2026 年 3 月 5 日

2025 年度　iTL 先端的プロジェクト奨学金

最終報告書

説明可能性を取り入れた人工知能開発手法についての考察

~NEAT アルゴリズムを利用した強化学習手法の提案~

中央大学　国際情報学部 4 年

氏名：岡本 和真

指導教員：須藤 修　教授

1. 課題設定

1.1 研究背景

第三次 AI ブームと呼ばれる現在では、ディープラーニングから端を発して様々な技術が開発されたことで、人工知能の性能は飛躍的に向上してきた。近年では、foundation model の開発によってより多くのタスクの実行が可能となり、OpenAI や Google などではマルチモーダル性が向上することによって、扱うことのできるデータの種類が格段に増加している。このような性能向上や利用可能範囲の拡大によって、企業での導入が進み社会全体においてもかなり普及が進んできた。一方で、現代の膨大なパラメータ数を持つ人工知能の活用にあたっては、説明可能性、倫理、バイアスなどの課題が指摘されており、ビジネスに人工知能を利用する企業にとっては、消費者の信頼を失いかねない深刻な問題となりうる。法的・技術的等の様々なアプローチからの解消が求められており、責任ある人工知能の開発・利用が国際的にも期待されている。実際に、「人工知能を利用したサービスや製品はデメリットよりもメリットの方が大きいか」と尋ねた調査では、国別に中国 83%、メキシコ 70%、インド 62%の回答者が、AI サービスや製品はメリットの方が大きいと回答しているのに対して、ヨーロッパや北アメリカなどの国々では同様の設問に対して日本 48%、イギリス 46%、オーストラリア 44%、カナダ 40%、アメリカ 39%の回答者のみに留まった[1]。この調査結果において、特に人工知能が生活に普及している先進国において人工知能への信頼度はかなり低いことがわかる。それにもかかわらず、より急速に人工知能は生活に溶け込んでいっており、人々の人工知能への信頼について今後さらに大きな課題となっていくことが予想される。

1.2 研究目的

そのような背景から、本研究では人工知能の信頼性に着目し、信頼性向上に関わる主要なテーマである説明可能性の向上を目的としている。また、生物学の分野において基礎的な考えとなっている「自然選択」という進化のメカニズムを模したアルゴリズムを活用することで従来とは異なるアプローチから責任ある人工知能の開発を目指し、調査・検討・考察を行っていく。具体的には、適合度と呼ばれる子孫のモデルの生存基準を変更することで、自然界における「環境」を規定することによるモデルの変化を調査する。説明可能性の指標及び出力の精度を向上させるように適合度を調整し、世代を経るごとにそれらの数値がどのように変化するかが調査対象となっている。今回は、NEAT アルゴリズムを利用した小規模なニューラルネットワークを対象として基本設計について検証を行う。

1.3 研究意義

本研究は現代で広く利用されている MoE アーキテクチャが今後極めて小さなエキスパートネットワークを100万個以上組み込んだモデルによって従来よりも優れた性能を実現するという Google Deepmind の先進的な先行研究をもとに、将来的に利用されるであろう小さなエキスパートネットワークをスコープに説明可能性を取り入れる強化学習手法を提案している。具体的には、従来精度向上を主な目的として利用されている NEAT アルゴリズムにおいて構造の複雑性を抑

制する選択圧をかけるために適合度関数を設定する方法により、説明可能性を同時に向上させる新たなアプローチを提示し、将来的な MoE モデルへの統合を視野に入れた概念実証としてスケーラブルな XAI 設計の可能性を示唆している。また、本提案手法は説明可能性に対する予防的アプローチの基礎研究として位置付けられる。本研究で対象としているトランスペアレント型 XAI による説明可能性の向上は企業等で利用される人工知能システムとして運用前にバイアスや誤動作の原因を追求することが可能となり、社会における人工知能の信頼性確保に対する要請に応えることができる。

2. 先行研究

2.1 説明可能性について

2.1.1 XAI (Explainable AI)

アメリカ国防総省は 2017 年から説明可能な人工知能に関する研究を行っている。このプロジェクトでは、ユーザがシステムの誤りを修正できるようにすることを目的とした説明可能性に関する研究を XAI と定義し、様々なアプローチから説明可能性の確保を目指した試みが行われている[2]。一般的に社会で広く利用されるシステムで誤った出力が行われた際には、原因を追求し再発防止のために修正を行ってからでなければ運用を再開することができない。他方で、人工知能の出力が誤っていた際に、その原因を突き止めて修正を行うことは非常に困難である。これは、人工知能では内部の構造が複雑でブラックボックスと呼ばれる状態にあるため、開発したモデルのニューラルネットワークを確認してもどのような理由でそのような出力が行われたのか把握することが難しいことに起因する。以上の理由から、社会で利用される人工知能において XAI 技術を用いて説明可能性を確保することが求められる。

恵木によると XAI には説明の目的、一般ユーザや研究者等の対象とする人物、微分可能性などの観点から非常に多くの種類がある。具体的には、学習済みのモデルの構築または出力が行われた後に説明するブラックボックス型、モデルの構造自体の解釈が可能であるトランスペアレント型に大別される[3]。前者はブラックボックスとなっているモデルに広く利用できる汎用性の高さからより一般的に研究されているが、後者には独自の利点が存在する。具体的には、差別や偏見などのバイアスのかかったデータを基に事前学習の行われたモデルでは、ニューラルネットワークの重みづけに影響することでモデルの出力にもバイアスが残存してしまう[4]。この際に、LIME[5]などのブラックボックス型の XAI 技術では事後的にニューラルネットワークのバイアスを発見することは可能なものの、内部構造から直接探ることはできないため、実際の運用上でバイアスのかかった出力をしてしまう可能性が存在する。一方でトランスペアレント型のモデルでは、事前学習データのバイアスに影響されていても、その解釈可能性から運用開始以前に内部構造の分析を行えば、事前に問題を発見することができる。このような利点から、今後トランスペアレント型のモデルの利用はさらに進んでいくことが予想されるため、本研究では社会的な需要を想定しモデル自体が解釈可能なトランスペアレント型として開発を行う。

2.1.2 出力精度と説明可能性の関係

深層学習では膨大なパラメータ数が寄与してより正確な出力に繋がっており、一般的には深層学習に限らずニューラルネットワークのノードやエッジの数を増加させて構造を複雑にすることでモデルの出力精度は向上する傾向にある。これは、モデルがより細やかに情報を処理できるようになることで入力値から正しい出力値を導きやすくなるためである。なお、学習量やモデルの規模、計算コストが増加するほどモデルの能力も向上することをスケーリング則と呼び、以上のようなパラメータ数の増加による精度向上もこれに該当する。一方で、モデルのエッジ数が増加するにつれて内部構造はブラックボックスに近づいていき、説明可能性は低下していく。これは、ニューラルネットワークが密になり、モデルの規模が拡大すると入力された情報がニューラルネットワーク内のどの経路を辿って出力ノードまで辿り着くのか、また各エッジにおける重みづけにどのような意味があるのか理解することが難しくなるからである。以上のように、モデルの表現力を向上させる一方で解釈を困難にさせるのは、ノードやエッジ、バイアスなどのパラメータと呼ばれるものであるが、本研究で前提としている細粒度化された MoE におけるエキスパートは非常に小規模な構成であり、パラメータのうち本論文で研究対象となっているノード及びエッジが非常に重要となる。それらの数量をなるべく小さくすることは、各エキスパートの説明可能性向上に直結し、それらを大量に組み合わせた MoE モデル全体の説明可能性向上にも寄与すると考えられる。なお、MoE の詳細は 2.6 にて説明する。以上のように、モデルの出力精度と説明可能性はトレードオフの関係にあるといえる。本研究でも例に漏れず両者は同様の関係にあるが、具体的な相関関係を明らかにし、可能な限り高水準な出力精度を維持しつつ、説明可能性を高めた最適なバランスを模索する。

2.2 Neuroevolution

本研究では、Neuroevolution と呼ばれる学問分野のアルゴリズムを利用している。生物において、子孫に受け渡される情報として非常に重要な役割を果たすのが遺伝子であり、その遺伝子の振る舞いを模して、変異等によって徐々にモデルを適応させていくことが遺伝的アルゴリズムの特徴である。具体的には、DNA の突然変異、逆位、交叉などの特徴が応用されており、1970 年代から利用されこれまで様々な分野で応用がなされてきた[6]。伊庭によると、Neuro evolution は人工知能のニューラルネットワークを構築する遺伝的アルゴリズムの総称である[7]。

Neuroevolution を実際に活用した事例として次のものが挙げられる。JAXA(宇宙航空研究開発機構)が自動車メーカーのマツダと共同研究を行った事例[8]では、複数車種における部品の共通化を目的として、「構造重量の最小化」、「および同一の厚みをもつ板で成形された自動車の組み立てパーツ数の最大化」をそれぞれ目的関数として多目的最適化が行われた。衝突安全性などの多くの制約条件があり、難易度の高い多目的最適化であったが、実際の車両構造設計よりも優れた設計を複数得ることができ、その中には大幅に重量を削減できるものや、重量を削減しつつ共通部品点数を向上させることのできる設計も含まれていた。なお、適合度の算出に必要な構造解析にはスーパーコンピュータの「京」が計算資源として利用され、設計最適化には十分な計算量が必要となることがわかるが、従来専門家が時間をかけても成し遂げられなかった網羅的な最適解の探索が可能となるという大きな意義が存在する。Neuroevolution アルゴリズムが利用された、この

例では、同分野のアルゴリズムが制約条件下における組み合わせ問題の解決能力の高さを窺い知ることが出来る。また、Neuroevolution における各アルゴリズムは強化学習に分類される。

