修士論文

強化学習を用いたターン制 RPG の多様なステージ自動生成

1610424　ナム サンギュ

主指導教員　池田 心
審査委員主査　池田 心
審査委員　飯田 弘之
長谷川 忍
白井 清昭

北陸先端科学技術大学院大学
先端科学技術研究科
(情報科学)

2019 年 08 月

Abstract

ゲームを対象とした人工知能領域の研究は，多様なゲームの強いプレイヤー生成を主な目的としていた．まだ多少の課題が残ってはいるが，チェスの DeepBlue，囲碁の AlphaGo，Starcraft の AlphaStar，Dota2 の OpenAI Five 等から強いプレイヤーの生成はある程度の成果が挙げられたと言える．それゆえ，近年では人間らしい AI プレイヤーや楽しませる AI プレイヤー，または教える AI プレイヤーを生成するなど多様な目的での AI の開発研究が行われている．加えて，ゲーム領域においては AI プレイヤー生成だけではなくゲームコンテンツ（シナリオ，敵味方の配置や強さ，地形，マップ，レベル，ダンジョンなど）の自動生成の研究も盛んである．

本研究では強化学習を用いた多様なコンテンツの自動生成を主な目的とする．あるアルゴリズムを用いてコンテンツを自動生成することは Procedural Content Generation（PCG）と呼ばれ，ゲーム領域における重大な研究課題の 1 つとして扱われている．特に PCG に関して近年活発に行われるアプローチは，機械学習を用いたコンテンツ自動生成で，Procedural Content Generation via Machine Learning（PCGML）と言う．本研究は PCGML の一例に属する．

画像や自然言語などの他の分野におけるコンテンツの自動生成に良い結果を見せた Variational Autoencoders, PixelCNN, Generative Adversarial Networks のような生成モデルはゲームのコンテンツ自動生成に活用するにはいくつかの難点がある．このモデルは主に大量のコンテンツデータを訓練データとして活用し，そのデータの分布や表現に従うコンテンツを生成する．しかし実際に新しいゲームを開発しようとすると，訓練用のデータは一般的に充分ではない上それを生成するには膨大なコストがかかる．従って，我々はデータを必要としない生成方法に着目し，強化学習を PCG の手法として用いることを考えた．

本研究では，ターン制 RPG を対象ジャンルに設定し，離散のマスで構成されている簡略化した“ステージ”を自動生成することを目指す．ターン制 RPG のステージは各マスのパラメータが前後のマスのパラメータとお互いに密接に関わっているため，望ましいステージを生成することは研究として難しい挑戦である．本研究では，ステージの良さを示す評価関数自体は何らかの形で与えられていると仮定したうえで，その評価値が高くなるようなステージを高速かつ多様に自動生成するための強化学習の枠組みを提案する．

強化学習は，状態空間・行動空間・報酬関数・状態遷移ルールからなるマルコフ決定過程もとで，期待報酬を最大化するような方策を学習する枠組みである．本研究ではまず，RPG のステージ生成をマルコフ決定過程にモデル化した．生成途中のコンテンツを状態，コンテンツの未生成部分を生成する行為を行動，評価関

数による完成コンテンツの評価値を報酬として定義して，ステージ生成方策を学習可能にした．本来は面白いステージの評価やプレイヤーモデリングからの評価法などこれをデザインすること自体が 1 つの重要な研究である．しかし，本研究では評価法のデザインが主な目標ではないため，目標勝率に近いほど高評価を出す勝率適切度と，ターン制 RPG に対して思われる一般的な面白さなどから定義した複合適切度の 2 つを評価関数として用いた．

古典的な強化学習では状態と行動ペアの Q 値をテーブル型として保存するが，本研究の生成対象であるステージは 3 x 7 のイベントマス（戦闘マスや非戦闘マス）で構成されていて，とりうる値の範囲も考慮すると全状態行動対のテーブルを保持することは不可能である．そこで，同様の課題によく用いられる Convolution Neural Network（CNN）を利用することで Q 値を近似することにした．まず，“ステージからその評価値を推定する ”という近似タスクが可能なのかを確認するため，予備実験として 40000 個の訓練データを用いて勝率適切度を推定する教師あり学習を行ったところ，評価値の誤差 MSE は 0.0137 となった．これは出力の範囲 0〜1 を考えればかなり精度が高いと言える．より複雑な指標である複合適切度に，より少ない訓練データ 5500 を用いた場合も MSE は 0.0203 であり，十分な近似性能を持っていると判断した．

CNN がステージの良さを近似できることが分かったので，これを用いてステージ生成の強化学習を行った．強化学習の手法として，方策の表現形式が異なる Deep Q-Network（DQN）と Deep Deterministic Policy Gradient（DDPG）をそれぞれ実装し，その特徴から PCG への適用時にありうる長所と短所を把握した．DQN と DDPG の生成モデルは，複合適切度の評価関数でそれぞれ平均評価値 0.78 と 0.85 のステージを生成することに成功し，その生成ヒストグラムから十分良好なステージが生成されることを確認した．

強化学習が良好な方策すなわち生成行動を学習できることは分かったが，ステージ生成においては良好なだけでなく「複数作成したときに多様であること」も重要である．そこで本研究では“評価値が最善となる行動 ”から，質をできるだけ落とさずにできるだけ遠くにずらすようなノイズの導入法を提案した．そのうえで，実際に複数のステージを生成し，それらが (1) 十分な評価値を持っているか，そのうえで (2) どの程度違うパラメータを持っているか，さらには (3) プレイヤの取るべき戦略がどの程度違っているか，について評価した．この結果，平均評価値は 0.82 と良好なまま，パラメータベクトル同士の MSE が 0.55 となる異なるステージを生成できていること，またそれにより取るべき戦略も大きく変わっていることが確認できた．

目 次

図 目 次

表 目 次

第1章　はじめに

近年，人工知能（AI）の劇的な発展とともにAIは多様な領域の応用先で効果的に用いられている．その一例としてコンテンツの自動生成が挙げられる．AIを用いたコンテンツの自動生成は画像 [1, 2]，自然言語 [2]，音楽 [3]，音声など様々な領域において活発に研究されている．人々は多様な文化生活において膨大なコンテンツを消費していて，十分で良好なコンテンツの提供は彼らの要求を満足させるために必要である．しかし，コンテンツが複雑化・高品質化するにつれ，これらを手動で生成するには量的な観点でも質的な観点でも高コストになってきている．これらの課題を解決するためにコンテンツの自動生成をアルゴリズムを用いて行うことをProcedural Content Generation（PCG）と呼ぶ．

コンテンツが要求されるのはゲームにおいても同様であり，PCGはゲームエンタテインメント領域において主要な研究分野の1つである．ここでいうゲームのコンテンツというのはゲームのほぼすべての要素が含まれ，地形，マップ，レベル，ストーリー，ダンジョン，キャラクター，アイテム，音楽，キャラクターのレンダリング，画像，思考ルーチン，ルール等まで対象となる．PCGはコンテンツ生成の省力化を目的としているものが多いが，それ以外にも多様な環境に適したAIプレイヤーの訓練 [4] のためにも使われている．

基本的にPCGはどのゲームでも適用可能であり，主にスーパーマリオブラザーズ [5, 6] のようなアクションゲームやレースゲーム [9, 10]，real time strategyゲームのマップ生成 [11]，あるいはパズルの問題生成 [12] など行われているが，古典的で有名なターン制ロールプレイングゲーム（ターン制RPG）に対しては比較的少ない研究しか行われていない．これらを踏まえ本研究では，PCGの中でもRPGを対象に“ステージ”を自動生成することを試みる．

大多数のターン制RPGにおいて，プレイヤーはゲームのストーリーを完成させる（例えば，最終ボスである魔王を倒す）ことを目的とし，キャラクターの役割を果たす．目的を達成するにはキャラクターの成長が不可欠であるため，連続して発生する戦闘において敵を倒すことで，報酬（経験値やアイテム）を得る必要がある．しかし，敵を倒すには資源（体力や魔法力，アイテムなど）を消耗する恐れがあり，出現する敵をどれだけ倒せばいいかが問題となる．例えば，もしプレイヤーが全ての敵を倒しキャラクターを限界まで成長させたとしても，全ての資源

を使い切ってしまっていた場合，最終ボスを倒すのは厳しいかもしれない．逆に全ての敵を無視して資源を温存していた場合，最終ボスを倒すにはキャラクターの成長が十分ではなくなる．それゆえ，プレイヤーはこのような問題を考慮して「この戦闘は回復アイテム一個を使って勝利する」「すぐに迫ってくるボス戦のため，アイテムや魔法力を保存する」「強そうな敵に勝つには資源の消耗が多いと判断して逃走する」などの自分なりの戦略を立てる必要があり，これがゲームの面白さに繋がる．

このような戦略が有効になるためには敵の配置，体力や攻撃力など強さを示すパラメータ，回復ができる場所の配置，アイテムの効果や出現頻度，出現する武器の強さなどを意味する“ステージ”のバランスを適切に設定することが重要になりうる．さらに，ゲームの面白さの観点では，1 つのステージに対して多様な戦略を取りうることのほかに，そもそもステージが多様に提供されることが望ましい．これにより，プレイヤーは毎回多くの選択肢の中から適切な戦略を選ぶ楽しみを得られる．これらのことから，本研究の目標は，良好なステージを生成し，かつそれを多様に生成することとする．本研究では「良好なステージ（群）とは何か」という research question を考究することはせず，それが定義できたとして，それを満たすように PCG を行うためにはどうしたらよいかを主要な research question とする．

PCG にはいくつかの接近法がある．その 1 つの例として Procedural Content Generation via Machine Learning（PCGML）[13] がある．PCGML はここ数年活発に研究されている．大多数の PCGML の研究では，既に用意されているゲームコンテンツを用いて，生成モデルを学習する．人間から作られた膨大なデータを入手することが簡単な場合，Variational Autoencoder（VAE）[14]，Pixel CNN[15] または Generative Adversarial Network（GAN）[16] のような生成モデルを用いて，データの分布に従う新しいコンテンツの生成が可能である．これらの一般的手法をゲームの PCG に適用する場合，大量の学習データを得ることのコストが課題となる．ゲームデザイナが大量の学習データを作ることには大きなコストがかかるし，そもそも大量のコンテンツがあるならそれをそのままプレイヤーに提供すればよいという話になりかねない．自然地形からマップを生成するなど，外部から流用可能なものならばこのような手法も使われているが，各ゲーム固有のルールに関係するようなコンテンツについては流用は難しい．

PCGML とは異なるアプローチとしては，生成検査法の一種である search-based PCG[18] が頻繁に用いられる．Search-based PCG はコンテンツ生成機構と評価機構の 2 つの部分を持ち，(1) 生成されたコンテンツを評価して良いものを選ぶ，さらには (2) 得られた良いものを少し改変または要素混合してより良いものを作る

ことを繰り返す，といった最適化システムを含んでいることが多い．Search-based PCG は学習データを必要としない代わりに，各コンテンツに対する評価機構（評価関数）を適切に定める必要がある．これは通常デザイナーが自分の持つ“望ましいコンテンツ”の方向性を表せるように設計し，さらにそれを各プレイヤーの熟練度や好みに合わせて調整することも可能である．最適化システムには多くの場合遺伝的アルゴリズム [21] が用いられるが，遺伝的アルゴリズムは多くの候補を進化させる必要があり，また原理的に“似たコンテンツ群”が得られる可能性が高いため，プレイヤーごとに適している多様なコンテンツをただちに提供するにはやや低速である．

以上のように，PCGML では大量のデータの必要性，Search-based PCG ではコンテンツのオンライン生成時の速度に課題があることが分かる．本研究では，この課題に対し，強化学習を用いることを考える．離散的に連なる n マスのステージを完成ステージとして仮定したとき，n マス以下のステージを「状態」，次の未生成 1 マスにパラメータを生成することを「行動」をしてみなす．ステージが完成すると，その良し悪しが与えられた評価関数で評価され，強化学習の「報酬」として与えられる．これを強化学習の学習手法を用いることにより，与えられた評価関数を最大化するようなステージが生成できるようになると期待できる．さらに，このままでは常に同じようなステージを提供することになり多様性に乏しくなってしまうため，評価値をあまり下げないままに大きく異なるようなステージを生成するための試みを行う．

本章に続き，第 2 章ではコンテンツ自動生成の関連研究を紹介し，また強化学習の枠組みを説明する．第 3 章ではターン制 RPG の構成と流れを説明し，どのようなステージが望ましいのかを述べる．第 4 章では対象とするターン制 RPG の研究プラットフォームを提案し，ステージやその評価の定式化を行う．第 5 章ではステージ生成を強化学習に定式化し，良好なステージ生成法および多様なステージ群の生成法を提案する．第 6 章では提案手法を実験し，最後に第 7 章で本研究を総括し，今後の展望や課題を述べる．

第2章　関連研究

本研究はコンテンツ生成のために強化学習を用いるというアプローチを取る．本章では，まず 2.1 節で Procedural Content Generation(PCG) の枠組みを紹介し，多様な手法があるなかでどのようなグループ分けが可能なのかを述べる．続いて 2.2 節と 2.3 節では代表的な PCG のグループである PCGML と search-based PCG について具体的な研究を紹介する．これらにはそれぞれ課題があるため，本研究では強化学習を用いるのであるが，強化学習の枠組みを 2.4 節で導入する．

2.1　ゲームにおける Procedural Content Generation

第 1 章で述べたように PCG は一般のコンテンツの自動生成の枠組みを指すが，仮にゲームを対象とした場合でも，その生成対象および生成方法や援用される AI 技術は非常に多岐にわたる．本節では，PCG に関する著名なサーベイ論文 [18][19] を参照し，ゲームにおける PCG がどのようにグループ化できるか，それぞれどんな傾向があるかを簡単に解説する．既存アプローチの全体像を知ることにより，本研究で提案する強化学習というアプローチの新規性と有望さを確かめることができる．

- Constructive vs Generate-and-test
  constructive アルゴリズムはコンテンツを一回生成するとそこで生成は終わりで，それゆえプレイ不可能なコンテンツを生成しないと保証された生成法を用いる必要がある．Song らは constructive な生成法を提案した [20]．Generate-and-test アルゴリズムは生成と検査の過程を含む．まず生成をしてそれが基準を満たしているのかの検査を行い，満たしていない場合にはまた生成を繰り返す. この過程は基準を超えたものが生成されるまで続けられる．

- Direct vs Simulation-based evaluation vs interactive
  Direct は生成されたコンテンツに対して評価関数により直接評価を行う．Simulation-based はゲームエージェントによるシミュレーションを行いそのログから評価をする．interactive は人間がステージを見てから評価を行うことである．それぞれ後ろのそのほど精度が高いが高コストである．

