修士論文

# ローグライクゲームにおける長期的戦略の学習

1610110　高橋 一幸

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
白井 清昭
長谷川 忍

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)

平成 31 年 2 月

**Abstract**

In recent years, the performance of computer game player is improved significantly due to the performance of hardware and the advancement of computation technique. DeepBlue(IBM) computer chess won against Kasparov who was world champion in 1997. AlphaGo of DeepMind won against the champion of the game of Go Ke Jie in 2017. DeepMind also presents Deep Q-Network(DQN) computer player of the arcade game Atari(which contain various type of games) in 2016 which is able to clear 49 games. And 29 games among that, the player is able to break a human professional record. Thus, it is shown that the performance of AI in term of strength already surpass human professional level not only in classic board game but also the Arcade game.

Almost all games can be the target of research, but in some genres of games, it is often difficult to develop/reproduce the game environment for research, or there often are too much/complex rules in a game. So, for academic research, "open source academic platform" have been frequently developed and employed. Such as MetaStone, FightingICE, or Mario AI Benchmark which is very similar to original games, TORCS, digital curling, or aiwolf which simplify original game rules.

"Roguelike" is one of the popular game genres which also known in Japan as "Mystery dungeon series". In the "Roguelike games", the player needs to concern long term resource management, short-term fighting strategy, the dilemma of exploring/harvest, and dense/sparse behavior selection. Each task is very challenging and necessary for AI development. Such task is also appearing in the other complex game such as StarCraft, however the more complex of a game is, the more difficult it is to analyze and develop. Thus, the development of roguelike research platform with moderate complexity is beneficial for research subjects.

In this research, we developed an academic research platform and proposed/created game AI for the rogue-like game. In order to develop a platform for research, we enumerated and organized the element features and extracted the basic essential elements in common for many titles. The rules of the game are formulated based on "Berlin Interpretation" which is a published definition of a rogue-like game.

By using the platform, we firstly tried rule-based player with the decision tree of if-then as the baseline of the other method. This decision tree represented the thinking while playing out in priority order, which is not a smart AI. Thus, the winning (game clear) percentage was about only 70.8%. We confirmed the behavior and found that the player was unable to escape to the aisle when surrounded by multiple enemies.

In order to improve the tactical aspect of the rule base player, the Monte Carlo method (depth limited Monte Carlo method, DLMC) is introduced to improve short-term decision-making. In DLMC simulation, since it is terminated at a fixed depth, very small calculation cost for the simulation operation can be expected. However, unlike the primitive Monte Carlo method, a state evaluation function of the leaf node is needed. Here, we prepared and adjusted the value function manually. As a result, a winning percentage of 82.3% was obtained. Improvements such as "escape into the aisle and make a one-on-one situation" can be confirmed in AI behavior, however, wasteful use of items at unnecessary scenes was also seen.

Finally, we attempted to improve the state evaluation function of DLMC, which had been created and adjusted manually, by one of the supervised learning methods i.e. averaged perceptron. There are options other than data number, learning parameters, etc. for of training data preparation, or training process. For example, when "data collection is performed", "how to express the state", "How to handle rare data that is difficult to collect in normal play" etc. It is difficult to cover overall. Therefore, in this research, we narrowed down and compared just some of them. In this research, the collecting data was done at "only when descending stairs" And "every battle is over". We found no significant difference between them. Thus subsequent data collection was taken at the end of a battle, due to the natural nature of the timing and a large number of data that can be collected. Next, we also compared "feature representation quantities between linear representation and one hot table" and "incensement of rare data". As a result, we obtained the same performance as DLMC using state evaluation function created by hand.

