

強化学習を用いたターン制 RPG のステージ自動生成

ナムサンギョ^{†,1} 池田心^{†,2}

概要: ターン制 RPG はドラゴンクエストシリーズ等で有名なゲームジャンルで、様々な強さを持つ複数の敵グループを工夫しながら撃破していくことが中心的課題である。簡単すぎず難しすぎず、工夫や努力が報われるように敵味方のパラメータのバランス調整を行うことはゲームの面白さにとって大変重要である。近年ではプレイヤーの飽きを防ぐために毎回ランダムにステージを生成することも行われ、そうするとプロのデザイナーの代わりにコンピュータにバランス調整を行わせる必要がある。従来良く使われる Search-based PCG にはオンライン生成に向かないという制約があるため、本研究では強化学習の枠組みを試みる。未完成のステージを状態、次に戦わせる敵グループを行動として、ステージが完成したらその良さを報酬として与える。学習後は任意の未完成ステージについて良いステージを高速で完成させられるという狙いである。本論文では非常に単純化した環境において、特定の目的プレイ勝率を与えた場合に誤差 5% 程度のステージを作成できるような強化学習に成功した。

Automatic Generation of Stages in Turn-Based RPG Using Reinforcement Learning

SangGyu NAM^{†,1} Kokolo IKEDA^{†,2}

Abstract: Turn-based RPG is a game genre which is famous as Dragon Quest series, and it has a main goal which is defeating many enemy groups with diverse strong considering many strategies. It is very important for entertainment of a game to balance the parameters of player and enemy, not so easy and not so hard or rewarding player's effort. Recently in order to keep players not be bored, some RPGs generate random stages at every playing times, to do that it is needed to adjust parameters by computer instead of pro-designers. In this research, we tested a reinforcement learning method, because Search-based PCG which is often used conventionally is not suitable for online generation. We considered reinforcement learning elements like "incomplete stage as a state", "next battle parameters as an action", "evaluation goodness of complete stage as the reward". After finishing learning, it will be possible to generate good stages quickly, when given random incomplete stages. In this paper, we succeeded to create a stage with about 5% error when giving a specific target winning rate, in very simplified environment.

1. はじめに

コンピュータゲームにおいて、ダンジョン、マップ、キャラクタ、アイテムなどから、音楽、グラフィック、思考ルーチン、ストーリーまで含む要素をまとめて“コンテンツ”と呼ぶことが多い。これらは全部手作業で作成されることが多かったが、(1) ゲームの複雑さに伴うコスト増を抑制するため、(2) ユーザーに毎回違うプレイ経験を与えるため、または(3) データ圧縮等のために、コンテンツをアルゴリズムから自動生成することがしばしば試みられる。これは Procedural Content Generation (以下 PCG) と呼ばれ近年活発に研究されている領域である。

PCG に関する研究はスーパーマリオなどのアクションゲーム[1]、レースゲーム[2]、RTS ゲームのマップ生成[3]、あるいはパズルの問題生成[4]など広い対象に行われている。一方で、特に日本で人気のターン制ロールプレイングゲーム (以下 RPG) についての研究は比較的少ない。ターン制 RPG は敵小グルー

プとの戦闘を、休憩を挟んで連続的に行うため、「この戦闘に安全に勝つ」「先を見通してアイテムや魔法力を温存する」「勝つのが無駄な敵からは退却を試みる」などの戦略の選択が難しく、それゆえに面白い。

複数の有力な選択肢が存在するような状況を作るためには、敵や休憩場所の配置、彼我のパラメータ設定などの“ステージ”のバランスが適切である必要がある。本研究はこの「バランスの良いステージを自動生成する」PCG 手法を研究対象とする。

PCG には様々な接近法がある。人間が作ったコンテンツが多量に入手可能なら、教師あり学習や GAN[5][8]などの方法で“新しい似た”コンテンツを作ることが可能かもしれない。これとは別に、生成検査法を用いることもよく行われる。この方法は、学習データを必要としない代わりに評価関数を必要とするが、各プレイヤーの実力や好みに合わせて評価関数を変更できることは利点である。生成検査法では、遺伝的アルゴリズムを用いて評価値の最適化を行うことがしばしば行われる[6]。しかし、遺伝的アルゴリズムはプレイヤーごとにただちにコンテンツを提供するにはやや低速であり、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。

この不満を解決するため、我々は強化学習を用い

[†] 北陸先端科学技術大学院大学
Japan Advanced Institute of Science and Technology, Nomi, Ishikawa
923-1211, Japan

¹ s1610424@jaist.ac.jp

² kokolo@jaist.ac.jp

ることを考える。n 個に連なるステージを想定し、作成途中のステージを「状態」、次の 1 ステップの作成を「行動」とみなす。ステージ完成をゴールとみなし、ステージの良さ悪さを「報酬」として与える。学習を進めると、良いステージが作成できるようになり、かつ確率的政策を用いて多様なステージが低コストで得られるようになって考えている。