- オンライン vs オフライン
  ゲームをプレイする途中コンテンツの要求があるとき，要求に応じた生成を行うオンライン生成とゲームの開発の段階で先に生成を行うオフラインの生成がありうる．オンライン生成の場合，プレイヤーの好みやタイプに応じて生成を行う場合が多く，多様な生成を高速で行う必要がある．

- 自動生成 vs 協力生成
  生成アルゴリズムによりコンテンツが直接生成されるかまたはアルゴリズムと生成者が協力しながら生成をするのかという分け方ができる．後者では生成側が生成した部分からエージェントが生成を補充したり，追加で一定部分を生成したりする．そこからまた生成側が生成を行いこの過程を繰り返す．

- Generic vs Adaptive
  コンテンツ生成はプレイヤの技量や好みを反映させた方が良い場合もある．Generic な PCG ではこういったことを行わず「大多数に支持される良いコンテンツ」を作ろうとするが，Adaptive な PCG では，例えば各プレイヤの技量を推定したうえで，易しすぎず難しすぎないようなコンテンツを生成するといったことを目指す．

他にも PCG にはさまざまな特徴づけが可能であり，ゲームの PCG に限っても高い多様性を持つ領域である．具体的な手法について見てみても，いわゆるヒューリスティックな if-then ルールが用いられる場合もあるし，オートマトン，フラクタル，あるいは遺伝的アルゴリズム，ニューラルネットワークなどの AI 技術が用いられることも多い．これらは，ゲームの種類，生成対象のコンテンツ，生成目的，速度やコストなどの要求，参照できるデータや利用できる人員などの条件によって，さまざまに使い分けられ，または組み合わされる．

多様なゲームの PCG の中でも，近年最も盛んに研究されまた用いられているのは，Search-based PCG[17] と呼ばれるグループと，PCG via Machine Learning (PCGML)[13] と呼ばれるグループである．続く 2.2 節・2.3 節ではこれら二つの代表的なグループについて，その研究例や長所短所を紹介する．

2.2 Search-based PCG

古典的な生成検査法（Generate-and-test）はコンテンツ生成機構と評価機構の2つの部分を持ち，ランダムにコンテンツ候補を生成しては評価し，十分に良いものができるまでそれを繰り返す枠組みである．しかし，生成対象が複雑になるにつれ，または求められる質が高くなるにつれ，ランダムにコンテンツを生成していては滅多に満足いくものが生成されないということが頻繁に起きるようになっていった．これを改善するために Search-based PCG[17][18] という手法が研究された．Search-based PCG は生成検査法の一種である．Search-based PCG の生成過程は評価関数による選別と，選別から何らかの生成手法を用いてより良い物を生成するとことが組み合わされている．数個の生成されたコンテンツの中で良いものを選別し，そこから評価がより高い物を生成しながら提供することになる．生成手法としては遺伝アルゴリズムが良く用いられる．

例えば，今回の対象であるターン制 RPG のステージを生成するなら図 2.1 のようになる．各ステージは遺伝子により表現され，ここでは 3 つの連続する戦闘を 3 つのマスで表す．遺伝的アルゴリズムでは解の候補は遺伝子の集団で持つ（世代G）．これらは定義済の評価関数により評価される（A,B,C,D の横の数字）．評価値の低いもの (B) は淘汰され，高いもの (A,C) は遺伝子を交換した子孫 (E) を残すことができる．あるいは単純に遺伝子に突然変異が起こることもある (D')．これらによって新しい世代 G' が作られる．これを繰り返すことで，生物の進化のように，優れたステージが生まれることを目指すのである．

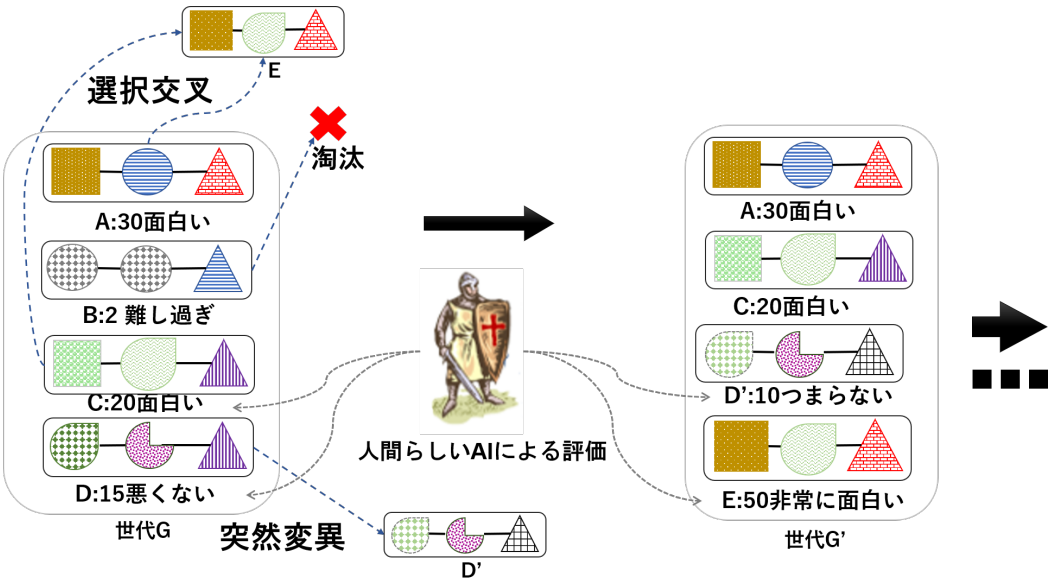


図 2.1: 遺伝アルゴリズムを用いた 3 マスサイズのステージ生成

評価関数をどのように定義するかは非常に難しい問題である．前節の 2 つめのグループ分けで示したように，生成したステージを何らかの特徴量や式で直接評価することも可能な場合はある．それが難しければ，コンピュータエージェント（AI プレイヤー）を作成して，解けるステージなのか，難しすぎたり簡単すぎたりしないかををテストする方法もある．より適切な評価値，すなわち「本当にそのステージは面白いのか」が欲しければ，人間の被験者を使う方法もありえる．しかし，３つの方法は後ろのものほど評価に大きなコストがかかることになる．遺伝的アルゴリズムは本質的に複数の個体を持ち複数の世代行われるため，評価コストは生成コストに直結するのである．

Loiacono と Togelius らはレースゲームにおけるトラック生成を遺伝アルゴリズムを用いて行った [9][10]．Togelius らはプレイヤーをモデリングし実装したテストプレイヤーのプレイ記録から面白さの推定を行ってステージを最適化した．Loiacono らは Togelius らの研究から発展をして，特定のプレイヤー向きのより多様なコンテンツを生成することを目指した．

通常，遺伝アルゴリズムは多数の解を保持する為に，多くの評価を必要とし，さらにモデルによっては多数の割には最終的に似通ったコンテンツが生成されてしまう．不特定のプレイヤーにオフラインでコンテンツを提供するには問題はない．しかし，プレイヤーのスキルや好みなどによって（Adaptive）その場で即時（オンライン）に多様なコンテンツを提供することは難しい．例えば，Loiacono らは，300 回の世代交代を行っていて，control point の数を 5，10，15 に分けて多様なトラックの生成に成功した．しかし，300 世代と異なる多様性を追求するためには最初から学習を行う必要があり，それには長い時間を要する [9]．

2.3 PCGML

前述のようにPCGMLはコンテンツ生成を機械学習を用いて行う枠組みである．何らかの方法で用意した大量のコンテンツを学習データとして，コンテンツの生成モデルを学習する．生成モデルは例えば「ここまでのステージ」を入力として，「次のステージ」を出力するような教師あり学習，またはデータからデータの分布を学習して分布に従うものを生成する教師なし学習である．前節のSearch-based PCGと比較するとPCGMLでは，生成されたコンテンツが目標コンテンツの空間を評価値により探索することで選別されるのではなく，直接生成モデルから生成される．

いくつかの代表的な研究を挙げると，Volzらはマリオにおけるレベルを GANを用いて生成を行った [6]．GANは敵対する2つのネットワークからデータの分布を学習を行う手法で，学習後潜在変数の入力から生成を行う手法である．生成されたマリオのステージの中ではプレイが不可能なものも生成されるので，AIのテストプレイから遺伝アルゴリズムを用いてプレイ可能な潜在変数を生成し，それをGANの入力として用いることで特性がありながらプレイが可能なステージを生成しようとした．Snodgrassらはマリオと icarusのゲームで二次元レベルに対して周りのタイルから markov chainを用いて次の生成タイルを生成した [8]．

Summervilleらは long short-term memory recurrent neural network（LSTM RNN）を用いることでマリオのレベルのタイル配置をした [5]．彼らはマリオの各タイルタイプを1つの文字としてみて，2次元のレベルを1次元に変換し，文字列として扱うことで次のタイルの確率が最も高いものを生成するようにした．結果物ではクリアができないレベルもあったので，A*で実装したエージェントが動いた経路データを入力情報として用いて，より多くのプレイ可能なレベルを生成しようとした．

PCGMLを用いてコンテンツを生成するためには「学習用として十分なデータを集めることができるか？」「集めたデータが望ましい分布を持っているか？」といった課題がありうる．ゲームから離れ一般のイメージ生成の場合，集めたデータは車，馬，鳥，などのように簡単にカテゴリとして分類ができる上に，その生成目標は各カテゴリの表現を学習させることである．しかし，ゲームにおけるコンテンツ生成の場合，例えばRPGのステージが生成目標であるなら集めたステージデータが難しいか，簡単なのか，適切な難易度なのか，悩ませるようなステージになっているのかなどを評価・分類することが望ましい．しかし，その作業自体がコンテンツの特徴を把握することを伴うので難しくコストが高い上に，ノイズが加わる可能性も高い．この問題以外にも，異なるゲームのコンテンツはお互いに構造が異なるため，対象ゲームの充分なコンテンツを集めることが難しくな

る可能性が高い．

Volz らと Summerville らの研究 [5][6] では大量のデータを用意できず，少量の学習データで生成を行っている．例えば，Summerville 等の研究では正確なデータ数は言及していないが，スーパーマリオブラザーズに既にあるステージ 1 を 173 分割して学習データとして用いる．その場合最大 4 個のステージを分割して 693 個のデータを活用していると考えられる．画像領域における簡単な生成対象である MNIST の場合にも 60000 個の学習データが，Flickr-Faces-HQ Dataset (FFHQ) の場合も 70000 個のデータが用いられることを考えると生成物は少量のデータの分布に大きく依存してしまう．そのため，真の意味での多様なコンテンツを生成することは困難であり，これは PCGML の手法がこれから解決すべき課題である．

本研究では学習データを用いず，オフラインでの学習の後，オンラインの短期間で多様なステージを生成できるようにする為の手法として強化学習に着目した．

2.4　マルコフ決定過程と強化学習

2.1 節から 2.3 節までは PCG の概要とこれまで主に用いられてきた手法についてその問題点と共に示してきた．我々はコンテンツ生成問題を“ マルコフ決定過程 ”にモデル化した上で，“ 強化学習 ”を用いて方策（コンテンツ生成関数）を学習する．そこで本節では，これらについての基礎概念を導入する．

マルコフ決定過程（Markov Decision Process, MDP）は強化学習の対象となる環境のモデル化に多く使われる定式化であり，実用上は以下のように定義されることが多い [7].MDP は (S, A, P, R, γ) と 5 つの要素で表現され，S は状態の空間，A は行動の空間，$P_a(s, s') = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$ はある時点 t において s から s' に遷移する確率，$R_a(s, s')$ はその遷移の時もらえる報酬，$\gamma \in [0, 1]$ は割引因子であって，未来の報酬をただちにもらえる報酬に比べてどれだけ割り引いて勘案するかを表す．π は s から取る行動 $\pi(s) = a$ を指定する関数，または行動をとる確率分布である．すべての次の状態 s' は現在状態 s と行動 a だけに依存し，過去状態とは独立であるマルコフ性を持つ．

マルコフ決定過程に対する主な問題設定は，与えられた環境に対して，良い方策 π を求めることである．MDP の要素 S, P, R 等については一部が明示的に与えられず，環境との相互作用によって獲得しなければならない場合も多い．良い方策とは具体的には，以下の式で表される“ 割引累積報酬 ”の期待値を最大化する π のことである．

$$\sum_{t=0}^{\infty} \gamma^t R_{\pi(s_t)}(s_t, s_{t+1})$$

ある状態と行動に対する Q 値というのはその状態から行動を取ったとき，方策 π に従う場合の割引期待報酬になる．Q 値が正しく学習できれば，状態ごとの最適な行動を求めることはたやすい.「現状態での最適行動の Q 値＝即時報酬＋割引率×次状態での最適行動の Q 値」となる（いわゆるベルマン方程式）ため，これを用いた学習法である Q 学習がしばしば用いられる．

$$Q^{\pi}(s, a) = E_{\pi}[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_0 = s, a_0 = a]$$

他の機械学習の一種である教師あり学習や教師なし学習に比べると強化学習は学習用のデータを必要としない変わりに，環境とのやり取りにより自己学習を行う大きな違いがある．まず，$Q(s, a)$ をテーブルで持つ場合は最適性が証明されているが，状態空間が大きくテーブルですべての Q 値を学習することができない場合には，CNN らを用いて近似することが良く行われる．Mnih らは 40 個の atari2600 ベームに対して DQN というモデルを用いて，大多数のゲームにおいて人間以上の

強い AI の生成に成功した [24].

強化学習の特徴をまとめると次のようになる.