2.2.1 アルゴリズムの基本的な流れ

NeuroEvolution に該当するアルゴリズムや遺伝的アルゴリズムでは次の流れで探索が行われる。初めに、初期集団として適当に個体群が生成される。それらの個体について評価が行われたのち、その結果をもとに優秀な個体は次の世代へ引き継がれ、引き継がれない個体は淘汰されていく。実際には、優秀な個体は親として、別の優秀な個体と特徴を次の世代の個体に受け渡し、ここで突然変異を模した新たな特徴の出現等の変異も行われる。このような作業により、次の世代の個体群が形成され、再び各個体の評価をもとに選別が行われるという流れが繰り返されていく。なお、初期個体の生成や各変異の内容等はランダム性に左右される。そのようなランダム性に左右されながらも、世代を経るごとに徐々に環境に適応した個体となっていく点が遺伝的アルゴリズムの最大の特徴であり、以上のようなプロセスが絶えず繰り返されてきた自然界では生物種の進化の根幹を担っている。

2.3 NEAT

Neuroevolution における主要なアルゴリズムの一つとして NEAT が挙げられる。Stanley によると、NEAT(Neuro Evolution of Augmented Topology)は人工知能のニューラルネットワークを進化的に構築するもので、小さなネットワークを世代ごとに複雑化させていくことができる[9]。具体的には、ニューラルネットワークの重みづけのみではなく、構造(ノードの数)も適応的に変化させられる。この特徴により事前にニューラルネットワークの構造を設定する必要がなくなる。これは、深層学習で一般的なバックプロパゲーションとの大きな違いとなっている。このアルゴリズムでは前述のように世代を経て変異や淘汰を繰り返していくことで学習データに適応したモデルとなっていく。また、同時に複数の個体が用意され、その間で自然選択が行われていくことで、優秀な個体の特徴が引き継がれつつ、さらなる好適な変異を持つ個体は適合度が上昇していく。ただし、変異にはランダム性が寄与しているため、各試行によって構築されるモデルの精度や構造の特徴が大きく異なることがある。

また、NEAT をさらに発展させてより大規模なニューラルネットワークを構築できるなどの特徴を持つ HyperNEAT も存在するなど様々な派生研究が進められており、画像認識などの対象とする問題によって多種多様なアルゴリズムに改良されている[10]。後述の自動運転シミュレータにおける研究では多目的最適化が試みられるなど、様々な性質が組み合わせた応用がなされている。Evgenia Papavasileiou によると、NEAT は近年盛んに研究され、ゲーム AI などに応用されている[11]。本研究でも強化学習テスト環境として提供されているシミュレーション問題を扱う。この研究では説明可能性の指標を適合度関数という NEAT の評価段階に取り入れることで説明可能性の向上を目指す。この関数の出力値が適合度であり、適応度が高いモデルは生き残り、別の優秀な個体とともにその性質を引き継いでゆく。具体的には、説明可能性がより高いと評価されたモデ

ルが有利に生き残り、子孫となるモデルを残す。そうでないモデルは淘汰されていくように適応度関数を設計する。

2.4 NEAT を用いた研究例

2.4.1 自動運転車の走行シミュレーション

Willigen の行った研究では、NEAT を利用して自動運転車の道路上での動きをコンピュータ上でシミュレーションしている[12]。構築されたニューラルネットワークは車のコントローラとして働き、速度の調整や車線変更などの操作を行う。シミュレーション環境には他の車も用意されており、より早く目的地に到着するためには頻繁に車線変更を行う必要がある。

2.4.2 多目的最適化

通常の NEAT を利用した研究では単一の指標について向上するように設計を行うが、この先行研究では多目的最適化を実施している。具体的には、問題解決能力を向上させる目的で一般的に用いられる「テストスコア指標」に加えて、「乗り心地」も同時に向上させられるように設計されており、本研究でも同様に、NEAT を用いた多目的最適化を行い、「テストスコア」と「説明可能性」を同時に向上させていくように設計する。この自動運転シミュレータの研究のように、NEAT アルゴリズムを用いた多目的最適化を実施している先行研究は複数存在するものの、説明可能性の向上を目的とした多目的最適化を行なっている例は存在しておらず、これらを組み合わせたアプローチを採用している点が本研究の独自性である。

また、前述の先行研究では主観的な評価である「乗り心地」を測る代理指標として、定量的に測定可能な「車線変更の回数」が利用されている。同様に、本研究では「ノード及びエッジの数」を説明可能性の代理指標として用いる。なお、第 2 章 1 節で述べたとおり、ニューラルネットワークにおけるノード数およびエッジ数の削減には、説明可能性を向上させる効果がある。

2.4.3 多目的進化アルゴリズム

先行研究では多目的進化アルゴリズムである SPEA2 (The Strength Pareto Evolutionary Algorithm 2)[13]を NEAT の適合度関数に統合することで多目的最適化が実現されており、多目的進化アルゴリズムによって複数の目的関数を同時に扱う多目的最適化が可能となっている。また、SPEA2 と同一の目的で利用されるものに、多目的進化アルゴリズムの NSGA-Ⅱ (nondominated sorting genetic algorithm II)[14]がある。これらは、前者が遺伝的アルゴリズムの各世代における個体選別の基準に各個体に特定の適合度を計算したものを用いるのに対し、後者は集団内の順位に基づいて評価を行うという点で異なる。多目的進化アルゴリズムを利用することで、複数の目的関数を同時に向上させることが可能となり、複数のパレート最適解を一度に導くことができる。パレート最適とは、トレードオフにある関係の指標のどちらも現状より良い解が存在しないことを指し、二つの目的関数を最大化する多目的最適化においては、どちらかの指標を下げなければもう一方の指標がより良い値を持つ解に辿り着くことのできない状態にあることを意味する[15]。本研究を例に挙げれば、ある「テストスコア」において最も少ない「ノードおよびエッジの

数」を持つこと、またはある「ノードおよびエッジの数」において、最も高い「テストスコア」を持つ状態のことをいう。先行研究では、それらのパレート最適解のうち、トレードオフの関係にあるどちらの指標もバランスよく両立されている一部のパレート最適解を取り上げ、評価している。ここでは、一台のコントローラとしてではなく、他車と共に隊列走行を行うことにより、到着に要する時間、快適性の2つの指標でともに人間を模したモデルよりも優れた結果が得られた。

2.4.4 スカラー化

本研究も同様の目的で多目的最適化を採用する一方で、先行研究では同一モデルによるシミュレーションの複数回実行時におけるスコアの不安定性が示唆されている。
先行研究で用いられた方法では SPEA2 によるパレート最適解は 30 回の実行結果から抽出されているものの、コントローラとして再度評価を行うと、より低いテストスコアとなってしまうことが多かった。統一した条件において複数回実行したデータから、分散等の統計量を算出し、適切に検定を行なった結果を採用する方法が望ましいと考え、本研究では多目的最適化のうち、多目的進化アルゴリズムの代替手段として「スカラー化」という手法を採用している。スカラー化は、複数の目的関数を1つの目的関数に数学的に変換する手法である。目的関数の数等に合わせていくつかの種類に分かれるが、今回は「テストスコア」という目的関数をなるべく高く維持しながら、説明可能性を向上させていくという性質は優先すべき目的関数が明確で定式化が比較的容易であることから、加重方式によるスカラー化を採用する。加重方式では、複数の目的関数から作成した重み付き線形和を最小化するように設計するが、今回はテストスコアの最大化およびノードおよびエッジの総数の最小化を NEAT における適合度を最大化させるプロセスで実現するため、「テストスコア」から「ノードおよびエッジの数」を減算した値を適合度とし、これが上昇していくことで実質的な加重方式を実現している。なお、具体的な設計については、次章にて詳述する。

2.5 Chain of Thought

本研究では細かな機能に特化したモジュールを作成する際に説明可能性を向上させるための手法を提案しているが、機能に特化したモジュールを利用する例として Chain of Thought が挙げられる。Chain of Thought という言葉は主にユーザがプロンプトを作成する際に思考のステップを明示するといった工夫を行うといった用例で広く使われるが、大規模言語モデル等の人工知能システムの内部設計として指示を分解して思考や検索または画像生成などの行動を段階的に行うように設計する方法として利用されるというもう一つの意味が存在する。本節では、後者に該当する内容を取り上げる。以前の大規模言語モデルでは、分子構造やテキストの言語、学問分野ごとに分類され、異なるモジュールが利用されることでモデルの汎用化及び効率化が可能となっていた。現在の大規模言語モデルでは公開されている情報が限られており、その詳細な設計は明らかにはなっていないものの、現在でも同様の仕組みが用いられることで更なるモデルの汎用化および大規模化が可能になっていると推察される。

Google の提供する最新の大規模言語モデルである Gemini 2.5 及び 3.0 では、同サービスの API 利用時に「Thinking Budget」という変数が用意されている[16]。これはプロンプトを受けて回答を生成する際に、計算資源的、時間的なコストを指すリソースをどの程度かけて論理的構築を行うか設定するための変数である。各モジュールの結果を組み合わせた回答をする上で論理的な妥当性や整合性を向上させる効果があり、実質的な Chain of thought の取り組みの一環として捉えられる。最先端の大規模言語モデルでも以上のように Chain of Thought の考えが利用されているが、技術的には MoE という方法で実現されている[17][18]。

2.6 MoE (Mixture of Experts)

本研究では、NEAT アルゴリズムを用いていることで小規模なニューラルネットワークの開発に限定されてしまうという課題が存在するが、モデル同士を結合することでより大きな全体のモデルを構成する先行研究が存在する。Robert A. Jacobs によると、細かなタスクに特化したモデルである「エキスパートネットワーク」を個別にトレーニングし、入力に応じて各ネットワークに処理を振り分けるような動作をすることで、学習の効率化が可能になる[19]。この MoE (Mixture of Experts) という設計思想は、パラメータ数の削減効果などがあり、数々の研究を経て現在では特に自然言語処理の分野で目立ったパフォーマンスを発揮し、GPT-4 などの主要な大規模言語モデルにも採用されている 16 [20]。実際に「Gemini 2.5 シリーズ[2]」及び「Gemini 3.0 シリーズ[3]」において MoE モデルがアーキテクチャ[4]として採用されている[16][17][18]。各能力に対するベンチマークスコアで比較すると、「Gemini 1.5 Pro」では、推論能力：58.1%，数学：17.5%であったのに対して、「Gemini 2.5 Pro」は推論能力：86.4%，数学：88.0%と高いスコアを記録しており、開発チームの発表資料では事前学習の安定性向上が各能力の進歩に寄与したと結論づけられており、同じ MoE アーキテクチャを利用していながらこのよう性能向上に至ったのはこの分野における研究の進展を表し、今後もますます広く利用されるようになると考えられる[16]。

