修士論文　　2019 年度

ベイズ理論を用いたビヘイビアツリーの中間ノードの評価に関する研究

東京工科大学大学院

バイオ・情報メディア研究科

メディアサイエンス専攻

義澤 勇輝

修士論文　　2019 年度

ベイズ理論を用いたビヘイビアツリーの中間ノードの評価に関する研究

指導教員　渡辺 大地 准教授

東京工科大学大学院

バイオ・情報メディア研究科

メディアサイエンス専攻

義澤 勇輝

論 文 の 要 旨

論文題目	ベイズ理論を用いたビヘイビアツリーの 中間ノードの評価に関する研究
執筆者氏名	義澤 勇輝
指導教員	渡辺 大地 准教授
キーワード	ゲーム AI、ビヘイビアツリー、ベイズ推定

[要旨]

近年、ゲーム AI は大きく進化した。家庭用ゲーム機が普及し始めた 1980 年頃のデジタルゲームではキャラクターは単純な行動しか起こせなかったが、近年では多種多様な動きを可能としている。また、ゲーム内の情報量が増え、ゲーム内でキャラクターが対処する状況が複雑化した。そのため、複雑な状況から適切な行動を起こす必要性が高くになり、キャラクターを制御する AI も複雑になった。複雑な制御を必要とするようになったキャラクター AI では、行動決定方法としてビヘイビアツリーが多く採用されるようになってきている。しかし、ビヘイビアツリーを用いて多くのキャラクター AI を制作し、プランナー等が求める挙動へと上手くビヘイビアツリーを調節することは容易ではない。また、現在のキャラクター AI の調節では AI を動かした結果を基に調節を行う。調節が不十分な場合は再び AI を動かし調節を行うため、確認と調節を何度も繰り返す必要がある。そのため、多くの労力を必要とする作業である。複雑化した近年のキャラクター AI では調節を行う際に調節の必要な部分を見つけることが難しいと言う問題がある。特にビヘイビアツリーを用いた AI では顕著である。

本論文ではビヘイビアツリーの AI で調節が必要な部分を簡単に見つけ、調節を行いやすくすることを目的としている。目的を達成する方法として、ベイズ推定を用いて確率的に評価することで、調節が必要な部分を簡単に見つけ出す方法を提案する。本手法ではキャラクターの行動決定に用いられるビヘイビアツリーを用いた AI を対象とし、ビヘイビアツリーの中間にあるノードをベイズ推定を使用し評価する。

検証を行った結果、本手法を用いることでビヘイビアツリーの中間ノードを確率に評価することが可能であることが分かり、調節が必要な部分を簡単に見つけ出すことができた。

A b s t r a c t

Title	Evaluation of Intermediate Nodes in Behavior Trees using Bayesian Theory
Author	Yuki Yoshizawa
Advisor	Taichi Watanabe
Key Words	GameAI, Behavior Tree, Bayesian Inference

[**summary**]

In recent years, digital game hardware has developed and performance has improved dramatically. As a result, recent digital games have become increasingly complex, such as those that require large amounts of memory and those that require computationally intensive processing. Compared to the past, game AI has evolved significantly. In the past, characters could only perform simple actions in digital games, but in recent years they have enabled a wide variety of actions. This complicates the control of the characters. As character AI becomes more complex and requires more control, behavior trees are increasingly being used as a method of behavior determination. However, it is not easy to create many character AI using the behavior tree and adjust the behavior well. In addition, the current adjustment of the character AI is based on the result of moving the AI. If the adjustment is insufficient, the AI will move again and the adjustment will be performed, so the confirmation and adjustment will need to be repeated many times. Therefore, it can be said that the work requires a lot of labor. There is a problem that it is difficult to find the part which needs the adjustment in the adjustment in the complicated character AI in recent years. This is especially true in AI using behavior trees. The purpose of this paper is to make it easier to find and adjust the parts of the behavior tree that need to be adjusted by AI. This is accomplished by probabilistic Bayesian estimation of the part that needs adjustment. In this method, the middle node of the behavior tree used for the action decision was evaluated using Bayesian estimation. As a result of the verification, it is found that it is possible to evaluate the intermediate nodes of the behavior tree in probability by using this method, and it can be said that it is possible to easily find out the part that needs adjustment.

目 次

図 目 次

表 目 次

第 1 章

はじめに

1.1 研究背景と目的

近年、デジタルゲームのゲームハードウェアである PC や家庭用ゲーム機が発達し、計算処理の性能や使用できるメモリ量が格段に向上した。そのため、最近のデジタルゲームでは膨大なメモリを用いることを前提としたゲームソフトや計算負荷の高い処理を必要とするゲームソフトを作ることが可能となった。多くのメモリや計算負荷の高い処理を用いることで数多くの種類のデジタルゲームが生まれ、多種多様なゲームが作られた。その結果さらなるデジタルゲームの面白さの追求が可能となった。また、多くのメモリを使うことや複雑な処理が可能になったことで、ゲーム内の状況が複雑化した。例えば、4K と呼ばれる超高画質での画面出力を可能としたゲームや、AR や VR 技術を用いたゲーム、他にも数十から数百というキャラクターを同時に動かしリアルタイムに処理を行うようなゲームなどが挙げられる。

デジタルゲームで大きく変化し進化したもののひとつとしてゲーム AI がある。有名な古いデジタルゲームの 1 つとして 1985 年に発売されたスーパーマリオブラザーズ [1] が挙げられる。このスーパーマリオブラザーズの敵キャラクターにクリボーという敵キャラクターがいるが、このクリボーを動かしているキャラクター AI はとても単純な処理しか行っていない。クリボーのキャラクター AI ではクリボーを真っすぐ進ませ、クリボーの前に障害物がある場合は進行方向を 180 度変更するだけである。他のデジタルゲームでもあらかじめ設定された道筋に沿って動くだけなど、家庭用ゲーム機などが普及し始めた 1980 年頃のデジタルゲームではとても単純なゲーム AI しか使用できなかった。しかし、近年のデジタルゲームのキャラクターでは多種多様な行動を可能とし、それらの行動を選択し実行することが可能となっている。そのため、キャラクター AI には複数ある行動の中からキャラクターが置かれている状況に応じて、適切に行動を決定することが求められるようになった。例えば、2016 年に発売されたファイナルファンタジー XV[2] の味方キャラクターや敵キャラクターのキャラクター AI では自分の置かれた状況や、味方や敵との距

離やステータス情報などをリアルタイムに取得し、まるで人間が操作しているのではないのかと思えるような行動決定を行う。近年のゲームでは使用できる行動の数や道具、敵キャラクターの種類、味方キャラクターの種類などが非常に多い。そのため、AI が行動決定を行うために必要な情報量も格段に増え、リアルタイムに変化する複雑な状況から行動決定をする必要性が高まった。このように近年のデジタルゲームではゲーム内の複雑な状況に応じてキャラクター AI が行動決定を行うことが必要となっている。また、プレイヤーがキャラクターの動きに違和感を覚える場合や納得できない動きをするキャラクターが存在する場合、プレイヤーの興味が薄れてゲームに熱中できないことや、ゲームの難易度によってプレイヤーのやる気がそがれてしまうことがある。つまりキャラクター AI の完成度はゲームへの熱中度ややる気などに深く関係があり、ゲーム全体の良さを左右する非常に重要なものである。そのため、近年のキャラクター AI には違和感のない行動となるように、状況に応じて納得感の高い行動決定を行うことが求められている。

その複雑な状況に対応するキャラクター AI を制作する方法として、ビヘイビアツリーを使用して行動決定を行うキャラクター AI が増えている。Halo2[3] において敵キャラクターを制御する方法として Damian によってビヘイビアツリーは作られたとされている。ビヘイビアツリーはステートマシンから環構造をなくし、末端のノードに向かって行動を選択していく階層型ツリーである。ビヘイビアツリーでの行動決定では開始となるノードから末端のノードまでたどり着くことで動作を決定し行動を起こす。ビヘイビアツリーを用いたキャラクター AI はツールを用いることでゲームデザイナーやプランナーのみで AI を制作することが可能であることや、2 回に 1 回特定の行動を起こす、キャラクターの体力が半分になったら特定の行動を起こすというような人間の直感に合わせて作りやすいことなどの利点がある。そのため、キャラクター AI の制作方法として近年多くのゲームタイトルでビヘイビアツリーが使用されている。

近年使用されることが増えたビヘイビアツリーを用いたキャラクター AI は制作することは比較的簡単である。しかし、ゲームバランスを整える調節作業の中でもビヘイビアツリーの調節作業

は容易ではないという問題がある。本論文での調節とは AI の条件分岐や構造などを変更し、ゲームバランスを整える作業のことを指しているが、ビヘイビアツリーの AI では特に調節が難しい。ゲームバランスを上手く整えることは非常に重要なことである。例えば、ゲームバランス悪いままプレイヤーがゲームを遊ぶ場合、初心者が序盤でゲームを進められなくなりそのゲームで遊ぶことをやめてしまうことや、難易度が高過ぎた結果ユーザにクソゲーなどと呼ばれ嫌われることなどがある。制作したゲームがこのような結果になってしまうことはゲーム制作者にとっても本意ではないため、ゲームバランスを整えることは非常に重要なことある。しかし、先述の通りビヘイビアツリーの AI では調節を行うことが難しい。ビヘイビアツリーの AI に関わらず、現在の調節作業ではゲームデザイナーやプランナーがキャラクターを実際に AI を用いて動かした結果を見て、その結果から調節を行う。調節を行った後、再びキャラクターを動かし調節の必要があるかを確かめ、調節が必要な場合は更に調節を行う。この確認と調節の工程を何度も繰り返し行うため、調節作業は時間と労力を非常に必要とする作業である。また、ビヘイビアツリーの AI が特に調節が難しい理由として、調節箇所を用意には見つけられず何度も確認と調節を繰り返す必要があるということが挙げられる。近年の複雑化したゲームでは状況に応じて多くの行動の中から行動を決定するが、行動決定をするためのビヘイビアツリーのツリー全体が非常に大きくなり膨大なノードの中から容易には調節箇所を見つけられないためである。

そのため、本研究ではビヘイビアツリーのノードをそれぞれ評価することでビヘイビアツリーの調節作業を行いやすくすることを目的とした。

ゲームバランスを上手く整えるためにゲームを評価し調節に役立てる研究として、眞鍋ら [4] が行ったソーシャルゲームの全体のゲームバランスを遺伝的アルゴリズムを用いて評価をした事例や、山本ら [5] の対戦型格闘ゲームで戦略の強さの評価を行った研究などがある。他にも自動でゲームを作成し、そのゲームバランスを評価した Michael Cook らの研究 [6] などが挙げられる。しかし、真鍋らの事例では遺伝的アルゴリズムを用いてゲームバランスの評価をしているが、キャ

ラクター毎の強さや武器などの道具の強さを対象とし相対的に評価するものである。そのため、キャラクターの AI を評価し、その結果から AI の調節に役立てることなどはできない。山本らの研究も同様に戦略の強弱のバランスを評価することを目的としており、ゲームバランスの調節のための判断材料にはなるが具体的に AI の調節に役立たせることは難しい。Michael Cook らの研究もゲーム全体のゲームバランスを評価することが目的であり、キャラクター AI の調節に役立てることは難しい。