- 報酬を最大化するように学習するので報酬の設計は大事である
- 学習には一般に長い時間がかかる．一方で，一度学習すれば行動決定は（GA や木探索に比べ）高速であることが多い
- 状態空間をどのように取るかによって，テーブル表現ができたりできなかったり，CNN を使うにしても関連性が高いものを近くに置くなど状態表現には工夫が必要である

第3章　ターン制RPGとステージ

本研究では，自作のターン制RPGを対象にそのステージ生成を行う．本章では，ターン制RPGの一般的な構造と，ステージに対する要求を述べたうえで，用いたターン制RPGの仕様を紹介する．ターン制RPGの例としてはドラゴンクエストシリーズ（2018年基準で，累計出荷数と配信数は7,600万本を超えた[25]），ファイナルファンタジーシリーズ（2018年基準で，累計出荷数と配信数は1億4200万本を超えた[26]）やポケモンシリーズ（2017年基準で，累計出荷数は3億本を超えた[27]）などが挙げられる．これらのゲームはそれぞれ異なる固有のシステムを持っているが，基本的なシステムでは共通する部分も多い．例えば，殆どのゲームは物語を進めながら登場人物を成長させ，最後にはボス敵を倒すといった要素を持つ．またその進行において，“戦闘シーン”と呼ばれるパートが非戦闘パートと分けられている点も共通する．

非戦闘パートは物語を進みながら，戦闘以外に起こりうるあらゆるイベントのことだと言える．この範疇にはゲームを進みながら寄る町や，ダンジョンの間に挟んでいる休憩ポイントで被害を受けたキャラクターを回復したり，町の鍛冶屋で武器を強化したり，次の探検に備えて回復アイテムを購入したり，途中に置かれている宝箱からアイテムを取得したりなどがありうる．このように，非戦闘パートではただ物語を進行するだけでなく，戦闘の難易度や戦略に影響するようなパラメータ変更が起こりうる．そのため，戦闘パートと非戦闘パートを完全に切り離して考えたりデザインしたりすることは好ましくない．

戦闘パートではプレイヤーチームと敵チームに分かれて，お互いに相手のチームを全部倒すかまたは逃走するまで戦い続ける．例外はあるが大体のゲームは戦闘後のキャラクターの状態が次のパートに引き継がれるため，ただ目の前の戦闘に勝つだけの短期的な戦略では十分ではなく，先を見据えた長期的な戦略が必要になる．この点は特に他のゲームジャンルよりRPGにおいて浮き彫りにされる点である．

以上のように，物語を進めたり戦闘の準備をする非戦闘パートと，そこから派生する戦闘パートを繰り返すことでゲームは進行する．プレイヤは通常，非戦闘パート・戦闘パートそれぞれでさまざまな意思決定を行う必要があり，逆に，“さまざまな意思決定を行う必要があるような”ステージ設定がなされていることが望

ましいとも言える．どのような選択肢を選んでも似たような結果しかもたらさないならば，それは主体的にゲームをプレイしているとは言えないためである．続いて次節から，戦闘パート・非戦闘パートのより詳細な説明と，望ましいステージ設定についての考察を行う．

3.1　戦闘パート

本節では，ターン制 RPG における戦闘パートの基本的な流れについて説明する．ターン制 RPG の戦闘はターンを中心に行われる．毎ターン各プレイヤーは行動順に 1 つの行動を行う．行動の順番を決め方はゲームによって異なる特徴を持っているが，多くの場合は各キャラクターは速度（素早さ）というパラメータを持っており，これによって行動順や行動頻度が決まる．以降は典型例としてドラゴンクエストシリーズに多い進行を例示すると，各キャラクターが行う行動はターンの最初に入力されまたはアルゴリズムによって決定され，速度に乱数を加えた値の大きい順にその行動が実行される．各キャラクターは速度以外にも，攻撃，魔法，技等の固有の行動を持っている．

図 3.1 に，2 対 2 の戦闘における 1 ターンの流れを例示する．図中，キャラクター絵の脇にある枠は，上・左から，キャラクターの名称，体力値／最大体力値，魔法力／最大魔法力，攻撃力，防御力，速度，そして取りうる行動の選択肢を表す．

1. まず，勇者の速度パラメータが一番高いので攻撃行動を幽霊に対して行う．
2. 続いて，次に速度が速い幽霊が特殊技である毒攻撃を勇者に対して行う．
3. 次に，龍が火炎の技を勇者に行う．
4. 最後に僧侶が被害を受けた勇者を回復する．

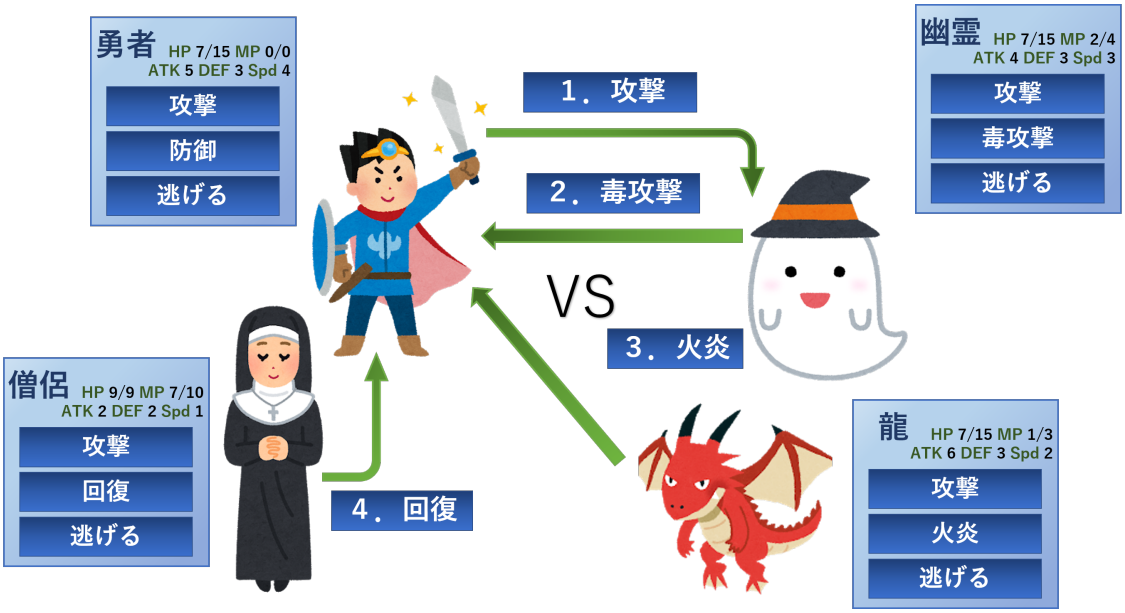


図 3.1: 2 対 2 の戦闘における 1 ターンの流れの例

ターンは，所属陣営の全キャラクターの体力パラメータが 0 になる「全滅」が敵・味方のいずれかに起きるか，プレイヤー陣営が戦闘から回避する「逃走」が起

きるまで続けられる．戦闘に勝つことでプレイヤーはキャラクターの強化と，アイテムや金銭などの報奨を得ることができる．

これらの設定はゲームごとにさまざまに異なるが，ターン制 RPG ではこの戦闘シーンにおける戦略決定の楽しさが重要であることは共通している．勝とうとすることはもとより，できるだけ体力を減らさずに勝ちたい，できるだけ魔法力を減らさずに勝ちたい，できるだけ（実時間で）早く勝ちたい，などプレイヤごと・状況ごとにさまざまな副次的な目的があり [28]，それらを達成するために多くの工夫が必要になるようにステージ生成が行われるべきである．

3.2　非戦闘パート

ターン制 RPG では町のような拠点と，フィールド，敵などが出現するダンジョンなどが分かれて構成されていることが多い．拠点では，買い物・回復などができるが，フィールドやダンジョンで回復をするためには手持ちの薬を消費するか魔法力を使って回復するしかない．

ゲームの最終的な目標は大ボス打倒であっても，それは節々に分かれ，拠点を中心にして概ね以下の４つの行動パターン（図 3.2）のどれかを繰り返すことになる．

(a) 拠点 ⇒ フィールドでレベリング・アイテムや金銭取得 ⇒ 拠点に戻る

(b) 拠点 ⇒ フィールドやダンジョンに移動 ⇒ 偵察したり宝を拾う

(c) 拠点 ⇒ 薬や魔法力を温存しながらダンジョンに移動 ⇒ 中ボスを倒す

(d) 拠点 ⇒ フィールド移動 ⇒ 別の拠点に到着する

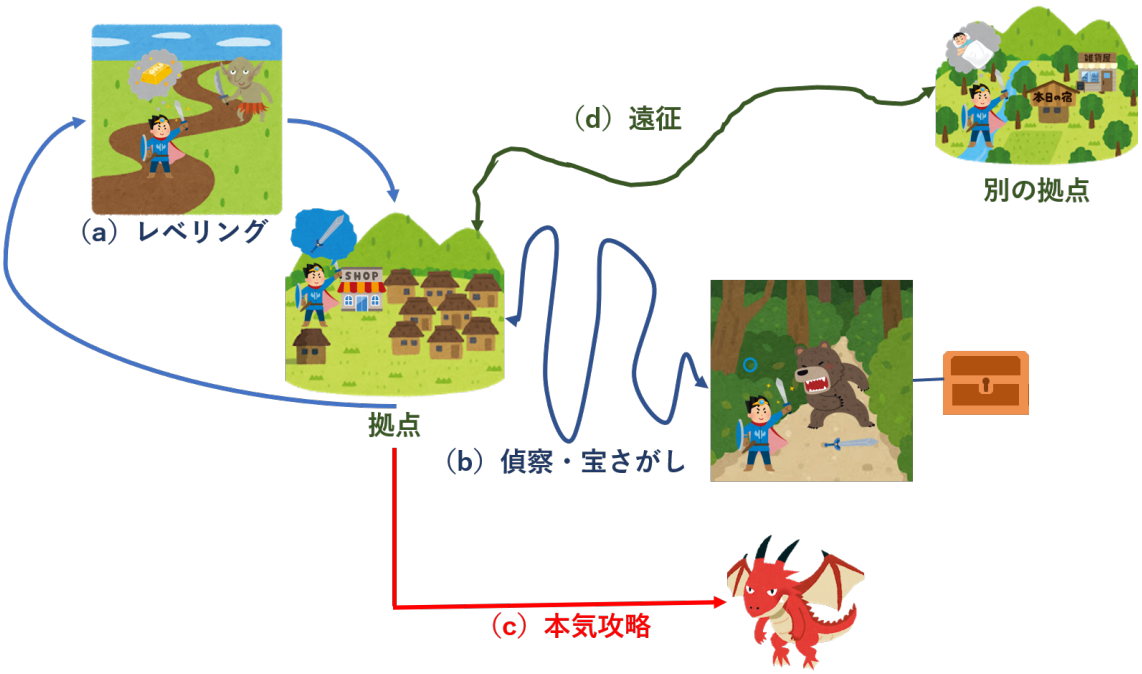


図 3.2: ターン制 RPG の物語の進み方の例

これらのモードごとに，プレイヤの達成したい目標やそのための戦略は異なる．(a) レベリングでは，多少リスクがあっても，または魔法力を消耗しても，早くに戦闘を終わらせたい場合が多い．そういった雑なプレイを容認するように，ステージも繊細に調整されることは少ない．(b) アイテム探しや (d) 拠点への移動は，戦

闘だけでなく，フィールド・ダンジョンの上での移動を伴う冒険や探索の意味合いが強く，例えば少し見つけにくい宝の隠し方であるとか，新しい世界が開ける楽しみであるとかを与えるようなコンテンツ生成が行われる．一方で，(c) 中ボスを倒しに行くようなモードでは，敵も概ね強く設計され，ちゃんとレベルを上げたうえでちゃんと薬や魔法力を温存してなんとか中ボスを倒せるように，挑戦が提供されていることが多い．このようにステージに求められるものはモードによっても異なるが，本研究では主に (c) のような状況を考えることにする．

3.3　望ましいステージ

3 章ではここまで，一般的な RPG の概要について述べてきた．本節では，物語部分や映像音声などを除いて，どのようなコンテンツを提供すればユーザを満足させやすいのかについて考察する．今まで RPG では手作業で作られたステージが主に遊ばれてきたが，近年では新しい経験をプレイヤーに提供するため，自動生成されたステージを提供するゲーム（例えば，Darkest Dungeon[29] や Slay the Spire[30] など）が急激に増加している．

通常ゲームコンテンツ生成ではプレイヤーにとって面白く感じさせるようなことを追求するのが当然であるが，どうすれば面白く感じるのかはプレイヤごとに異なり，常に成り立つ明示的な法則は存在しないと考える．多くの場合，ゲームデザイナーはさまざまな要素・条件を考慮に入れたうえでコンテンツを生成したり，AI に生成させたりするわけである．ドラゴンクエストのような RPG において，また拠点から中ボスを倒しに行くような状況を想定し，“ 中ボスを含めたそこまでの敵の配置や強さ ” を評価対象とするならば，以下のような点が考慮すべき内容であると考える．

- どんな戦略をもってもクリアできるほど易しい，またはどんな戦略も高い成功率を望めず運任せになるようなステージは面白くない．
- 自動生成されるのに，いつも同じようなワンパターンなステージであってはつまらない．
- 仮に，異なるパラメータを持つステージであっても，同じ戦略で攻略できてしまうと，面白いと言いづらい．状況によって多様な戦略が有効であるステージが望ましい．
- 取りうる行動や戦略が「明らかに良い」「明らかに悪い」と自明であるのは面白くない．プレイヤーに悩ませる状況の方が良い．
- 「どうしても倒せない敵が序盤に出て序盤に逃走するしかない」,「後半の敵が弱すぎて気が抜ける」などの，タイミングに合わない難易度の強さを持つ敵は不自然である．
- 中ボスに向けて進行する中で，一時的にかなり危険な状況に陥ったり，そこから回復場所に到達して乗り切ったりなど，メリハリのある緊迫した状況が生まれるようなステージは，面白く感じられることが多い．

本研究では，上記の要請を考慮した評価関数を定めて，コンテンツ生成時・生成後の評価に用いることにする．これらの要請はあくまで一般的な傾向として議論した結果であって，それを定式化・定量化したり，それらを満たすことによって実際に満足度が向上することを確かめたわけではない．実際のところ，そのような定式化・定量化・被験者実験などはそれだけで一本または数本の論文になるほどの内容であると考える．本研究の目的は「仮にそのような適切な定量化が可能であったとして，それを使ってどのようにステージを生成するか」であって，定量化そのものではない．そこで本研究では簡易の評価関数を用いることにし，その内容は 4.3 節で詳述する．

第4章　対象問題

この章では本研究において用いる対象問題について具体的に記述する．．4.1 節では研究ゲームの環境について述べ，4.2 節ではステージをどのように変数表現するのかについて，そして 4.3 節ではステージの評価法について述べる．

4.1　研究プラットフォーム

多くの商業用 RPG は濃厚なストーリーや多様なゲームモード，自由なマップなどが与えられていることが多いが，本研究ではこれらを対象とせず，戦闘に密接に関わっている，戦闘パートと休憩の非戦闘パートに限定して研究する．

研究プラットフォームを使用することにおいて機能を制限することは必要によって頻繁に行われる．例えば，Roguelike のゲームに対して高橋らと金川らはログライクのゲームに対して機能を制限したゲーム環境を提案している [31][32]．しかし，ターン制 RPG は適した環境がなかったため，この研究の延長線を考慮して多様な機能を持つターン制 RPG を実装し，それを基に単純化したものを利用する．

以下に本研究で用いたゲーム環境に登場する要素・概念システムなどの列挙する．表 4.1 は，一般の RPG における設定，実装した研究プラットフォームが扱える範囲，実際に実験に用いた設定，の 3 つを比較したものである．

- チーム数
  国取りゲームのようなものでは多くのチーム（人間/AI プレイヤ）が参加することも多いが，一般にターン制 RPG では，冒険者側のチームが，モンスター側のチームと 1 対 1 で戦い，勝ちぬいて行く形が多い．本プラットフォームでもこの形式を採用し，実験でもそれを用いる．

- ステージ構成
  一般のゲームではステージは二次元上を歩きまわれるようになっていることが多いが，ボスを倒しに行くモードでは最短ルートでボスを目指すことが多い．扱いの簡単さのため，本プラットフォームではステージは一方向一列の構造を持っていて n 個のマスで構成されている（図 4.1）ものとする．最後の