近年では、MoE における「エキスパートネットワーク」をより小規模に、より多数組み込むことを意味する細粒度化によりモデルの計算コストを維持しつつも性能向上が見込まれるという新たな知見をもとに研究が進められている。本研究で扱うニューラルネットワークと従来の MoE モデルに利用される「エキスパートネットワーク」を比較すれば、その規模に大きな差が存在するといった課題が存在する一方で、細粒度化により単一のノードのみを持つエキスパートを多数組み込むことで計算効率を維持しつつ、モデル全体の表現力向上が可能になるといった Google Deepmind による先行研究が存在する。この研究では、従来の MoE において最大で 128 個に限られていたエキスパ

---

[2] 「Gemini 3 Flash」，「Gemini 3 Pro Image」，「Gemini 3 Pro」を含む

[3] 「Gemini 2.5 Computer Use」，「Gemini 2.5 Deep Think」，「Gemini 2.5 Flash-Lite」，「Gemini 2.5 Flash and Gemini 2.5 Flash Image」，「Gemini 2.5 Pro」を含む

[4] モデルの構造設計を指す

ートネットワークを、この研究ではどのエキスパートネットワークを利用するか判断するためのルーティングを工夫することにより 100 万個以上にまで増加させることができることが明らかとなった[21]。従来の手法では、エキスパートネットワークの数に対して線形の計算コストがかかったのに対して PEER（Parameter Efficient Expert Retrieval）と呼ばれる学習可能なインデックス構造を採用することでここまで大胆な細粒度化を実現している。この研究では、モデルのサイズと学習データ量が増加するにつれてモデルの性能も向上していくことを示すスケーリング則に基づき、従来のアプローチから計算コストを大きく増やすことなくモデルのサイズを拡大させていくことが可能になることで、モデルの汎用性拡大や精度向上等が期待できる。実際に、利用できる計算コストの条件を統一して行った検証では MoE およびトランスフォーマーにおける従来の方法に比べて本アプローチを利用することでより高い精度であることが確認された。この手法のような細粒度化は DeepSeek 等で利用されており、Enric によってその有効性が確認されている[22]。一方で、モデル全体の中で入力に対して活性化するエキスパートの数を増やすことでそれらの再利用が発生し、効率的に対応できるタスクの幅が拡大するものの、活性化するエキスパート数の増加によって複雑なルーティングが強いられ、学習や推論に要する時間が長くなりすぎてしまうという欠点が指摘されており、それらの最適なバランスを保った設計が今後必要となると結論づけられている[22]。
さらに、大規模なモデル全体をまとめて学習させると不必要なエッジが増え、必要以上に複雑なモデルとなってしまう可能性が考えられる。また、不必要なエッジにより事前学習がスムーズに行われずに学習コストが増加するリスクも存在する。一方で、小規模なモデルを結合させてゆくことで、全体のエッジ数が減少し、結合後のモデル全体の構造の複雑性の低減が期待できる。小規模なニューラルネットワークにおいて、説明可能性の差異は大規模なものと比較して相対的に小さくなるが、以上のようなアプローチでは、説明可能性を担保しつつ、モデルの規模を大きくしていくことができるのではないか。ある機能に特化したモジュールとして開発できれば将来的に MoE を用いてそれらを結合させていくことができるのではないかと考えられる。以降の章では、MoE において利用されるごく小規模なエキスパートの開発を念頭に、多目的最適化における提案手法の有効性を確認する。ただし、実装難易度の高さから、本研究における MoE の具体的な実装は行なっていない。

3. 研究

3.1 検証タスク

本研究における検証では、以下の2つのタスクを用いる。検証では第一に、従来手法と、提案手法における構築されたモデル構造の違いを確認し、提案手法が有効に機能しているかどうかを検証する目的がある。加えて、性能と構造のシンプルさに関するトレードオフの関係について確認する。以上の理由からこれらのタスクを選定した。

3.1.1 XOR

入力 A	入力 B	出力 (A XOR B)
0	0	0
0	1	1
1	0	1
1	1	0

図 1: XOR の入出力パターン

上図のような4種類の入力パターンから、右の列のように出力が行われるかテストする問題であり、全て正しく出力するためにはニューラルネットワークにおいて、入力層及び出力層に加えて隠れ層が必要となることが特徴である[23]。

3.1.2 Lunar Lander

より複雑な制御問題として、OpenAI の提供する強化学習用テスト環境の「Lunar Lander」を採用した。これは月面に着陸する宇宙船をコントロールするシミュレーションであり、着陸船の座標や角度などリアルタイムの状態を入力値として取得し、出力値として左右および下方向のエンジンの噴射をコントロールして学習を行う[24]。このタスクでは、モデルのシミュレーション結果から機体の安定性や落下速度、着地地点に応じてテストスコアが算出されるが、280 点以上でかなり優れた能力を示しているといえる。

3.2 研究方法

本研究の実装にあたり、GitHub 上で公開されている既存のライブラリである neat-python[25]を使用する。本研究独自のアプローチとして、適応度関数においてモデルの複雑さを抑制するための変数を導入した。具体的には、ニューラルネットワーク内における「ノードの数」を複雑さの指標とし、これをペナルティとして問題に対するテストスコアから減算したものを適応度とする手法を採用した。このペナルティが進化に与える影響を検証するため、ペナルティの重さとなる係数を調整し、各条件で 15 回ずつ試行を行った。また、実行結果のデータの検定及び分析を行う。

この検証設定によって、ペナルティの係数を大きく設定した場合、モデルの複雑さを抑制する選択圧がより強くかかるため、テストスコアの最大化よりも構造のシンプルさが優先されると考えられる。反対に、ペナルティの係数を小さく設定した場合、テストスコアを重視する選択圧がはたらき、結果としてより複雑な構造を持つモデルが構築されると推察できる。前節で述べた「XOR」および「Lunar Lander」のタスクを用いて、検証を行う。なお、いずれのタスクも neat-python ライブラリに対応したコードが公開されており、これらのコードを適宜改良して検証を行う。

3.2.1 適合度関数におけるペナルティの導入

NEAT では、適合度関数の設定することで、どのようなモデルが構築されるようになるかコントロールすることができる。今回は「XOR」および「Lunar Lander」のタスクでそれぞれ適合度関数を改変してペナルティを導入した。「XOR」において適合度は、各個体の出力の正誤結果のスコアから隠れノードとエッジの個数を減算したものとして規定した。他方で、「Lunar Lander」においてはタスクがより複雑なものとなることから、きめ細やかな条件設定が可能となるようにノード、エッジそれぞれのペナルティの重みづけを別々に指定できるように改変した。具体的には新たに、入力および出力ノード以外を隠れノードとしてその数をカウントするノード数と、ノード間を繋ぐ有効なエッジの総数をカウントするエッジ数の変数を用意し、テストスコアからそれぞれの変数の値に任意の数値を掛け合わせたものを掛け合わせたものを減算して適合度とする。掛け合わせる任意の数値はペナルティの重みづけとして機能し、各数値が大きくなるほど隠れノードとエッジがそれぞれ減少するように選択圧が働く。なお、XOR においてペナルティはノード、エッジそれぞれ1つにつき適合度が 1 下がるように設定しており、Lunar Lander における具体的なペナルティ係数の設定は次に説明する。

3.2.2 Lunar Lander におけるペナルティ係数の選定

Lunar Lander は、XOR と異なりそのタスクの複雑性からペナルティがより柔軟なものとなるようにノード及びエッジそれぞれに対する個別の重みづけが可能な設計を採用している。この項では、適合度関数において重みづけにあたるペナルティ係数の選定基準及びその経緯について説明する。ペナルティ係数はその設定に応じて、構築されるモデルの構造は多様に変化することが期待される。また、どのような設定からどのようなモデルが構築されるかを明らかにするためには本来、各ペナルティ設定を網羅的に検証実施すべきであるが、各実行に要する時間を考慮するとあまり多くの回数を実行することはできないため、以上のように満遍なくデータを収集することは叶わなかった。本研究では、限定的な計算コストを鑑み、ペナルティー係数を 7 通り設定した上で各条件における検証を複数回実行したデータを分析する。なお、本検証にあたり、事前に予備調査を実施し、適切なペナルティ係数設定を探った。予備調査では通常の NEAT による複数回の実行結果から構築されたモデルのニューラルネットワークを観察し、そのノード数およびエッジ数からペナルティ係数を適当に設定した。実際には、ノード: 5、エッジ: 1 として 5 回実行し、この予備調査の結果をもとに、より広範囲にペナルティ係数が客観的な設定となるよう意識して設計を行なった。

初めに、ノードとエッジのペナルティの重みづけ比率を固定し、それらを定数倍した 3 つの条件を設定した。この設定では予備調査よりも重いペナルティと軽いペナルティのどちらも存在するように設計し、(ノードのペナルティ係数, エッジのペナルティ係数)は (1.5, 1)、(3, 2)、(4.5, 3)と規定した。さらに、過度に重みづけを行った場合について、前出のものよりも大きなペナルティ係数としてノード:10、エッジ 5 の条件で検証している。さらに、(1.5, 2)、(4.5, 2) を追加することで、等間隔の重みづけや、一定のエッジの重みづけにおけるノードの重みづけの差異による結果の比較

