にならい，ガウス過程回帰モデルを用いる．事前データとして，評価点 x と評価値 y の N 個のペア

$$D = \{(x_1,y_1),(x_2,y_2),\ldots,(x_N,y_N)\} \tag{8}$$

が与えられたとき，未評価点 x^* における評価値 y^* をガウス過程回帰によって予測する場合を考える．評価値 $y=f(x)$ は，平均 $\mu(x)$，共分散関数 $k(x,x')$ のガウス過程に従うとする．

$$f(x) \sim \mathrm{GP}(\mu(x),k(x,x')) \tag{9}$$

ガウス過程回帰モデルは，評価点が似ていれば，評価値も似ているという直感的な仮定に基づく．共分散関数 k は，評価点 x の類似度をカーネルで定義する．

次式で示すカーネル関数は，ガウスカーネル (Gaussian Kernel) または動径基底関数 (Radial Basis Function: RBF) カーネルとよばれ，最もよく用いられている．

$$k(x,x') = \theta_1 \exp\left(-\frac{|x-x'|^2}{\theta_2}\right) + \theta_3 \delta(x,x') \tag{10}$$

θ_1, θ_2, θ_3 はこのカーネル関数の性質を決めるパラメータであり，$\delta(x,x')$ は $x=x'$ のとき 1，それ以外は 0 を返す関数である．

このカーネル関数を用いて，事前データと未評価点から次のカーネル行列とベクトルを生成する．

$$\boldsymbol{K} = \begin{bmatrix} k(x_1,x_1) & \ldots & k(x_1,x_N) \\ \vdots & \ddots & \vdots \\ k(x_N,x_1) & \ldots & k(x_N,x_N) \end{bmatrix} \tag{11}$$

$$\boldsymbol{k}_* = (k(x^*,x_1),k(x^*,x_2),\ldots,k(x^*,x_N))^T \tag{12}$$

未評価点 x^* における y^* はガウス分布に従うため，

$$p(y^*|x^*,D) = \mathcal{N}(\mu(x^*),\sigma^2(x^*)) \tag{13}$$

と表現できる．このとき，予測分布の期待値 $\mu(x^*)$ と分散 $\sigma^2(x^*)$ は次のように与えられる．

$$\mu(x^*) = \boldsymbol{k}_*^T \boldsymbol{K}^{-1} \boldsymbol{y} \tag{14}$$

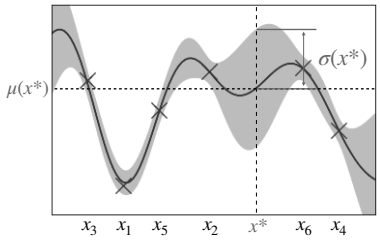


Fig. 2 Example of Gaussian process regression

$$\sigma^2(x^*) = k(x^*,x^*) - \boldsymbol{k}_*^T \boldsymbol{K}^{-1} \boldsymbol{k}_* \tag{15}$$

ガウス過程回帰の例を Fig. 2 に示す.

得られた予測分布の不確かさは，ベイズ最適化 (Bayesian Optimization)[16] によって定量化される．本研究では，次式で与えられる信頼性上限関数 (Upper Confidence Bound: UCB)[17] を用いる．

$$\mathrm{UCB}(x) = \mu(x) + \kappa\sigma(x) \tag{16}$$

ここで，κ は期待値と分散のどちらを重視するのかを決定するパラメータである．

3.2 適合性距離カーネル

ガウス過程は，(10) 式が計算できる場合，すなわち個体間の類似度を測る尺度が存在する場合，未評価解の予測ができる．しかし，TWEANN ではネットワーク構造が変化するため，入力空間が一定ではない．そこで，A. Gaier らは NEAT で用いられる適合性距離を個体間の距離尺度とした，適合性距離カーネルを提案した [10].

$$k(\boldsymbol{x},\boldsymbol{x'}) = \theta_1 \exp\left(-\frac{d(\boldsymbol{x},\boldsymbol{x'})^2}{\theta_2}\right) + \theta_3 \delta(\boldsymbol{x},\boldsymbol{x'}) \tag{17}$$