ビヘイビアツリーについても多くの研究が行われている。ゲームの AI 以外への使用方法として、Petter[7] の無人航空機の制御にビヘイビアツリーを用いるための研究が挙げられる。他にも、Shuang ら [8] は衣服のデザインを容易に行う方法としてビヘイビアーツリーを用いる研究をしている。また、ビヘイビアツリーの評価を行う研究としては Michele Colledanchise ら [9] のロボット制御に使用されるビヘイビアツリーを評価する研究が挙げられる。この研究ではロボットの目的に合わせて各ノードで平均実行時間と成功確率を算出し、その値から各ノードを評価して最終的にビヘイビアツリー全体の評価を行うという研究である。しかし、ゲームのキャラクター AI では状況に応じて目的が変わることや、実行時間や成功確率を容易に算出できないなどの理由から、ゲームのビヘイビアツリーへ Michele Colledanchise らの手法をそのまま適応することは難しい。

他にもボードゲームのチェスや囲碁などのゲームでは、ツリー構造を持つゲーム木を用いて意思決定を行う AI が多い。しかし、ボードゲームに使用されるモンテカルロ木探索やアルファ・ベータ法などの手法や研究では同じツリー構造を使用してはいるが、ビヘイビアツリーとはツリーの意味合いが大きく異なるため適応することは難しい。

AI の調節という観点では、Yang ら [10] の動的に AI の意思決定を調節するという研究や、石原ら [11] はモンテカルロ木探索を用いることで格闘ゲームのキャラクター AI の強さを容易に調節する手法の研究を行った。他にも、佐藤 [12] はユーティリティベースの意思決定を行うキャラクター AI を用いて First-Third-Person-Shooting ゲームの難易度を簡易的に調節する方法の研究

を行った。仲道ら [13] は将棋の AI において相手の強さに合わせて AI の強さを動的に自動調節する方法を研究した。中川ら [14] は格闘ゲームの AI を機械学習を用いて調節する方法を研究した。しかし、これらの研究はビヘイビアツリーの AI を目的としていないことや、機械学習でほぼ自動で調節を行っているため、ビヘイビアツリーの調節を人間が行う際に使用することは難しい。上記の AI の調節とは調節へのアプローチの仕方が多少違う研究もある。杉本ら [15] はあらかじめ複数用意した AI を状況に応じて変更することでキャラクターの AI を調節する手法を研究した。しかし、この研究でも複数の AI を用意する必要があることや、単体の AI で根本的に調節を行うことはできない。

また、AI の調節に似た研究として、AI の自動生成についての研究も挙げられる。人間が直接調節をすることなく、機械学習を用いて AI を自動で調節しつつ自動生成する研究や、遺伝的アルゴリズムで進化的に AI を自動生成する研究が多くある。星野ら [16] のプレイヤーの飽きを軽減する方法としてプレイログから行動を模倣し、成長していく AI の研究を行った。藤井ら [17] は生物学的制約を課した機械学習により、人間らしい NPC を自動生成できることを示唆した。張ら [18] は Multiplayer Online Battle Arena でキャラクターの行動を機械学習を用いて自動生成する研究を行っている。加納ら [19] らは深層強化学習を用いてローグライクゲームを自動攻略できる AI を制作する方法を研究した。Grammatical Evolution を用いて新たなビヘイビアツリーを生成することで人間による調節ではなく、より良い AI を進化的に自動生成する研究としては、Chong-U ら [20] の研究や、Diego ら [21] の研究、Michele Colledanchise ら [22] の研究、QI ら [23] の研究などがある。他にも、上田 [24] は遺伝的アルゴリズムを用いて特色の違う AI を自動生成することで、プレイヤーのレベルに AI を合わせる研究を行った。福嶋ら [25] も同様に遺伝的アルゴリズムを用いることで多様な振る舞いを持つ AI を生成する研究を行った。しかし、これらの AI を自動生成する研究ではプランナー等が制作する場合と違い、自動で AI が生成されていくため、本来プランナー等が意図したキャラクターの挙動を持つ AI を制作することができるとは

言えない。

上記の通り、ゲームバランスを評価する研究や AI の自動生成、自動調節を行う研究は多くある。しかし、ゲーム AI に使用されるビヘイビアツリーの調節が難しいという問題解決を目指したものは少なく、ビヘイビアツリーを用いた AI の調節が非常に難しいという問題は解決されていない。そのため、本研究ではビヘイビアツリーを用いた AI の調節を容易に行えるようにすることを目的とした。

調節を容易に行う方法として、調節の必要性が高い部分を簡単に見つけ出すことにした。本手法では、ビヘイビアツリーの中間ノードに良い働きをする良い中間ノードと、悪い働きをする悪い中間ノードがあると考え、ビヘイビアツリーの各中間ノードを評価することで、調節の必要性が高いと考えられる悪い働きをする中間ノードを見つけ出す。

各中間ノードの評価にはベイズ推定を用いることにした。各中間ノードをベイズ推定を行い、確率分布を用いて確率的に評価することで調節が必要な部分を簡単に見つけ出す。キャラクターが行動を起こす前後で、どれだけ状況を変化させたのかで、その行動が良い行動と悪い行動のどちらであったのか評価関数を用いて判定し、この行動の良し悪しからベイズ推定を行う。ベイズ推定の尤度は、二項分布を用いて各中間ノードが悪い行動となる悪い選択をどの程度起こしたのかで尤度関数として設定した。ベイズ推定は繰り返し行うことで、確度を高めることができるため、キャラクターが行動を起こす毎に、ベイズ推定を繰り返し行う。事前確率は情報なしとして、一様分布を用いる。ベイズ推定を繰り返し何度も行うことで各中間ノードの評価を行い、調節の必要性が高いと考えられる悪い中間ノードを確率的に見つけることが可能である。

本手法の検証を行った結果、本手法を用いることでビヘイビアツリーの中間ノードを確率に評価することが可能であることが分かった。ベイズ推定と確率分布を用いることで、各中間ノードの評価ができ、その推定結果の確度の高さもあわせてわかるため、推定結果の信用度の高さも容易に知ることができる。そのため、調節の必要性が高い部分を簡単に見つけ出すことができ、そ

の見つけ出した調節の必要性が高い部分がどれだけ信用できる推定結果によるものであるのかを知ることも容易にできるため、単純な頻度による確率的な評価よりも有用である。

1.2　論文構成

本論文の構成は，以下の通りである。第 2 章では、ゲーム AI やビヘイビアツリー、ベイズ推定について説明する。第 3 章では、本研究の提案手法について述べる。第 4 章では、検証として本手法を用いてビヘイビアツリーを評価した結果を述べ、第 5 章でまとめについて述べる。

第 2 章

ビヘイビアツリー

本章ではビヘイビアツリーについて記述する。本章を記述するにあたりゲーム AI について記述された書籍 [26][27][28] を参考にしている。

ビヘイビアツリーとはデジタルゲームの AI としてキャラクターの行動決定に用いられている手法のひとつである。第 1 章で述べた通り、Halo2 において敵キャラクターを制御する方法として Damian Isla 氏によってビヘイビアツリーは作られたと言われている。当時、多くの AI に使用されていた行動決定手法はステートベースと呼ばれる有限状態マシンに基づいて行動決定を行う手法である。ステートベースの AI では行動がパターン化しやすいことや 2 つの状態だけを頻繁に遷移しやすい場合があるなどの問題があった。そのため、ビヘイビアツリーはこれらの問題を解消することとともに更なる拡張性を求めて作られた。

ビヘイビアツリーはゲームの AI だけでなく他にもロボットの行動決定に用いられることなどもあるが、本論文ではデジタルゲームのキャラクター AI に用いられるビヘイビアツリーを対象としている。

ビヘイビアツリーの構造は何層にも積み重なったツリー構造をしている。root ノードと呼ばれる開始地点となるノードから選択され、ノードに設定された選択ルールに従い下位のノードを選択していく。最終的に末端にあるノードが選択されると、その末端のノードに設定されたアクションを起こす。本論文では、この root ノードから末端の行動ノードまで選択され、キャラクターがアクションを起こすまでの一連の流れを 1 回の行動とした。

また、本論文ではビヘイビアツリーは行動決定を行うための開始地点となる root ノードと下位のノードへの分岐を持つ中間ノード、末端にありアクションを保持するアクションノードがある。このアクションとはキャラクターが起こせる最小限の行動のことを指す。例えば、仮にスーパーマリオブラザーズのマリオのアクションであるなら、前に進むことやジャンプすることがアクションにあたる。ブロックを叩きキノコを取得するといったような一連の行動ではない。図 2.1 は簡単にビヘイビアツリーを表した図である。

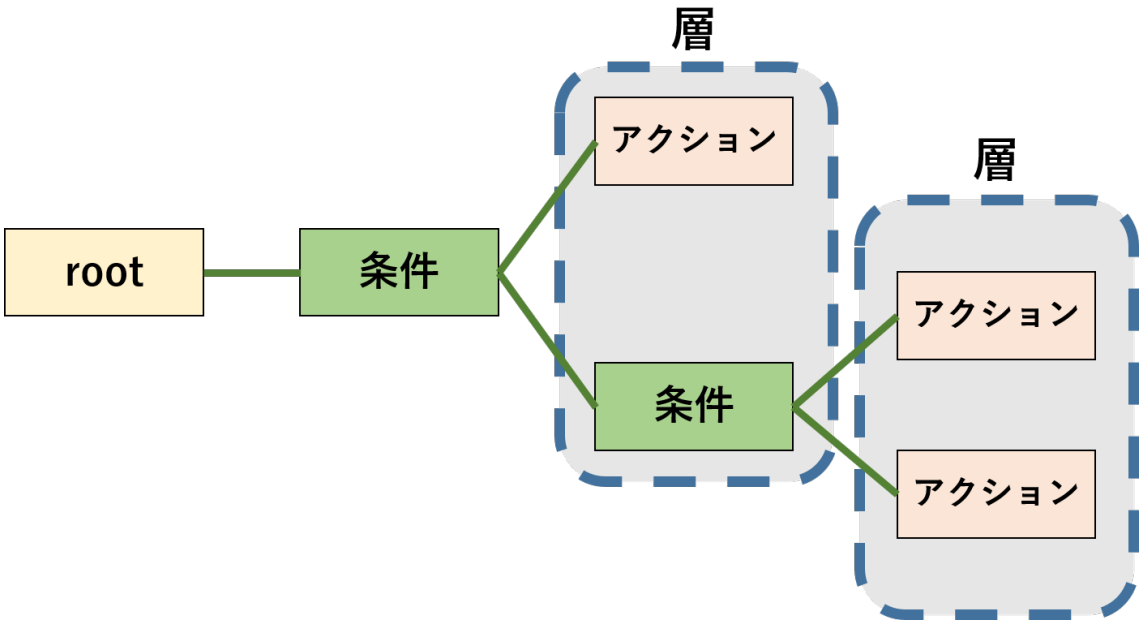

図 2.1　ビヘイビアツリーの例

中間ノードではあらかじめ設定された条件に従い下位のノードを選択する。この中間ノードに設定された条件のことを本論文では選択ルールとする。中間ノードに設定できる選択ルールは複数あり、本論文では下記の 3 つの選択ルールを対象とした。

- 条件ルール
- 順番ルール
- ランダムルール

ゲームに使用するビヘイビアツリーの中間ノードの選択ルールには上記の 3 つ以外にも他の選択ルールが存在する場合がある。また、内容に多少差がある場合もあるが、本研究では上記の 3 つのみを対象とし選択ルールの内容は下記のような条件であるとした。