## 概要

近年，ハードウェアの高性能化やそれに伴う手法の高性能化により，コンピュータゲームプレイヤ（ゲーム AI）は目覚ましい発展を遂げている．1997 年にチェスにおいて IBM DeepBlue が世界チャンピオン Kasparov に勝利したことをはじめ，2017 年には囲碁でも DeepMind の AlphaGo が世界チャンピオン柯潔に勝利している．このように，ボードゲームにおいてゲーム AI は人間と同等以上の強さを示した．このゲーム AI の研究はボードゲームに限らず，アクションゲームや FPS などをはじめとするビデオゲームもその対象としている．2016 年に DeepMind 社の Deep Q-Network(DQN) が，49 種類のゲーム中 29 種類のゲームにおいて人間と同等以上の成績を収め，2018 年には同社の For The Win(FTW) が FPS において人間以上の成績を収めている．これらをはじめとするビデオゲームには，不完全情報性，多人数性，リアルタイム性など，ボードゲームとは異なる困難さを持つものも多く，それぞれ新しい解決手段が必要なため，これからも研究の重要性は高い．

多様なジャンルのゲームが研究対象とされている一方で，そのジャンルによっては “ゲーム環境の構築が面倒”，“ルールが煩雑” などの問題が存在する．そのため，学術研究では AI 部分のみを開発できるようにした統一のプラットフォームが用いられることが多い．その例として，実ゲームに近いものとしては MetaStone, FightingICE, MarioAI BenchMark など，ルールを単純化したものとしては TORCS，デジタルカーリング，人狼などが挙げられる．

我々は多様なゲームジャンルの中でも，人気の高い “ローグライク (Rogue like) ゲーム” に着目する．これはある種の一人用ダンジョン探索ゲームの総称であり，国内では “不思議のダンジョン” の名前で知られ，1993 年の第一作発売以降，いくつかのタイトルは 100 万本以上を売り上げている．非常に知的で面白いゲームとして知られる一方で，一人用ゲームであるために囲碁などに比べるとゲーム AI の実益性は低く，学術研究も十分に行われていない．しかし我々は，このゲームには「長期的なスケジューリング」「短期的な戦術」「探索と収穫のジレンマ」「疎密のある行動選択」など AI が取り組むべき複数の課題があると考える．これらの課題は，例えば StarCraft などのより複雑なゲームにも登場するが，ゲームが複雑なほどその学習や探索，実験や分析は質的にも量的にも困難になることが多い．そのため，研究対象として中程度の複雑さのゲームも有益であると考える．

本研究では，ローグライクゲームを対象として，学術研究用プラットフォームを新規作成し，それを用いてゲーム AI の作成を行った．

学術研究用プラットフォームを作成するにあたり，ローグライクゲームが持つ要素を列挙・整理し，多くのタイトルに共通する基盤的要素を抽出した．さらに，Berlin Interpretation と呼ばれるローグライクゲームの定義を踏まえたうえで，研究基盤用のルールを策定・プラットフォームの作成を行った．

これを用いて，まず初めに，他手法のベースラインとして if-then の決定木からなるルールベースプレイヤの作成を行った．この決定木は自身がこのゲームをプレイする際に考えていることやその優先順位を書き起こしたもので，必ずしも賢いとは言い難く，ゲームにおける勝率は 70.8%程度であった．挙動を確認してみると複数の敵に囲まれた際に通路に逃げ込むことができないなど，戦術面での弱さが目立った．

そこで次に，ルールベースプレイヤの戦術面を改善すべく，戦闘場面において探索の深さを限定したモンテカルロ法（深さ限定モンテカルロ法，DLMC）を用いることで，短期的な意思決定の改善を図った．DLMC ではシミュレーションを一定の深さで打ち切るため，シミュレーションにかかる計算コストが小さく高速な動作が期待できる．しかし，原始モンテカルロ法と異なり終局までゲームを行わないため，リーフノードの状態を評価する関数が必要となる．ここではその評価関数を手作業により作成および調節を行った．その結果，82.3%という勝率が得られた．挙動を確認すると,「通路に逃げ込み，一対一の状況を作る」ような改善があった一方で，必要のない場面でのアイテムの無駄遣いなども見られた．