ハイパーパラメータ $\boldsymbol{\theta} = (\theta_1,\theta_2,\theta_3)$ は，次式で示される尤度関数を最大化することで最適化される．

$$\log p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\theta}) = -\frac{1}{2}\boldsymbol{y}^T \boldsymbol{K}^{-1}\boldsymbol{y} - \frac{1}{2}\log|(\mathbf{K})| \tag{18}$$

ここで，適合性距離は微分不可能であるため，共分散行列適応進化戦略 (Covariance Matrix Adaptation Evolution Strategy: CMA-ES)[18] を用いて，尤度関数を最大化する $\boldsymbol{\theta}$ を求める．

NEAT における個体の適応度を適合性距離カーネルにより形成されたサロゲートモデルによって得られる近似解で代替した手法が SA-NEAT である．本研究では，この技術を MBEANN に適用した SA-MBEANN を提案する．

4. Surrogate-Assisted MBEANN

MBEANN では交叉を用いないため，ノードやシナプス結合に全探索集団を通して固有の番号は付与されていない．そのため，SA-MBEANN では適合性距離を計算できるよう刷新番号を導入した．SA-MBEANN のアルゴリズムは以下の通りである．

(1) トレーニングセットの生成
世代 $t=0$ のとき，SA-MBEANN は，MBEANN と同様に最小限のネットワーク構造を持つ初期集団 P_0 から始まる．集団サイズは M とし，トレーニングセットを形成するまでの世代数を T とすると，$P_i (i=0,1,\ldots,T)$ のすべての個体は，真の目的関数 $F(\boldsymbol{x})$ をもとに評価され，サロゲートモデルのトレーニングセット D を形成し，サロゲートモデルを構築する．

(2) サロゲートモデルを用いた進化
個体の適応度 $\boldsymbol{y_t} = [y_1,y_2,\cdots,y_M]_t$ をもとに，MBEANN のアルゴリズムに従って子個体 $P_{t+1} = [\boldsymbol{x_1^*},\boldsymbol{x_2^*},\cdots,\boldsymbol{x_M^*}]$ を生成する．子個体は，サロゲートモデルと (16) 式の獲得関数により，近似適応度 $\boldsymbol{y'_{t+1}}$ が与えられる．次の世代の子個体 P_{t+2} を生成する際，適応度の代わりに近似適応度 $\boldsymbol{y'_{t+1}}$ を用いて進化のプロセスが繰り返される．

(3) モデルの更新
集団の中で，近似適応度 $\boldsymbol{y'_t}$ が最も高い個体から順に，設定した数 n だけ選択する．選択された個体 $[\boldsymbol{x_1},\boldsymbol{x_2},\ldots,\boldsymbol{x_n}]$ は Infill Individuals とよぶ．これらは真の目的関数 $F(\boldsymbol{x})$ で評価され，トレーニングセット D に追加される．トレーニングセット D には，個体 $\boldsymbol{x}$ とその適応度 y のペアをストックできる上限値 N が定められており，新しいサンプルの追加により上限値 N を超える場合は，最も古いサンプルと置き換える．

(4) 集団の更新
トレーニングセットは，サロゲートモデルの形成以外にも，進化の再出発点となりうる個体を保存する役割がある．SA-MBEANN では，進化が進むにつれてネットワークが大規模化し，解の探索が十分に行われる前に個体の次元数が増加する．これにより，モデルが予測可能な既知の解 D から遠ざかり，近似適応度 $\boldsymbol{y'}$ の精度が低くなる．そこで，SA-NEAT と同様にトレーニングセットの個体 $[\boldsymbol{x_1},\boldsymbol{x_2},\cdots,\boldsymbol{x_N}]$ を再び集団 P_t に戻すことで，個体の次元数を引き戻す．個体は，トレーニングセットに新しく追加されたものから順に選択される．これらの個体 $[\boldsymbol{x_{n+1}},\boldsymbol{x_{n+2}},\cdots,\boldsymbol{x_M}]$ と Infill Individuals $[\boldsymbol{x_1},\boldsymbol{x_2},\ldots,\boldsymbol{x_n}]$ で新しい集団 P_t が構成される．そして，手順 (2) に戻り進化が行われる．