2. ターン制 RPG

ターン制 RPG の代表としてはドラゴンクエスト、ファイナルファンタジー、ポケモンシリーズが挙げられる。RPG の多くは戦闘パートと非戦闘パートに分かれる。戦闘パートとはプレイヤーチームと敵チームに分かれ、どちらかの全滅または退却により終了する。戦闘パートの最終状態は引き継がれるため、プレイヤー側にとっては「ただ勝てばよい」というわけではない点が多く、ゲームと異なる。非戦闘パートでは傷を回復したり、装備を買ってユニットを強化したりする。「弱い敵から金を奪い装備を買ひ、敵本拠地に入り込んで魔法力を温存してボスを倒す」などの中長期的な目標があり、進撃と撤退、魔法の使用不使用など、さまざまな判断が求められる。言いかえれば、これらの判断が要求されるようなステージが生成されることが望ましい。

2.1 戦闘パート

ターン制 RPG の戦闘はターンを中心に全キャラクターが行動をする。行動の順番はゲームそれぞれで微妙に違うが、基本的に速度のパラメータが高い順番に行動を取る。各キャラクターは攻撃・魔法など固有の可能行動を持ち、それぞれ自分の役割に合わせた戦略を用いて「より良い勝利」を目指す。このように各キャラクターに個性や役割がある点が、Role (役割) Playing と呼ばれる所以である。

例えば図 1 は 1 ターンの流れの図である。

- 戦士の速度パラメータが一番高く、攻撃行動を幽霊に対して行った。
- 次に速度が速い幽霊が毒攻撃を戦士に与えた。
- 次に「トレント」が戦士を攻撃した。
- 最後に、僧侶が戦士を回復した。

こうして、全キャラクターが一回ずつ行動を取ることによって 1 ターンが終わる。これらのターンはプレイヤーまたは敵のチームが全滅または退却するまで続けられる。戦闘パートで勝利すると、キャラクターの成長、金や装備などの戦利品が得られることが多い。

2.2 非戦闘パート

非戦闘パートについてはターン制 RPG の間でもゲ

ームそれぞれに異なる点が多いが、いくつかの共通するところがある。ストーリーを進める、宿屋で休憩を取って回復する、武器屋で武器を買ひキャラクターを強化する、道具屋で回復アイテムを買う、宝箱からアイテムを得るなどの基本要素は大体のターン制 RPG では共通する点である。

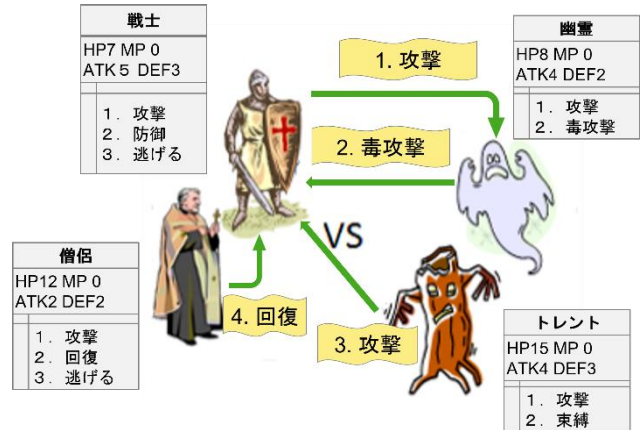


図 1 ターン制 RPG の戦闘パートのイメージ

2.3 RPG のステージ評価関数

RPG のステージ生成は人手で行うことが多かったが、最近はプレイヤーに新鮮な経験を与えるため、これらの設定をプレイする度に新しく生成できるゲームも増えてきている。しかし、これらターン制 RPG のパラメータは相互に緊密に関わり合っており、単にステージをランダムに決定するだけでは満足なコンテンツを生成することが難しい。例えば

- 1) どうやってもクリアできる、どうやっても運任せでしかクリアできないなど難易度が極端になればゲームとしては面白くない。
- 2) 敵の配置や彼我のパラメータが多少変わったとしても、結局ワンパターンな攻略手順になるのでは面白くない。状況に応じて多様な戦略があり得るほうが良い。
- 3) 取りうる行動や戦略が「明らかに良い」「明らかに悪い」といつでも明白であるのは面白くない。適度に悩む状況があるほうが良い。

といったことを考える必要があり、これらのようなステージの満足度を何らかの評価関数を用いて数値化することは重要な課題であるため、これ自体が研究の対象である。本論文では基本的には評価関数は与えられているものとし、生成法に焦点を当てる。

3. 研究環境

市販の RPG はゲームの特性上、濃密なストーリーや多くのゲームモード、移動や自由なマップなどが

与えられていることが多いが、本研究ではこれらの要素ではなく、戦闘関連の部分の主たる興味の対象とする。このような機能を限定した研究用プラットフォーム環境を研究に用いることは他のジャンルでも普通に行われるが、我々は RPG では適したものを見つけられなかったため、以下のような独自のプラットフォームを実装した。この枠組みで少なくとも、人間プレイヤーが楽しく悩めるようなステージを人間が作成できることは確認している。