最後に，手作業により作成･調節していた DLMC の状態評価関数を，平均化パーセプトロンを用いた教師あり学習により求めることを試みた．学習データの作成及びその学習には，データ数や学習パラメータなどの他にもオプションが存在し，それは例えば,「データの収集はどのタイミングで行うか」「どのように状態の表現をするか」「通常のプレイでは収集しづらいレアな状態のデータを与えるか」など様々あり，すべてを網羅することは難しい．そのため，本研究ではその中のいくつかに絞って比較を行った．まず，データの収集タイミングを “階段降下時のみ” と “戦闘終了ごと” の 2 つで比較を行ったところ，有意な差はみられなかった．そのため，タイミングの自然さ・収集できるデータ数の多さから，以降のデータ収集は戦闘終了ごととした．続いて，“線形表現を用いてきた特徴量をワンホット表現にする試み” と，“レアデータの補完の試み” の 2 つを行った．その結果，複数の特徴量をワンホットに表現し，レアデータの補完を行うことで，特定の階層に限ってだが，手作業で作成した状態評価関数を用いた DLMC と同程度の性能を得ることができた．

# 目 次

# 図 目 次

# 表 目 次

# 第1章　はじめに

近年のコンピュータゲームプレイヤ（以下，ゲーム AI と呼ぶ）の発展は目覚ましく，1997 年にチェスにおいて IBM 社の DeepBlue が世界チャンピオン Kasparov に勝利 [1] したことをはじめ，2017 年には囲碁でも DeepMind 社の AlphaGo が世界チャンピオン柯潔に勝利している [2]．このように人間と同等以上の強さを示しているゲーム AI の研究はボードゲームに限らず，アクションゲームや FPS[1] などをはじめとするビデオゲームもその対象としている．2016 年に DeepMind 社の Deep Q-network(DQN) が，49 種類のゲーム中 29 種類のゲームにおいて人間と同等以上の成績を収めた [3, 4]．また，2018 年には同社の For The Win(FTW) が FPS において人間以上の成績を収めている [5]．これらをはじめとするビデオゲームには，不完全情報性，多人数性，リアルタイム性など，将棋や囲碁とは違った困難さを持つものも多く，それぞれ新しい解決手段が必要なため，これからも研究の重要性は高い．

多様なジャンルのゲームが研究対象とされている一方で，そのジャンルによっては “ゲーム環境の構築が面倒”，“ルールが煩雑” などの問題が存在する．そのため，学術研究では AI 部分のみを開発できるようにした統一のプラットフォームが用いられることが多い．実ゲームに近いものとしては StarCraft[6], MetaStone[7], FightingICE[8]，MarioAI BenchMark[9] など，ルールを単純化したものとしては TORCS[10]，デジタルカーリング [11]，人狼 [12]，我々の研究室で提供している崩珠 [13]，TUBSTAP[14] なども挙げられる．

我々は多様なゲームジャンルの中でも，人気の高い “ローグライク (Rogue like) ゲーム” に着目する．これはある種の一人用ダンジョン探索ゲームの総称であり，国内では “不思議のダンジョン” の名前で知られ，1993 年の第一作発売以降，いくつかのタイトルは 100 万本以上を売り上げている．非常に知的で面白いゲームとして知られる一方で，一人用ゲームであるために囲碁などに比べるとゲーム AI の実益性は低く，学術研究も十分に行われていない．しかし我々は，このゲームには「長期的なスケジューリング」「短期的な戦術」「探索と収穫のジレンマ」「疎密のある行動選択」など AI が取り組むべき複数の課題があると考える．これらの課題は，例えば StarCraft などのより複雑なゲームにも登場するが，ゲームが複雑なほどその学習や探索，実験や分析は質的にも量的にも困難になることが多い．そのため，研究対象として中程度の複雑さのゲームも有益であると考える．