(5) トレーニングセットの初期化
手順 (4) の終了後，新しい個体がトレーニングセットに十分に追加されているにもかかわらず，それまでに獲得した最高適応度 $y_{\max}$ が更新されない場合，トレーニングセットを初期化する．このとき，トレーニングセットを初期化するまでに追加する個体の数を Stagnation とよぶ．そして，この時点における集団全体を真の目的関数 $F(\boldsymbol{x})$ で評価し，

最高適応度 $y_{\max}$ より高い適応度を獲得しているにもかかわらず，近似適応度 $\boldsymbol{y'}$ が低い個体が集団の中に存在するかを確認する．存在する場合，手順 (2) に戻る．存在しない場合，最高適応度 $y_{\max}$ が更新されるまで，サロゲートモデルを用いない従来の MBEANN による進化が行われる．ここで，真の目的関数 $F(\boldsymbol{x})$ で評価された個体はすべてトレーニングセットに追加される．ただし，上限値 N を超えた場合は，最も古いサンプルと置き換える．最高適応度 $y_{\max}$ が更新された場合，手順 (2) に戻る．

5. 評価実験

本実験では，提案手法である SA-MBEANN を MBEANN, NEAT, SA-NEAT と比較する．ベンチマークとして OpenAI Gym[19] により提供されている CartPole-v0, HalfCheetah-v4, Ant-v4 を用いる．これらのタスクにおいて，より高い適応度を獲得できる ANN を進化的に設計する．NEAT, SA-NEAT は neat-python[20] を用いて実装し，SA-NEAT は文献 [10] を参考に作成した．

5.1 タスク設定

本節ではベンチマークタスクの設定について述べる．各タスクにおける目的，制御器への入出力情報，報酬設定について概説する．

5.1.1 CartPole-v0

CartPole-v0 の概観を Fig. 3 (a) に示す．本タスクでは, ポールが取り付けられた台車を左右に移動させ，ポールを所定の角度内に倒立させることを目的とする．制御器への入力は四つの情報からなり，それぞれ台車の位置と速度，ポールの角度と角速度である．これらの情報はタイムステップごとに取得される．制御器の出力は, 台車の左右方向どちらに力を加えるかを決定する．報酬は，ポールの角度 θ が $-\frac{\pi}{15} \leq \theta \leq \frac{\pi}{15}$ にあるとき，タイムステップごとに 1 与えられる．なお，ここでは OpenAI Gym におけるデフォルトの設定 [19] から以下の変更を加え，ポールの振り上げ問題とすることでタスクの難化を図った．

- ポールの角度 θ が $-\frac{\pi}{15} \leq \theta \leq \frac{\pi}{15}$ の範囲外でも終了しない．
- ポールの角度 θ の初期状態を $\theta = \pi$ を中心とした一様乱数とする．