2.1　条件ルール

選択ルールが条件ルールである中間ノードは設定された条件に従って下位のノードを選択する中間ノードのことである。例えば、選択ルールが条件ルールである中間ノードがある。その設定された条件がキャラクターの体力が半分以下であるかどうかという場合を考える。この中間ノードは選択された際に現在のキャラクターの状況を参照し下位のノードを選択する。中間ノードの下位には 2 つのアクションノードがあるとし、明示的に A のアクションノードと B のアクションノードとする。この場合、ノードが選択されたときにキャラクターの残り体力が半分以下なら A のアクションノードを選択し、体力が半分よりも多い場合は B のアクションノードを選択する。このような条件によって下位のノードを選択する選択ルールが条件ルールである。

図 2.2 に上記で挙げた選択ルールが条件の中間ノードの例を示す。

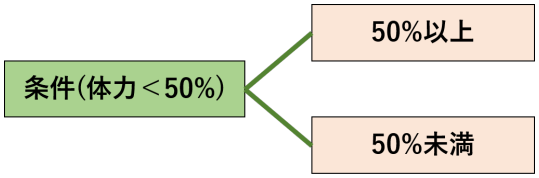

図 2.2　条件ルールの例

2.2　順番ルール

選択ルールが順番ルールである中間ノードはその中間ノードが選択された回数によって下位のノードを選択する中間ノードのことである。例えば、ある中間ノードの選択ルールが順番であり、そのノードが選択された回数が 1 回目なら 1 つ目の下位のノードを選択する。2 回目なら 2 つ目の下位のノードを選択する。下位のノードが 2 つしかない場合でノードが選択された回数が 3 回目なら最初に戻り、1 回目と同じく 1 つ目の下位ノードを選択する。

図 2.3 に 3 つの下位ノードを持つ選択ルールが順番となっている中間ノードの例を示す。この

中間ノードでは 1 回目は 1 つ目の下位ノード、2 回目は 2 つ目の下位ノード、3 回目は 3 つ目の下位ノードを選択する。4 回目以降はまた 1 つ目の下位ノード、2 つ目の下位ノードと繰り返し選択する。

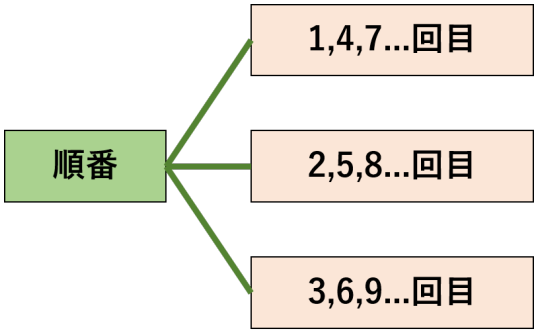

図 2.3　順番ルールの例

2.3　ランダムルール

選択ルールがランダムルールである中間ノードとは下位のノードをランダムに選択する中間ノードである。ランダムに選択する確率は等確率である場合や、あらかじめ設定できるものなどがある。

図 2.4 に選択ルールがランダムの中間ノードの例を示す。この中間ノードではそれぞれ 50%,25%,25% の確率でランダムに下位のノードを選択する。

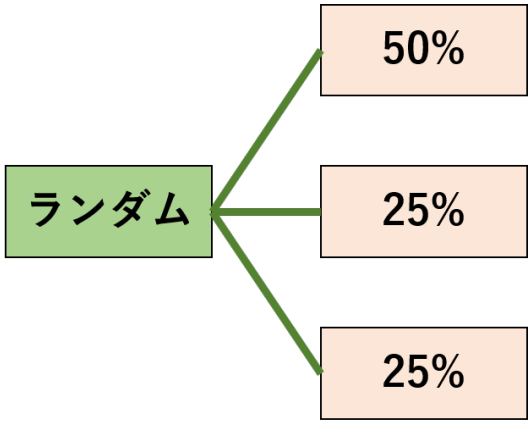

図 2.4　ランダムルールの例

2.4　ビヘイビアツリーの問題点

先述の通りビヘイビアツリーは近年使用されることが増えてきている。その理由のひとつとしてはビヘイビアツリーを用いることで比較的簡単に AI を作れることが挙げられる。昔はプログラマーが仕様書に則り AI 作成の全作業を行っていたが、ビヘイビアツリーを用いることでプログラマーではなくゲームデザイナーやプランナーが AI を制作することが可能である。また、AI を作る際にキャラクターの体力が半分以下になった場合といったように制作者が直感的に作りやすいことも、近年ビヘイビアツリーが多く使用される大きな理由である。

しかし、簡単に AI を作ることができるビヘイビアツリーにも問題点がいくつかある。そのひとつとして挙げられるのはビヘイビアツリーに限った話ではないが、AI の制作中ではその AI が上手く作れているのか容易には判別ができないことである。そのため実際にキャラクターを動かした際に想定通りに上手く動くとは限らない。例えば、人間が操作する場合は敵キャラクターに必ず勝てるというキャラクターを AI に操作させるとほぼ必ず負けてしまうということや、RPG などのゲームでストーリーの中盤に出てくるボスキャラクターが強い技を多用してきて想定よりも強すぎることや、逆にゲームの終盤に出てくるボスが明らかに意味のない攻撃を連発してきて弱いことなどが挙げられる。このように AI を作り実際にキャラクターを動かしてみるまで AI の出来の良さや悪さが分からないため、後からゲームバランスを整えることはゲーム開発では頻繁に起こる出来事である。ゲームバランスを整える方法としてはキャラクターのステータスなどのパラメータを調節する場合も多くあるが、キャラクターの AI 自体を調節することも非常に多くある。

現在のゲームのバランス調節は実際にキャラクターなどを動かしてゲームデザイナーやプランナーが確認し、確認した結果から再び調節を繰り返すという調節と確認を繰り返し行う作業である。また、ひとつのゲームでもキャラクターは複数存在し、キャラクターの数だけ調節が必要で

あることも少なくない。そのため、ゲームのバランス調節は非常に労力の必要な作業である。

特にビヘイビアツリーの AI を調節することは容易ではない。その最たる理由として挙げられるのが、ビヘイビアツリーのどの部分を調節する必要があるのか分からない点である。

ビヘイビアツリーの調節方法として考えられるのは大きく分けて 2 つある。1 つはビヘイビアツリーの構造自体を変更することである。もう 1 つは中間ノードの選択ルールや条件を変更することである。しかし、複数個存在する中間ノードの中からどの中間ノードを調節するべきかを判断することは難しい。近年の大型タイトルのゲームで使用されるビヘイビアツリーの AI ではノードの数が膨大であり、中間ノードの数も数十から数百もの量となるためである。その膨大な量のノードからなるビヘイビアツリーの中から調節の必要な中間ノードを見つけ調節するのは多くの時間と労力を必要とする。例えば、比較的少ない 10 個の中間ノードを持つビヘイビアツリーの調節が必要となった場合でも、10 個の中間ノードからどの中間ノードを調節するべきか判断するのは非常に難しい。図 2.5 が中間ノードを 10 個持つ比較的単純なビヘイビアツリーの例を示した図である。

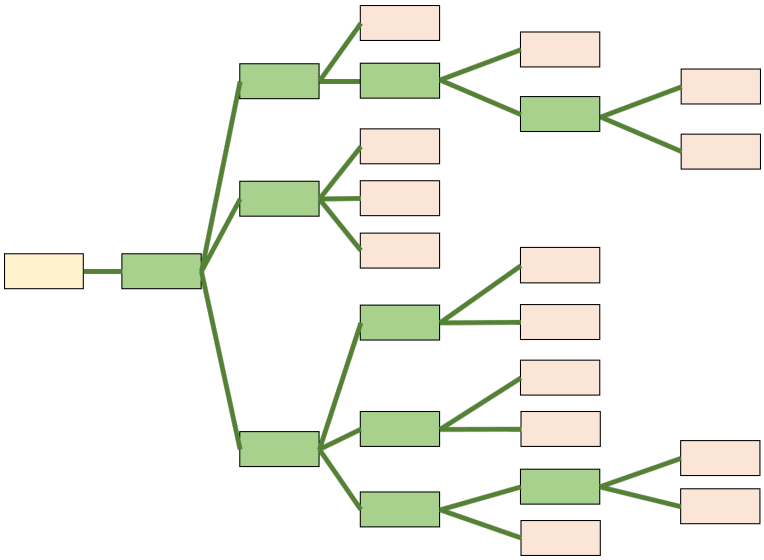

図 2.5　調節箇所が分かりづらい例

このように調節する箇所を見つけ出すだけでも困難なビヘイビアツリーの調節を行いやすくす

るために、本論文では調節箇所の必要な中間ノードを見つけ出す手法について提案する。

第 3 章

提案手法

3.1　中間ノードの選択結果に対する良し悪し

まず、本手法が中間ノードを評価する際に使用する行動選択の良し悪しの考え方について記述する。本手法では、キャラクターが行動を決定するまでの各中間ノードが行う選択に対して良し悪しがあるとした。キャラクターが行動を起こす前後で状況が改善した場合に、その行動を起こすまでに行った中間ノードの選択を良い選択とする。行動の前後で改善しなかった、あるいは状況を改悪した場合の中間ノードの選択を悪い選択とする。例えば、キャラクターの体力が残り少ない状況で回復行動を行う場合は状況が改善するため良い選択となる。しかし、体力の残量が多い状況で行う回復行動は状況を改善しているとは言えないため悪い選択となる。

3.2　中間ノードの良し悪し

本手法のが行う中間ノードの評価の方法は第 3.1 節で定義した中間ノードの選択の良し悪しを、各中間ノードがどの程度しやすい傾向にあるのかを確率的に推定することで行う。

例えば、同じキャラクターで AI に使用しているビヘイビアツリーが違う 2 種類のビヘイビアツリーで行動をすると仮定する。この 2 つのビヘイビアツリーは構造自体は変わらないが中間ノードに設定された選択ルールの条件ルールが持つ条件が一部が違うものとする。図 3.1 と図 3.2 で 2 つのビヘイビアツリーをそれぞれ示す。この図で表したビヘイビアツリーの AI を持つキャラクターは残りの体力によって回復と攻撃を選択し行動を起こす。2 つのビヘイビアツリーの回復と攻撃への分岐条件の違いは、残り体力が 50% 以下であるのか、25% 以下であるのかの違いとする。表 3.1 と表 3.2 に、この 2 つのビヘイビアツリーでそれぞれキャラクターが行動を数回起こした際の行動の良し悪しの結果を示す。表 3.1 が 1 つ目のビヘイビアツリーで行動を起こした場合の結果で、8 回中 2 回悪い行動を起こすとする。表 3.2 が 2 つ目のビヘイビアツリーで行動を起こした場合の結果で、8 回中 0 回が悪い行動であるとする。この場合は悪い選択を起こす回数