- ① ステージの構成: ステージは n マスからなる一方向一列の構造を持つ (図 2)。各マスは、敵・成長・休憩・ボスなどからなる
- ② ユニットの種類: 彼我のチームは複数のユニットからなり、ユニットは体力・攻撃力・速度などのパラメータ、回復スキル・全体攻撃スキルなどを有する。ボス戦以外はプレイヤー側は逃走も選べる。逃走すると HP の減りを抑えられる半面、成長ができないので後半に悪影響がある。
- ③ 行動順: 各ユニットは 1 ターンに 1 回行動し、速度パラメータの高い順に行動を実施できる

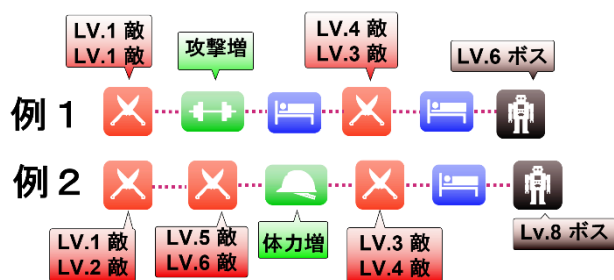


図 2 ステージの構成例

・研究の大目標と本論文での目標

本研究の最終的な目標のために必要な技術・過程は以下のようなものがある。

- 1) 単純化されていながらも、工夫や思考のしがいのある RPG プラットフォームの作成
- 2) ステージの面白さを検査するための人間らしい AI の作成
- 3) その AI のテストプレイによる、さまざまな要素を考慮したステージの評価関数の作成
- 4) 強化学習を用いた PCG
- 5) Search-based PCG との被験者実験による比較

これらの高水準な完了は少し先の話と考えており、本稿ではこの研究の第一歩として次のように単純化して実験を行うことにする。

- 1') 1) をベースに単純化したプラットフォーム
- 2') 単純な思考ルーチンによる検査

3') 適切な難易度のみに着目した評価関数

4') 4) と同じ

5') 比較ではなく、作成した問題の主観的な確認
具体的な単純化環境は次のようになる

1') 単純化したプラットフォーム

最終的には多数対多数の戦闘で、使用する行動も多様であるような環境を利用したいが、今回は「1つの戦闘内の挙動」ではなく「戦闘の連なり、ステージとしての評価と生成」に着目するため、生成したプラットフォームにさらに制限を加えた。厳密には、ここまで単純化したゲームは Role Playing をしているとは言い難いかもしれないが、将来的な展開を見据えての単純化ということでご容赦いただきたい。

- ・敵味方のキャラクタは 1 つに制限する
- ・体力と攻撃力のみをパラメータとして持つ
- ・可能な行動は攻撃と逃走のみとする。HP が減っても成長のために攻撃するか、強い敵からは逃げるか、が迷う部分となる。

2') 単純な思考ルーチンによる検査

本来戦略は人間のプレイスタイルの分析の後、それらに対して真似ができる人間らしい AI からその戦略を用いることが目標であるが、それ自体が一つの研究として大きな課題を持っている。1' で述べたように可能な行動は 1 つの敵について 2 通りだけであるため、思考ルーチンは単純化できる。

ボス戦は退却不可のため、これを除いた m 個の戦闘マスがあるステージに対して、戦闘マスで取れる m 個の (攻撃だけするまたは逃走) 行動の一連を戦略として見る。つまり m 個の戦闘に対してありうる戦略の数は 2^m 個になる。この様に戦略を定義することで、戦略空間の全探索が可能になる。

3') 適切な難易度のみに着目した評価関数

2.3 節で述べたように、本来ステージを評価することはそれだけで難しい課題であり、さまざまな要素を考慮する必要がある。本稿では、その第一段階として、難易度のみに着目する。2' で述べたようにプレイヤーには 2^m 個の戦略しかない。これら全ての帰結を調べることで、そのステージが「どうやっても解ける」「どうやっても解けない」ものでないかを確認することができる。具体的には、全戦略のうち何回ボスを倒すことができたかの割合を“勝率”と呼ぶことにする。90%であればほぼどうやってもクリアできるものであり、5%であれば相当に先読みをして慎重に行動を選ばなければクリアできないものということである。

4. 既存研究

PCG に関する既存手法は概ね「何らかの方法でコンテンツを生成する」「場合によっては生成されたものを評価して、良いものだけ使う」という構造になっている。生成法としては次のようなものがある。

- ーヒューリスティック
- ーオートマトンやフラクタルなどの科学モデル[7]
- ー既存データの一部を再利用する
- ーGAN などの教師あり学習を用いた生成モデル[8]

しかし、例えば地形生成のようにゲームの勝敗に小さな影響を与えないものならばともかく、パラメータが前後に複雑に影響しあうターン制 RPG では、ただ生成しただけでは満足いくステージを保証できない。