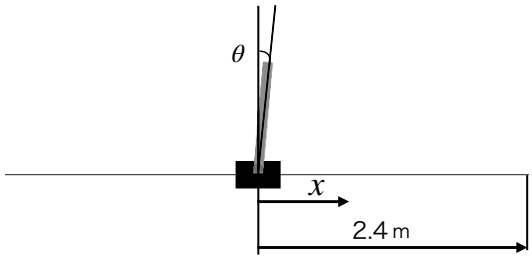
タスクの終了条件を以下に示す．

- 台車中心の x 座標が ±2.4 m を超過する．
- 200 タイムステップ経過する．

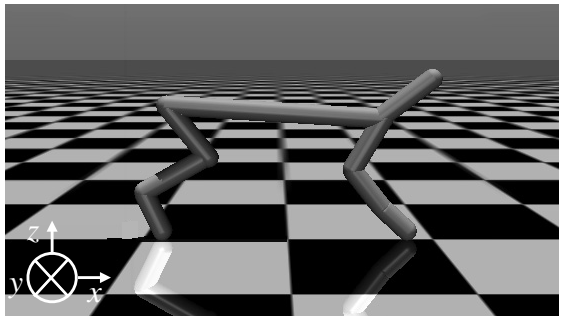
制御器の適応度として，獲得報酬の総和を用いる．

5.1.2 HalfCheetah-v4

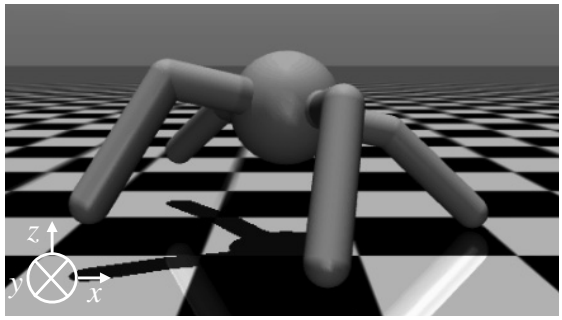
HalfCheetah-v4 の概観を Fig. 3 (b) に示す．HalfCheetah-v4 は 2 脚のロボットであり，各脚は 3 関節からなる．タスクにおける目的は，図中の x 軸方向にロボットを出来るだけ速く移動させることである．制御器への入力数は 17 であり，各関節の速度と角速度，頭部の位置（z 座標）と速度などから構成される [19]．制御器の出力は計六つであり，各関節にかかるトルクに対応する．報酬は以下の式で与えられる．



(a) CartPole-v0



(b) HalfCheetah-v4



(c) Ant-v4

Fig. 3 OpenAI Gym task environments

$$reward = forward - 0.1 ctrl\_cost \tag{19}$$

ここで $forward$ はロボットの速度から算出される．$ctrl\_cost$ は各関節のトルク値の二乗和であり, 罰則として作用する．タスクの終了条件は 1,000 タイムステップ経過することである．制御器の適応度は CartPole-v0 と同様に報酬の総和である．

5.1.3 Ant-v4

Ant-v4 の概観を Fig. 3 (c) に示す．Ant-v4 は 4 脚の歩行ロボットであり，各脚は 2 関節からなる．タスクの目的は HalfCheetah-v4 と同様に，図中の x 軸方向へ移

動させることである．制御器への入力数は 27 であり，各関節の速度と角速度，胴体中心の座標（z 座標）と姿勢角などから構成される [19]．制御器の出力は計八つであり，各関節にかかるトルクに対応する．報酬は以下の式で与えられる．

$$reward = healthy + forward - 0.5ctrl\_cost \quad (20)$$

$forward$ および $ctrl\_cost$ は HalfCheetah-v4 と同様に与えられる．$healthy$ はタスクの終了条件を満たさない間，タイムステップごとに与えられる報酬である．タスクの終了条件を以下に示す．

- 胴体中心の z 座標値が [0.2, 1.0] を超過する．
- 1,000 タイムステップ経過する．

制御器の適応度は獲得報酬の総和である．

5.2 アルゴリズム設定

本研究で使用した，NEAT, MBEANN, SA-NEAT, SA-MBEANN のパラメータを Tables 1∼4 に示す．NEAT および SA-NEAT は (μ,λ) 選択，エリート選択を用いる．MBEANN および SA-MBEANN はトーナメント選択を採用し，エリート選択は採用していない．隠れ層および出力層におけるバイアスのステップサイズは，シナプス結合荷重値におけるステップサイズと同じ値を用いる．Ant-v4 は CartPole-v0，HalfCheetah-v4 と比較すると困難なタスクであるため，集団サイズを 500 とした．すべての手法において，ノード付加突然変異率は 0.03，シナプス結合付加突然変異率は 0.3，シナプス結合およびバイアスの突然変異率は 1.0 と設定した．MBEANN と比較して NEAT および SA-NEAT には，特有のパラメータが存在する．MBEANN にないハイパーパラメータは，文献 [4] と neat-python[20] を参考にして設定した．なお，HalfCheetah-v4 および Ant-v4 では，初期世代からサロゲートモデルを構築した場合，局所解に陥ることが確認されたため，HalfCheetah-v4 はトレーニングセットを形成する世代数を $T = 20$ とし，Ant-v4 はトレーニングセットを形成する世代数を $T = 60$ とした．