が多い 1 つ目のビヘイビアツリーの中間ノードの方が悪い中間ノードであることが分かる。

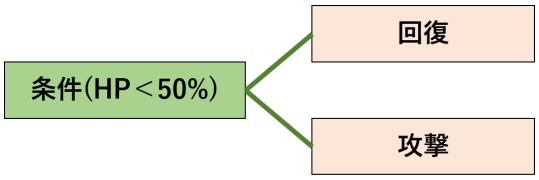

図 3.1　行動の良し悪しの傾向例 1

表 3.1　行動の良し悪しの傾向例の表 1

ターン	図 3.1 の行動	良し悪し
1	攻撃	良い
2	攻撃	良い
3	回復	良い
4	攻撃	良い
5	回復	悪い
6	攻撃	良い
7	攻撃	良い
8	回復	悪い

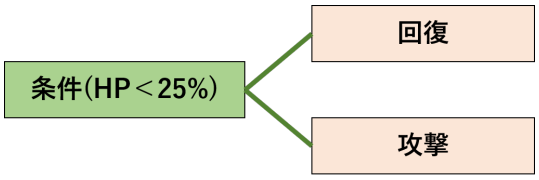

図 3.2　行動の良し悪しの傾向例 2

表 3.2　行動の良し悪しの傾向例の表 2

ターン	図 3.2 の行動	良し悪し
1	攻撃	良い
2	攻撃	良い
3	攻撃	良い
4	回復	良い
5	攻撃	良い
6	攻撃	良い
7	攻撃	良い
8	回復	良い

このように中間ノードでどの程度良い選択と悪い選択をする傾向にあるのか知ることができれば悪い選択のしやすい中間ノード、つまり調節の必要性が高い中間ノードを見つけることができると推測可能である。そのため、本手法では各中間ノードがどの程度良い選択と悪い選択をする傾向にあるかを推定することで中間ノードの評価を行う。推定方法としてはベイズ推定用いた。

3.3　ベイズ推定

ベイズ推定とはベイズ理論に基づき、新たに得た観測事象から過去の情報を用いて不確実な事象を確率的に推定することが可能な推定方法である。有名な使用例として迷惑メールフィルタなどが挙げられる。本節を記述するにあたり Web サイトの情報 [29][30][31] を参考にしている。

ベイズ推定はベイズの定理から考えられている。ベイズの定理は下記の式 (3.1) である。

$$P(B|A) = \frac{P(B) \cdot P(A|B)}{P(A)} \tag{3.1}$$

A と B は事象を表し、$P(A)$ は事象 A の起こる確率である。$P(A|B)$ は事象 B が起きたときに事象 A が起きる確率を表す条件付き確率である。

上記の式 (3.1) を展開しベイズ推定は下記の式 (3.2) として一般的に扱われている。

$$P(B_i|A) = \frac{P(B_i) \cdot P(A|B_i)}{\sum P(B_j) \cdot P(A|B_j)} \tag{3.2}$$

この式 (3.2) から連続的な確率分布を扱う場合は下記の式を用いて推定を行う。

$$\pi(\theta|x) = \frac{f(x|\theta)\pi(\theta)}{\int_\theta f(x|\theta)\pi(\theta)d\theta} \tag{3.3}$$

上記の式 (3.3) の x は事象を示し、θは確率変数である。また $\pi(\theta)$ は事前分布、$\pi(\theta|x)$ は事後分布、$f(x|\theta)$ は尤度、$\int_\theta f(x|\theta)\pi(\theta)d\theta$は周辺尤度である。

事前確率とは推定を行う前の想定していた確率のことで、事後確率は事象を観測し推定を行い事前確率を更新したものである。この事前確率から事後確率に更新することをベイズ更新と呼ぶ。尤度とはもっともらしさのことである。例えば、コインを 2 回投げ 2 回とも表が出たとする。この場合にこのコインを投げて表になる確率のもっともらしさである尤度は確率 p が 0 から 1 である場合 p =1 が最大となり、p =1 から離れるほど小さくなる。コインを 2 回投げ 1 回が表でもう 1 回は裏であったとすると、p =0.5 が最大となる正規分布となる。周辺尤度とは観測事象の出やすさを表しており、事後分布を正規化するためのものである。

3.4 本手法でのベイズ推定の使用方法

ベイズ推定を行う上で必要なものは尤度と事前確率である。本手法では尤度を尤度関数とし、二項分布を用いて定義した。二項分布とはベルヌーイ試行と呼ばれる 2 つの事象のみが起きる確率モデルであるときに、成功する確率が p である試行を n 回した際に k 回の成功が出た場合、その結果がどれだけ出やすいのかを確率的に算出する方法である。二項分布は下記の式 (3.4) から求められる。この式 (3.4) の p を 0 から 1 で変化する確率変数とすることで本研究では尤度関数として扱う。

$$L(n, k, p) = {}_nC_k p^k (1-p)^{n-k} \tag{3.4}$$

図 3.3 で 10 回の試行を行い、成功が 5 回出た場合の尤度関数の例を示す。横軸が確率変数 p の値を左から右へ 0 から 1 の範囲で表し、縦軸がその確率変数 p が真値である確率を表している。この 10 回中 5 回成功が出た場合、尤度関数は横軸の中央である 0.5 付近の確率が一番高く、左右に広がるにつれて小さくなる正規分布となる。

図 3.4 で 10 回の試行を行い、成功が 2 回でた場合の尤度関数の例を示す。この場合は先ほどの 0.5 付近が一番高い図 3.3 の例とは違い 0.2 付近の確率が一番高くなる。

図 3.3　尤度関数の例 1

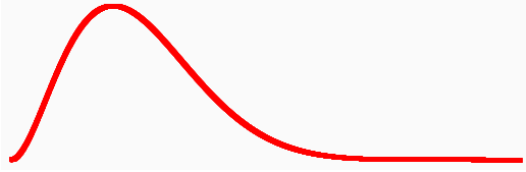

図 3.4　尤度関数の例 2

本研究の尤度関数の算出ではキャラクターを実際に動かし、各中間ノードでその中間ノードが

選択され行動が実行された回数と悪い選択を行った回数から尤度関数を算出する。式 (3.4) を用いて各値を、中間ノードが選択された回数を n として、n 回の間に悪い選択をした回数を k とする。確率 p の部分は 0 から 1 をとる確率変数とし、p が悪い選択の出しやすさの確率となる。この n, k, p から尤度関数を算出する。本研究では式 (3.4) から求められる、この $L(n, k, p)$ を悪い選択の出やすさのもっともらしさである尤度関数として扱う。

事前確率は事前情報なしとして、各中間ノードに一様分布として保持しておく。事前確率は $B(p)$ とし、確率変数 p の値から p に対応する確率を算出できる。この尤度関数と事前確率からベイズ推定を行う。

選択の良し悪しを評価関数を用いて観測事象から導く。選択が悪かったという事象を X、良かったという事象を Y とする。一般的に行動や選択の良し悪しはプランナーの意図により変化する。そのため、本手法では意図に沿うような評価関数を設定し判定を行う。評価関数は行動を起こす前後で状況をどれだけ変化させたのか判定し、行動選択の良し悪しを算出する。

3.5 ベイズ更新

第 3.4 節で述べた通りキャラクターを動かし尤度関数を生成するが、ベイズ推定も同時に行う。キャラクターが行動をしたときの行動 1 回 1 回で選択された各中間ノードでベイズ推定を行い事前確率から事後確率にベイズ更新を行っていく。本手法ではベイズ更新に悪い選択だった事象 X の場合の更新と、良い選択だった事象 Y の場合の更新がある。選択の良し悪しは先述の通り評価関数を用いて行う。悪い選択である事象 X の場合は 式 (3.3) の観測事象 x が悪い選択となり、θは確率変数であるため p を代入する。また $\pi(\theta)$ の事前分布は $B(p)$ となり、$f(x|\theta)$ である尤度は尤度関数として $L(n, k, p)$ となる 事後確率を $A(p|X)$ すると、上記を代入した場合の式が式 (3.5) となり、悪い選択の場合のベイズ更新には (3.5) を用いる。

$$A(p|X) = \frac{L(n,k,p)B(p)}{\int_p L(n,k,p)B(p)dp} \tag{3.5}$$

良い選択の場合には式 (3.5) とは尤度関数の部分を変える必要がある。$L(n,k,p)$ は悪い選択をする確率が確率 p であるときのもっともらしさであるため、良い選択をするもっともらしさは $1 - L(n,k,p)$ とする。

よって式 (3.6) となる。そのため、良い選択の場合のベイズ更新には (3.6) を用いる。

$$A(p|Y) = \frac{(1 - L(n,k,p))B(p)}{\int_p (1 - L(n,k,p))B(p)dp} \tag{3.6}$$

これを 1 回の行動で選択された各中間ノードで行う。ベイズ更新は繰り返し行うため、次のベイズ更新を行う際の事前分布は、今回算出された事後分布を用いる。行動をするたびに繰り返しベイズ更新を行うことで中間ノード毎に確率分布が更新され、各中間ノードの選択の良し悪しの傾向がわかる。

3.6 調節の必要な中間ノードの発見

中間ノードの選択結果が悪い選択となる確率は事後分布の確率分布から知ることができる。事後分布の分布は悪い選択となる確率を表す p の確率分布であるため、この分布で p の確率が高い部分が $p = 1$ に近いほど悪い選択を行いやすい中間ノードである。そのため確率分布の形状から悪い選択を起こしやすい中間ノードを見つけることができる。

図 3.5 と図 3.6 はベイズ更新を行った例である。横軸で左側が 0、右側が 1 である。図 3.5 の方では $p = 0.36$ 付近の山が一番高く、鋭くとがっている。図 3.6 の方では $p = 0.32$ 付近が高くなっているが緩やかな山となっている。

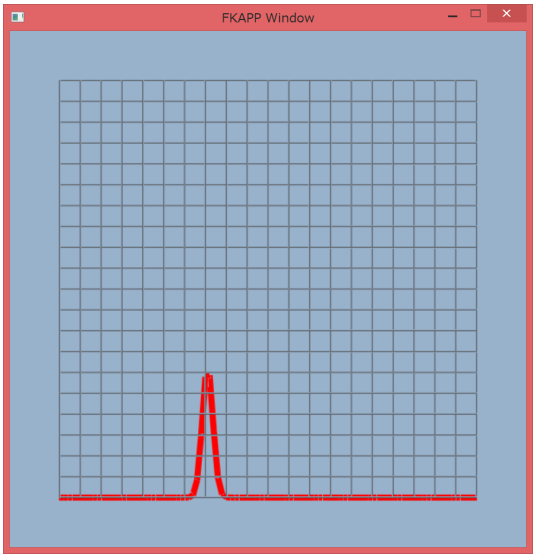

図 3.5　ベイズ更新の例 1

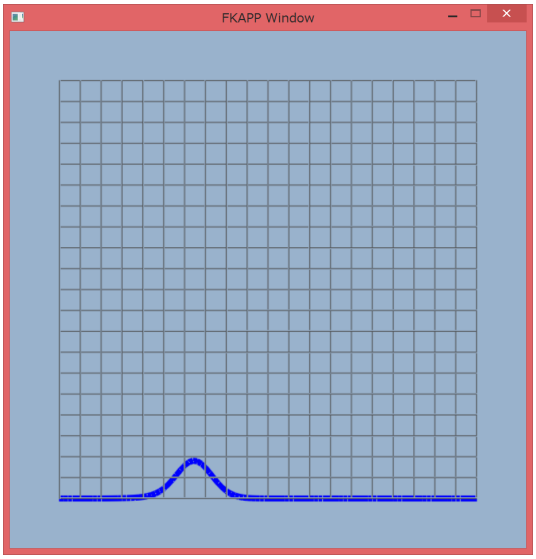

図 3.6　ベイズ更新の例 2

分布に現れる山の高さと鋭さによりその値 p が真値であるのかを知ることができる。山の形状が鋭い場合はベイズ推定がある程度収束したことを表し、形状が穏やかな山の場合はベイズ推定により推定を行ってはいるが、まだ収束していないため十分に推定ができておらず確度があまり高くないことを表している。そのため、図 3.5 と図 3.6 では p が $p = 0.36$ と $p = 0.32$ と近しい値で分布の山を形成しているが、より鋭い山の形状をしている図 3.5 の方が確度が高く信頼性が高い。