そこで、何らかの生成法に加えて評価関数による選別を組み合わせた“生成検査法”が用いられることが多い。コンテンツ x の良さを表す評価関数 $f(x)$ を準備し、多くの x を生成したうえで $f(x)$ が良好なものを提示するような方法である。図 3 は 3 ステップからなるステージを遺伝的アルゴリズムで最適化し、生成する例である。まず、4 つの個体がランダムに生成される（図左）。これらは人間テストプレイヤを模した AI によりプレイされ、その難易度や面白さが評価される（図数字）。良い個体は交叉操作で次世代に残し、悪い個体は淘汰される（図中央）。これが繰り返されることで、良いステージを得ることができる。

この手法には 3 つの課題がある。1) 人間にとっての面白さを評価する関数が適切に作れるか、2) 最適化によって十分関数値を高くする解が得られるか、3) 解を高速または多様に作れるか、である。特に 3) は遺伝的アルゴリズム PCG に特有の問題であり、オンラインでのコンテンツ生成を困難にしている。すなわち、開発者側で一度作っておいて多数のプレイヤで共有すれば良い場合は問題ないが、各プレイヤ側で好みや実力に合わせて自分専用の多様なステージをただちに作成することが難しくなるのである。

遺伝的アルゴリズムは多数の解を保持するために通常多くの評価回数を必要とし、さらに世代交代モデルによっては多数の解の割に最終的には似通った解に収束してしまうこともある[9]。我々は、オフラインでの長時間の学習のあと、オンラインでは短時間で多様なステージを生成できるようにするために、強化学習を用いることを考える。

5. 適用手法

我々は、未完成ステージをマルコフ決定過程における状態、未完成ステージを完成に近づける操作を

行動とし、完成ステージをゴール状態、その評価関数値を報酬とする強化学習である。n 駒の完成ステージに対して、任意の n より少ない駒数の初期状態と確率的な方策選択を用いれば、学習後には多様で良好な完成ステージをオンラインで高速に作成することが可能になると考える。

そのため、手法として強化学習の DQN を選んだ。DQN は膨大な状態空間を対象にしている多数のゲームに対して、良い結果を得ていることが知られている[10]。RPG のステージはそれぞれ複数のパラメータを持ち、マスが連続して構成されているので、状態空間が大きい。行動もまた、「次の駒（敵等）をどのように設定するか」を表し、そのパラメータの範囲にもよるが行動空間も大きくなる。そのため、強化学習 MDP の定式化、ステージの表現、状態空間の汎化、行動の定義など、考慮すべき問題が多い。

発展的課題になるが、膨大な状態と行動からの Q 値の汎化推定精度によっては作成されたステージが必ずしも適切なものにならないと思われる。その場合は再度評価関数を用いることで確認、再生成を行う。このような処理を組み合わせても、最初から遺伝的アルゴリズム等で生成検査法を実装することに比べれば計算時間は短くなると考える。

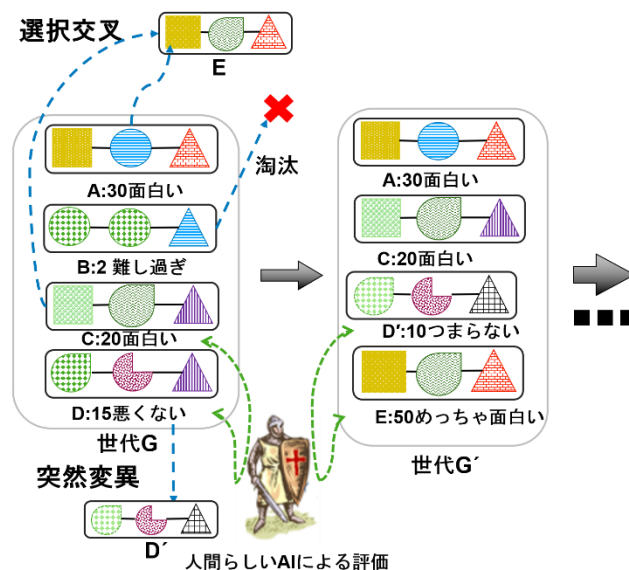


図3 遺伝的アルゴリズムによるステージ生成

5.1 強化学習を用いたステージ、MDP の定式化

本節では我々の提案手法をMDPの定式化から説明する。まず、n ステップからなる完成ステージ $s^* \in S^*$ について評価値 $f(s^*)$ が存在して、その評価値は AI プレイヤのテストプレイから得られることとする。この部分は「難しさとは何か」「面白さとは何か」「人間の盲点がコンピュータに分かるか」などの難しい課題を持つが、それは従来の生成検査法と同じであ

る。

続いて、 n より少ないステップ数の未完成ステージの集合 S を状態空間とし、1 ステップを追加して行く操作（未完成ステージを完成ステージ完成に近づける操作）を行動 $a \in A$ とする。MDP のエピソードは完成ステージ $s^* \in S^*$ に到達すると終了し、報酬として評価値 $f(s^*)$ が与えられるとする。このような MDP の定式化により、ステージ作成問題を報酬最大化の強化学習に帰着することができる。これを Q 学習することができれば、任意の未完成ステージ $s \in S$ と行動 $a \in A$ に対する Q 値 $Q(s,a)$ が学習でき、これが最大または次善となるような a によってステージを構成していくことができる。