[1] First Person Shooter

そこで本研究ではローグライクゲームを対象として,（1）**学術研究用プラットフォームを新規作成**し，それを用いて（2）**高性能なゲーム AI の作成**を目指す．具体的には,（1）これらが持つ要素を列挙・整理し，多くのタイトルに共通する基盤的要素を抽出する．この基盤的要素を基に，新しいプラットフォームを提案し，そのルールを提示する．その上で，他手法の基盤となる（2-1）ルールベースプレイヤを作成，それを基に疎な意思決定を行う（2-2）モンテカルロプレイヤの作成を行う．さらに,（2-2）において手作業で調節していた状態評価関数を（2-3）機械学習により求めることを試みる．

本章に続き，第 2 章では本研究で対象とするローグライクゲームを構成する要素やゲームルールの概要を紹介した上で，ローグライクゲームを対象とした学術的論文や，様々なゲームジャンルにおける既存の研究用プラットフォームについて紹介する．第 3 章では（1）研究用プラットフォームを構築するにあたって，基盤となるルールやその構成について述べる．第 4 章では，第 5-7 章にて紹介するゲーム AI のアプローチの概要ついて述べる．第 5 章では,（2-1）ルールベースプレイヤ，第 6 章では（2-2）モンテカルロプレイヤ，第 7 章では（2-3）モンテカルロプレイヤにおける評価関数の機械学習について述べる．最後に，第 8 章で本研究を総括し，今後の展望や課題を述べる．

なお，本論文の一部は投稿済みの論文 [15] を加筆・再構成したものである．

# 第2章　対象ゲームと関連研究

本章では，本研究で対象とするローグライクゲームを構成する要素やゲームルールの概要を紹介した上で，ローグライクゲームを対象とした学術的論文や，様々なゲームジャンルにおける既存の研究用プラットフォームについて紹介する．

## 2.1　ローグライクゲームとは

ローグライクゲームとは，1980 年に Michael らによって発表されたコンピュータ RPG[1] である “Rogue”（図 2.1）のような性質を持つゲームの総称である．概ね，1 人の冒険者がダンジョンを探検し，アイテムを集めそれを駆使しながら敵を倒し，力を強め，複数階層を突破してゴールを目指す形態を取る（図 2.2 左）．マップや敵・アイテムの配置がランダムであることが特徴の一つであり，プレイヤは毎回異なる状況に合わせて作戦を立てる必要がある．