マス以外の各マスは戦闘，休憩のどれかに当てはまり，最後のマスはボス戦になっている．戦闘マスは敵キャラで構成されていて，体力，攻撃力の各パラメータが一定有効範囲の中から決められる．休憩マスは 0〜100 の回復率をパラメータとして持ち，プレイヤーがここにたどり着くとその分体力パラメータが損失分から上がる．ボスマスは基本的に戦闘マスと同一であるが，各パラメータの有効範囲が戦闘マスより高く設定されている．

図 4.1: 様々なステージの例

- 勝利条件
  前節で述べたように，一般のゲームでは中ボスを倒すことは物語全体の一部のモードの目的に過ぎない．本プラットフォームでは各マスを敗北せず進み，最後のマスのボスを倒すことで勝利であると定義する．

- チーム人数・キャラクター種類
  一般のゲームでは登場キャラクタは敵味方多岐にわたり，またそれぞれが異なる役割を持つ複数でチームが構成されることが多い．本プラットフォーム

も，任意のパラメータを持たせたキャラクタを任意の数戦わせることができる．しかし本論文ではこれを非常に単純化して用い，各チームが 1 人のみで構成される場合で実験を行う．キャラクターは冒険者，途中の敵，ボス敵の３種類であり，それぞれとりうるパラメータの範囲が異なる．

- パラメータ
  一般のゲームでは時には 20 種類以上のパラメータが一つのキャラクタに設定されることも多いが，本プラットフォームでは，最大体力・攻撃力・速度のみを採用することにした．追加は容易なように設計してある．さらに本論文の実験では速度も用いていない．体力はキャラクタの生命力を意味し，攻撃を受けると攻撃者側の攻撃力分だけ減少し，0 になると死亡と扱われる（チームが 1 名のみならば，それが敗北を意味する）．回復マスにおいて体力は回復するが，最大体力を超えることはない．

- 行動順序
  一般のゲームでは各キャラクターに速度パラメータが設定され，それによって順序やさらには行動頻度が定まることが多い．本プラットフォームでは，速度やそれによる行動順序の決定を実装したが，本論文の実験では，常に冒険者サイドが先行するように単純化した．

- 行動
  3.2 節で例示したように，一般のゲームでは各キャラクターはその個性に応じてさまざまな戦闘行動をとりうる．本プラットフォームでも，単体攻撃・全体攻撃・単体回復・全体回復・防御・アイテム利用・逃亡などさまざまな行動がとれるように枠組みは実装したが，本論文で用いているのは単体攻撃と逃亡のみである．敵キャラクター側は単体攻撃のみを行う．

- 戦闘結果
  一般のゲームでは，勝利によってキャラクターが成長したりアイテムを得たりすることが多い．本プラットフォームでは，キャラクターの成長のみを実装した．具体的には，戦闘に勝てば攻撃力が 10 %増加する．一方で，戦闘に勝つためには体力がかなり減少することも多く，プレイヤには逃亡という選択肢も与えられる．逃亡した場合には攻撃力の増加はなく，さらに体力も 15 %減少する．

- 取りうる選択肢の数
  一般のゲームでは，キャラクターの行動選択肢数は攻撃対象の選択なども含めてかなり多く，またそれが複数キャラクターで何ターンも繰り返されるた

め，一戦あたりの戦略の可能性は膨大である．一方で，本論文で用いた単純化した設定においては，一戦あたりの戦略は２パターンしかない．すなわち，攻撃するか逃げるかである（明らかなように，一度「攻撃して，敵から攻撃を受ける」ことをしてから逃亡することには価値がない）．

	代表的な ターン制 RPG	プラットフォーム 仕組み	今回用いた設定
チーム数	1 対 1	1 対 1	1 対 1
ステージ構成	多様なマップ構造 戦闘	一直線の戦闘 休憩マス	一直線の戦闘 休憩マス
勝利条件	長い旅ののちに ボスを倒す	マスを一直線に 進みボスを倒す	マスを一直線に 進みボスを倒す
チーム人数 キャラクター種類	任意・任意	任意・任意	1・3
パラメータ	20 以上	任意設定	体力・攻撃力・速度
行動順序	速度や乱数により決定	速度により決定	プレイヤー先攻
行動	多様な技	多様な技	攻撃逃走のみ
勝利結果	成長 金銭・アイテム	成長	攻撃力増加
1 戦あたりの 選択肢の数	膨大	膨大	2

表 4.1: 代表的な RPG，プラットフォーム，本論文で用いる環境の比較

4.2　ステージ表現

本プラットフォームでは，一直線上に敵マスや回復マスが並んで最後にボスマスが控えるようなステージを扱う．図 4.2 上部に，敵-回復-ボスの 3 マスからなる単純な例を示す．敵・ボスはそれぞれ体力と攻撃力を持ち，回復マスは回復率を持つ．これらの値は一定の範囲内で可変であり，例えば通常の敵は 20〜100 の体力と 5〜30 の攻撃力を持つ．ボスの場合は 60〜180 の体力と 20〜50 の攻撃力である．

このようなステージをプログラム上でどのような形式で表現するかにはいくらか任意性があり，それぞれ扱いやすさや学習時の汎化のしやすさなどに一長一短がある．本論文では，深層強化学習を用いたときに入出力される値の範囲が 0〜1 の間にあると学習の際に都合が良いという予想から，体力や攻撃力をそのままの値ではなく，取りうる値の範囲内で正規化することにする．例えば図 4.2 の中ほど，敵の体力は 70 であるが，これは体力の範囲 20〜120 のちょうど中間に位置するから，0.5 と表現する．

さらに，敵マスと回復マスはその性質に大きな違いがあることに注意したい．敵マスは 2 つのパラメータを持ち，回復マスは 1 つのパラメータしか持たず，その意味合いも大きく異なる．これらを一直線に並べてしまうと，深層学習の汎化能力を期待することは難しくなる．そのため，便宜的に「敵マスの次には回復マスが常にある」と考え，これをセットにしてしまう工夫を行った．図 4.2 の例で言えば，(0.5,0.2) の敵マスの次には回復マス (0.7) があり，ボスマス (1,0.8) の次には回復マスはない (-1)，ということである．こうして，もともとは 3 マスのステージが，2 セットの（体力，攻撃，回復）からなるステージ，3 行 2 列の配列で表現されることとなる．

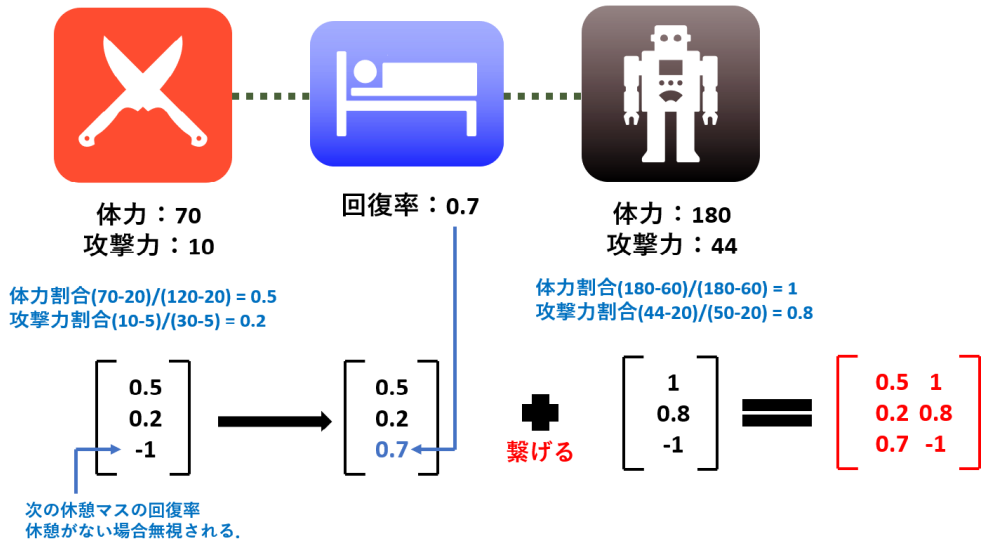


図 4.2: 戦闘－休憩－ボスのマスで構成されたステージの行列変換例

4.3 評価関数

3.3 章で記したように望ましいステージというのは考慮すべきことが多いのでそれを正確に反映する評価をデザインすること自体は難しいことである．この研究ではまず，最初のステップとして難易度をベースにしてそこに RPG における一般的な面白さを簡単に反映した評価関数を定義した．

ステージの難易度を評価する方法はいくつかありうるが，ここでは「どうやってもクリアできない」「どうやってもクリアできてしまう」などを避けるため，勝利につながる選択肢の選び方（戦略）がどの程度の割合存在するのかを難易度と定義することにする．4.1 節最後で示したように，この単純化した環境では 1 戦あたりの選択肢は 2 つである．ということは，ボス戦（戦うしかない）を含む戦闘マスが m 個あるとき,「ステージ全体をどうプレイするか」即ち戦略の数は 2^{m-1}個あることになる．それぞれについて，このプラットフォーム上でボスを倒すことができるかどうかは簡単にまた確定的に（確率的にではなく）判定することができる．そこで，ボスを倒せた戦略の数/2^{m-1}を，“ 勝率 ”と呼ぶことにして，難易度を表す値として用いる．

もし，勝率が 90 %であったら，それはほぼどんな戦略を用いてもステージを攻略できるという意味で，それはつまらないことになる．逆に勝率が 5 %であったらプレイヤーは慎重に行動を選択しないと負けてしまう．場合によってはこういった難易度のものが好まれることもあるかもしれないが，本研究では仮に 30 %くらいが最も好ましい勝率であると定義し，これに近いほど高い評価値が出るような

計算式を用いたうえで，この評価値を“勝率適切度”と呼ぶ．勝率適切度の具体的な形状は図 4.3 の通りである．

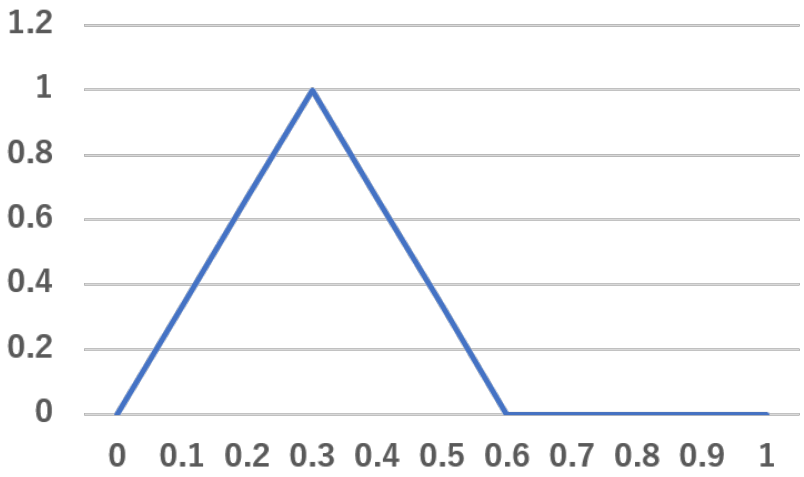


図 4.3: 勝率（横）による勝率適切度（縦）

4.3.1 複合適切度

難易度はステージの面白さのために最も大事なものではあるが，3.3 節で述べたように他にも大事な要素は数多くある．さらにそれは，面白さやプレイスタイルなどはプレイヤごとに異なる [28]．これらを全て含めて，人間プレイヤの満足度を正確に予想できるようにすることは本論文の対象ではない．しかしながら，仮にそれら複雑な要素が多数入ってきた場合には，深層ニューラルネットワークでもステージからの値の推測が困難になることが予測される．そこで本研究では，ある程度複雑な要素が入ってきてもステージ生成がちゃんとできるのかを確認するため，勝率適切度だけではなく，以下の要素を線形和した“複合適切度”も用いることにする．

- 勝率適切度（重み 0.4）
- 体力が少ないキャラクターが回復マスでの回復により九死に一生を得るとき（体力が 3 割以下のとき，5 割以分上回復される），プレイヤーは面白さを感じる場合が多いとして九死に一生の頻度の割合が高いほど高評価にする（重み 0.1）
- 序盤（最初の 3 敵）の敵に対して逃走する戦略が有効である場合はゲームの流れとして不自然と思われる可能性があるので低評価する．あるステージについ

てクリア可能な戦略（攻撃と逃走の組み合わせ）がn通りあり，そのうち最初の敵に逃走するものがn_1通り，次の敵がn_2通り，3番目の敵がn_3通りあったとする．このとき，そういうものが含まれる重み付き割合$(3n_1+2n_2+n_3)/6n$に重み0.05をかけて減点を行う．

- 後半（最後の2敵）の敵が弱すぎるとそれは不合理に思われるので低評価にする．平均パラメータが0.7以下である時弱いと判断し，重み0.1 × (0.7-平均パラメータ)/0.7ぶんだけ減点を行う．

- 攻撃だけや逃走だけでクリアできるような単調な戦略で勝つことは，ステージ攻略がつまらないという意味なので低評価にする．これらの戦略1つごとに，0.025減点する．

- 極端なパラメータ値（1割以下のパラメータや9割以上のパラメータだけ持つ）を持つ敵やイベントは不自然に思われるので低評価にする．より極端であるほど低評価にする．例えば下位1割のパラメータの場合は0，上位5％の場合0.1にする．（重み0.2）

- ボスを倒した後のプレイヤーの状態が万全に近いと，ボスがつまらなく簡単に倒せるということなので低評価にする．ボスクリア後の体力が4割以上であるとき，重み0.1 × (残り体力割合 -0.4) / 0.6 の評価値を引く．（重み0.1）

図4.4はランダムに生成した時のステージを設計した複合適切度で評価した時の評価値分布図になる．様々な要素を考慮していて，ランダムな生成では高い評価のステージを得ることは難しいことが分かる．生成したステージのうち評価値0.05以下のものが4割近くを占め，0.9以上のものは1000回に1回も生成されない．したがって，単純な生成検査法や遺伝的アルゴリズムなどでも，良いステージを得るには大きなコストがかかることが予想できる．