を行うことのできるように設定した。なお、本検証で得たモデルに関するデータは、本論文の次節以降[5]に示している。

3.2.3 各タスクにおける条件設定

本研究では、広く利用されている neat-python ライブラリにて公開されている上記の二つのタスクに対応したコードをそれぞれ利用する。これらは NEAT アルゴリズムにおける標準的な条件設定となっているため、可能な限り設定を改変せず研究の客観性を維持する。一方で、Lunar Lander における予備調査ではこの条件において、構築されたモデルの結果に対するばらつきが大きいといった課題が存在した。また、2.4 で述べた先行研究においても、実行時と検証時でスコアに差異があることが示唆されており、本研究では検証結果の正確性を確保するために該当するタスクにおける以下のような条件の変更を行った。なお、重力値やランダムに出現する風等のシミュレーション環境における条件変更は一切行っていない。

初めに、適合度が200点のままでは機体がふらついたり、地面に衝突するシミュレーション結果を示すモデルでも、数十世代で終了条件に到達していたため終了条件の厳格化を実施した。具体的には「fitness_threshold」変数を既定値の 200 から 280 に上昇させることで、適合度が 280 点以上になった場合に探索が終了するように変更した。この変更により、探索の上限となる世代数の既定値では終了条件に達しない事例が多く確認されたため、この条件を設定する引数を 200 増加させて 500 世代とした。なお、この時点で従来の NEAT を用いた手法と、今回提案するペナルティ付き多目的最適化手法のいずれも調査を行なっており、前者は終了条件を満たすようになり、後者は 500 世代完了時に終了条件に達しない適合度であった。本調査における詳細なデータや各手法の比較は次節にて示す。

以上の変更により従来よりも適合度が改善されたものの、世代を経る中で適合度が上がるだけではなく、下がる動きが見られた。通常探索の途中で適合度が下がりつつもアーキテクチャが変化し局所最適解に陥るのを防ぐことができるが、この例ではあまりに多く適合度が下がっていく動きがみられた。これ評価回数が少ないことによって優秀な個体の評価が正しくできていないことに起因している可能性が考えられるため、同一個体における評価回数を意味する「runs_per_net」変数を 5 回から 10 回に増加させた。この変更により、探索途中のモデルの集合全体における平均値が世代を経るごとに着実に上昇していき、学習がうまく進むようになった。他方で、評価回数の増加に伴い、より厳密に適合度が算出され、適合度が低く出やすくなるようになった。これはモデルの正確な性能を表しているものの、通常の NEAT によるアプローチでも再び終了条件に達しないモデルが出現したため、タスクの難易度が上がったことによる探索範囲の不足を解消することを目的として改良を実施した。具体的には、各世代における個体数を表す「pop_size」変数を 150 から 300 にし、計算コストを増大させて探索空間の拡大を実施した。さらに、各世代の最上位の個体を無条件に次世代へ保護する個体数を規定する「elitism」変数を2から 5 に増加させることで、各世代の優秀な個体を次の世代へ残りやすくするとともに、モデルの特徴から異なる種として管理する

[5] 末尾の補足資料を含む

「compatibility_threshold」を 3 から5に変更し、同一種とみなされやすくすることでその種内における進化を促し、適合度の上昇を図った。これらの改良によって基準となる通常の NEAT 手法において規定の 500 世代に達する以前に終了条件を満たす傾向を確認した。上で述べた条件設定は検証における両手法において統一し、厳密に一致している。
以上がライブラリで用意されているソースコードを改変した全箇所に関する説明である。

3.3 構築されたモデルの分析

本節では、実際に検証を行い構築されたモデルの構造について幾つか取り上げて紹介する。

3.3.1 XOR

初めに、通常の NEAT 手法を用いた実行結果を示す。以下の図は XOR タスクにおいて構築されたモデルのニューラルネットワークを表したものである。

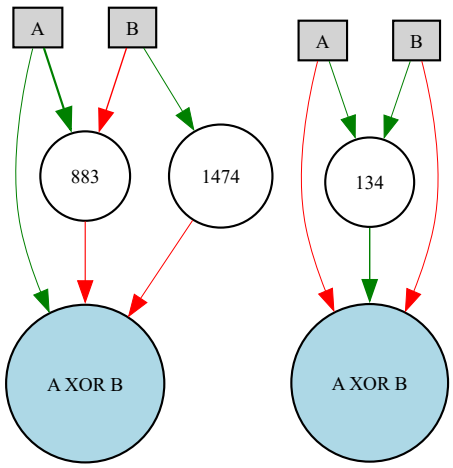


図 2　XOR: 通常の NEAT の実行結果 1, 2

図において、灰色のボックスが入力ノード、水色の円が出力ノードを表しており、白い円が今回追加された隠れノードを、矢印はエッジを意味している。なおノード内の数字は 1 回の実行において何番目に作成されたノードであるかを示す ID 番号であり、重みづけは表示されていない。なお、以降の XOR タスクにおける図では各図形や記号は同じ意味を表す。

上の図では、左右それぞれが実行結果のモデルであり、どちらも隠れノードが存在することがわかる。これらのモデルでは 4 つの入力パターンに対していずれも正しい値を出力することができたが、適切に隠れノードが存在することがその要因である。この図は以降の構造の複雑さを抑制したモデルに対する基準となる。
次に、ノードとエッジの総数に 0.01 の重みづけを掛け合わせた非常に弱いペナルティを導入した実行結果を示す。

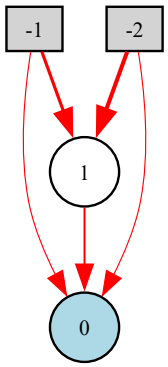


図 3　XOR: 弱いペナルティ付き NEAT の実行結果

上のモデルでは、図2の結果とあまり変わらないニューラルネットワークを持っている。この条件では、ペナルティの効果が限定的で、ペナルティ係数が 0.001 では通常の NEAT 手法と大差ないことがわかる。
同様に、係数が 0.3 のより強いペナルティを課した実行結果を示す。

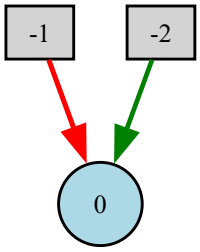


図 4　XOR: 強いペナルティ付き NEAT の実行結果

図4では隠れノードがなくなり、エッジ数も少なくなっていることがわかる。これは各入力値を利用する最小のネットワーク構造であり、図3の条件よりもペナルティが強くなっていることから、このようなか結果が生まれた。本提案手法を採用した上の二つの図からは、ペナルティ係数の差異により構築されるニューラルネットワークの変化が見て取れ、この手法が有効に機能していることがわかる。一方でどちらも 4 パターン中 3 パターンのみの正答にとどまっている。図4のモデルでは隠れノードが存在しないため、全問正答には隠れノードが必要な XOR 問題について完全に解くことはできないものの、図3のモデルでは隠れノードが存在し、全問正答を果たしている図 2 のモデルと同じアーキテクチャを持っているにも関わらず、このような結果に至ったのは、エッジの重みづけの学習がうまくいかなったのが原因と考えられるが、適合度関数におけるペナルティの導入が学習に影響している可能性も考えられる。

3.3.2 Lunar Lander - 通常の NEAT

本項では、Lunar Lander タスクにおけるモデルを条件別にいくつか取り上げる。まず初めに、通常の NEAT 手法を用いた実行結果を示す。なお、以降では明瞭性確保のため構築されたモデル

の適合度を小数点以下第二位で四捨五入している。ただし、グラフにおけるデータはこの限りではない。

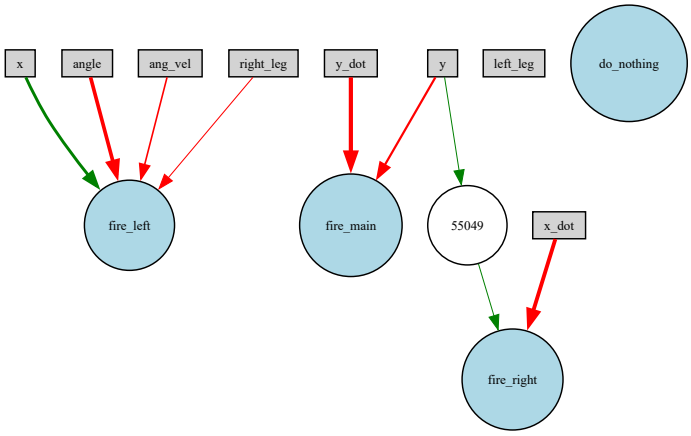


図 5　Lunar Lander: 通常の NEAT の実行結果 1

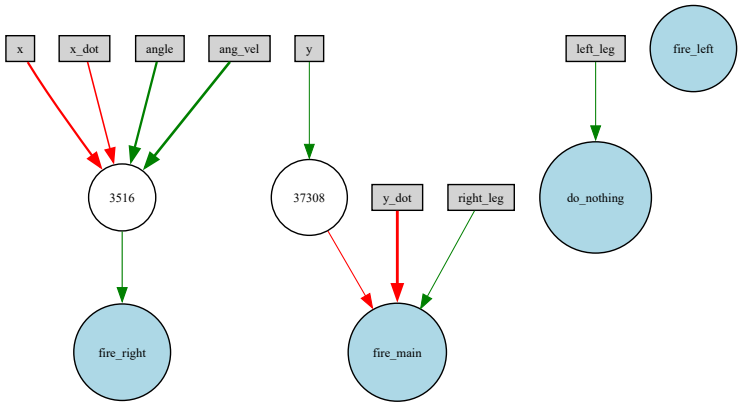


図 6　Lunar Lander: 通常の NEAT の実行結果 2

図において、灰色のボックスが入力ノード、水色の円が出力ノード(行動の選択肢)を表しており、白い円が今回追加された隠れノードを、矢印はエッジを意味している。なおノード内の数字は 1 回の実行において何番目に作成されたノードであるかを示す ID 番号であり、重みづけは表示されていない。なお、以降の Lunar Lander タスクにおける図では各図形や記号は同じ意味を表す。

2つの図からは、どちらも隠れノードが存在し、エッジが 9 本または 10 本存在することが見て取れる。テストスコアはそれぞれ図5のモデルが 280.2、図6のモデルが 286.3で、いずれも 200 世代までに終了条件に達しており、順調に学習が進んだことがわかる。