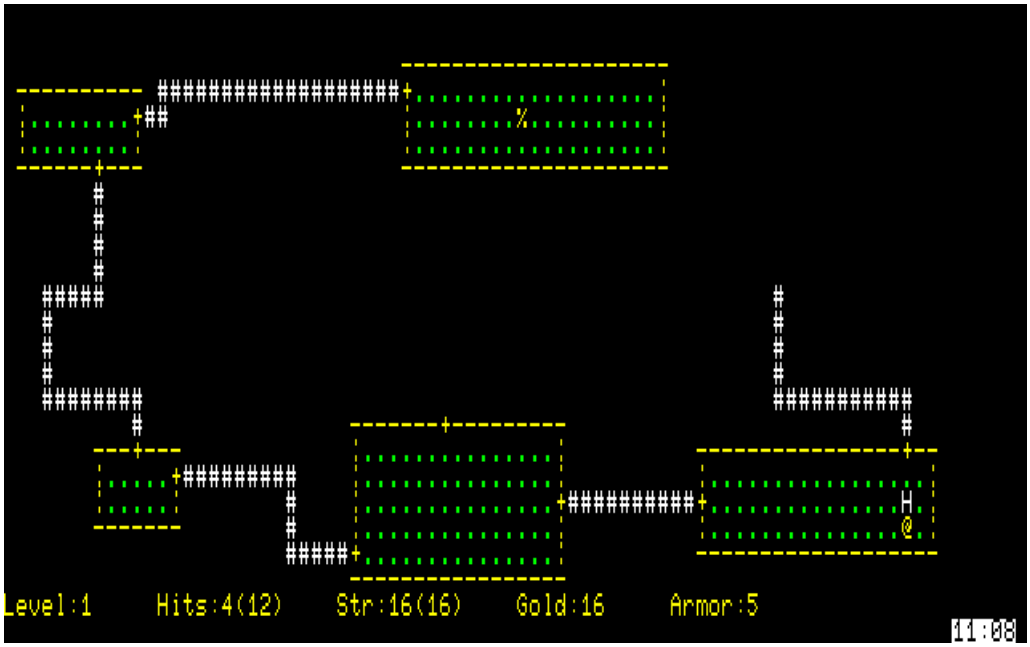



**図 2.1:** Rogue のスクリーンショット．5 つの部屋が通路 “#” で結ばれている．
プレイヤは “@” の文字で表され，敵 “H”，階段 “%” がある．

[1] Role Playing Game. ロールプレイングゲーム

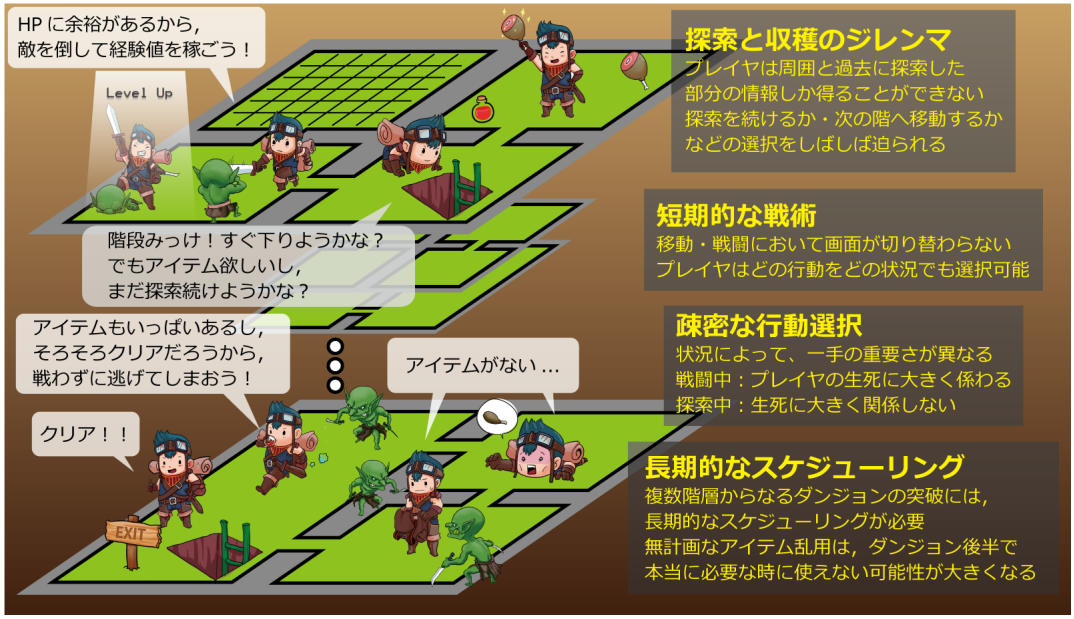



**図 2.2:** ローグライクの概略図．敵を倒し，アイテムを拾いながら複数の層を攻略し，最終層のゴールを目指す．

このローグライクゲームは，知的で面白いゲームとして長らく人々から愛され，Rogue のほか Angband, Nethack, 不思議のダンジョンシリーズなど数多くの作品が公開・発売されてきた．その数はフリーゲームを除いても数十にのぼる．当然ルールも様々であり，そもそも何をもってローグライクゲームと呼ぶのかですら明確ではない．

そこで，本研究では 3.2 節においてローグライクゲームを構成する要素の抽出を試みる．この要素の中には，

- 通路・部屋からなるフロア，そのフロアが複数層重なったダンジョンをプレイヤは探索する
- ダンジョンはマス目状で構成される
- アイテムには，自身を回復するものや敵を攻撃するものなど様々存在する

といったものが挙げられる．また，3.3 節では Berlin Interpretation と呼ばれる Roguelike を構成する要素を定義する試みについて紹介する．この Berlin Interpretation は High value factors と Low value factors から構成されており，High value factors では，