5.2 ステージ行列表現

各ステージは n 個の戦闘マスで構成されていて、各マスの m 個のパラメータをベクトルで表現する。そのベクトルを繋げた行列 ($m \times n$) を入力として使う。具体的な数値は以下の通りである。

・戦闘・ボスマス

敵ユニットは体力と攻撃力を持つ。これは、作者によってそれぞれ上下限が定められている。体力は $HP_{min} \sim HP_{max}$ 、攻撃力は $ATK_{min} \sim ATK_{max}$ である。これに対して、下限からの割合

体力割合 = (体力- HP_{min}) / (HP_{max} - HP_{min})

攻撃力割合 = (攻撃力- ATK_{min}) / (ATK_{max} - ATK_{min})

をベクトルの 2 要素として持つことにする。

・休憩マス

戦闘が終わったあとに、一定割合体力を回復できるマスを設ける場合がある。これは n 個のうちの 1 つのマスのみとして扱わず、戦闘マスの中に情報を入れる。具体的には、回復の割合をベクトルの 3 番目の要素とする。

図 4 は生成範囲が体力 (20~120)、攻撃力 (5~30) における、本質的には 3 マス、行列表現としては 2 マスのステージに対する例である。

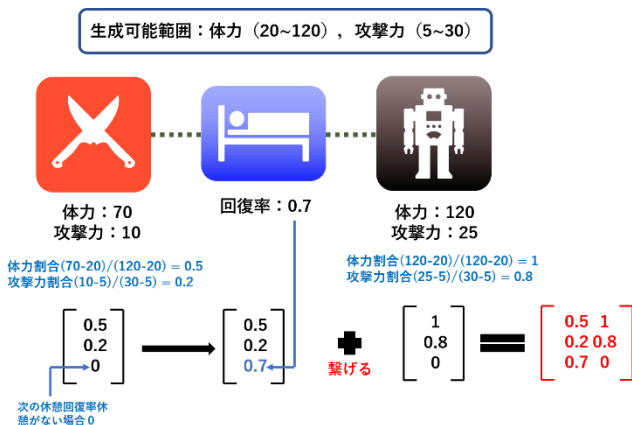


図 4 戦闘ー休憩ーボスのステージの行列表現例

5.3 行動の選択

5.2 節で状態の行列表現を説明した。続いて、DQN の出力である行動の実装を説明する。自然に考えれば、次の敵のパラメータ（体力割合、攻撃力割合）の組み合わせを行動とすべきだが、これだと行動空間が大きくなることを踏まえ、体力割合と攻撃力割合を別々に生成することにする。

例えば、「3 匹目までの敵とその後の休憩」まで作成された状態ならば、出力は「4 匹目の敵の体力割合」のみとする。出力ノードは、0~100 の離散値すなわち 101 ノードとした。その次の行動はその体力を踏まえての「4 匹目の敵の攻撃力割合」、続いてもし休憩があるなら「4 匹目の敵の後の回復割合」を同様に 0~100 の離散値で出力する。割合 0~100 を全て離散値の別のノードとして扱うことは効率が悪いかもしれない。例えば 54 と 55 では殆ど同じパラメータであるのに、学習時には 1 つしか更新されないためである。この点については工夫の余地がある。

5.4 多様さのステージ評価関数設定

本研究で最終的に目的にしているステージは、簡単すぎず難しすぎず、多様な戦略を比較検討する必要があるような、やりがいのある面白いステージである。これらを反映するステージを生成するためには、そのための評価関数をデザインする必要がある。

本来評価関数に入れるべきものは沢山あるが、今回は最初の実験として有効な戦略の比率のみに着目した。3 章最後で説明したように、この単純ゲームではプレイヤーが取りうる戦略（行動列）は 2^n 通りに限定される。その中で何個がゲームクリアに通じるのかを“勝率”と呼ぶ。勝率が高すぎても低すぎても、好ましいステージとは言えない。

この勝率がどの程度が良いかは場合によるだろうが、本稿では目標値を 30% と規定し、このとき“勝率適切度”としてステージの評価値 $f(s)=100$ を与えることにする。勝率が 0% と難しすぎる場合は評価値 0、その間は線形に補間する。また勝率が 60% 以上と簡単すぎる場合も評価値 0 とし、その間は線形に補間する。簡単なステージや難しいステージを作るとは簡単であり、勝率 30% になるようなステージを作るとはそれよりは難しいことであると考えられる。

6. 予備実験：勝率の教師あり学習

今回、我々が提案するステージ生成モデルは未完成ステージを含む全ステージ S に対して、 Q 値（もしくは状態評価値）を表現・推測することが可能であることが前提条件である。完成ステージに対しては報酬を計算するための評価法（AI プレイヤーによるシ

ミュレーション)を用いることができるが、それ以外の状態では Q 値を何らかのモデルで持つしかない。テーブルで持つには大きすぎるため、汎化が必要になる。しかし、ステージは前後に大きく関連しており、その評価値をパラメータのみから推測できるのかには疑問が残る。