3.3.3 Lunar Lander – ペナルティ付き NEAT

ペナルティを導入した手法により構築されたモデルのニューラルネットワーク図は以下の通り。

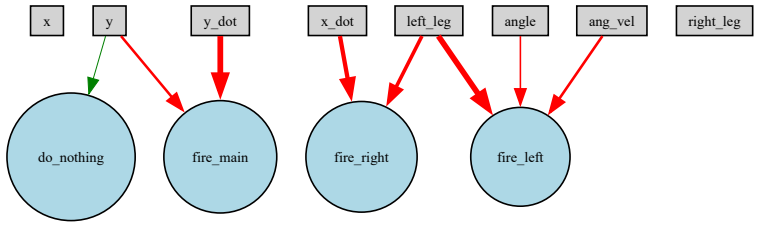


図 7　Lunar Lander:　NEAT の実行結果 1 (node: 5, edge: 1)

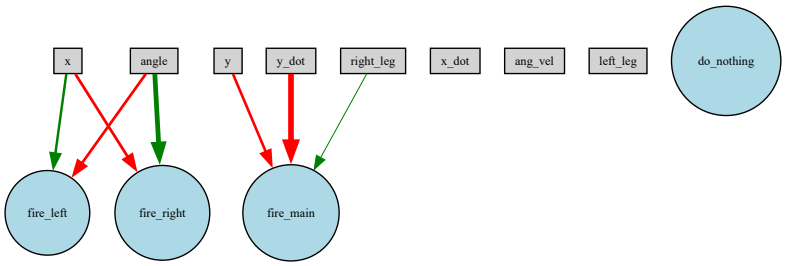


図 8　Lunar Lander:　ペナルティ付き NEAT の実行結果 2(node: 5, edge: 1)

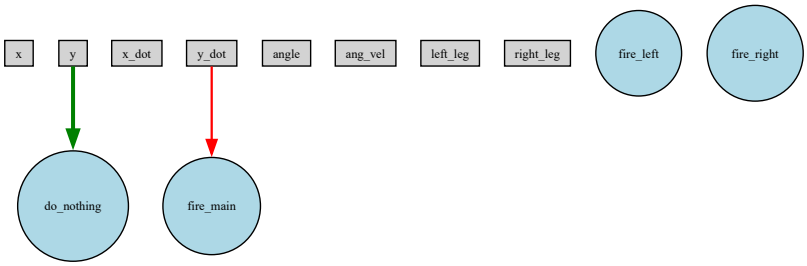


図 9　Lunar Lander:　強いペナルティ付き NEAT の実行結果 1 (node: 10, edge: 5)

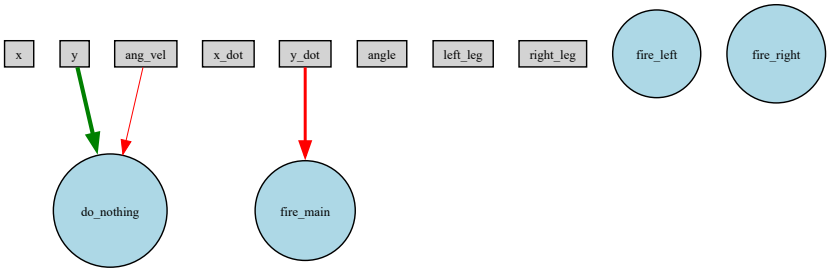


図 10　Lunar Lander:　強いペナルティ付き NEAT の実行結果 2 (node: 10, edge: 5)

図 7 及び図 8 はノード: 5,　エッジ: 1のペナルティ係数を、図 9 及び図 10 はノード: 10,　エッジ: 5 の強いペナルティ係数を設定して実行を行った結果である。両条件によるニューラルネットワーク図を比較すると、隠れノードはどちらも存在せず、エッジ数には顕著な差が見られることがわかる。

さらに、図 7 及び図 8 と通常の NEAT 手法の図 5 及び図 6 を比較するとノード数に違いが見られる一方で、エッジ数にはあまり差がない。これは 1 ノードあたり5点の比較的強いペナルティと、1 エッジあたり1点という比較的軽いペナルティによる適合度に与える影響の差によるものという可能性が考えられる。また、これらの関係に図 9 及び図 10 の強いペナルティ付き NEAT 手法を加えて分析すると、強いノードペナルティによるノードの削減効果に加えて、もう一方の標準的なペナルティ設定よりも比較的強いエッジペナルティによって更なるエッジ削減効果があったと言える。なお 3.5 で示す通り、上で述べたどちらのペナルティ付き手法も通常の NEAT 手法による実行結果に対して「ノード及びエッジの総和」における有意差がマン＝ホイットニーの U 検定により確認されている($p<0.05$)[6]。

3.4 Lunar Lander: 条件ごとの散布図による比較

この節では、Lunar Lander タスクにおける検証結果を散布図にまとめてその傾向を分析する。初めに、すべての検証結果をまとめて記した散布図を取り上げる。以下のグラフでは、本研究で行なった Lunar Lander における検証の全結果を散布図にプロットしている。なお、基準となる通常の NEAT 手法および提案手法のすべての条件で 15 回ずつ行った。

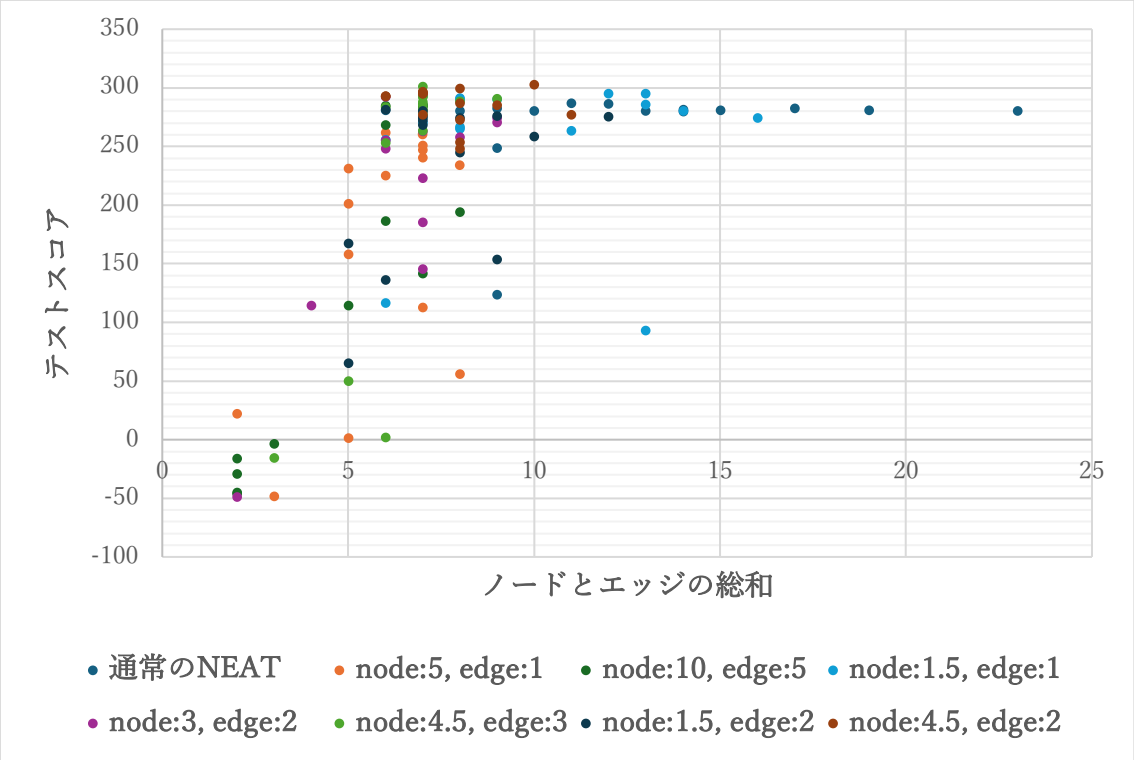


図 11: 各条件で構築されたモデルの比較

[6] ノード: 5、エッジ: 1 (U = 9, p = 0.0113858*10^-9)、
ノード: 10、エッジ: 5 (U = 8, p = 0.00721245*10^-9)

上図は前節の各モデルの特徴を散布図にプロットしたものである。ただし、凡例の node 及び edge の右側にある数字はそれぞれ一つ増えるごとに適合度から減算される値を示している。例えば、「node:5, edge:1」であれば、隠れノード（入出力ノード以外）の総数に5をかけた数値と、エッジの総数に 1 をかけた数値の総和がテストスコアから引かれたものが適合度として計算される。また、縦軸の「テストスコア」は各モデルの出力精度、横軸の「ノードとエッジの総和」は構造の複雑さ、すなわち説明可能性を表している。また、以降の説明に現れる各モデルの詳細なテストスコアや各モデルの世代数などの情報は、補足資料における各モデルの実行結果をもとに作成した表に記載されている。

図 11 において、黄緑色の点が示す、通常の NEAT 手法では 13/15 のモデルのスコアが 500 世代に達する前に終了条件である 280 点に到達しており、安定したスコアを残していることがわかる。一方で、散布図の横軸に注目すると、サンプルサイズに違いがあるものの、ペナルティ付きの NEAT と比較して、より多くのノード及びエッジを持っていることがわかる。また、少なくともこれらのデータからはペナルティ係数の大きな個体ほど左下に位置し、横軸の数値が 5 を下回ると急激にテストスコアが下がる傾向が見られる。この現象から、このモデルの構造が入力値をうまく生かして正確な出力に結びつけることのできる限界値である可能性が示唆される。
さらに、ノード: 10、エッジ: 5 の強いペナルティの NEAT では、ほとんどのモデルがマイナスの点数を示しているものの、最も良いモデルは 280 を超えるテストスコアを記録している。このモデルでは、ノード数が 0、エッジ数がこの条件において最も多い7であり、適合度では 35 点ものペナルティを受けながらも着実にエッジの重みづけを学習していった稀有な例であることが伺える。対照的に、良いモデルを構築しているノード: 4.5、エッジ: 3 の条件でも 1.9 点というスコアを記録したモデルがあり、散布図からは条件の優劣に関わらず学習が停滞してしまったモデルでは同条件の中でも外れ値として低いスコアとなってしまう可能性が考えられる。しかしながら、これらの条件の中で最もスコアの中央値が高いノード: 4.5、エッジ: 2 の条件では 250 を下回るスコアを記録しておらず、重みづけの組み合わせによっては安定して高いスコアを残すことができる可能性が示唆される。他方で、通常の NEAT 手法と同程度のスコアを記録するモデルでも、構築に最大世代数まで要することがほとんどであり、ペナルティ導入に多目的最適時の学習効率の向上には別途、アルゴリズムを改良する必要があることを示唆している。