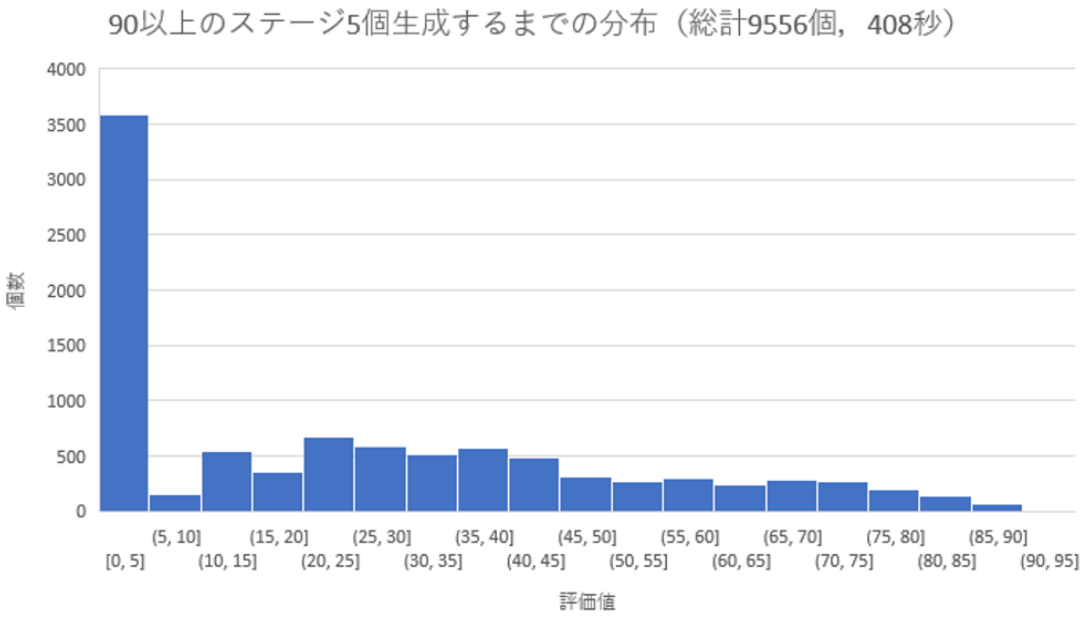


図 4.4: 複合適切度を評価関数としてステージを完全ランダムで
0.9 評価値のステージを 5 個生成するまでの分布
（本来の値域は [0,1] であるが，ここでは便宜上 [0,100] のラベルを振っている）

第5章　提案手法

第4章で生成対象とするステージの形式と評価関数を定義した．本章では，(1) これをどのように強化学習問題として扱うのか，(2) その強化学習問題をどの手法を用いて解くのか，(3) そこから多様なステージを生成するにはどうしたらよいのか，について提案手法を説明する．　簡単に言えば，(1) では生成途中の未完成ステージを状態，パラメータの操作を行動，完成されたステージをゴール状態，そしてステージの評価関数を報酬とすることでマルコフ決定過程に定式化する．(2) では性質の異なる2つの手法 Deep Q-Network(DQN)[24] と Deep Deterministic Policy Gradient(DDPG)[33] を用いる．(3) ではステージの初めの部分をランダムに作成することと，行動決定に質を落とさないようなノイズを加えることを提案する．

5.1　MDP 定式化

このセクションでは MDP の定式化について説明する．まず，完成ステージ $s^* \in S^*$ に対して，それを評価する評価値 $f(s^*)$ が存在し，これは設計された評価関数により得られるものとする．一般の場合にこの評価関数をどのように定めるのかは重要な課題だが，本論文の場合は 4.3 節で定義された“勝率適切度”または少し高度な“複合適切度”を用いることにする．

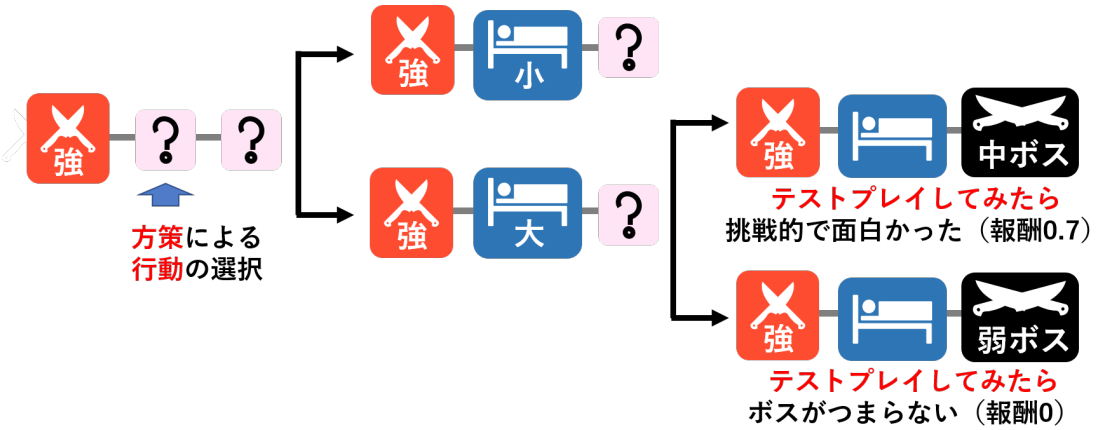


図 5.1: ステージ生成におけるマルコフ決定過程

次に，m サイズのステージを完成ステージとするとき，未完成ステージを含む m 以下の全ステージの集合を S を状態空間とし，未完成ステージに 1 ステップ分のパラメータを生成（完成ステージに近づける）する操作を行動 $a \in A$ とする．MDP のエピソードは未完成ステージ s が完成ステージに $s^* \in S^*$ 辿り着くと終了し，評価値 $f(s) \in R$ が報酬として与えられる（図 5.1）．

このような MDP の定式化により，ステージ生成問題は強化学習における報酬を最大化する方策の学習問題に置換できる．任意の状態とその時の行動に対する予測評価値である Q 値が正しく学習ができれば，各状態で最大の Q 値を持つ行動を辿っていくことにより，評価値の高いステージを生成できると期待できる．ただし，常に最大の Q 値を選び続けると，同じ（未完成）ステージからは常に同じ（完成）ステージが生成されてしまうことになることに注意したい．実際には「良好かつ多様な」ステージ群を生成したいのであって，それについては 5.3 節で方法を述べる．

5.2 強化学習モデルと行動選択

4.2 節で述べたように，本論文のステージは 3 行 m 列（m は敵マスの数）の行列で表す．5.1 節では，このステージ生成問題をマルコフ決定過程で定式化した．そしてその報酬は 4.3 節で定義した勝率適切度または複合適切度を用いる．従って，これで強化学習を行う準備は全てできた．

さて強化学習手法としては Q 学習などが有名であるが，Q 値をテーブルとして持つ古典的な Q 学習では多くの状態行動対の可能性を持つ問題を効率的に解くことはできない．そこで，Q 値をニューラルネットワークで表現して，近い状態行動対では近い Q 値をもつような汎化が行われるようになった．

その代表的な例が Deep Q-Network(DQN)[24] であり，atari ゲームへの適用結果などでその効果が示された．DQN では，状態をニューラルネットワークに入力すると，行動の数だけの Q 値が出力されるような形式を用いる．atari ゲームの場合，行動空間はゲームコントローラの操作（上下左右やボタン）を表しており，これは離散的で数も少ない．

一方で本論文の場合，行動は「次のマスのパラメータ」であって，例えば敵キャラクターならば 20〜120 の体力と 5〜30 の攻撃力を出力する．これを素直に DQN で実装すると，ネットワークは 101 × 26 個の出力を持たなければならない．例えば 20 の体力と 21 の体力などは多くの場合殆ど差がないはずなのに，違う行動としてしまうことは効率が良いとは言えない．

そこで，このような“連続的な行動”にも対応できるような強化学習手法として，Deep Deterministic Policy Gradient(DDPG)[33] も用いることとした．離散的な行動が得意な Q 学習に対して連続的な行動が得意な強化学習手法としては Actor-Critic という枠組みが昔からあり，これを Deep CNN で実装したものが DDPG である．DDPG では，Q 値も学習する (Critic) 一方で，状態から直接行動を出力するような機構 (Actor) も持つ．

アルゴリズムとしては原論文 [24][33] と同じものを用いる．DQN と DDPG では扱うのに適した行動空間が異なるので，具体的には以下のように行動空間を設計した．

DQN の場合，ネットワークの構造（出力次元数）を固定したいという要請から，常に 101 個の行動があるとみなすことにする．101 個の行動はそれぞれ，体力・攻撃力・回復率それぞれに定められた下限値から上限値を 1 %刻みで内分したものとなる．例えば下限が 100，上限が 400 の何かがあれば，行動 0 は 100 を表し，行動 1 は 103 を表し，行動 100 は 400 を表すということである．この 101 個の行動それぞれに Q 値が与えられ，例えば Q 値が最大のものを選んでステージを構築していく．敵には体力と攻撃力というパラメータが 2 つあるが，これは別々に扱う．

すなわち,「i 番目の敵の体力・攻撃力が決まってない未完成ステージでは，体力を定める」→「i 番目の敵の体力が決まり，攻撃力が決まってない未完成ステージでは，攻撃力を定める」，といった順である．このように別々に扱ってステップバイステップで決めていくことによって，出力次元数を抑えることができる一方で，本来まとまった意味を持つものを分けるという不自然さも発生してしまう．

DDPG の場合，行動空間は任意の次元数の連続の数値のベクトルを扱うことができる．そこで本論文では，4.2 節の行列の 1 列分，つまり（敵の体力，敵の攻撃力，回復マスの回復率）をまとめて $[0, 1]^3$ の 3 次元行動とした．例えば，20〜120 の体力，5〜30 の攻撃力，0〜1 の回復率が範囲と定められている場合に，DDPG が $(0.5, 0, 0.8)$ を出力したならば，70 の体力，5 の攻撃力を持つような敵マスと，0.8 の回復率を持つ回復マスが生成されるということである．

5.3　多様なステージ生成

強化学習の通常の目的は割引期待報酬を最善にすることであるから，その行動パターンがいつも同じであっても問題はない．具体的には，状態 s に対して最も高い Q 値 (s,a) を取るような行動 a*を取る greedy 方策を用いるのが通常である．しかし本研究の目的はステージ生成であり，しかも“一つの良いステージ”を生成するのではなくて“多様な良いステージ群”を作りたいのであるから，いつも同じような行動（マスのパラメータ）が選ばれることは好ましくない．そこで我々は多様性を持たせるための 2 つの方法を試した．

5.3.1　任意の初期ステージ

強化学習モデルを学習させるとき，エピソードの初期ステージとして任意性がある未完成ステージを与えて，多様な初期ステージに対応できるように学習させる．さらに，学習を終わった時点においても，任意性がある初期ステージを使うことで多様なステージが生成できるようにした．こうすれば，多様な初期ステージを与え，それによって後続のステージも毎回異なるものになることが期待できる．

ただし，固定した初期ステージを与えたい場合にはこの方法は使えないし，場合によっては初期ステージが異なっても後半部分は結局似たようなものになっていく恐れもあるため，これだけでは十分とは言えない．

5.3.2　確率ノイズ方策

初期ステージの多様化だけでは不十分と考え，仮に同じ状態であっても異なるステージを作れるような拡張を提案する．常に最善行動を取る greedy 方策と比べ，確率的にノイズのある行動を取るため，これを確率ノイズ方策 (stochastic noise policy) と呼ぶことにする．ここで重要なことは適当にノイズを取ってしまうと評価値が低い結果が出る可能性が高い点である．ゆえに，行動の期待される報酬, つまり良さを示す Q 値から“悪くはないが異なる行動”を選ぶことにする．

“悪くはないが異なる行動”は，Q 値の分布を用いて選択することにする．DQN の場合は全ての行動の Q 値が保持されており，また DDPG の場合でも Q 値のサンプリングは可能であるため，近似的に“悪くないが異なる行動”を選び出すことはできるはずである．選択のための具体的な手順と結果については 6.4 節で述べ，そのための予備実験は 6.3 節で述べることにする．

第6章　実験

本研究では，目的の異なる多数の実験を行ったので，これらを以下の通り 6.1 節から 6.4 節にまとめる．

まず 6.1 節では，予備実験として，完成ステージを入力，３種類の評価値を出力とするような教師あり学習を行った結果を示す．これは，DQN や DDPG で用いるニューラルネットワークが，複雑な入出力関係を表現できるか確かめるものである．

そして 6.2 節では DQN と DDPG を適用してステージ作成を行った結果を示す．まずは多様性を気にせず，良好な解が得られているかを確かめるものである．

続いて 6.3 節では，強化学習で得られた Q 値の推測がどの程度正しいのかを調べる実験の結果を示す．多様なステージを生成するためには次善の行動を候補にする必要があるが，強化学習ではえてして最善の行動以外の推定誤差は大きくなるため，これを確認する必要がある．

最後に 6.4 節で，多様なステージを生成するための２つの方法を提示し，結果を評価・考察する．

6.1　予備実験：完成ステージの評価値の教師あり学習

6.1.1　実験の目的

我々の強化学習のモデルでは，未完成ステージ s についてそこで取るべき行動 a やその良さ $Q(s, a)$ を推定できなければいけない．その推定モデルには，ニューラルネットワークが用いられる．CNN をはじめとする近年のニューラルネットワークは非常に高い推定精度・汎化性能を持つことが知られるが，その対象の多くは画像や音声など連続の位相が仮定できるものである．本論文で対象とする RPG のステージすなわち 3 行 m 列の入力は，それぞれが複雑に関連しており，望むような推定精度が出る保証はない．これを，強化学習よりも単純な枠組みである教師あり学習でまず確かめることが本実験の目的である．

そこで，完成ステージとその評価値のペアを多数与え，ニューラルネットワークを用いた教師あり学習を行う．評価値としては，最も単純な勝率，それを図 4.3 のように非線形に変換した勝率適切度，さらに 7 つの要素を組み合わせた複合適切度の 3 種類を用いる．

6.1.2　実験設定

ステージは 9 個のマスで図 6.1 のように構成されている．ボス戦を除いた戦闘マスは 6 つがあるのでありうる戦略数は $2^6 = 64$ 個 がある．入力の形式は 7 戦闘にパラメータが 3 つあるので 3 行 7 列になる．

学習データとするステージを完全にランダムに作ってしまうと，その難易度に偏りが出ることが予測されたため，学習データの 1/4 は勝率が 0〜25 %のもの，1/4 は 25〜50 %のもの，とある程度均等になるように収集を行った．教師あり学習であるので，訓練データを一定数与えたあと，テストデータを別に与える必要がある．勝率の推定には 12000 の訓練データと 4000 のテストデータ，勝率適切度の場合，40000 個の訓練用データを 12000 テストデータ，複合適切度に対してはそれぞれ 5500 個を与えた．なおデータ数の違いは作成上の都合で意図したものではない．

ネットワークは 2 つの 32 ノードの convolution 層と 7 つの 256 ノードの fully connected 層で構成されている．各層は ReLU の活性化関数 [34] を用いて過学習を防ぐため dropout[35] と batch normalization[36] と early stopping[37] を使用した．100 個ずつ batch して，40 エポック分学習を行った．