5.3 実験結果

各手法を用いて 15 試行ずつ実験を行った．Fig. 4 に解評価回数に対する各試行の最高適応度 ($y_{\max}$) の 15 試行分の平均値の推移とその標準偏差を示す．ここで，サロゲートモデルによる近似解評価は，解評価回数に含まれていない．Fig. 4 から，CartPole-v0 では，すべての手法において，比較的類似した推移を示した．CartPole-v0 では，MBEANN が 20,000 回の解評価で獲得した適応度に，SA-MBEANN はおおよそ 6,600 回ほどの解評価で到達している．HalfCheetah-v4 では，おおよそ 4,800 回ほどの解評価で MBEANN が最終的に獲得した適応度に到達している．Ant-v4 では，おおよそ 30,000 回ほどの解評価で MBEANN が最終的に獲得した適応度に到達している．これらの結果から，SA-MBEANN は MBEANN よりも少ない解評価回数で同等以上の性能を獲得したことがわかる．

Table 1 Settings of NEAT (CP-v0: CartPole-v0, HC-v4: HalfCheetah-v4)

Parameter	CP-v0	HC-v4	Ant-v4
Population size	200	200	500
(μ,λ)	(40,198)	(40,198)	(100,498)
Elite size	2	2	2

Table 2 Settings of MBEANN

Parameter	CP-v0	HC-v4	Ant-v4
Population size	200	200	500
Tournament size	20	20	50

Table 3 Settings of SA-NEAT

Parameter	CP-v0	HC-v4	Ant-v4
Population size	200	200	500
(μ,λ)	(40,198)	(40,198)	(100,498)
Elite size	2	2	2
Number of infill individuals	10	20	20
Stagnation	200	200	200
Training set size	500	600	800

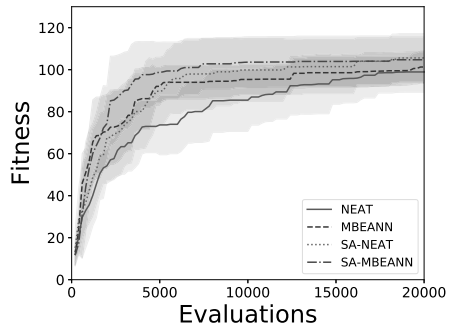
Table 4 Settings of SA-MBEANN

Parameter	CP-v0	HC-v4	Ant-v4
Population size	200	200	500
Tournament size	20	20	50
Number of infill individuals	10	20	20
Stagnation	200	200	200
Training set size	500	600	800
Initial step size	0.5	0.01	0.005
Step size	[0.01,5.0]	[0.001,0.1]	[0.001,0.1]

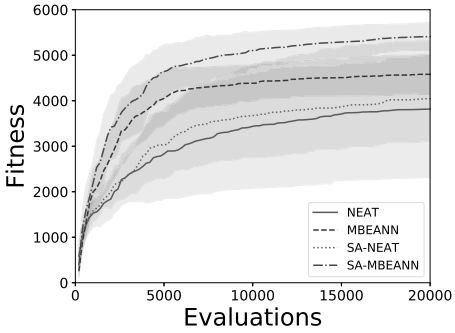
獲得した適応度に差がみられた HalfCheetah-v4 および Ant-v4 において，15 試行の各進化試行で記録された最も高い適応度を Fig. 5 に示す．各手法が獲得した最高適応度の比較のため，Kruskal-Wails 検定と Bonferroni 法で補正した Mann-Whitney の U 検定を有意水準 5% で行った．CartPole-v0 では有意差は確認されなかったが，HalfCheetah-v4 および Ant-v4 では有意差が確認され，SA-MBEANN が最も高い適応度を獲得した．これらの結果から，HalfCheetah-v4, Ant-v4 では，提案手法が有効であると考えられる．

6. 考察

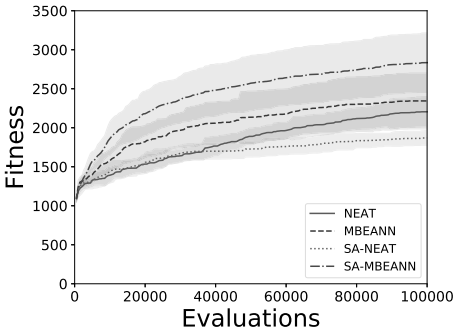
本章では獲得されたネットワーク構造に着目する．SA-MBEANN が優れた性能を示した Ant-v4 において，