3.5 Lunar Lander タスクの実行結果における検定

「ノードおよびエッジの総和」は比例尺度の離散データであり、本研究で構築された各モデルのように遺伝的アルゴリズムにより出力された情報は、正規分布に従わないことから、t 検定は利用できない。t 検定の代替方法としてノンパラメトリック検定であり、2 群間の値の比較を行うマン＝ホイットニーの U 検定を採用する。各条件と通常の NEAT 手法における構造の複雑さにおける差を明らかにするため、通常の NEAT 手法における実行結果に対して、各条件における実行結果を「ノー

ドおよびエッジの総和」についてそれぞれ検定を実施した。以下のように、すべての条件で通常の NEAT 手法に対する有意差が確認された($p<0.05$)。
ただし、前述のとおりサンプルサイズは各条件で 15 となっている。

ノード: 1.5、エッジ: 1($U = 73, p = 0.009583327$)、
ノード: 1.5、エッジ: 2($U = 29, p =0.043469*10^{-6}$)、
ノード: 4.5、エッジ: 2 ($U = 29, p =0.043469*10^{-6}$) 、
ノード: 3、エッジ: 2 ($U = 17, p = 0.037731*10^{-8}$)、
ノード: 4.5、エッジ: 3 ($U = 14.5, p = 0.0130067*10^{-8}$) 、
ノード: 5、エッジ: 1 ($U = 9, p = 0.0113858*10^{-9}$)、
ノード: 10、エッジ: 5($U = 8, p = 0.00721245*10^{-9}$)

ノード: 1.5、エッジ: 1 を除く条件では、$p<0.001$ という結果となった。このことから、NEAT アルゴリズムにおける構造の複雑さの抑制する目的において、本提案手法が有効であることが確認された。

3.6 研究の限界

以上のような検証結果を得ることができたが、さらなる研究に向けて本研究ではいくつか課題が存在する。第一に、各条件における実行結果のサンプルサイズが小さいことが挙げられる。今回はペナルティ係数設定のバリエーションを増やしたことで、各条件におけるサンプルサイズは小さくなってしまい、各知見を他研究でも再現可能であると断定するには不十分であった。計算資源的な限界はあるものの、さらなる条件設定の広範囲化、サンプルサイズの拡大による研究の信頼性向上は今後の課題としたい。第二に、本研究が将来的に想定している MoE アーキテクチャによるエキスパートネットワークの統合について、具体的な統合方法を考慮していないことである。MoE モデルの構築は非常に難易度が高いものの、実際にそのようなモデルとして実装することは今後の課題としていきたい。

4. 結論

4.1 提案手法の有効性

モデルごとのニューラルネットワークにおける比較による検証では、構築されるモデルのノード数およびエッジ数に本提案手法の多目的最適化が有効に機能していることが確認できた。また、Lunar Lander タスクにおいては 7 つの条件においてマン＝ホイットニーの U 検定によりそれぞれ通常の NEAT と有意差があることが確認された。また、ペナルティ係数の大小がモデル構造に与える影響については、構造の複雑性を抑制する方向に働く選択圧を強めることでノードやエッジの削減効果があることが明らかになった。対照的に、選択圧がほとんどかかっていない条件では、従来手法とあまり変わらない結果となったことで、適合度関数における選択圧をかけることによる構造

の複雑さのコントロールは行うことができたといえるのではないか。一方で、そのペナルティ係数の条件設定がどれだけ実行結果に与えるかどうかについては、目的とするタスクにより大きく変わると考えられ、本研究では明確な基準は明らかとなっていない。

4.2 トレードオフの関係

XOR タスクでは各条件におけるモデル構造の比較から、Lunar Lander では散布図からトレードオフの関係を読み取ることができた。

今回はペナルティ係数設定のバリエーションを増やしたことで、各条件におけるサンプルサイズは小さくなり、本研究における結果を一般化することはできない。更なるデータの収集を行い、結果の信頼性を向上させるとともに、さまざまな角度からデータ分析を行うことでデータからより多くの知見を提供することができると考えられる。

4.3 今後の展望

本研究では、将来的な MoE アーキテクチャにおけるエキスパートの小規模化に着目し、ごく小規模なニューラルネットワークを対象にした説明可能性向上を目指すものであった。本研究の課題であるサンプルサイズの増加に加えて、これまでに得たデータのさらなる分析を行うことでより多くの知見を得ることができるのではないかと考える。また、MoE に関するさらなる情報収集を進め、エキスパートネットワークの構造の複雑さを抑制することによってモデル全体の説明可能性向上に貢献するより具体的な研究に発展させてゆきたい。

謝辞

iTL 先端的プロジェクトに選定いただいた先生方や多大なるご指導を賜った須藤修教授には深く感謝いたします。

参考文献

[1] Maslej, Nestor et al. The AI Index 2025 Annual Report. Stanford University, Institute for Human-Centered AI, AI Index Steering Committee, 2025. https://doi.org/10.48550/arXiv.2504.07139. (参照 2025-12-26)

[2] Gunning, D, et al. DARPA's explainable AI (XAI) program: A retrospective. Applied AI Letters, vol. 2, no. 3, 2021. https://doi.org/10.1002/ail2.61. (参照 2025-12-26)

[3] 恵木 正史. XAI(eXplainable AI)技術の研究動向. 日本セキュリティ・マネジメント学会誌. 2020, vol.34, no.1, p.20-27. https://www.jstage.jst.go.jp/article/jssmjournal/34/1/34_20/_pdf. (参照 2025-10-20)

[4] Riccardo Guidotti, et al. A Survey of Methods for Explaining Black Box Models. ACM Computing Surveys (CSUR), vol. 51, no. 5, p.1-42, 2018. https://doi.org/10.1145/3236009. (参照 2025-12-26)

[5] Marco Tulio Ribeiro, et al. “Why Should I Trust You?": Explaining the Predictions of Any Classifier. Association for Computing Machinery. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16). p.1135-1144, 2016. https://doi.org/10.1145/2939672.2939778. (参照 2025-12-26)

[6] 東京大学工学教程編纂委員会 編, 伊庭斉志, Danushka Bollegala 著.「知識情報処理」. 丸善出版. 2016.

[7] 伊庭斉志.「進化計算と深層学習 創発する知能」. オーム社. 2016.

[8] 大山 聖. 多目的進化アルゴリズムの最前線. 日本機械学会, vol. 123, no. 1217, 2020. https://www.jstage.jst.go.jp/article/jsmemag/123/1217/123_14/_pdf. (参照 2025-12-26)

[9] Kenneth O. Stanley, Risto Miikkulainen. “Evolving Neural Networks through Augmenting Topologies.”. Evol Comput vol.10, p.99–127, 2002. https://doi.org/10.1162/106365602320169811. (参照 2025-10-20)

[10] Kenneth O. Stanley. Neuroevolution: A different kind of deep learning. O’REILLY. 2017. https://www.oreilly.com/radar/neuroevolution-a-different-kind-of-deep-learning/. (参照 2025-10-20)

[11] Evgenia Papavasileiou, Jan Cornelis, Bart Jansen. “A Systematic Literature Review of the Successors of ‘NeuroEvolution of Augmenting Topologie’”. Evol Comput vol.29, p.1–73, 2021. https://doi.org/10.1162/evco_a_00282. (参照 2025-10-20)

[12] W. H. van Willigen, et al. Evolving intelligent vehicle control using multi-objective NEAT. 2013 IEEE Symposium on Computational Intelligence in Vehicles and Transportation Systems (CIVTS). p. 9-15, 2013. https://doi.org/10.1109/CIVTS.2013.6612283. (参照 2025-11-26)

[13] Zitzler, Eckart, et al. SPEA2: Improving the strength pareto evolutionary algorithm. ETH Zurich, Computer Engineering and Networks Laboratory, 2001. https://doi.org/10.3929/ethz-a-004284029. (参照 2025-12-26)

[14] K. Deb, et al. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation, vol. 6, no. 2, p.182-197, 2002. https://doi.org/10.1109/4235.996017. (参照 2025-11-26)

[15] 坂和正敏. 「離散システムの最適化」. 森北出版, 2000.