本手法では確度の高く悪い選択となる確率を表す p が 1 に近い、つまり山が鋭い分布を持ち、その山が右側にある中間ノードほど悪い選択を高確率で起こしやすい中間ノードである。そのため、調節の必要性が高い中間ノードであると考えられる。このようにベイズ更新を行い事後分布を確認することで悪い選択を行いやすい中間ノード、つまり調節の必要性が高い中間ノードを見つけ出すことができる。図 3.7 が上記をまとめた図である。

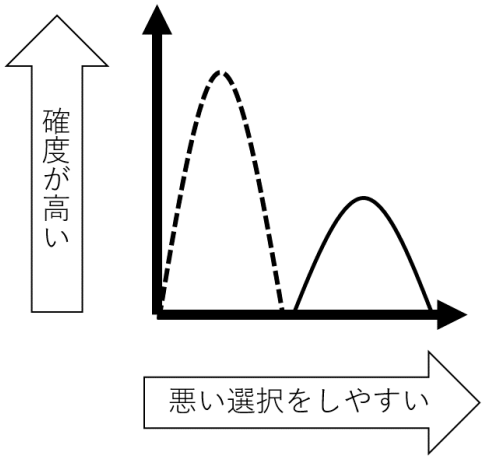

図 3.7　確率分布からわかること

第 4 章

検証

本章の検証は C#を用いて作成した簡易的なコマンド式ターン性バトルのゲームを用いて行った。ターン性のバトルを採用した理由としては、リアルタイムのアクションゲームではキャラクターの 1 回の行動を明確に定義しづらいためである。そのため、今回はターン性のバトルを採用し 1 ターンに起こしたアクションを行動とした。このアクションを起こす前後の状況から行動の良し悪しの評価を行う。

4.1 作成したゲーム

検証で用いる作成したゲームについて簡単に記述する。作成したゲームは簡易的なコマンド式ターン性バトルを採用している。プレイヤー側キャラクターと敵側キャラクターの 2 つの陣営に分かれて戦闘を行い、相手陣営のキャラクターすべての体力を 0 にすることで戦闘に勝利する。本検証ではプレイヤー側のキャラクターもビヘイビアツリーの AI によって行動を決定する。

各キャラクターはステータスとして体力を表す HP、アクションを行う際にコストとして消費される MP、物理的な攻撃力を示す ATK と魔法攻撃力の $MATK$、物理防御力を示す DEF と魔法防御力の $MDEF$、速さを示す SPD を持つ。表 4.1 が、上記のステータスをまとめた表である。

表 4.1 キャラクターのステータス

名称	内容
HP	キャラクターの体力を表し 0 になると負ける
MP	アクションを行う際のコストとして消費される
ATK	物理攻撃力
$MATK$	魔法攻撃力
DEF	物理防御力
$MDEF$	魔法防御力
SPD	行動を起こす速さ

戦闘によるダメージの計算は物理攻撃なら攻撃を行うキャラクターの ATK 引く、攻撃をされ

るキャラクターの DEF で算出し、算出された値が 0 より大きい場合にその数値分だけ攻撃をされるキャラクターの HP を減らす。魔法攻撃の場合は攻撃を行うキャラクターの $MATK$ 引く、攻撃をされるキャラクターの $MDEF$ で算出を行う。

各キャラクターは 1 ターンに 1 回ずつ行動を起こす。行動は SPD の数値が高い順に行う。ベイズ推定を行う際のキャラクターの行動選択の良し悪しは、プレイヤー側の HP 量と敵側の HP 量、プレイヤー側と敵側のキャラクターのステータスの数値差から判定した。

4.2 利点の検証

プレイヤー側のキャラクター 2 種と敵側のキャラクター 1 種を戦わせ、敵側のキャラクターのビヘイビアツリーの評価を行った。プレイヤー側のキャラクターはそれぞれ P_1と P_2 と表記し、敵側のキャラクターは E と表記する。

P_1 は物理攻撃を主体とし、P_2 は回復と魔法攻撃を行うようにした。E は物理攻撃、魔法攻撃、回復を平均的に使用するようにした。

P_1 には 5 つのアクションがあり、弱物理攻撃、強物理攻撃、MP 回復、バフ、デバフの 5 つである。表 4.2 が上記のアクションをまとめた表である。

表 4.2　P_1 のアクション

名称	効果
弱物理攻撃	ATK の数値分の物理攻撃を行う
強物理攻撃	MP を 5 消費し ATK の 3 倍の物理攻撃を行う
MP 回復	MP を 20 回復する
バフ	MP を 10 消費して自分のステータスを一時的に上げる
デバフ	MP を 5 消費して相手のステータスを一時的に下げる

P_1 のビヘイビアツリーは開始となる root ノードから始まり、root ノードの下位に選択ルールが条件ルールの中間ノードがひとつある。この中間ノードを N_{11} とする。N_{11} の条件は MP の

値によって分岐を行う。N_{11} は 1 つの中間ノードと 1 つのアクションノードを持つ。MP の値が 5 より大きい場合は下位の中間ノードを選択する。この中間ノードを N_{12} とする。MP の値が 5 より小さい場合は下位のアクションノードを選択する。このアクションノードに設定したアクションは MP 回復である。

N_{12} の選択ルールは条件ルールである。N_{12} は 1 つのアクションノードと 1 つの中間ノードを持つ。MP の値が 10 より大きく、現在自分にバフが掛かっていない場合は下位のアクションノードを選択する。このアクションノードに設定したアクションはバフである。上記の場合以外は下位の中間ノードを選択する。この中間ノードを N_{13} とする。

N_{13} の選択ルールはランダムルールである。N_{13} は 3 つのアクションノードを持つ。N_{13} は 3 つのアクションノードを等確率で選択する。各アクションノードに設定したアクションは弱物理攻撃、強物理攻撃、デバフである。図 4.1 が上記の構造を表したビヘイビアツリーである。

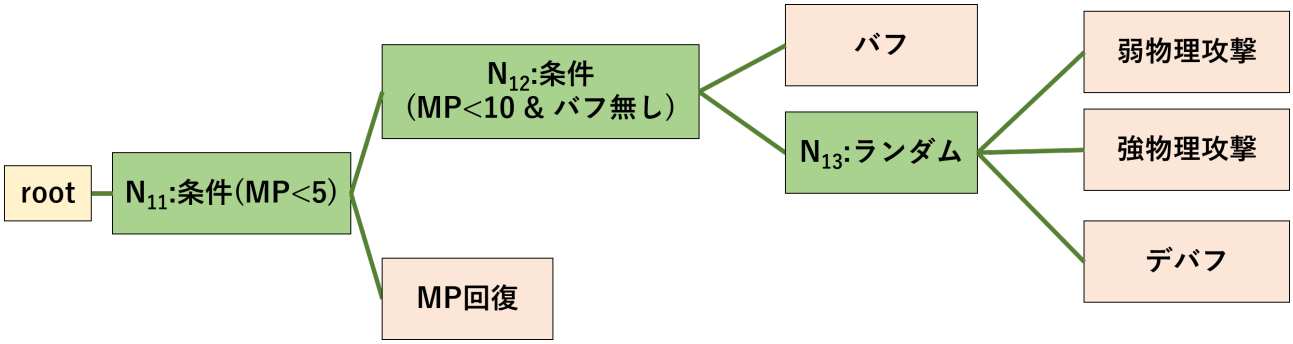

図 4.1　$P1$ のビヘイビアツリー

P_2 には 7 つのアクションがあり、物理攻撃、弱魔法攻撃、強魔法攻撃、MP 回復、弱 HP 回復、強 HP 回復、全体 HP 回復の 7 つである。表 4.3 が上記のアクションをまとめた表である。

表 4.3　P_2 のアクション

名称	効果
物理攻撃	ATK の数値分の物理攻撃を行い、MP を 5 回復
弱魔法攻撃	MP を 3 消費し $MATK$ の数値分の物理攻撃を行う
強魔法攻撃	MP を 10 消費し $MATK$ の 3 倍の物理攻撃を行う
MP 回復	MP を 30 回復する
弱 HP 回復	MP を 5 消費し対象の味方キャラクターの HP を 20 回復
強 HP 回復	MP を 15 消費し対象の味方キャラクターの HP を 50 回復
全体 HP 回復	MP を 20 消費し味方キャラクター全員の HP を 20 ずつ回復

P_2 のビヘイビアツリーは開始となる root ノードから始まり、root ノードの下位に選択ルールが条件ルールの中間ノードがひとつある。この中間ノードを N_{21} とする。N_{21} は 1 つのアクションノードと 2 つの中間ノードを持つ。MP の値が 15 より小さい場合は下位のアクションノードを選択する。このアクションノードに設定したアクションはを MP 回復である。MP の値が 15 より大きく、1 キャラクターでも HP が半分以下の味方キャラクターがいる場合、または味方キャラクター全体の合計 HP が半分以下の場合は下位の中間ノードのひとつを選択する。この中間ノードを N_{22} とする。上記の場合以外は下位にあるもうひとつの中間ノードを選択する。この中間ノードを N_{23} とする。

N_{22} の選択ルールは条件ルールである。N_{22} は 1 つのアクションノードと 1 つの中間ノードを持つ。MP の値が 20 より大きく、味方キャラクター全体の合計 HP が半分以下の場合は下位のアクションノードを選択する。このアクションノードに設定したアクションはを全体 HP 回復である。上記の場合以外は下位の中間ノードを選択する。この中間ノードを N_{24} とする。

N_{23} の選択ルールは条件ルールである。N_{23} は 1 つのアクションノードと 1 つの中間ノードを持つ。MP の値が 30 より小さい場合は下位のアクションノードを選択する。このアクションノードに設定したアクションはを物理攻撃である。上記の場合以外は下位の中間ノードを選択する。この中間ノードを N_{25} とする。

N_{24} の選択ルールは条件ルールである。N_{24} は 2 つのアクションノードを持つ。MP の値が 15 より小さい、または味方キャラクターの HP が 25% より多い場合は下位の 1 つ目のアクションノードを選択する。このアクションノードに設定したアクションはを弱 HP 回復である。上記の場合以外は下位にあるもうひとつのアクションノードを選択する。このアクションノードに設定したアクションはを強 HP 回復である。

N_{25} の選択ルールは条件ルールである。N_{25} は 2 つのアクションノードを持つ。MP の値が 30 より小さい場合は下位の 1 つ目のアクションノードを選択する。このアクションノードに設定したアクションはを弱魔法攻撃である。上記の場合以外は下位にあるもうひとつのアクションノードを選択する。このアクションノードに設定したアクションはを強魔法攻撃である。図 4.2 が上記の構造を表したビヘイビアツリーである。