そこで、予備実験として、完成ステージを入力、勝率または評価関数値を教師出力とする教師あり学習を行ってみた。もし、完成ステージだけに対しても満足な推定精度が出ないようなら、強化学習における Q 値を正しく学習することはとても望めないためである。

6.1 実験条件

6.1.1 ステージとそのサイズ

9 マスサイズの戦闘—戦闘—戦闘—休憩—戦闘—戦闘—戦闘—休憩—ボス。ボス戦を除くと $2^6 = 64$ 戦略がありうる。休憩マスは戦闘マスと同じ場所に表現されるので、入力は 3×7 サイズのステージ行列になる。

6.1.2 ネットワーク設定

ネットワークの隠れ 2 層の Convolution (32-32 filter size3) と AveragePooling (pool size3) と 7 層の fully connected 層 (256 ノードの 7 層) で構成され、活性化関数としては ReLU、最適化は adam を用いた。これはステージが長く前後のマスが緊密に関わっていて、層を深くする必要があると考えるためである。過学習を防ぐために Dropout と Batch Normalization を用いた。

6.1.3 教師データ

一つめの実験では、生の勝率を出力とした。12000 個の訓練ステージと 4000 個のテストステージを与えるが、勝率がほぼ均等に分布するように生成した。

二つめの実験では、勝率の適切度を出力とした。40000 個の訓練ステージと 12000 個のテストステージを与えた。同様に“勝率を”均等分布にした。この場合、勝率適切度 100 に近いものは少なくなる(勝率 30%前後のみ)ことに注意されたい。

6.2 結果

実験 1, 勝率に関する最終的な validation loss は 55 となった。これは勝率の推定誤差が大体 8% ぐらいで、64 戦中の勝ち数にすると ± 5 勝程度である。

勝率に関しては、0%のものを 10%と推定することは、60%のものを 70%と推定するより罪が深いという関係がある。図 5 には、3 本の線 ($y=x$ が正しい推

定、ほかに $y=1.5x$ と $y=1/1.5x$) を記入している。この範囲内には 92%のテストデータが含まれており、勝率推定が誤っていても 1.5 倍や 1/1.5 になることは少ないことが分かった。

実験 2, 適切度に関する最終的な validation loss は 135 となった。適切度の定義の関係で、小さい誤差が大きな loss に見えるが、これは推定した勝率適切度が実際の勝率適切度と平均 12 ぐらいの差がある意味で平均的に勝利した戦略の誤差は 2 個分にすぎない。

図 6 は実際の勝率適切度に対するネットワークからの予想値である。大体のデータが望ましい $y=x$ の線形関係になっている。適切度 83 以上に推定しているものがないのは、そのような訓練データがそもそも少ないからと考えている。

実際のステージ生成においては、「適切度低いと思っていたものが高かった(過小評価)」よりは、「評価値高いと思っていたものが低かった(過大評価)」のほうが問題である。そこで、60 以上と推定したものが実際に 60 を超えている割合を見てみると 84%であった。十分とは言えないがある程度は信用できるものであると言える。

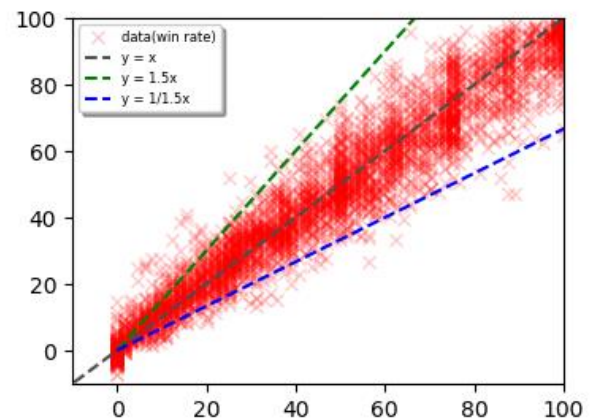


図 5 実際の勝率（横軸）と推定勝率（縦軸）

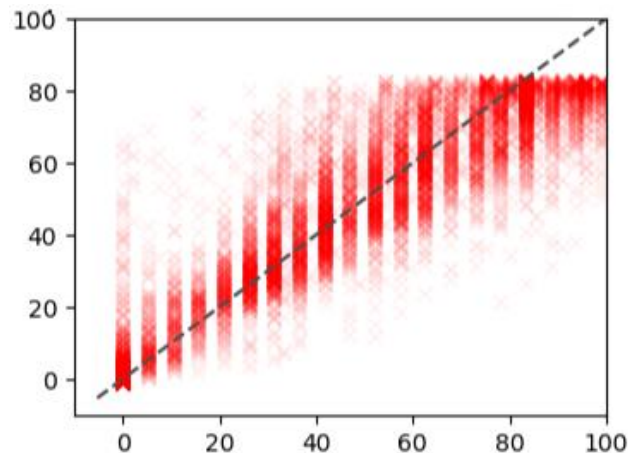


図 6 実際の適切度（横軸）と予想適切度（縦軸）

これらの結果から、近代的なネットワークを用いれば、ステージパラメータだけからでもある程度複雑な勝率などの評価値を表現・推定できると判断し、強化学習を用いる実験に進むこととした。