[16] Gheorghe Comanici, et al. Gemini 2.5: Pushing the Frontier with Advanced Reasoning, Multimodality, Long Context, and Next Generation Agentic Capabilities. p.2-13, 2025. https://arxiv.org/abs/2507.06261. (参照 2026-1-9)

[17] Google. Gemini 2.5 Pro Model Card. p. 1, 2025. https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-2-5-Pro-Model-Card.pdf. (参照 2026-1-9)

[18] Google. Gemini 3 Pro Model Card. p. 1, 2025. https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-Pro-Model-Card.pdf. (参照 2026-1-9)

[19] Robert A. Jacobs, et al. “Adaptive Mixtures of Local Experts”. Neural Comput. vol.3, p.79–87, 1991. https://doi.org/10.1162/neco.1991.3.1.79. (参照 2025-11-27)

[20] Dave Bergmann. 専門家の混合とは. IBM. https://www.ibm.com/jp-ja/think/topics/mixture-of-experts. (参照 2025-11-27)

[21] Xu Owen He. Mixture of A Million Experts. 2024. https://doi.org/10.48550/arXiv.2407.04153. (参照 2025-12-26)

[22] Enric Boix-Adsera, et al. The power of fine-grained experts: Granularity boosts expressivity in Mixture of Experts. arXiv preprint, 2025. https://doi.org/10.48550/arXiv.2505.06839. (参照 2025-12-26)

[23] 斎藤 康毅.「ゼロから作る Deep Learning」. オライリー・ジャパン. 2016

[24] Oleg Klimov. Lunar Lander. Gymnasium Documentation. https://gymnasium.farama.org/environments/box2d/lunar_lander/. (参照 2025-11-17)

[25] CodeReclaimers. CodeReclaimers/neat-python. GitHub. https://github.com/CodeReclaimers/neat-python?tab=readme-ov-file. (参照 2025-11-26)

Appendix

以下に今回の研究で得られた結果を示す。

通常の NEAT	スコア	説明可能性	隠れノード数	エッジ数	世代数
1	280.641403	13	4	9	213
2	280.237101	10	1	9	161
3	286.249118	12	2	10	116
4	280.067439	14	3	11	203
5	282.409296	17	6	11	358
6	280.868351	15	3	12	224
7	280.44807	8	1	7	126
8	280.901374	19	6	13	145
9	248.939947	9	0	9	500
10	282.437955	9	1	8	181
11	123.556304	9	2	7	500
12	281.04963	6	0	6	52
13	287.010721	11	1	10	124
14	280.627185	23	9	14	421
15	281.36096	14	4	10	418
平均	269.120324	中央値	280.868351		
分散	1584.62555	標準偏差	39.8073555		

node:5, edge:1

	スコア	説明可能性	適合度	隠れノード数	エッジ数	世代数
1	234.193673	8	226.193673	0	8	500
2	260.073351	7	253.073351	0	7	500
3	251.116602	7	244.116602	0	7	500
4	225.138983	6	219.138983	0	6	500

5	-48.3040134	3	-51.3040134	0	3	500
6	56.3360666	8	44.3360666	1	7	500
7	158.410417	5	153.410417	0	5	500
8	240.609517	7	233.609517	0	7	500
9	22.3360056	2	20.3360056	0	2	500
10	112.695161	7	105.695161	0	7	500
11	231.193673	5	226.193673	0	5	500
12	246.973911	7	239.973911	0	7	500
13	201.152477	5	196.152477	0	5	500
14	261.826434	6	255.826434	0	6	500
15	1.26836031	5	-3.73163969	0	5	500
平均	163.668041	中央値	225.138983			
分散	10638.6721	標準偏差	103.143939			

node:10, edge:5

1	-15.9608933	2	-25.9608933	0	2	500
2	-3.51578815	3	-18.5157882	0	3	500
3	-45.9517833	2	-55.9517833	0	2	500
4	293.271298	7	258.271298	0	7	500
5	141.929356	7	101.929356	1	6	500
6	285.028497	7	250.028497	0	7	500
7	194.179542	8	149.179542	1	7	500
8	285.838717	7	245.838717	1	6	500
9	268.595931	6	238.595931	0	6	500
10	284.069265	7	249.069265	0	7	500
11	-29.0470168	2	-39.0470168	0	2	500
12	-44.9382207	2	-54.9382207	0	2	500
13	186.415806	6	156.415806	0	6	500
14	114.232566	5	89.2325665	0	5	500

15	284.869855	6	254.869855	0	6	500
平均	146.601142	中央値	186.415806			
分散	18097.0681	標準偏差	134.525344			

node:1.5, edge:1

1	263.52719	11	252.02719	1	10	500
2	274.452617	16	256.452617	4	12	500
3	280.724598	14	265.724598	2	12	500
4	256.081024	6	250.081024	0	6	500
5	290.229782	9	280.729782	1	8	234
6	270.758762	7	263.758762	0	7	500
7	265.071923	8	257.071923	0	8	500
8	291.507251	8	283.507251	0	8	68
9	294.985222	12	281.985222	2	10	355
10	285.922278	13	271.422278	3	10	500
11	93.2803728	13	78.2803728	4	9	500
12	294.991154	13	280.491154	3	10	485
13	267.015467	8	259.015467	0	8	500
14	116.828186	6	110.828186	0	6	500
15	290.373375	8	282.373375	0	8	165
平均	255.716613	中央値	274.452617			
分散	3654.97799	標準偏差	60.4564139			

node:3, edge:2

1	262.744749	7	248.744749	0	7	500
2	114.270739	4	106.270739	0	4	500
3	248.13177	6	236.13177	0	6	500
4	185.511	7	171.511	0	7	500

5	255.112205	6	243.112205	0	6	500
6	223.241455	7	209.241455	0	7	500
7	145.828593	7	130.828593	1	6	500
8	-48.688535	2	-52.688535	0	2	500
9	270.559118	9	252.559118	0	9	500
10	292.581605	6	280.581605	0	6	423
11	295.05479	7	281.05479	0	7	315
12	258.252012	8	241.252012	1	7	500
13	295.35456	7	281.35456	0	7	267
14	274.693845	9	256.693845	0	9	500
15	279.747677	7	265.747677	0	7	500
平均	223.493039	中央値	258.252012			
分散	8060.14809	標準偏差	89.7783275			

node:4.5, edge:3

1	263.68079	7	242.68079	0	7	500
2	291.022414	9	264.022414	0	9	500
3	1.92523948	6	-17.5747605	1	5	500
4	301.313162	7	280.313162	0	7	460
5	288.814726	8	263.314726	1	7	500
6	285.196525	7	264.196525	0	7	500
7	280.993257	7	259.993257	0	7	500
8	277.944411	7	256.944411	0	7	500
9	283.600663	6	265.600663	0	6	500
10	-15.4923966	3	-24.4923966	0	3	500
11	281.658911	7	259.158911	1	6	500
12	292.376695	7	271.376695	0	7	500
13	50.1036168	5	33.6036168	1	4	500
14	253.356825	6	235.356825	0	6	500

15	287.930278	7	266.930278	0	7	500
平均	228.295008	中央値	281.658911			
分散	11952.4137	標準偏差	109.327095			

node:1.5, edge:2

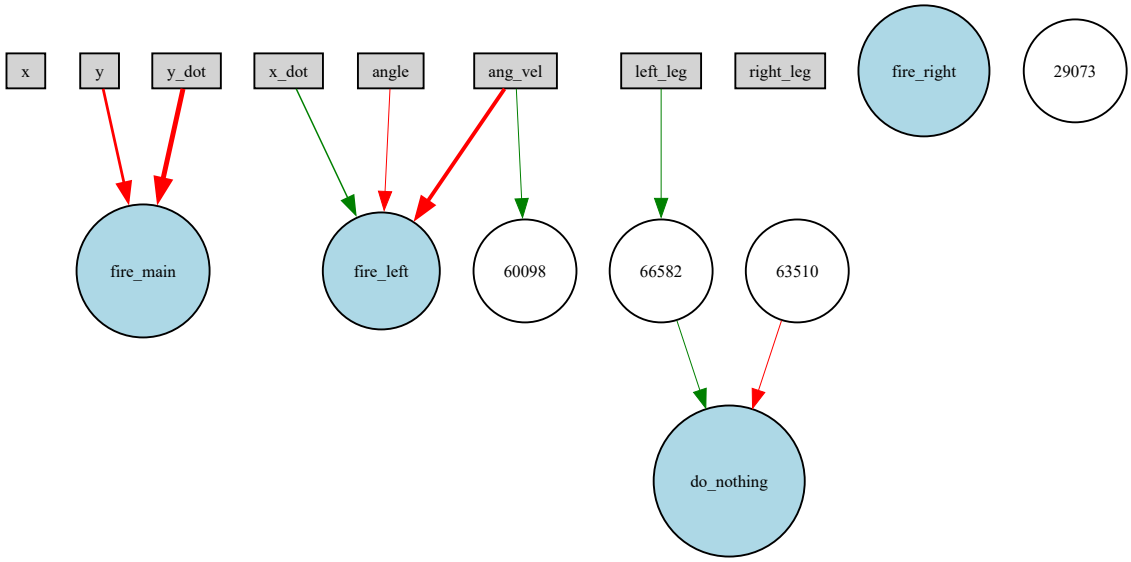
1	275.794216	12	252.294216	1	11	500
2	274.321687	8	258.321687	0	8	500
3	275.821687	9	258.321687	1	8	500
4	153.645079	9	136.645079	2	7	500
5	274.584842	7	260.584842	0	7	500
6	280.570225	7	266.570225	0	7	500
7	244.857509	8	229.357509	1	7	500
8	281.39476	6	269.39476	0	6	500
9	258.488919	10	239.488919	2	8	500
10	167.686637	5	157.686637	0	5	500
11	136.576998	6	124.576998	0	6	500
12	276.058998	7	262.058998	0	7	500
13	268.428523	7	254.928523	1	6	500
14	65.4373462	5	55.4373462	0	5	500
15	272.010035	7	258.010035	0	7	500
平均	233.711831	中央値	272.010035			
分散	4337.21361	標準偏差	65.8575251			

node:4.5, edge:2

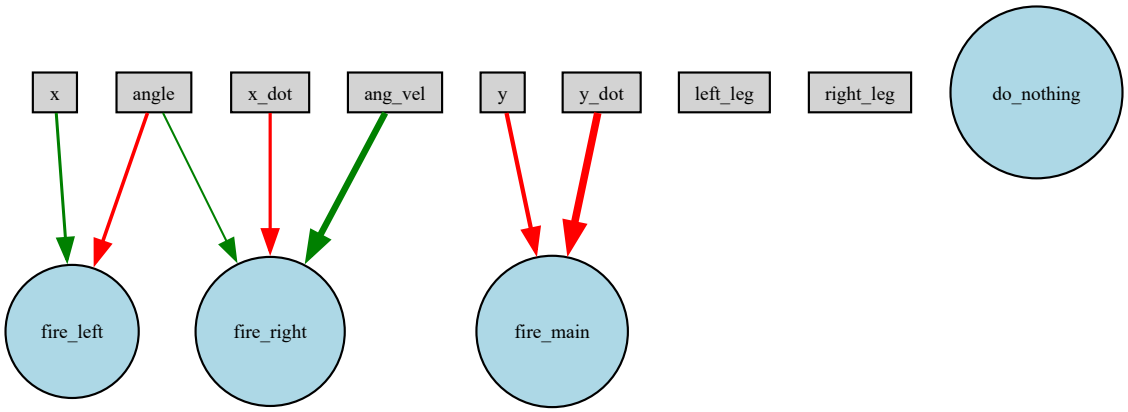
1	285.414817	9	264.914817	1	8	500
2	292.842564	6	280.842564	0	6	426
3	293.188545	6	281.188545	0	6	132
4	295.1791	7	281.1791	0	7	287