- マップやアイテム・敵の配置がランダムである
- プレイヤと敵が交互に行動するターンベースで，リアルタイムの判断・操作を要求されない
- アイテムは有限で，利用が制限される
- 移動と戦闘のシーンは別個ではなく，すべての行動はどの状況でも選択可能

といった要素が必要であるとしている．ローグライクの構成要素・定義について，詳細は 3.2 節・3.3 節にてそれぞれ後述する．

上記のような要素を持つこのゲームを研究対象としてみた時，ゲーム AI 作成には以下のような課題があると考える（図 2.2 右）．

- 探索と収穫のジレンマ
  探索を続ければ，アイテムを発見できるかもしれないし，敵を倒してレベルを上げられる可能性もある．一方で，満腹度が減少したうえでそれを回復するアイテムは発見できないかもしれないし，敵に攻撃され HP を失うことも考えられる．探索を続けるか，可能性を見限って次に進むかの選択をしばしば迫られる．
- 疎密な行動選択
  状況によって一手の重要さが異なる．戦闘中の一手はプレイヤの生死に大きく関わるためその重要さは大きく，一手一手慎重に考慮し選択する必要がある．一方で，ダンジョン探索中はその限りではなく，一手一手考慮することの必要性は低い．そのため，数手まとめたような行動を選択することができる．
- 短期的な戦術
  ゲームのどのような場面においても，攻撃する／移動する／アイテムを使うなど行動には様々な選択肢があり，それはダンジョン探索中や目の前に敵がいるときでも変わらない．敵に遭遇したからといって戦闘用画面に移行することはないため，プレイヤは様々な選択肢の中から，状況に合わせた行動を選択しなければならない．
- 長期的なスケジューリング
  アイテムを使わないで死ぬぎりぎり手前までダメージを食らいつつ敵を倒すより，アイテムを使いダメージを受けずに敵を倒すことは良い選択の様に見える．しかしそのアイテムを使ったことにより本当に必要な時に窮してしまうなど，長期的に見ると悪い選択であった可能性もある．ゲーム後半を見据えたうえでの行動選択が必要である．

これらの課題は他のゲーム，例えば StarCraft や Civilization などにもみられるが，リアルタイム性や，ゲーム自体が複雑であることなどから，学術研究の対象としてみるとその難易度は高い．その点，ローグライクゲームは将棋・囲碁などよりは複雑で，StarCraft や Civilization などよりは単純であり，上記のような興味深い課題を持つことから，研究対象として有益であると考える．

## 2.2　ローグライクゲームに関する研究

近年，様々なジャンルのゲームが学術研究の対象とされる中で，ローグライクゲームも，その「不完全情報性」「報酬の遅延」「ゲーム自体の適度な複雑さ」などの観点から，着目されつつある．

田中らはローグライクゲームに着目し，AI コンペティション開催のためのローグライクベンチマークを提案している [16]．このベンチマークは “10 年以上経過した現在でも盛んにプレイされている” “内部情報の解析が行われており，設計が容易” などの観点から “風来のシレン　女剣士アスカ見参” をベースとし，作成された．サンプルコントローラアルゴリズムとして，if-then ルールで作成したルールベースプレイヤを作成している．店，モンスターハウスが未実装であるものの，装備や壺・巻物といった多様なアイテムを実装しているため，ゲーム AI 作成の難易度は低くないと考える．