7. 強化学習によるステージ生成

6章の結果から、少なくともこのような単純化された環境では Q 値の表現と推定が可能であると判断し、次に強化学習を用いてステージを生成する実験を試みた。

7.1 実験条件

7.1.1 ステージ構成

6.1.1 と同様、9 マスサイズの戦闘—戦闘—戦闘—休憩—戦闘—戦闘—戦闘—休憩—ボス。ボス戦を除くと $2^6 = 64$ 戦略がありうる。休憩マスは戦闘マスと同じ場所に表現されるので、入力は 3×7 サイズのステージ行列になる。

7.1.2 強化学習要素設定

未完成ステージは、「3 マスまで完成」「4 マスまで完成」などの状況ごとに異なるネットワークを用いるのが適切かもしれないが、本稿の実験では、ステージは全て同一のネットワークで表現することにした。

3×7 行列のうちまだ値が定まっていない部分については、仮に 0 を埋めた。このことは「体力・攻撃力が最小値」とも読めるため、本来適切ではない可能性が高い。

報酬はステージを完成させる行動以外には 0、ステージ完成後には、評価関数である勝率適切度の値 0 ~ 100 を与えた。

7.1.3 初期ステージ設定

全く何もないところから、勝率が 30% に近づくようにステージを作るのはある意味簡単かもしれない。また、それではステージの多様性を持たせにくいかもしれない。そこで本論文では、最初の数個のマスは完全にランダムにパラメータを設定したうえで、残りを強化学習によって学習および生成させることにした。

ランダムに生成する初期未完成ステージのサイズは 8,5,2 とした。8 の場合は残り決めなければいけないのはボスの体力攻撃力だけであり、単純な問題である一方で自由度がないとも言える。2 の場合は逆に、殆どのステージを自分で生成しなければいけない、もしくはすることができる。

7.1.4 ネットワーク設定

ネットワークの隠れ 2 層の Convolution (32-64, filter size3) と AveragePooling (pool size3) 1 層と 4 層の fully connected 層 (128-256-256-256) で構成され、活性化関数としては ReLU, 最適化は RMSprop を用いた。Replay memory のサイズは 100000 で、学習率は 0.00025, Batch サイズは 128, gamma は 0.9, target ネットワークの更新周期は 2000, 行動の選択には最初の段階で ϵ -greedy 方策を用いた。 ϵ パラメータ値は 1 から始めて 0.1 まで減少させた。これらの設定は暫定的なものである。

7.2 結果

図 7 に、サイズ 8 の初期ステージを与えた場合の学習の進捗を示す。この設定ではあまりにも初期状態のランダム性が強く、「どう頑張っても簡単すぎる・難しすぎるステージにしかない」ケースが多い。そのため、平均報酬は 50 程度までしか上がらなかった。

図 8 のサイズ 5 の初期ステージに対しては概ね平均 70 程度、図 9 のサイズ 2 の初期ステージに対しては概ね平均 75 程度まで、勝率適切度をあげることができた。ある程度のステージ生成の自由があれば、“この単純化した環境では” “勝率のみに限っていえば” 良い問題を強化学習で作成できることが示せた。

当然ながら、今回作った問題がただちに遊んで面白い問題になっているとは言えない。ボスが異常に弱くても、辿りつくのが難しく結果的に勝率 30% になっている問題や、「当然攻撃」「当然退却」などが明確な場合もあるだろう。これらを組み込んだ実験は早急に行う予定である。

図 10 には、作成したステージの例を示す。まず、上部は完全にランダムで生成した 9 マス (7 戦闘) ステージである。これはどうやってもクリアできるようなステージであり、評価関数値 (勝率適切度) は 0 であった。

ここから、最初の 5 マス (4 戦闘) だけを切りだし、残りの 4 マス (3 戦闘) を学習した Q ネットワークで生成させたのが、図 10 下部である。比べてみると、5 匹目 6 匹目の敵のパラメータが増強されており、難しいステージにできたのだと想像できる。

ただし、図 10 のステージについて、64 通りの戦略を個別に見てみると、実際には「全て逃げる」という戦略でもボスを倒すことが出来てしまうことに気付く。ボスの攻撃力が弱すぎるためである。これは、我々が設計した勝率適切度の意味では 91 点と高得点であり「学習がうまくいった」と言えるが、実際に