(a) CartPole-v0

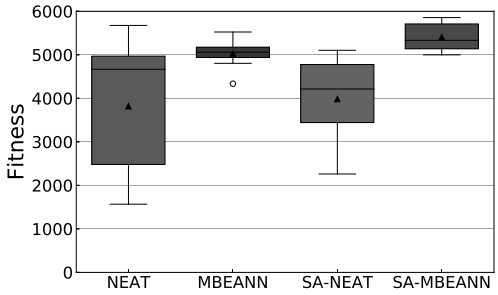


(b) HalfCheetah-v4

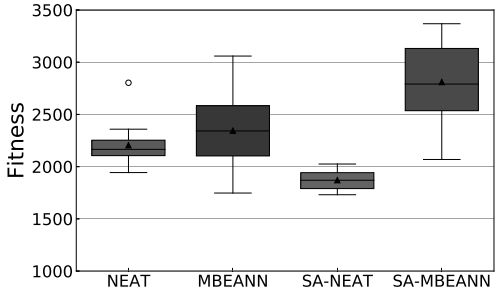


(c) Ant-v4

Fig. 4 Transition of the average fitness value of 15 trials



(a) HalfCheetah-v4



(b) Ant-v4

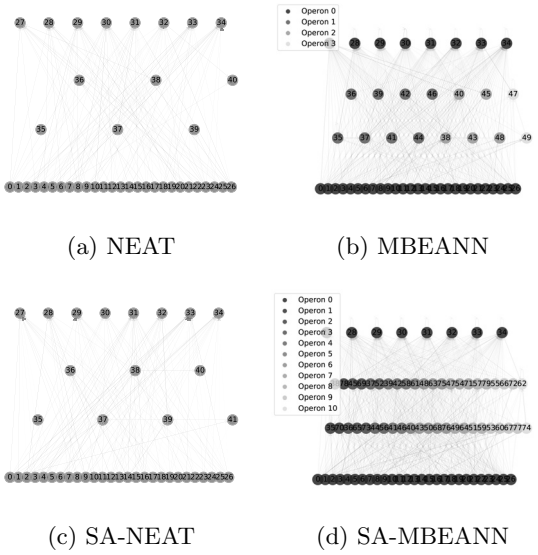
Fig. 5 Boxplots of the best fitness value from 15 trials

各アルゴリズムが獲得したネットワークの構造を Fig. 6 に示す．ここでは，最高適応度を獲得した進化試行において，最終世代におけるネットワーク構造を例示している．MBEANN, SA-MBEANN はそれぞれ NEAT, SA-NEAT と比較して多くのノード，リンクを有している．

解評価回数に対する 15 試行の各試行の平均ノード数および平均シナプス結合数の推移を Figs. 7, 8 に示す．HalfCheetah-v4 では，SA-MBEANN と MBEANN が獲得したノード数，シナプス結合数に大きな差は見られない．一方で Ant-v4 では，SA-MBEANN は MBEANN と比較してノード数には大きな差は見られないが，少ないシナプス結合数を獲得していることがわかる．これは MBEANN では，十分に探索が行われる前に構造が肥大化され，SA-MBEANN と比較して低い適応度を獲得したが，SA-MBEANN では，次元の引き戻しが有効に働いたためだと考えられる．MBEANN では，構造突然変異が operon ごとに働くため，評価回数が増加するにつれて operon 数が増加し，ネットワーク構造が肥大化しやすくなる．一方，SA-MBEANN では，過度にネットワーク構造が肥大化したとき，サロゲートモデルの予測範囲から遠ざかるため，近似適応度の精度が低くなる．このため，第 **4** 章の手順 (4) に示すように，トレーニングセットの個体を再び集団に戻すことで個体の次元数を引き戻している．この操作が SA-MBEANN において有効に働き，SA-MBEANN は MBEANN よりも少ないシナプス結合数を獲得したと考えられる．HalfCheetah-v4 では，MBEANN の肥大化ペースでも十分に解けたので，MBEANN と SA-MBEANN のノード数およびシナプス結合数に大きな差が出なかったと考えられる．