図 6.1: ステージの構成

6.1.3 結果

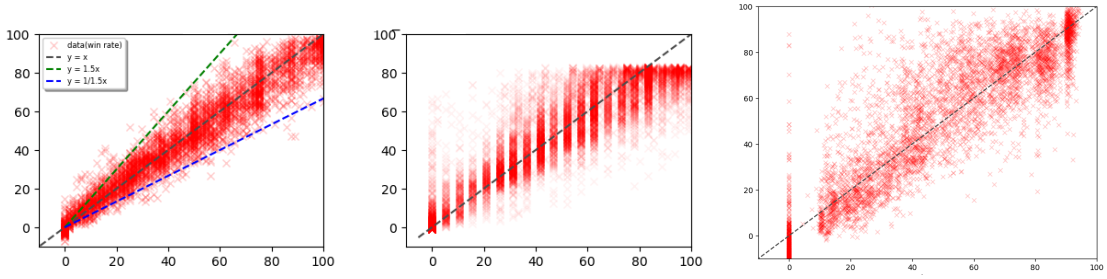
図 6.2 に推定の結果を示す．横軸が実際の値，縦軸が推定された値を表す散布図である．この図の本来の値域はどちらも [0,1] であるが，ここでは便宜上 [0,100] のラベルを振っている．

まず最も単純な勝率の推定について見てみると（図 6.2 (a)），概ね y=x の線に沿って，多少上下に誤差があるような推定ができている．test loss は 0.0055 であり，つまり勝率としては平均的に 7〜8 %程度の誤差があるということである．図 4.2(a) には，y=1.5x と 1/1.5・x の点線も表示しており，概ね 1.5 倍や 1.5 分の 1 の範囲に収まっていることが分かる．

続いて勝率適切度について見てみると（図 6.2(b)），これは (a) の場合よりも誤差が大きいものが目立つことが分かる．test loss も 0.0137 と平均的にも悪化している．図 4.3 を見れば分かるように，勝率のわずかなずれが勝利適切度には大きな違いをもたらすため，このように誤差が大きくなったと考える．

最後により複雑な複合適切度について見てみると（図 6.2(c)），こちらも大きなずれが目立ち，test loss も 0.0203 と悪化した．ただし実用上これがどの程度問題になるかは明確ではない．(c) で適切度が 0.95 以上に評価されたような（つまり実際に選ばれるような）ステージのほとんどは実際の適切度も 0.90 程度であり，良いステージを作るだけならばさほど問題はないかもしれない．

いずれにせよ，上記の結果から，3 行 7 列というステージ入力に対してそれなりに評価値推定ができていることが分かったため，強化学習へと進むことにした．



(a) 勝率　(b) 勝率適切度　(c) 複合適切度

図 6.2: 各評価値に対する予想値と実際値関係

6.2 DQNとDDPGによる良好なステージ生成実験

6.2.1 実験の目的

深層ニューラルネットワークで完成ステージの評価値の推定はある程度正確に行えることが分かったので，6.2節では,「良好なステージを作成する」ための強化学習を行う．手法としては，行動出力の形式が異なる二つの手法，DQNとDDPGを比較し，ステージ生成というこれまで強化学習が用いられてこなかった対象に対してどのくらいの性能が出るのかを確認することを目的とする．本節ではまだ本来の目的である多様なステージという部分については考慮しない．

6.2.2 実験設定

対象となるステージは前節と同じ，9マス（敵6，回復2，ボス1）からなるものであり，これは3行7列の行列で表現される．このうち，先頭の2マスはパラメータをランダムに定めることにする．できるだけ多様な状態を学習させるためにはランダムに定めるマスを多くしたいが，一方でもっと多くのマスをランダムに定めてしまうと，いくら後ろを工夫したところで良好なステージが作成できなくなってしまう（Appendix A.3参照）ため，2マス分とした．

DQNとDDPGの大きな違いは，5.2節で述べたようにその行動形式である．DQNでは，パラメータ一つ一つについて取りうる範囲を100等分する，すなわち101の離散の行動を持つ．先頭2マスが定まっているので，ステージ完成までに行われる行動選択回数敵マスについて$2\times\ 5$，休憩マス2で合計12回となる．一方でDDPGでは，敵マスと回復マスを合わせて3つの連続値ベクトルとして出力するので，7-2回の行動選択が行われる．

DQN のネットワークは 36-64 個の 2 層の convolution 層と 128-256-256-256 個の 4 層の fully connected 層で活性化関数は ReLU で構成される（図 6.3）．パラメータの設定は表 6.1 に示す．

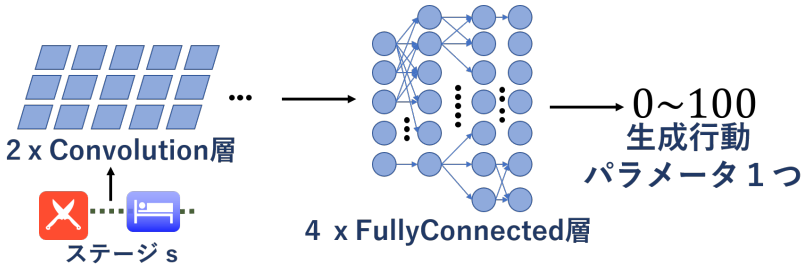


図 6.3: DQN ネットワークの構造

パラメータ	値
エピソード数	20000
メモリサイズ	100000
学習率	0.25×10^{-4}
Batch サイズ	128
γ（gamma）	0.9
target network 更新周期	2000
ε（epsilon）	$1 \rightarrow 0.1$
評価関数	勝率適切度，複合適切度

表 6.1: DQN 実験のパラメータ設定

DDPG は 2 つのネットワーク，行動を決定するための Actor と，Q 値を推測するための Critic からなる．Actor ネットワークは 128-256-512 の convolution 層と 128-256-256-512 の fully connected 層で構成されていて出力層は 3 次元の sigmoid を用いて 0〜1 の出力を持つ（図 6.4）．critic ネットワークは 128-128-256 の convolution 層と 128-256-256-512 の fully connected 層で構成されている（図 6.5）．パラメータの設定は表 6.2 のようになる．

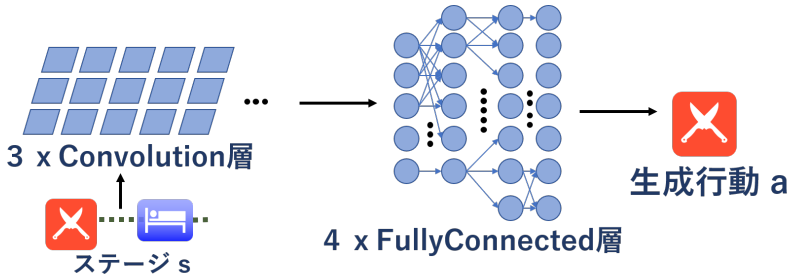


図 6.4: actor ネットワークの構造

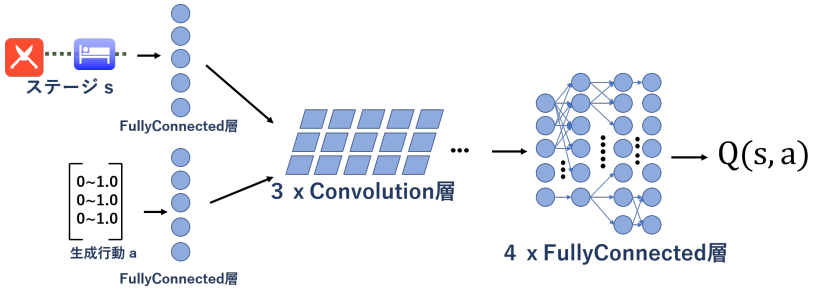


図 6.5: critic ネットワークの構造

パラメータ	値
エピソード数	100000
メモリサイズ	300000
Actor, Critic 学習率	$5 \times 10^{-5}, 5 \times 10^{-4}$
TAU(τ)	0.001
Batch サイズ	64
γ	0.99
exploration	OU noise[38] × epsilon
epsilon	$1 \rightarrow 0.1$
評価関数	複合適切度

表 6.2: DDPG 実験のパラメータ設定

6.2.3　結果

図 6.6 と図 6.7 はそれぞれ DQN と DDPG を用いて生成した，50 エピソード（ステージ）ごとの評価値の平均値の推移を表したものである（学習中の数値であり，ε =0 にして評価したものではない）．DQN は 2 万エピソード，DDPG は 10 万エピソードであることに注意されたい．DQN の場合は平均評価値が 0.78 程度まで至ったが，その後は停滞または悪化しており，これ以上のエピソードを重ねることに意味はない．一方で DDPG の場合は平均が 0.82 程度までいたり，まだわずかながら上昇しているように見える．DQN の設定は表 6.1 や図 6.3 に示した以外にもさまざまに試しており，これよりも良い結果は得られていないため，DDPG のほうがこの問題に対しては適した手法であると言えそうである．

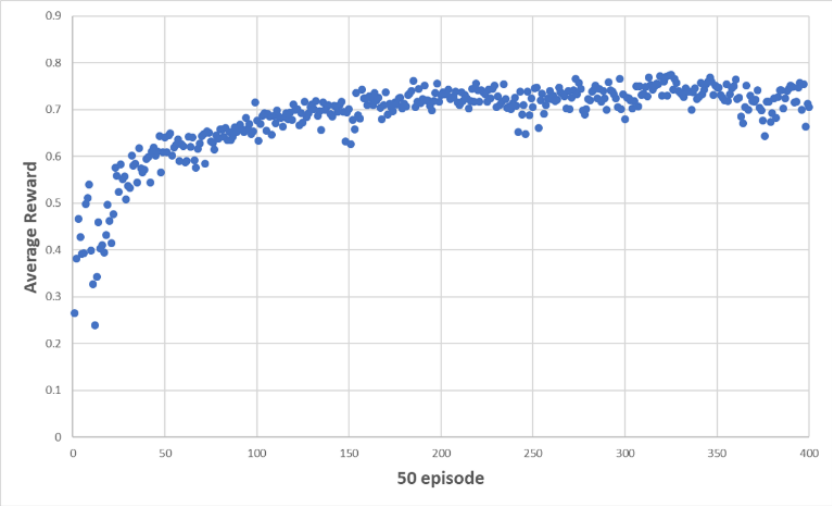


図 6.6: DQN のテージ生成の 50 エピソードの平均評価値（20000 エピソード）

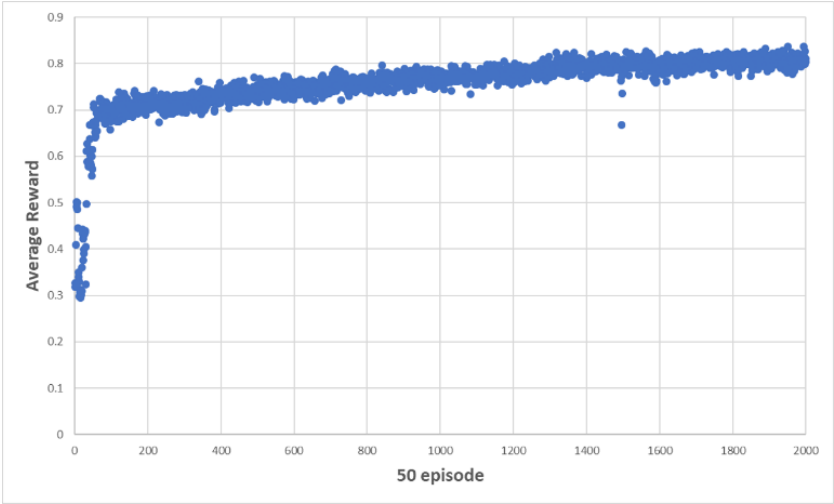


図 6.7: DDPG のステージ生成の 50 エピソードの平均評価値（100000 エピソード）

図 6.8 はランダムに生成したステージとそのステージの 2 ベクトル分（2 戦闘マス）を取り初期ステージとして DQN で生成した時の例になる．上側のステージは全体的なパラメータが低くなっていてどのような戦略を用いても簡単に攻略できるもので勝率適切度は 0 となった．一方上側のステージの一部を初期ステージとして DQN を用いて生成したステージは全体的に敵パラメータが上昇して難しくなっていること，休憩の回復率も上がってメリハリがついていることが分かる．

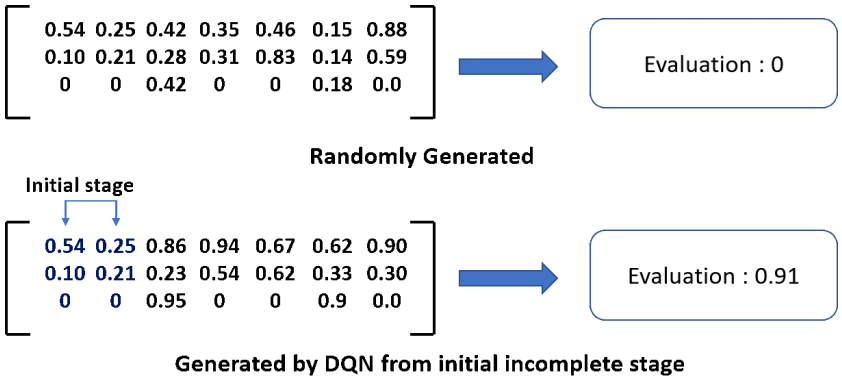


図 6.8: DQN から学習中に生成されたステージの例 (勝率適切度)

図 6.9 には，DDPG を用いて学習した最後の 20000 エピソードで作成されたステージの評価値ヒストグラムを示す．図 4.4（ランダムに生成したステージ）に比べると，ランダムに生成した 2 マスを出発地点としているにもかかわらず，多くの場合最終的には辻褄を合わせて高い評価値のステージを作成することができていることが分かる．

遺伝アルゴリズムによる生成ではないが，図 4.4 からランダム生成検査法だと，満足いくものを作るのに 2000 回くらいランダム生成つまり，80 秒くらいかかる．これに比べて，RL で学習したものは満足いくものを作るのに，3-4 回くらい一部のステージをランダムに作れば良くそれは 弱 0.2 秒でできる．

これに比べて，RL で学習したものは，満足いくものを作るのに，3-4 回くらいランダムに作れば良い．で，多分それは 0.2 秒とかでできる．

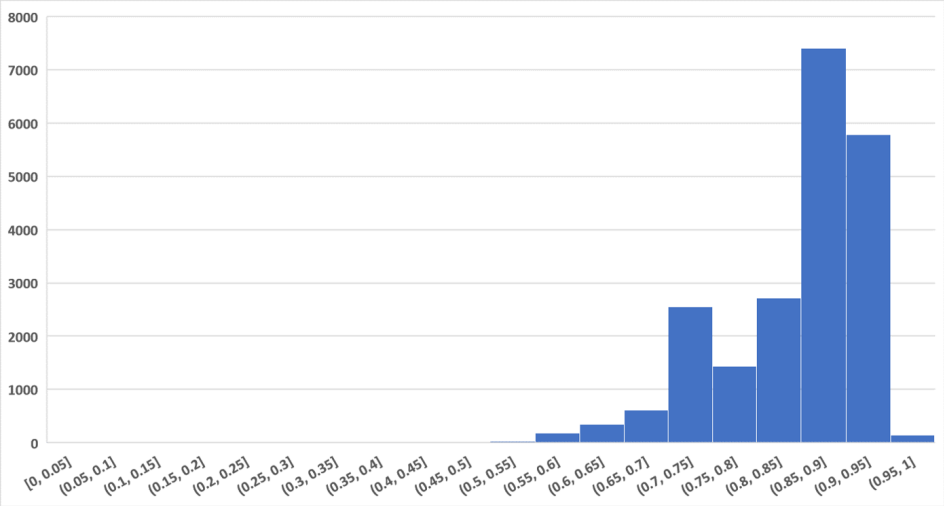


図 6.9: DDPG で学習した最後の 20000 エピソード分の
ステージ評価値のヒストグラム

以上のように，評価関数を与えさえすれば，それが高くなるようなステージを生成するような方策が強化学習で訓練できることが分かった．現実的な規模の問題や，実際に使われるレベルの評価関数で本当に質の高いものが作成できるのかは今後検証していくが，我々の手法が有望であることは示せたと考える．次節以降では，良好なだけでなく多様なステージを作成するための方法について述べる．