は適切度の設計をもう少し複雑にしなければいけないという例にもなっている。

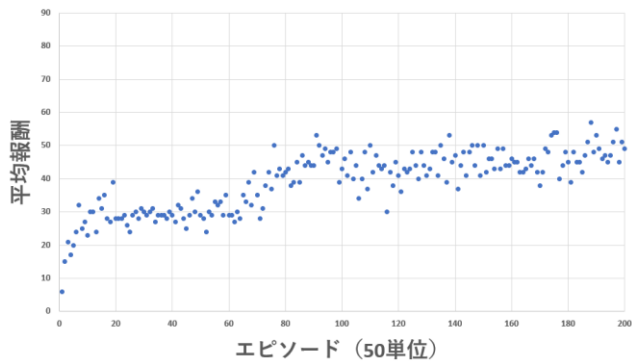


図7 サイズ8の初期ステージにおいて、10000 エピソードの50 エピソード平均報酬

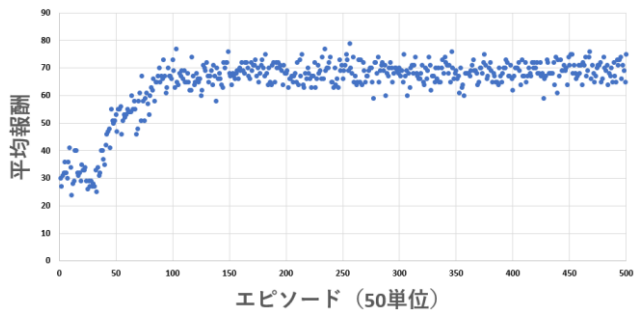


図8 サイズ5の初期ステージにおいて、25000 エピソードの50 エピソード平均報酬

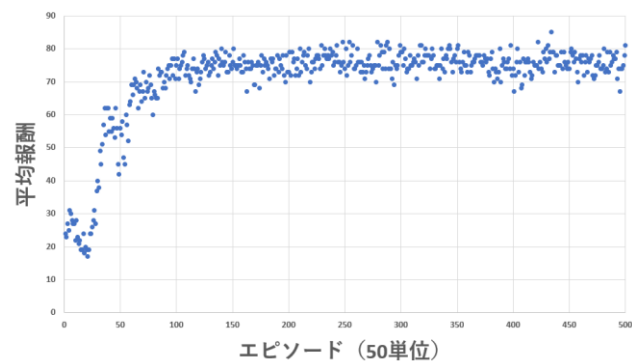


図9 サイズ2の初期ステージにおいて、25000 エピソードの50 エピソード平均報酬

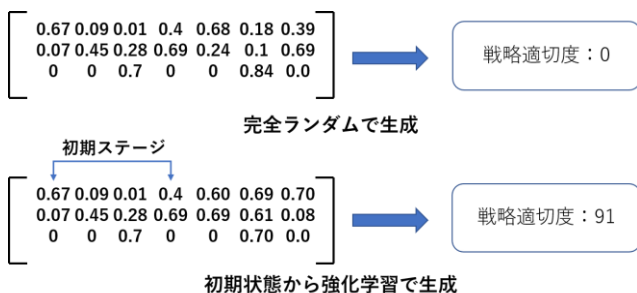


図10 ランダムに生成したステージに対して、その途中からを強化学習で生成した例

8. まとめと今後の課題

本論文では、ターン制 RPG ゲームを対象に、強化学習を用いてステージを生成する手法を提案した。非常に単純化したパラメータや戦略を用いる場合、十分に勝率適切度が表現・推測できることを確認した。ただし、作成されたマップは、勝率についてのみ適切であり他の部分が考慮されていないという予想通りの問題点もある。

今後は、人間らしいテストプレイヤを用いて評価を可能にすること、被験者実験を通じてどのような部分が面白さ推定の評価関数に必要なのか知ること、より複雑なルール設定やステージ設定を用いること、などを順次進めていきたい。

参考文献

- [1] A. Summerville, M. Mateas. "Super Mario as a string: Platformer level generation via lstms." In DiGRA and FDG 2016
- [2] D. Loiacono, et al. "Automatic track generation for high-end racing games using evolutionary computation." In TCIAIG, 3 (3):245-259, 2011.
- [3] T. Mahlmann, J. Togelius, and G. N. Yannakakis, "Spicing up map generation." in EvoApplications, vol. 7248. Springer, 224-233, 2012.
- [4] 高橋竜太郎, 池田心, 連鎖構成力向上のためのぶよぶよの問題作成, 情報処理学会 第39回ゲーム情報学(GI)研究発表会, 2018-3, 東京大学
- [5] I. J. Goodfellow, J. Pouget-Abadie, et al. "Generative adversarial nets." In Proceedings of NIPS, 252-259, 2014
- [6] M. Stephenson, J. Renz, "Procedural generation of complex stable structures for angry birds levels." In CIG 2016.
- [7] Noor Shaker, Julian Togelius, et al. "Fractals, noise and agents with applications to landscapes." In Procedural Content Generation in Games. 57—72, 2016
- [8] V. Volz, J. Schrum, et al. "Evolving mario levels in the latent space of a deep convolutional generative adversarial network." In Proc. GECCO, 2018
- [9] 上田陽平, 池田心, 遺伝的アルゴリズムによる人間のレベルに適應する多様なオセロ AI の生成, 第27回ゲーム情報学研究會(2012-03)
- [10] Mnih Volodymyr, et al. "Human level control through deep reinforcement learning." Nature, 518(7540):529–533, 2015.