7. おわりに

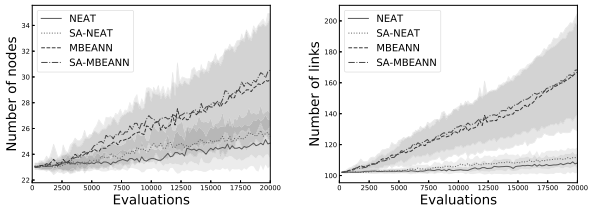
本研究では，自己適応型突然変異法を導入した MBEANN にサロゲートモデルを適用した SA-MBEANN を提案した．MBEANN に刷新番号を導入することで，A. Gaier らが提案した適合性距離カーネルを MBEANN において計算可能とした．SA-MBEANN は，NEAT と MBEANN に比べ，少ない解評価回数で同等以上の結果を得ることができた．また，HalfCheetah-v4 では，SA-MBEANN は MBEANN と比較して同程度の



(a) NEAT (b) MBEANN

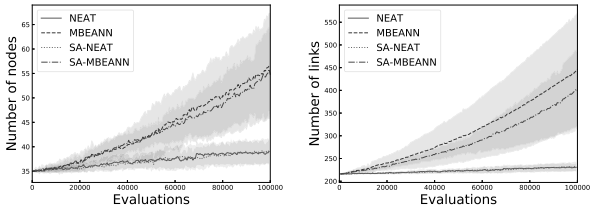
(c) SA-NEAT (d) SA-MBEANN

Fig. 6 Structure of ANN of each algorithm in Ant-v4. They are from the last generation in the best evolutionary trial.



(a) The number of nodes (b) The number of links

Fig. 7 Number of nodes and links of neural networks evolved in HalfCheetah-v4



(a) The number of nodes (b) The number of links

Fig. 8 Number of nodes and links of neural networks evolved in Ant-v4

平均シナプス結合数および平均ノード数を獲得しており，Ant-v4 では SA-MBEANN は MBEANN と比較して少ない平均シナプス結合数および同程度の平均ノード数を獲得していることがわかった．これは，MBEANN では，十分に探索が行われる前に構造が肥大化され SA-MBEANN と比較して低い適応度を獲得したのに対して，SA-MBEANN では次元の引き戻しが有効に働いたためだと考えられる．今回のアプローチでは，適合性距離によりカーネルを計算したが，ほかにも TWEANN に適用可能な距離尺度が考案されている [21]．今後は，MBEANN に適した距離尺度について調査したい．

参 考 文 献

[1] A. K. Jain, J. Mao and K. M. Mohiuddin: Artificial neural networks: A tutorial; *Computer*, Vol. 29, No. 3, pp. 31–44 (1996)

[2] X. Yao: Evolving artificial neural networks; *Proceedings of the IEEE*, Vol. 87, No. 9, pp. 1423–1447 (1999)

[3] K. O. Stanley, J. Clune, J. Lehman and R. Miikkulainen: Designing neural networks through neuroevolution; *Nature Machine Intelligence*, Vol. 1, No. 1, pp. 24–35 (2019)

[4] K. O. Stanley and R. Miikkulainen: Evolving neural networks through augmenting topologies; *Evolutionary Computation*, Vol. 10, No. 2, pp. 99–127 (2002)

[5] 平賀, 渡辺, 大倉: 二重倒立振子制御問題への TWEANN アプローチ—NEAT と MBEANN の特性比較; システム制御情報学会論文誌, Vol. 35, pp. 126–132 (2022)

[6] K. Ohkura, T. Yasuda, Y. Kawamatsu, Y. Matsumura and K. Ueda: MBEANN: Mutation-based evolving artificial neural networks; *Advances in Artificial Life, ECAL 2007*, pp. 936–945 (2007)

[7] T. Ohta: Slightly deleterious mutant substitutions in evolution; *Nature*, Vol. 246, pp. 96–98 (1973)