6.3 Q 値と評価値の関係把握実験

6.2 節では評価関数による評価値が高いステージを生成することが主眼であった．しかし，我々の主目標は良好で多様なステージの生成である．この目標のためには良好さは維持しながら多様なステージを生成するための方法を考えなければならない．

強化学習では一般的に訓練後は最善の行動を取っていくが，ここで敢えて最善行動とは違う行動を選択することになる．違う行動をとったときにどのくらい得られる報酬（ステージの良さ）が変わるのかというのは未知の問題であり，(1) 最善行動から少し離れると Q 値がすぐに悪くなる，(2) 最善行動から離れると，学習量が少ないために Q 値が信用できない，という 2 つの課題がありうる．

そこで本節ではまず，ある状態における Q 値の分布がどのようになっているのかを調べ，またその学習された Q 値が本当の評価値と比べてどの程度信用できるのかについて調べる実験を行う．

6.3.1 実験設定

6.2 節の結果からは DQN よりも DDPG のほうが性能が良いと想像され，また 3 つのパラメータを同時に生成できる利便性からも，今後の実験は DDPG だけを用いることにする．評価関数は複合適切度であり，その他の設定は表 6.2 と同じである．

対象とするステージ構成も 6.2 節のものと同じで，最初の 2 マス分があるランダムな値を取り，それ以降の 5 列分（敵 4 マス，休憩 2 マス，ボス 1 マスの 7 マス）は DDPG の Actor による最善行動でステージが生成されている．そのうえで，最善行動“以外”の行動に対して Critic がどのような Q 値を持っているかを観察する．

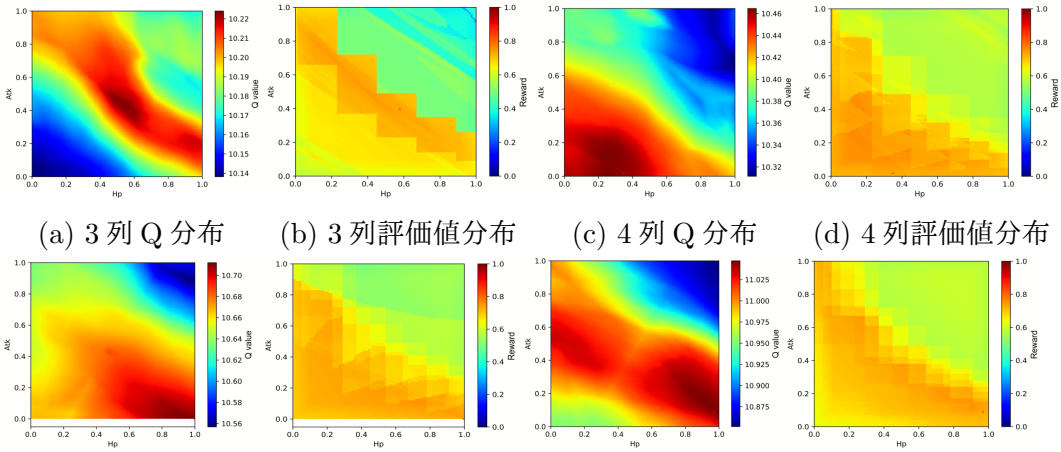
6.3.2 各敵マスでの Q 値と複合適切度の分布

DDPG は連続値の行動ベクトルを扱うことができ，本論文では（敵体力，敵攻撃力，回復率）を $[0, 1]^3$ の 3 次元ベクトルで表す．ある状態において Actor は 3 次元ベクトルの最善行動を提示するが，それとは別に，Critic を参照することで他の行動の Q 値を知ることができる．3 次元の分布は視覚しづらいので，ここでは回復率は（最善の値に）固定して，敵の体力と攻撃力をさまざまに変えた場合の Q 値と複合適切度を見て行くことにする．

図 6.10(a) は，最初の 2 マスが定まり，3 マス目の敵パラメータを決める際の Q 値の分布を表す．赤いほど高い Q 値，すなわち複合適切度の意味で好ましいパラ

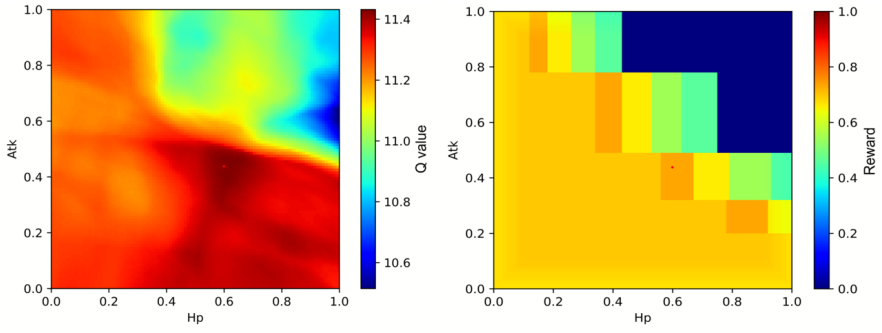
メータであることを意味する．画面中央付近に最も良いパラメータがあるが，全体に右肩下がりの方向に良いパラメータがあるのは自然なことである．なぜなら，体力も攻撃力も両方高い敵は強すぎ，両方低い敵は弱すぎてどちらも適切でない場合が多いからである．なお，Q 値のスケールは 10.14 から 10.22 と「評価値よりも高いところで差が小さい」ものになっているが，このような Q 値の課題評価は DDPG の原著 [33] でも起こりうることが指摘されている．

図 6.10(b) は，同じ 3 マス目の敵パラメータを同様に $[0, 1]^2$ の範囲で変更したうえで，それ以降のパラメータはすべて Actor が最善行動を選んでステージを完成させる，ということをやった場合の，複合適切度の値である．これはつまり，(a) の時点で (1.0, 1.0) のような値を用いてしまうと，それ以降頑張っても複合適切度は 0.3-0.4 程度にしかならないことを表す．逆に，(a) で中央からは離れた (0.0, 0.9) くらいの値を用いた場合は，それ以降を頑張れば 0.7 前後の複合適切度は得られることが分かる．



(a) 3 列 Q 分布　(b) 3 列評価値分布　(c) 4 列 Q 分布　(d) 4 列評価値分布

(e) 5 列 Q 分布　(f) 5 列評価値分布　(g) 6 列 Q 分布　(h) 6 列評価値分布



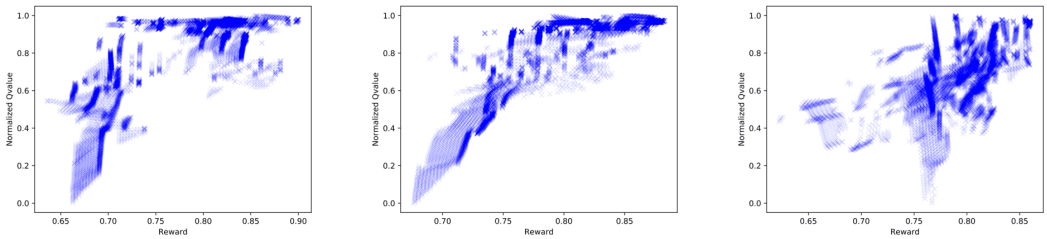
(i) 7 列 Q 分布　(j) 7 列評価値分布

図 6.10: 近似的な全パラメータからの Q 値の分布図, 赤い点：最善のパラメータ

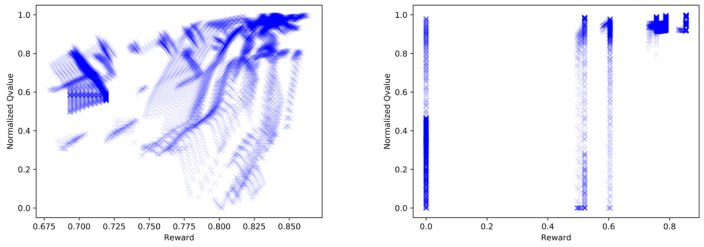
図 6.10(a) と 6.10(b) を比較すると，全体的な傾向は概ね似通っている．4 マス目

の 6.10(c)，6.10(d)，5 マス目の 6.10(e)，6.10(f), 以降も全て概ね同じである．つまり，あるマスで最善行動を取らなかった場合の Q 値は,（その後を最善行動した場合の）実際の評価値をある程度推測できているということである．二つの図を正確に見比べるのは難しいので，各点（行動）について，横軸を実際の評価値，縦軸を Q 値（[0,1] に正規化）としてプロットしたのが図 6.11 である．綺麗に 2 つが対応しているとは言い難いが，概ね「Q 値が低いところ（グラフ左側）は実際の評価値も低く，選ぶべきではない」ということは言えそうである．

本節の実験を通し，多様性を持たせるために“Q 値が悪くなりすぎないように，遠い行動を選ぶ”ことは有望であろうと判断した．



(a) 3 番目ベクトルに対する Q 値と評価値の関係　(b) 4 番目ベクトルに対する Q 値と評価値の関係　(c) 5 番目ベクトルに対する Q 値と評価値の関係



(d) 6 番目ベクトルに対する Q 値と評価値の関係　(e) 7 番目ベクトルに対する Q 値と評価値の関係

図 6.11: ノイズパラメータによる Q 値と評価値の関係性
縦：正規化 Q 値　横：評価値）

6.4　確率ノイズ方策の詳細

6.3 節では，途中の 1 つのマスで“ Q 値が悪くない，最善からは遠い行動 ”を取れば，最終的な評価値もある程度良くなる可能性が高いことが示唆された．そこで本節では，良好かつ多様なステージ“ 群 ”を作るためのノイズ導入アルゴリズムを提示する．

1. ステージの構造を決め，そのどこにいくつのノイズを導入するかを決める．これは決め打ちでもよいしランダムに決めるのでもよいが，本論文の実験では 3 マス目と 6 マス目にノイズを入れることに固定した．

2. 学習は基本的には通常通り行うが，出来るだけ多くの状況を経験してもらって適切な Q 値を学習するために，6.2 節では 2 マスに固定し ていたランダム初期化マスの個数を，本節の実験では 1〜5 マスにした．

3. 生成ステージ数 m, 生成パラメータ n, d を定める．

4. 実際にステージの生成を行う．1. で決めたマス以外では，通常通り Actor に従って次のマスを定める．1. で決めたマスの場合は，n 個のランダムな $[0, 1]^3$ のベクトルを次のマスの候補として作成し，そこからまず Actor の出力との Euclid 距離が d 以下のもの（最善行動に近いもの）を却下したうえで，最も Q 値が高いものを選ぶことにする．

5. 4. を m 回繰り返し，m 個のステージを作成する．将来的にはここに，作成されたステージ間が近すぎないかをチェックする機構を入れる方が良いかもしれない．

6.4.1　多様性の評価

上記アルゴリズムで作成したステージ群が，実際に多様なものであるのかを評価する．多様さというのは数学的に定義できるものと，人間プレイヤが感じるものでは異なってくるが，本論文では，これらに近いと考える 2 つの指標を用いることにする．

一つは parameter mse と呼ぶもので，3 行 7 列のステージ行列 21 個のパラメータについて，各 2 つのステージ間で Mean Square Error を取ったものである．これが大きければ，2 つのステージは離れた敵体力・敵攻撃力・回復率を持っているということになる．

もう一つは，相違有効戦略数と呼ぶものである．これは，プレイヤーが取りうる戦略（本論文の実験の場合は，敵マス 6 つに対する攻撃・逃走，2^6 通り）のうち，各 2 つのステージ間でどれだけ食い違いがあるかを求めたものである．これが大きければ，ステージ A では通用しないがステージ B では通用する戦略，またはその逆が多いということになり，つまり遊んでいてワンパターンではなくなるということである．

6.4.2 結果

提案手法は，パラメータ d と n を持つ．d は，Actor の最善行動と近すぎる行動を却下するものであり，これが大きいと遠い解しか許容しない一方で，その質は悪くなりすぎる可能性もある．n は，候補にする行動数であり，これが大きいとより良い評価値のものを探し出せる一方で，d の値によっては結局最善行動と近すぎるものが選ばれるかもしれない．本実験では，d の値は 0.2 に固定したうえで，n について 5,10,20,50 と値を変化させて，どのような違いが出るかを確認した．

図 6.12 は，m=50 個のステージを作成した場合の，横軸 4 通りの n について，縦軸を評価値平均（青線），Parameter mse（赤線），相違有効戦略数平均（緑棒）としたものである．まず評価値平均について見てみると，n が大きいほど高い評価値となっていることがわかる．これは候補手の多い中から最善のものを選んでいるため自然なことである．一方で，n が大きいほど Parameter mse や相違有効戦略数は減少している，つまり多様でなくなっていることが分かる．これらがトレードオフの関係にあるのは自然なことであり，実際にどのようなパラメータを選ぶかは現実的にどの程度の評価値や多様さが求められるのかによって変わってくるであろう．

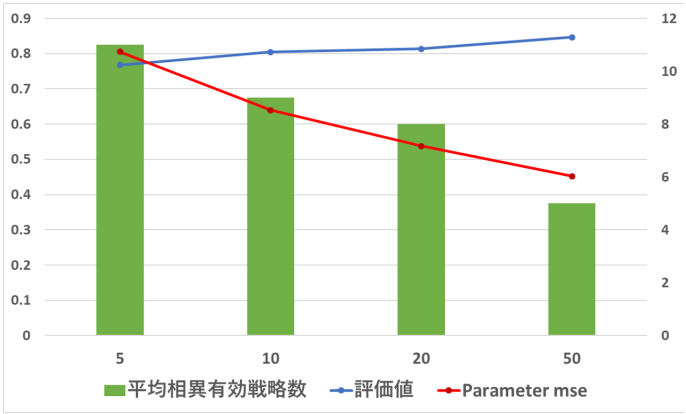


図 6.12: 横軸：n（5，10，20，50）縦軸：候補と最善ステージ同志の平均（緑）異なる有効戦略数,（赤）parameter mse,（青）評価値

最後に，図 6.13 には DDPG で最善行動をとった場合（上側）と，ノイズを入れた場合（下側）の例を示す．最初の 2 マスは同一であるが，その後に 2 回ノイズ行動を入れることにより，評価値はほぼ同じながら違うステージが生成されていることが分かる．実際この二つは，3 番目の敵と戦うべきかどうかなどを含めて 5 つの相違有効戦略がある．