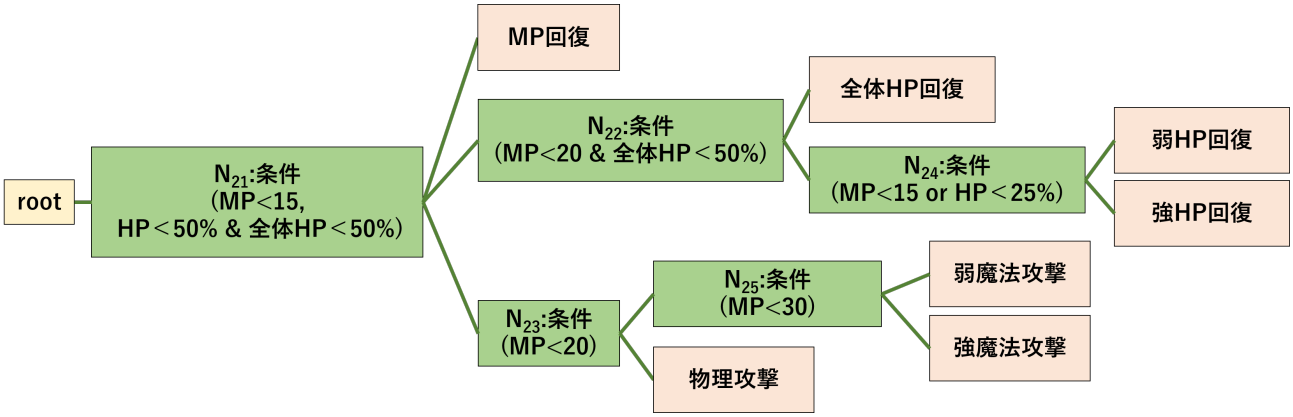

図 4.2　$P2$ のビヘイビアツリー

E には 9 つのアクションがあり、単体物理攻撃、単体魔法攻撃、全体物理攻撃、全体魔法攻撃、ランダム物理攻撃、ランダム魔法攻撃、バフ、デバフ、HP 回復の 9 つである。表 4.4 が上記のアクションをまとめた表である。

表 4.4　E のアクション

名称	効果
単体物理攻撃	ATK の数値分の物理攻撃を行う
単体魔法攻撃	$MATK$ の数値分の魔法攻撃を行う
全体物理攻撃	相手全員に ATK の半分の数値分の物理攻撃を行う
全体魔法攻撃	相手全員に $MATK$ の半分の数値分の魔法攻撃を行う
ランダム物理攻撃	ランダムに相手へ ATK の半分の数値分の物理攻撃を 3 回行う
ランダム魔法攻撃	ランダムに相手へ ATK の半分の数値分の魔法攻撃を 3 回行う
バフ	自分のステータスを一時的に上げる
デバフ	相手のステータスを一時的に下げる
HP 回復	HP を回復

E のビヘイビアツリーは開始となる root ノードから始まり、root ノードの下位に選択ルールが順番ルールの中間ノードがひとつある。この中間ノードを N_{31} とする。N_{31} は 3 つの中間ノードを持ち、選択される順番から下位の 3 つの中間ノードを N_{32}, N_{33}, N_{34} とする。

N_{32} の選択ルールはランダムルールである。N_{32} は 1 つのアクションノードと 1 つの中間ノードを持つ。下位のアクションノードと中間ノードを等確率で選択する。アクションノードに設定したアクションは単体物理攻撃であり、下位の中間ノードは N_{35} とする。

N_{33} の選択ルールはランダムルールである。N_{33} は 1 つのアクションノードと 1 つの中間ノードを持つ。下位のアクションノードと中間ノードを等確率で選択する。アクションノードに設定したアクションは単体魔法攻撃であり、下位の中間ノードは N_{36} とする。

N_{34} の選択ルールはランダムルールである。N_{34} は 1 つのアクションノードと 1 つの中間ノードを持つ。下位のアクションノードと中間ノードを等確率で選択する。アクションノードに設定したアクションは HP 回復であり、下位の中間ノードは N_{37} とする。

N_{35} の選択ルールは順番ルールである。N_{35} は 2 つのアクションノードを持つ。下位にある 2 つのアクションノードを順番に選び、設定したアクションはそれぞれ全体物理攻撃とランダム物

理攻撃である。

N_{36} の選択ルールは順番ルールである。N_{36} は 2 つのアクションノードを持つ。下位にある 2 つのアクションノードを順番に選び、設定したアクションはそれぞれ全体魔法攻撃とランダム魔法攻撃である。

N_{37} の選択ルールは条件ルールである。N_{37} は 2 つのアクションノードを持つ。自分にバフが掛かっていない場合はバフを設定したアクションノードを選択し、既に自分にバフが掛かっている場合はデバフを設定したアクションノードを選択する。図 4.3 が上記の構造を表したビヘイビアツリーである。

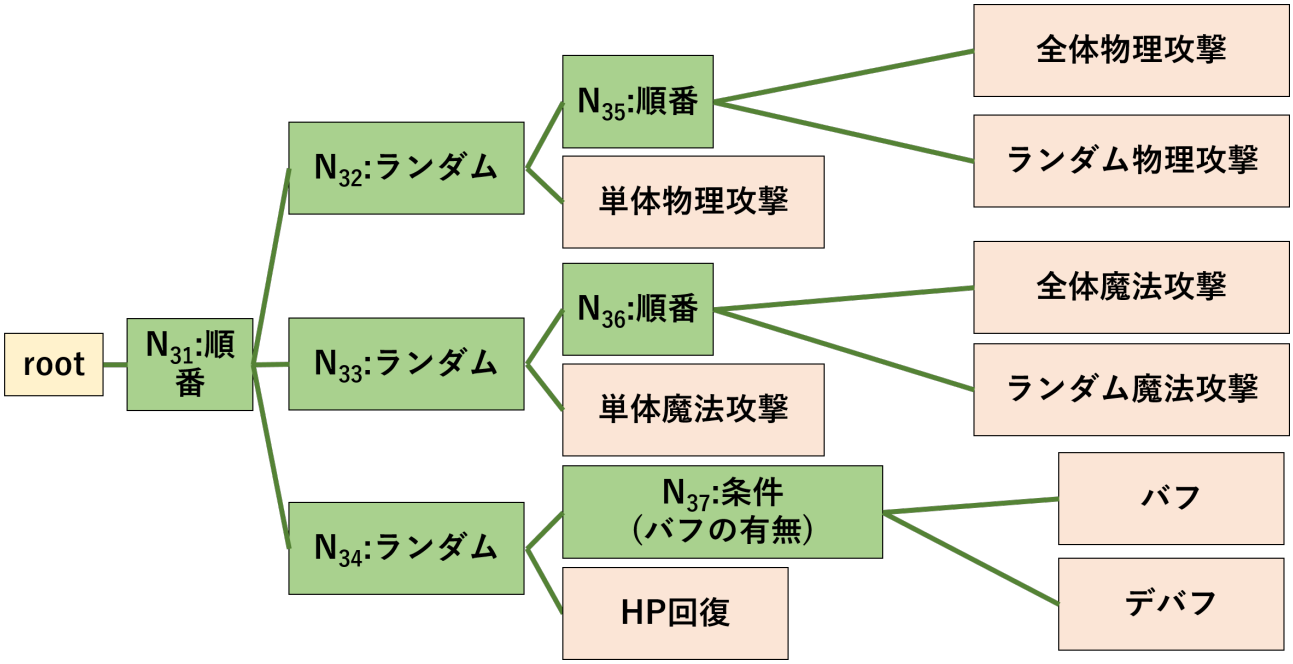

図 4.3　E のビヘイビアツリー

この上記の 3 キャラクターを戦わせ、E のビヘイビアツリーの評価を行った。本節の検証では 3 キャラクターのステータスを下記の表 4.5 にある数値で検証を行った。

表 4.5　各キャラクターのステータス

キャラクター	HP	MP	ATK	$MATK$	DEF	$MDEF$	SPD
P_1	300	100	10	0	4	1	10
P_2	200	200	1	10	0	5	10
E	1000	–	10	10	1	1	5

本節の検証ではベイズ推定の結果が収束しやすいように比較的ターン数が多くなるように数値を設定した。ランダムに行動を選択する中間ノードがあるため複数回評価を行い、ベイズ推定後の確率分布を比較した。評価の結果は 1 回目から n 回目を順に $T_1, T_2, ...T_n$ として表記する。

図 4.4 が T_1 の評価結果を示したものである。

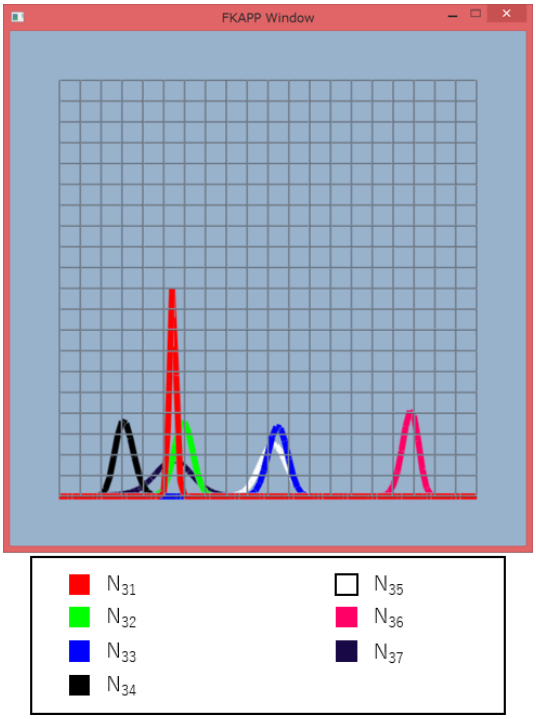

図 4.4　T_1 の評価結果

T_1 では 269 ターンでプレイヤー側が勝利した。図の各曲線が確率分布を表し、中間ノードをそれぞれ示している。T_1 では図 4.4 から分かるように 1 つの分布の山が右側である $p = 1$ の近くになった。この分布が指す中間ノードが悪い選択を起こしやすい中間ノードであると考えられる。