5	296.703318	7	282.703318	0	7	493
6	253.946298	8	237.946298	0	8	500
7	277.189003	11	250.189003	2	9	500
8	272.841206	8	256.841206	0	8	500
9	277.019594	7	263.019594	0	7	500
10	296.395705	7	282.395705	0	7	405
11	295.311881	7	281.311881	0	7	400
12	248.165658	8	232.165658	0	8	500
13	286.917415	8	268.417415	1	7	500
14	299.711772	8	283.711772	0	8	208
15	302.578377	10	282.578377	0	10	323
平均	284.893684	中央値	292.842564			
分散	248.539161	標準偏差	15.7651248			

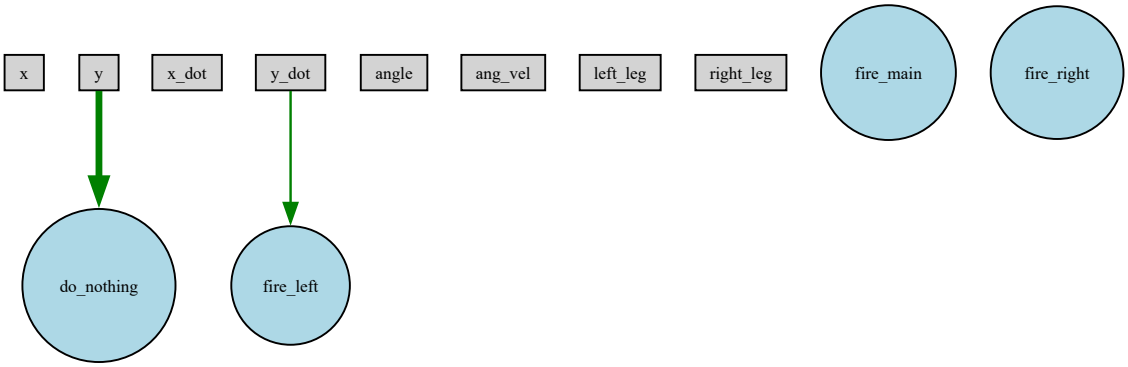
以下に Lunar Lander タスクおける今回の研究で構築されたモデルのニューラルネットワーク図を示す。



XOR: 通常の NEAT の実行結果 3



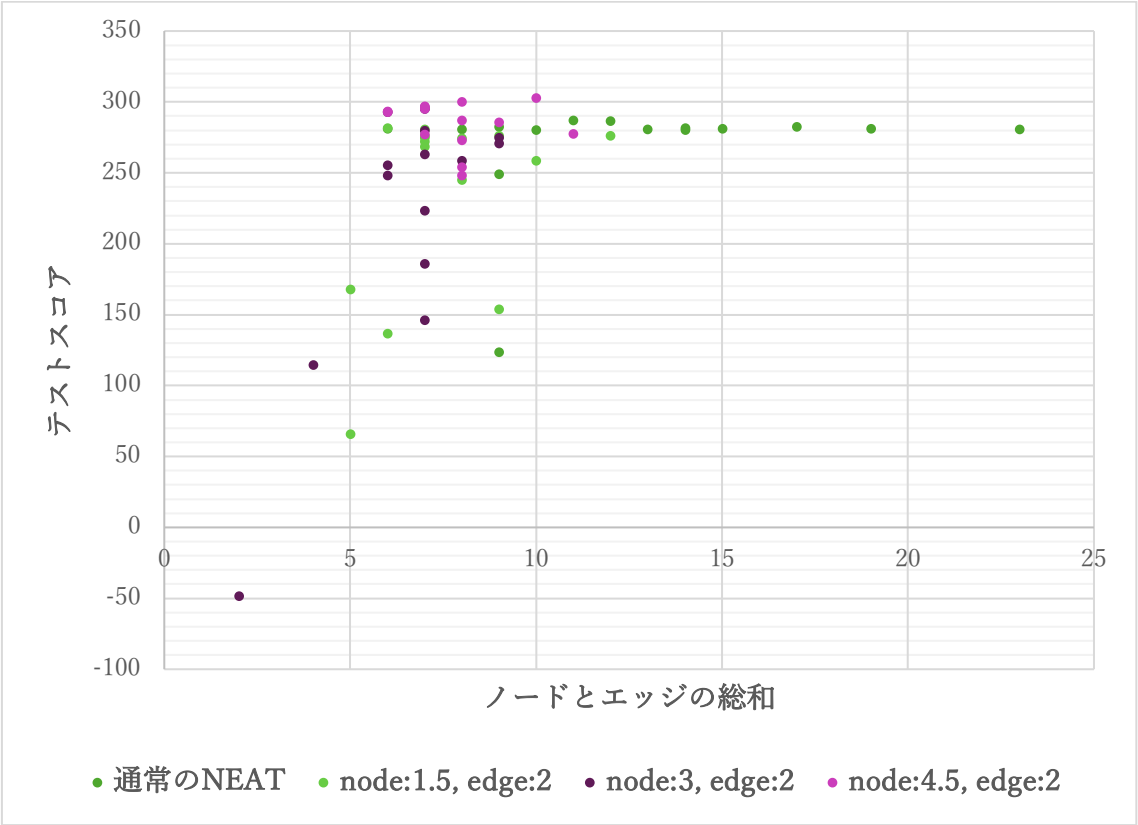
ペナルティ付き NEAT の実行結果 3 (node: 5, edge: 1)

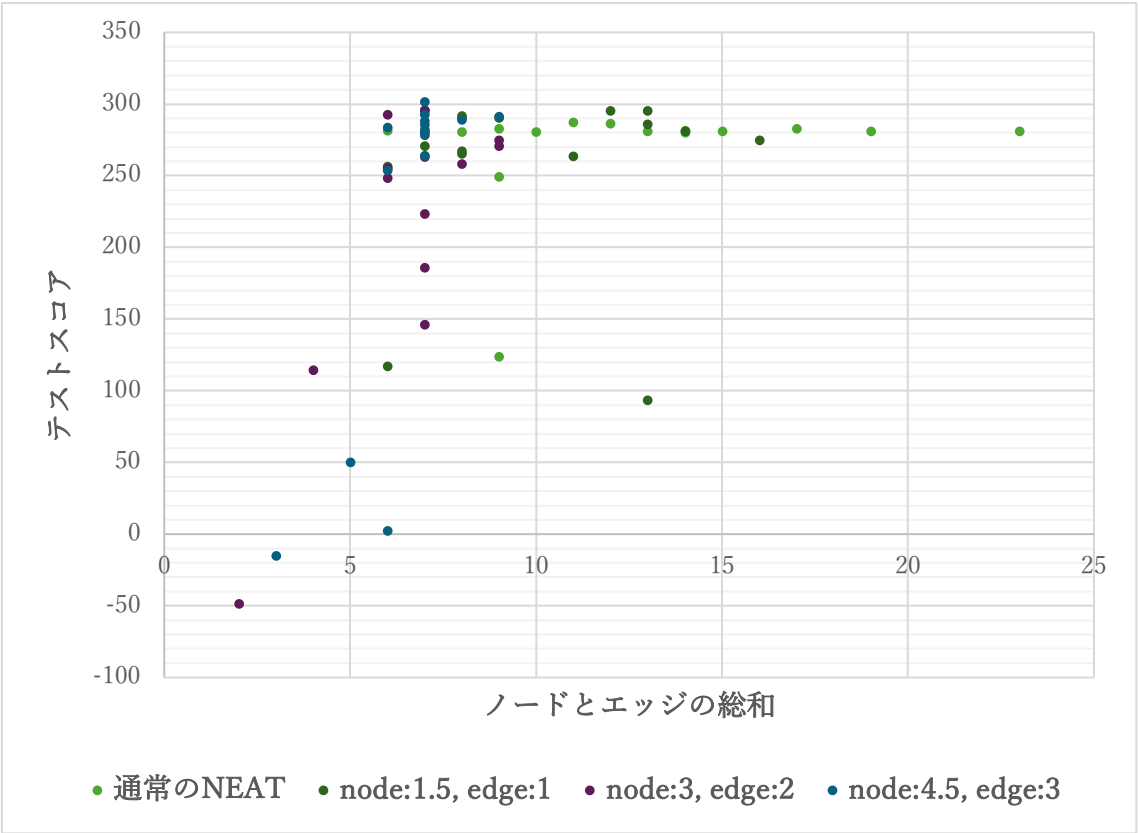


強いペナルティ付き NEAT の実行結果 3 (node: 10, edge: 5)

最後に、Lunar Lander タスクおける異なる条件の結果を比較した図を示す

以下の図はエッジの重みづけ係数を固定して比較した条件とペナルティを等倍した条件での結果をそれぞれプロットしたものである。


350
300
250
200
150
100
50
0
-50
-100
0
5
10
15
20
25
テストスコア
ノードとエッジの総和
通常のNEAT
node:1.5, edge:2
node:3, edge:2
node:4.5, edge:2


350
300
250
200
150
100
50
0
-50
-100
0
5
10
15
20
25
テストスコア
ノードとエッジの総和
通常のNEAT
node:1.5, edge:1
node:3, edge:2
node:4.5, edge:3