図 6.13: 上：DDPG の方策によるステージ，下：確率ノイズ方策を用いたステージ

第7章　おわりに

この研究では，ターン制 RPG におけるステージ生成をテーマに，従来使われている方法の問題点を指摘したうえで，強化学習を用いることを提案した．ステージの良さを表す評価関数は与えられているものとして今回の研究対象には含めず，それを報酬として用いて，良いステージが生成できるかを確かめた．

強化学習手法としては，方策表現の異なる DQN と DDPG を採用し，比較を行った．それぞれの手法には一長一短あるが，DQN が離散・少数の行動を得意としているのに対して，本論文で扱うような問題では連続値ベクトルを使うことができる DDPG のほうが適切であることが分かった．

単に良好なステージを生成するだけでなく，多様なステージ群を生成することが本来の目的であるため，“最善行動に比べて，遠いが，悪くはない”ような行動を取れるような手法を提案し，それが実際に多様で良いステージを作れることを確認した．

今後の研究の展開としては，それぞれの手法を改善していくこと，それによってより大規模複雑なステージやルール設定に対応できることを示すことが挙げられる．もう一つは，ステージの評価関数そのものの設計法である．被験者実験などを通じて，ステージの評価関数の学習法とその効果なども検証していきたい．

発表論文リスト

1. Nam SangGyu, Ikeda Kokolo, Generation of Diverse Stages in Turn-Based RPG using Reinforcement Learning, In Conference On Games (COG), 2019-8, London

2. ナム　サンギュ，池田心，強化学習を用いたターン制 RPG のステージ自動生成，第 23 回ゲームプログラミングワークショップ (GPW-18)，2018-11，箱根セミナーハウス

3. テンシリリックン　シラ，高橋一幸，ナム　サンギュ，池田　心，コンピューターゲームプレイヤにおける人間らしさの調査，情報処理学会　第 40 回ゲーム情報学 (GI) 研究発表会，2018-6，高知工学大学

謝辞

本研究を進めるにあたり，ご指導・鞭撻を頂きました指導先生である池田心准教授に御厚情に深謝いたします．また，副指導教員の飯田弘之教授，今回の論文の作成における日本語の修正の協力をくれた石井，原口君を含め研究室のメンバーに感謝いたします.

参考文献

[1] Liqian Ma, Qianru Sun, Stamatios Georgoulis, Luc Van Gool, Bernt Schiele, Mario Fritz. Disentangled Person Image Generation, The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.99-108 (2018)

[2] Xu, K. et al. Show, attend and tell: Neural image caption generation with visual attention, In Proc. International Conference on Learning Representations http://arxiv.org/abs/1502.03044, (2015)

[3] Jean-Pierre Briot, Gatan Hadjeres, and Franois Pachet. Deep learning techniques for music generation-a survey, arXiv preprint arXiv:1709.01620, (2017)

[4] Niels Justesen, Ruben Rodriguez Torrado, Philip Bontrager, Ahmed Khalifa, Julian Togelius, Sebastian Risi. Illuminating Generalization in Deep Reinforcement Learning through Procedural Level Generation, arXiv preprint arXiv:1806.10729v5 (2018)

[5] A. Summerville, M. Mateas. Super Mario as a string: Platformer level generation via LSTMs, Proc. 1st Int. Joint Conf. DiGRA/FDG, (2016)

[6] V. Volz, et al. Evolving Mario Levels in the Latent Space of a Deep Convolutional Generative Adversarial Network, in GECCO, (2018)

[7] https://en.wikipedia.org/wiki/Markov_decision_process

[8] S. Snodgrass and S. Ontan. Learning to Generate Video Game Maps Using Markov Models, in IEEE Transactions on Computational Intelligence and AI in Games, vol. 9, no. 4, pp.410-422 (2017)

[9] D. Loiacono, L. Cardamone, P.-L. Lanzi. Automatic track generation for high-end racing games using evolutionary computation, IEEE Trans. Comput. Intell. AI Games, vol.3, no.3, pp.245259 (2011)

[10] J. Togelius, R. De Nardi, and S. M. Lucas. Towards automatic personalised content creation in racing games, in Proc. IEEE Symp. Comput. Intell. Games, pp.252259 (2007)

[11] T. Mahlmann, J. Togelius, G. N. Yannakakis. Spicing upmap generation, in EvoApplications, vol.7248, pp.224-233 (2012)

[12] A. Liapis, C. Holmgard, G. N. Yannakakis, J. Togelius. Procedural personas as critics for dungeon generation, European Conference on the Applications of Evolutionary Computation, pp.331-343, 2015.

[13] A. Summerville et al. Procedural Content Generation via Machine Learning (PCGML), IEEE Transactions on Games, vol.10, pp.257270 (2018)

[14] Diederik P Kingma, Welling Max. Auto-encoding variational bayes, arXiv preprint arXiv:1312.6114, (2013)

[15] A. van den Oord, et al. Conditional image generation with pixelcnn decoders, In Advances in Neural Information Processing Systems, pp.47904798, (2016)

[16] I. Goodfellow, et al. Generative adversarial nets, In Advances in Neural Information Processing Systems, pp.26722680 (2014)

[17] J. Togelius, G. N. Yannakakis, K. O. Stanley, and C. Browne,Search-based procedural content generation, in Proc. EvoAppl, vol. 6024 (2010)

[18] J. Togelius, G. N. Yannakakis, K. Stanley, C. Browne. Search-based Procedural Content Generation: A Taxonomy and Survey, IEEE Trans. Comput. Intell. AI Games, vol.3, no.3 pp.172186 (2011)

[19] B. De Kegel and M. Haahr, Procedural Puzzle Generation: A Survey, IEEE Transactions on Games, vol.1, no.1 (2019)

[20] P. Song, C.-W. Fu, and D. Cohen-Or, Recursive interlocking puzzles, ACM Transactions on Graphics (TOG), vol.31, no.6, pp.128 (2012)

[21] M. Stephenson, J. Renz. Procedural generation of complex stable structures for angry birds levels, in 2016 IEEE Conference on Computational Intelligence and Games, pp.18, (2016)

[22] M. Guzdial, N. Liao, M. Riedl. Co-creative level design via machine learning, arXiv preprint arXiv:1809.09420 (2018)

[23] M. Guzdial, et al. Friend, Collaborator, Student, Manager: How Design of an AI-Driven Game Level Editor Affects Creators, arXiv preprint arXiv:1901.06417 (2019)

[24] Mnih, V. et al. Human-level control through deep reinforcement learning, Nature 518, pp.529533 (2015)

[25] https://ja.wikipedia.org/wiki/ドラゴンクエストシリーズ

[26] https://ja.wikipedia.org/wiki/ファイナルファンタジーシリーズ (2019/08/05)

[27] https://www.4gamer.net/games/383/G038332/20171124003/ (2019/08/05)

[28] N. Sato, K. Ikeda and T. Wada. Estimation of player's preference for cooperative RPGs using multi-strategy Monte-Carlo method, IEEE Conference on Computational Intelligence and Games (CIG), pp.51-59, (2015)

[29] https://www.darkestdungeon.com/ (2019/08/05)

[30] https://www.megacrit.com/ (2019/08/05)

[31] 高橋 一幸，Temsiririrkkul Sila， 池田 心. ローグライクゲムの研究用プラットフォーム GAT2018 論文集, (2018)

[32] Kanagawa, Y., Kaneko, T. Rogue-Gym: A New Challenge for Generalization in Reinforcement Learning, CoRR, abs/1904.08129, (2019)

[33] T. Lillicrap, et al. Continuous control with deep reinforcement learning, arXiv preprint arXiv:1509.02971 (2015)

[34] Vinod Nair and Geoffrey Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines, ICML (2010)

[35] Srivastava N., et al. Dropout: A Simple Way to Prevent Neural Networks from Overfitting Journal of Machine Learning Research, vol.15, pp.1929-1958 (2014)

[36] Ioffe Sergey, Szegedy Christian. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, arXiv:1502.03167 (2015)

[37] Prechelt L. Early stopping — but when? In: Orr GB, Muller OR, editors.Neural networks: Tricks of the trade, Springer-VerlagTelos, pp.5769, (1999)

[38] Uhlenbeck, George E and Ornstein, Leonard S. On the theory of the brownian motion. Physical review, vol.36, no.5 pp.823, (1930)

[39] Silver David, et al. Deterministic policy gradient algorithms, In ICML, (2014)

付 録A章　Appendix

A.1　Deep Q-Network

DQN は多層の neural network が状態 s を入力として与えられたとき，ネットワークパラメータ θ により状態を近似させ全行動の $Q(s, a, \theta)$ を出力する．DQN では 2 つの重要な要素があって，遷移記録 $(s_t, a_t, r_{t+1}, s_{t+1})$ を保存する Experience replay と毎 T ステップで θ を複製した θ ' を持つ target network である．そこから次の loss 関数を最適化するように θ の更新を行う．

$L_\theta = E_{s,a,r,s' \sim D_\theta}(r + \gamma \ max_{a'} Q(s', a', \theta') - Q(s, a, \theta))^2$

A.2　Deep Deterministic Policy Gradient

DDPG は continuous control のため提案されたアルゴリズムになる．DQN と同様に Experience replay を用いて，target network も使用する． Critic と Actor の 2 つのモデルを使用すして，Critic は状態と行動を入力としてその Q 値を出力し，Actor は状態を入力し行動を出力する方策ネットワークである．

Critic ネットワークの更新は DQN と同様に target と Q 値の最適化する方向で更新を行い．Actor ネットワークは次の Deterministic Policy Gradient[39] からの更新を行う．

$\nabla_{\theta^\mu} \mu = E_{\mu'}[\nabla_a Q(s, a, \theta^Q)|_{s=s_t, a=\mu(s_t)} \nabla_{\theta^\mu} \mu(s|\theta_\mu)|_{s=s_t}]$

A.3　初期サイズによる評価値把握の実験

実験の目的

我々は 6.1 の結果から我々が実装したターン制 RPG 環境ではパラメータから Q 値を推定することが可能だと判断した．次の段階として望ましいステージを生成す

る，つまり強化学習を用いて評価値が高いステージを生成する実験を行う前に初期サイズによる生成結果の影響を知るため異なる初期サイズから学習を行った．

実験設定

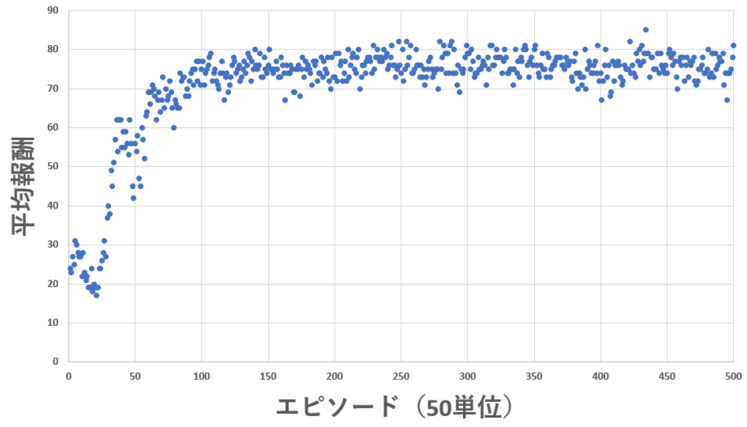
ステージの構成は 6.1.2 と同様であって 3×7 の行列で表現される．詳細の設定は表 A.1 のようになる．

パラメータ	値
学習モデル	DQN
初期ステージサイズ	2, 4, 6
エピソード数	25000
メモリサイズ	100000
学習率	$0.25\text{x}10^{-4}$
Batch サイズ	128
γ （gamma）	0.9
target network 更新周期	2000
ε （epsilon）	$1 \rightarrow 0.1$

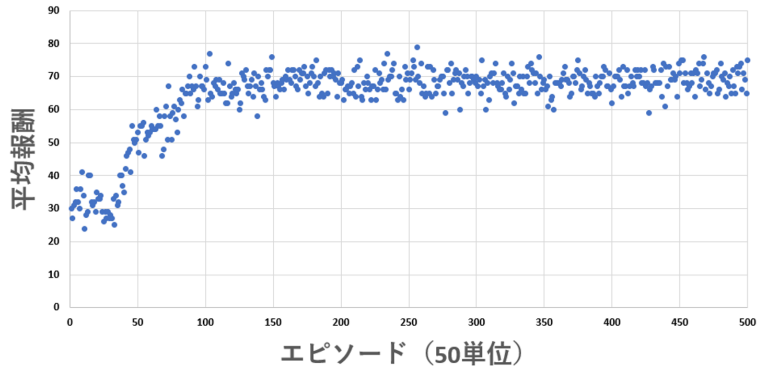
表 A.1: 異なる初期ステージの実験のパラメータ設定

結果

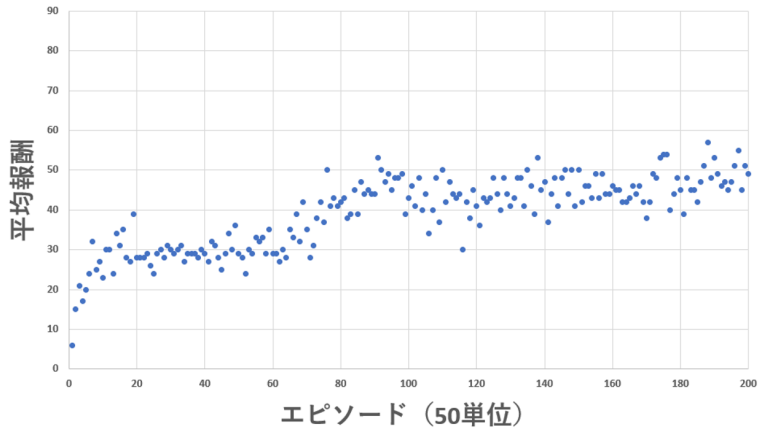
図 A.1 は DQN で勝率適切度を評価関数として異なる初期サイズから学習を行った時の評価値（0 100）になる．初期ステージのサイズが大きいということはランダム性が多くなり，多様性は保証するがランダム性が大きい分，生成される物の評価値が低い（図 A.1c）．逆に初期ステージのサイズを減らすことで多様性は減るがランダム性が減ることで生成されるステージの平均評価値は上がる（図 A.1b）（図 A.1a）．



(a) サイズ 2



(b) サイズ 4



(c) サイズ 6

図 A.1: DQN の初期ステージのサイズによる生成されるステージの評価値