次に T_2 の評価結果である。図 4.5 が T_2 の評価結果を示したものである。

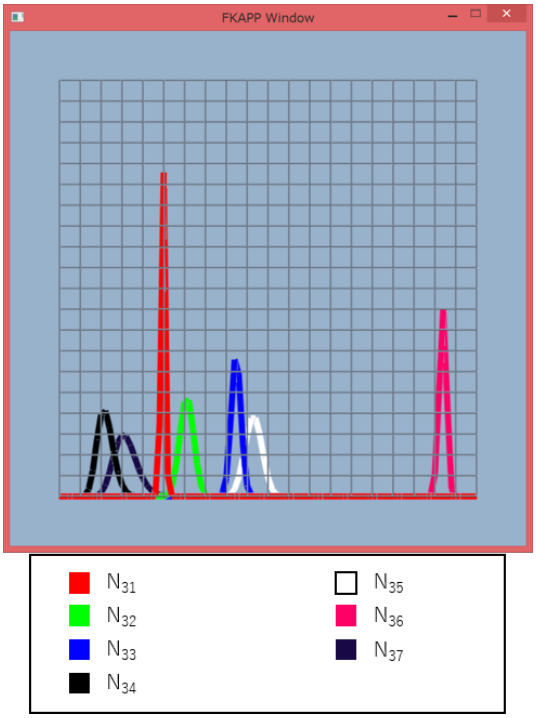

図 4.5　T_2 の評価結果

T_2 では 289 ターンでプレイヤー側が勝利した。T_1 と比較すると多少の差はあるが、各分布で確率 p の値がかなり近い値となっている。

次に T_3 の評価結果である。図 4.6 が T_3 の評価結果を示したものである。

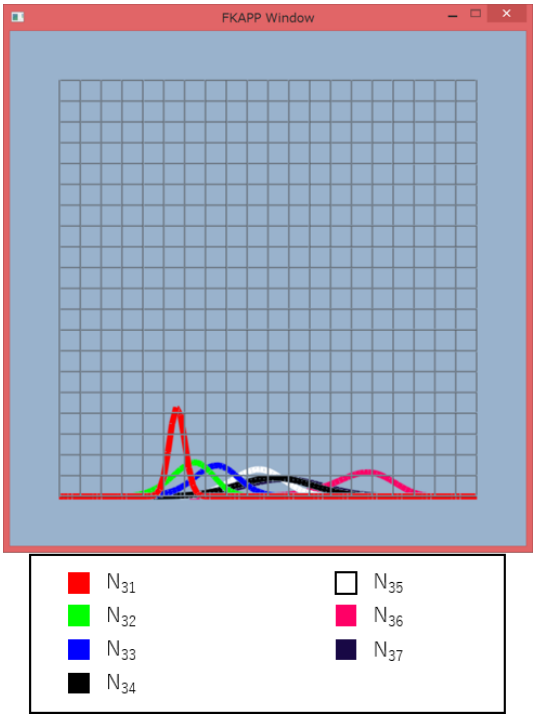

図 4.6　T_3 の評価結果

T_3 では 93 ターンでプレイヤー側が勝利した。T_1 と T_2 の評価で算出した確率 p が真値であると仮定し、比較すると図 4.6 の分布では真値から外れた場所で山を形成している中間ノードがあることが分かる。T_3 では T_1 と T_2 の分布と違い、山の形状がかなり緩やかなものとなっている。このことからベイズ推定があまり行えず、推定が不十分であることが分かる。頻度による算出では数値のみのため信用度を知ることは難しいが、本手法を用いた中間ノードの評価では確率分布を用いるため容易に信用度を知ることが可能である。そのため、頻度による評価よりも本手法を用いる方が信用度を容易に知ることができるため有用である。

4.3　調節への有用性

T_1 と T_2 の結果から悪い選択を行いやすい中間ノードが分かった。その調節の必要性が高い中間ノードは N_{36} である。この調節の必要性が高い N_{36} を調節することで AI を上手く調節できるのか検証を行った。

調節の方法としては中間ノードの選択ルールを変更することや選択確率の変更を行うことで中間ノード自体を調節する方法、その中間ノードの下位にあるノードを調節する方法の 2 つが考えられる。まず中間ノード自体の調節を行った。選択ルールをランダムルールにし、選択確率を変更し再度評価を行った。結果は中間ノードの選択ルールを変更しただけでは改善できなかった。改善できなかった理由としてその中間ノードの下位にある 2 つのアクションが相手に攻撃を行うアクションであるが、このアクション自体が相手側にダメージを与えることがあまりできない比較的弱いアクションであるため、状況に関わらずどちらのアクションが選ばれた場合でも高確率で悪い行動であると評価されることであるとわかった。そのため次の調節方法として、下位にある 2 つのアクション自体を強くなるように調節を行った。その結果中間ノードの悪い選択確率は下がり AI の改善ができた。この結果を T_4 とし、図 4.7 で示す。

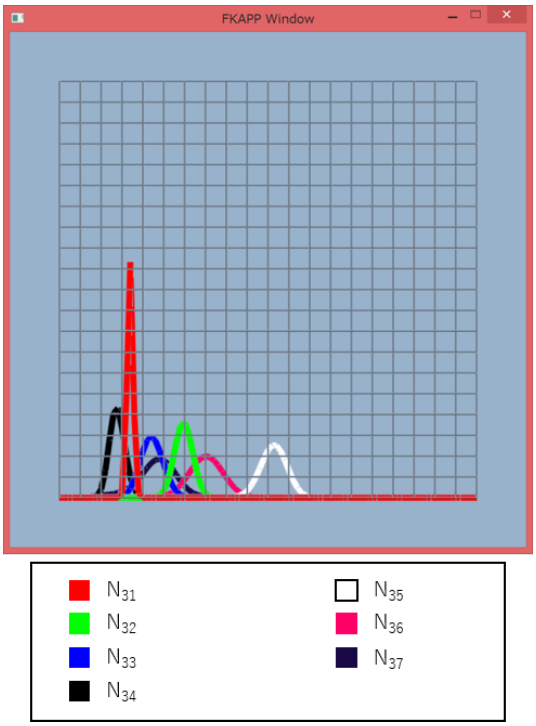

図 4.7　T_4 の評価結果

図から分かるように T_1 と T_2 の分布では右側にあった分布が中央付近になり、悪い選択をする確率が減ったことが分かる。

本検証に用いたプレイヤー側と敵側のキャラクターで戦闘を行う場合、必ずプレイヤー側が勝利するように作成した。そのため、ターン数を計測することで敵側のキャラクターの強さを判定できると考えられる。調節前のビヘイビアツリーと調節後のビヘイビアツリーを用いて 100 回ずつ戦闘を行い平均ターン数を計測した。その結果、調節前は平均 192.63 ターン、調節後では 223.71 ターンとなり約 30 ターンも増加したことが分かった。よって、キャラクターを強くしようと AI を調節した場合上手く調節ができたといえ、本手法による評価は十分有用性が高いと考えられる。

4.4　巨大なビヘイビアツリーでの評価

確度の高い中間ノードの確率推定のためにはベイズ推定を多く行う必要がある。ベイズ推定を行うのは各中間ノードが選択されたときである。つまり選択される回数が少ない場合は確度の高い推定ができない。そのため本節では上記の検証に用いたビヘイビアツリーよりも大きなビヘイビアツリーを持つ AI ではどの程度のターン数が必要であるか検証を行った。

検証には中間ノードを 25 個、アクションノードを 30 個持つビヘイビアツリーを使用した。図 4.8 は使用したビヘイビアツリーを表したものである。

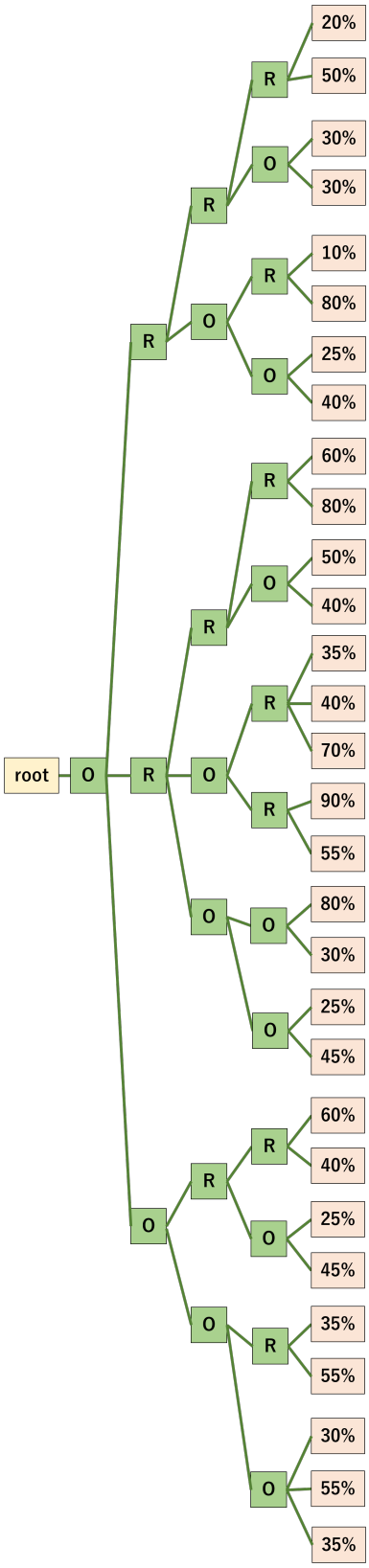

図 4.8　巨大なビヘイビアツリー

中間ノードに表記してある O は、その中間ノードの選択ルールが順番ルールであることを示している。また、中間ノードに表記してある R は、その中間ノードの選択ルールがランダムルールであることを示している。アクションノードに表記してある確率はそのアクションが選択された際に悪い選択であると判定する確率である。

このビヘイビアツリーの中で最も下位の階層にある 14 個の中間ノードの評価を行った。図 4.9 が評価を行い算出された確率分布を示したものである。

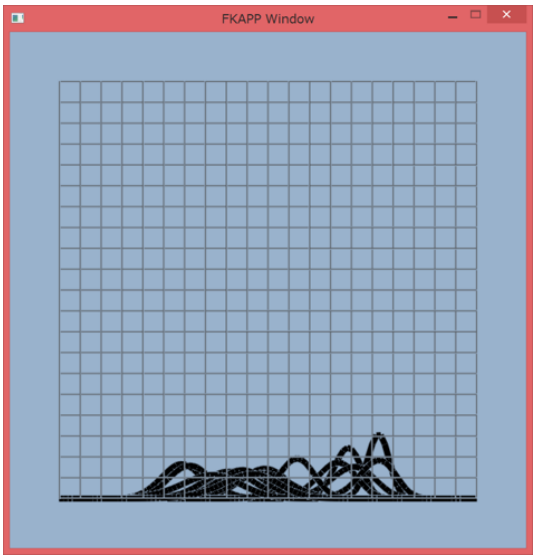

図 4.9　300 ターンの状態

図 4.9 では 300 ターン行動を起こしベイズ推定をした結果である。第 4.2 節の検証結果では既に高い確度となっていた 300 ターンだが、巨大なビヘイビアツリーの場合は全ての分布の確度がまだ低いことが分かる。図 4.10 で、さらに行動を起こし 1000 ターン行動を起こした結果を示す。

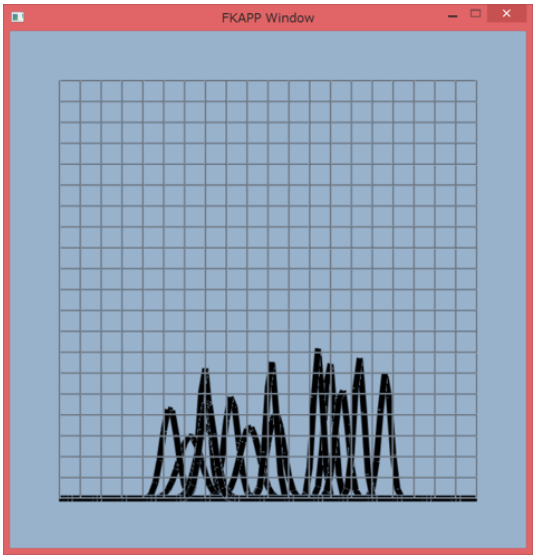

図 4.10　1000 ターンの状態

ベイズ推定を繰り返し行ったので確度が先ほどと比べ高くなったが、確度が十分高くなったとはまだ言えない。ベイズ推定をさらに続けて行い、図 4.11 で合計 1500 ターンまで行動を起こし評価したものを示す。