[8] J. Y. Li, Z. H. Zhan and J. Zhang: Evolutionary computation for expensive optimization: A survey; *Machine Intelligence Research*, Vol. 19, No. 1, pp. 3–23 (2022)

[9] Y. Jin: Surrogate-assisted evolutionary computation: Recent advances and future challenges; *Swarm and Evolutionary Computation*, Vol. 1, No. 2, pp. 61–70 (2011)

[10] A. Gaier, A. Asteroth and J. B. Mouret: Data-efficient neuroevolution with Kernel-based surrogate models; *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 85–92 (2018)

[11] A. E. Eiben and J. E. Smith: *Introduction to Evolutionary Computing*, Springer, pp. 56–60 (2015)

[12] D. Floreano, P. Dürr and C. Mattiussi: Neuroevolution: From architectures to learning; *Evolutionary Intelligence*, Vol. 1, No. 1, pp. 47–62 (2008)

[13] M. Hiraga and K. Ohkura: Topology and weight evolving artificial neural networks in cooperative transport by a robotic swarm; *Artificial Life and Robotics*, Vol. 27, No. 2, pp. 324–332 (2022)

[14] A. I. Forrester and A. J. Keane: Recent advances in surrogate-based optimization; *Progress in Aerospace Sciences*, Vol. 45, No. 1-3, pp. 50–79 (2009)

[15] 持橋, 大羽: ガウス過程と機械学習, 講談社 (2019)

[16] P. I. Frazier: A tutorial on Bayesian optimization; arXiv preprint, arXiv:1807.02811 (2018)

[17] N. Srinivas, A. Krause, S. Kakade and M. Seeger: Gaussian process optimization in the bandit setting:

No regret and experimental design; *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pp. 1015–1022 (2010)
[18] N. Hansen: The CMA evolution strategy: A comparing review; *Towards a new evolutionary computation: Advances in the estimation of distribution algorithms*, pp. 75–102 (2006)
[19] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang and W. Zaremba: OpenAI Gym; arXiv preprint, arXiv:1606.01540, `https://github.com/openai/gym` (2016)
[20] A. McIntyre, M. Kallada, C. G. Miguel, C. Feher de Silva and M. L. Netto: neat-python; `https://github.com/CodeReclaimers/neat-python`
[21] J. Stork, M. Zaefferer, T. Bartz-Beielstein and A. E. Eiben: Surrogate models for enhancing the efficiency of neuroevolution in reinforcement learning; *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 934–942 (2019)

著 者 略 歴

小村 真央 (学生会員)

2023 年 3 月広島大学工学部第一類（機械・輸送・材料・エネルギー系）卒業，同年 4 月広島大学大学院先進理工系科学研究科博士課程前期に進学し，現在に至る．

宮本 明治

2023 年 3 月広島大学工学部第一類（機械・輸送・材料・エネルギー系）卒業，同年 4 月広島大学大学院先進理工系科学研究科博士課程前期に進学し，現在に至る．

平賀 元彰 (正会員)

2022 年 3 月広島大学大学院工学研究科博士課程後期修了．2021 年 4 月日本学術振興会特別研究員 (DC2)，2022 年 4 月同研究員 (PD) へ資格変更．2023 年 4 月京都工芸繊維大学助教となり，現在に至る．スワームロボティクス，進化ロボティクスなどの研究に従事．博士（工学）．

森本 大智 (正会員)

2023 年 9 月広島大学大学院先進理工系科学研究科博士課程後期修了．2021 年 4 月日本学術振興会特別研究員 (DC1)，2023 年 10 月同研究員 (PD) へ資格変更．2024 年 4 月九州工業大学助教となり．現在に至る．博士（工学）．

大倉 和博 (正会員)

1990 年 3 月北海道大学大学院工学研究科情報工学専攻修士課程修了．同年 (株) 富士通研究所，1993 年神戸大学助手，1998 年英国サセックス大学客員研究員，2000 年神戸大学助教授，2006 年広島大学教授．EC, RL, MAS, SR, 生産システムなどの研究に従事．博士（工学）．日本機械学会，計測自動制御学会，IEEE などの会員．