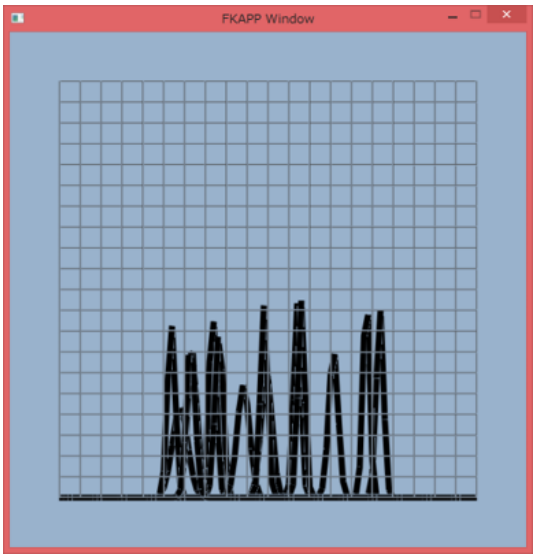

図 4.11　1500 ターンの状態

1500 ターン繰り返した結果、分布の山が鋭くなり確度が高まったことが分かる。このようにビヘイビアツリー全体が大きくなった場合、細部まで評価を行うには多くの回数が必要となること

が分かった。また、処理時間の測定も行った。表 4.6 が測定を行った際の実行環境である。

表 4.6　実行環境

CPU	Intel(R) Core(TM) i5-5200U CPU @ 2.20 GHz
メモリ	4.00GB
GPU	Intel(R) HD Graphics Family

各ターン数で 10 回計測を行い、その平均時間を算出した。300 ターンでは平均 1.860 秒、1000 ターンでは平均 4.421 秒、1500 ターンでは平均 6.098 秒となった。

4.5　結論と考察

検証結果から本手法を用いることでビヘイビアツリーの各中間ノードを評価することが可能であると分かった。また、頻度による確率の算出よりも本手法の確率分布を用いることで、その算出した確率がどの程度信頼できるのか容易に判断できるため、本手法の方が有用である。本手法では巨大なビヘイビアツリーの AI では中間ノードの評価を行う際に非常に多くの行動回数を必要とする性質がある。

本論文ではターン性コマンド式のバトルを用いて検証を行ったが、リアルタイムアクションのゲームや他のゲームなどで使用するビヘイビアツリーの AI に本手法を適応する場合、1 回の行動の定義など多少工夫の必要があると考えられる。また、本手法では行動の良し悪しの判定に評価関数を用いているため、評価関数によってベイズ推定の結果が大きく左右される可能性が高いと考えられる。そのため、評価関数を上手く作成する必要性が高いという欠点がある。

結論としては、本手法を用いることで中間ノードの評価を容易に行うことが可能となり調節の必要なノードを簡単に見つけることができる。そのため、調節作業を行いやすくするという本研究の目的を達成できた。

第 5 章

まとめ

最近のハードウェアなどの技術が向上した。そのため、デジタルゲームは昔と比べ複雑化しそれに合わせて AI も複雑になった。AI の調節はゲーム全体の完成度を大きく左右する要因であるが、近年使用されることが増えたビヘイビアツリーを用いた AI では調節を行うことが非常に困難である。

そこで本研究は調節の難しいビヘイビアツリーの調節を行いやすくするために、ビヘイビアツリーの各中間ノードを評価し調節の必要な箇所を簡単に発見する手法について提案した。

本手法ではベイズ推定を用いることで中間ノードの評価を行った。中間ノードの評価では中間ノードがどの程度悪い選択をするのかを推定することで行った。推定結果から中間ノード毎に悪い選択のしやすさを検出することができた。また、本手法では確率分布で算出するため頻度による確率の算出より信用度を知ることができるため有用である。

本手法では中間ノードを評価し調節の必要性が高い中間ノードを見つけることができ、その中間ノードを調節することで上手く AI を調節することができた。ビヘイビアツリーが大きくなるほど精度の高い推定を行うまでに時間が多くかかることが分かった。

しかし、本論文の検証では限定的なゲームでの検証しか行えていないため、他のゲームシステムを持つゲームに適応できるのかは不明である。

謝辞

本研究を締めくくるにあたり、ご指導下さいました先生方に心より感謝致します。

様々な相談や息抜きに応じて下さった、研究室のメンバーに深く感謝致します.

最後に、研究という非常に辛い状態の中で共に頑張り駆け抜けた仲間たちに感謝と最大の讃辞を送ります。ありがとう！そしてよく頑張った！

参考文献

[1] スーパーマリオブラザーズ. https://www.nintendo.co.jp/ngc/sms/history/sm1/. 参照:2019.11.13.

[2] FINAL FANTASY XV （ファイナルファンタジー 15） — SQUARE ENIX. http://www.jp.square-enix.com/ff15/. 参照:2019.11.13.

[3] Damian Isla. GDC 2005 Proceeding: Handling Complexity in the Halo 2 AI. http://www.gamasutra.com/view/feature/130663/gdc_2005_proceeding_handling_.php. 参照:2019.11.15.

[4] 眞鍋和子, 三宅陽一郎. 遺伝的アルゴリズムを用いたプレイヤーエージェントによるデジタルゲームのバランス調整. *SIG-SAI*, Vol. 30, pp. 1–6, 2017.

[5] 山本界人, ターウォンマットラック. StGA を適用した観察用 AI を用いた自動ゲームバランス分析. 2014 年度 情報処理学会関西支部 支部大会 講演論文集, Vol. 2014, , 2014.

[6] Michael Cook, Simon Colton, and Jeremy Gow. The ANGELINA Videogame Design System—Part I. *IEEE Transactions on Computational Intelligence and AI in Games*, Vol. 9, pp. 192–203, 2016.

[7] Petter Ögren. Increasing Modularity of UAV Control Systems using Computer Game Behavior Trees. *AIAA Guidance, Navigation, and Control Conference 2012*, 2012.

[8] Shuang Liang, Eddie C L Chan, George Baciu, and Rong-Hua Li. Cognitive garment design interface using user behavior tree model. *9th IEEE International Conference on Cognitive Informatics (ICCI'10)*, pp. 496–500, 2010.

[9] Michele Colledanchise, Alejandro Marzinotto, and Petter Ögren. Performance analysis of stochastic behavior trees. *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3265–3272, 2014.

[10] Yang Li and Daiwen Xu. A game AI based on ID3 algorithm. *2016 2nd International*

Conference on Contemporary Computing and Informatics (IC3I), pp. 681–687, 2016.

[11] Makoto Ishihara, Suguru Ito, Ryota Ishii, Tomohiro Harada, and Ruck Thawonmas. Monte-Carlo Tree Search for Implementation of Dynamic Difficulty Adjustment Fighting Game AIs Having Believable Behaviors. *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, pp. 1–8, 2018.

[12] 佐藤悠. 協調ユーティリティを用いたゲーム AI の調整の簡易化. 第 75 回全国大会講演論文集, Vol. 2013, No. 1, pp. 105–106, 2013.

[13] 仲道隆史, 伊藤毅志. プレイヤの技能に動的に合わせるシステムの提案と評価. 情報処理学会論文誌, Vol. 57, No. 11, pp. 2426–2435, 2016.

[14] 中川明紀, 逢坂翔太, 柴崎智哉, Ruck Thawonmas. ニューラルネットワークによる格闘ゲーム AI の難易度調整及び行動多様性向上手法. ゲーム学会和文論文誌, Vol. 3, No. 1, pp. 35–40, 2009.

[15] 杉本直樹, 鶴岡慶雅. 戦略の動的推定による 2 人対戦ゲーム接待 AI の提案. ゲームプログラミングワークショップ 2018 論文集, Vol. 2018, pp. 114–119, 2018.

[16] 星野准一, 田中彰人, 濱名克季. 模倣学習により成長する格闘ゲームキャラクタ. 情報処理学会論文誌, Vol. 49, No. 7, pp. 2539–2548, 2008.

[17] 藤井叙人, 佐藤祐一, 中嶌洋輔, 若間弘典, 風井浩志, 片寄晴弘. 生物学的制約の導入による「人間らしい」振る舞いを伴うゲーム AI の自律的獲得. ゲームプログラミングワークショップ 2013 論文集, pp. 73–80, 2013.

[18] 張輝陽, 星野准一. プレイヤ行動の模倣に基づく AI キャラクタ行動ルールの自動生成. 情報処理学会研究報告. GI, [ゲーム情報学], Vol. 2014, No. 5, pp. 1–4, 2014.

[19] 加納由希夫, 鶴岡慶雅. 内部報酬と Hybrid Reward Architecture を用いたローグライクゲームの強化学習. ゲームプログラミングワークショップ 2018 論文集, No. 2018, pp. 64–71,

2018.

[20] Chong-U Lim, Robin Baumgarten, and Simon Colton. Evolving Behaviour Trees for the Commercial Game DEFCON. *Applications of Evolutionary Computation*, pp. 100–110, 2010.

[21] Diego Perez, Miguel Nicolau, MichaelO' Neill, Anthony Brabazon. Evolving Behaviour Trees for the Mario AI Competition Using Grammatical Evolution. *Proceedings of the 2011 International Conference on Applications of Evolutionary Computation*, Vol. 1, pp. 123–132, 2011.

[22] Michele Colledanchise, Ramviyas Parasuraman, and Petter Ögren. Learning of Behavior Trees for Autonomous Agents. *IEEE Transactions on Games*, Vol. 11, pp. 183–189, 2019.

[23] Qi Zhang, Kai Xu, Peng Jiao, and Quanjun Yin. Behavior Modeling for Autonomous Agents Based on Modified Evolving Behavior Trees. *2018 IEEE 7th Data Driven Control and Learning Systems Conference (DDCLS)*, pp. 1140–1145, 2018.

[24] 上田陽平. GA を用いたプレイヤーのレベルに適応する多様なオセロ AI の開発. 修士論文, 北陸先端科学技術大学院大学情報科学研究科情報科学専攻, 2012.

[25] 福嶋良平, 藤井叙人, 片寄晴弘. GA を用いた NPC の多様な振舞いの生成. エンタテインメントコンピューティングシンポジウム 2015 論文集, Vol. 2015, pp. 197–205, 2015.

[26] 三宅陽一郎. 人工知能の作り方 「おもしろい」ゲーム AI はいかにして動くのか. 株式会社技術評論社, 東京, 2017.

[27] 株式会社スクウェア・エニックス『FFXV』AI チーム. FINAL FANTASY 15 の人工知能 ゲーム AI から見える未来. 株式会社ボーンデジタル, 東京, 2019.

[28] 三宅陽一郎. ゲーム AI 技術入門 広大な人工知能の世界を体系的に学ぶ. 株式会社技術評論

社, 東京, 2019.

[29] masa. ベイズの定理の導出と考え方をわかりやすく解説 — 全人類がわかる統計学. https://to-kei.net/bayes/bayes-theorem/. 参照:2019.11.13.

[30] ベイズ統計学基礎 — logics of blue. https://logics-of-blue.com/%E3%83%99%E3%82%A4%E3%82%BA%E7%B5%B1%E8%A8%88%E5%AD%A6%E5%9F%BA%E7%A4%8E/. 参照:2019.11.14.

[31] 10-4. ベイズの定理 — 統計学の時間 — 統計 web. https://bellcurve.jp/statistics/course/6444.html. 参照:2019.11.14.