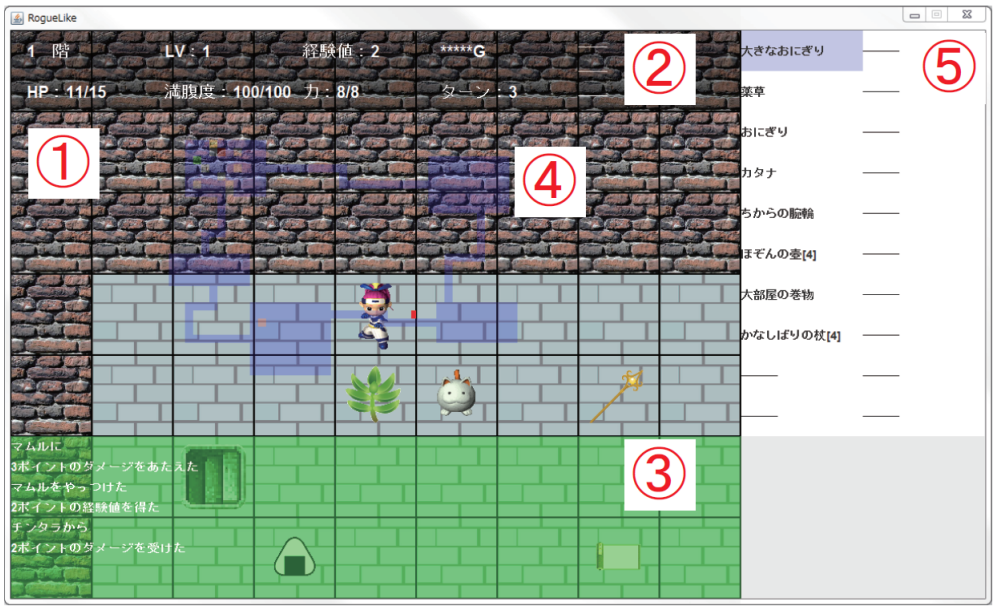



**図 2.3:** 田中らのローグライクベンチマークのスクリーンショット．①ゲーム画面，②プレイヤステータス情報，③ダメージ等のメッセージ情報，④透過表示された全体マップ，⑤所持アイテム．

Vojtech らはローグライクゲームの一つである Desktop Dungeons を題材とし，ゲーム AI および Procedural Content Generators（PCGs）作成のためのプラットフォーム（Desktop Dungeons への API と，AI および PCG 開発のための Java フレームワーク）を提案，進化計算アルゴリズムにより調整した貪欲法を用いてゲーム AI の作成を行った [17]．ゲーム AI は約 75%でゲームをクリアでき，平均的な人間プレイヤと同等の性能を示した．Desktop Dungeons は 3.3 節にて後述するローグライクの定義 Berlin Interpretation の High Value Factors の要素を踏襲している

が，オリジナルの Rogue や不思議のダンジョンシリーズと比較すると，「複数階層性はなく一階層のみ」「敵は移動しない」という点が我々には不満である．

複雑な状況をとるようなゲーム環境を扱う強化学習は，学習が遅く安定しない傾向にある．加納らはこのような特徴を持つゲームの一つとしてローグライクゲームに着目した [18]．強化学習を用いたゲームを自動攻略できる AI の作成を目的とし，簡単なローグライクゲームの環境を用意，強化学習のアルゴリズムである Asynchronous Advantage Actor-Critic(A3C) に，内部報酬を生成する Intrinsic Curiosity Module(ICM) を組み合わせた学習を行い，ICM が学習に与える効果を検証した．実験環境は「毎回異なるダンジョンを探索する」「ダンジョンが進むほどその形状が明らかになっていく」という 2 つの要素を意図してゲームが単純化されていたが，敵やアイテム，体力や複数階層性などの拡張の余地があると考える．

金川らは部分観測性や報酬の遅延（一般的に，強化学習では報酬とそれを誘発した行動の隔たりが大きいほど，学習が困難になるとされている）を持つゲームとして Rogue に着目し，Rogue を基に様々な規模の実験に対応できるよう設計した実験環境 Rogue Gym を提案した [19]．Rogue Gym では，ダンジョン中で取得したゴールドが報酬となり，下の階に降りた際に疑似報酬が与えられる．また実際にこれを使用し，深層強化学習の標準的なアルゴリズムの 1 つである Double DQN を用いて実験を行っている．結果，階を降りる際に疑似報酬を与えていたために，ダンジョンを探索せずに階を降りるように学習されてしまったものの，ベンチマーク環境としての有用性を示した．この段階において，Rogue Gym には加納らと同様に敵やアイテム等の要素が存在せず，拡張の余地があると考える．

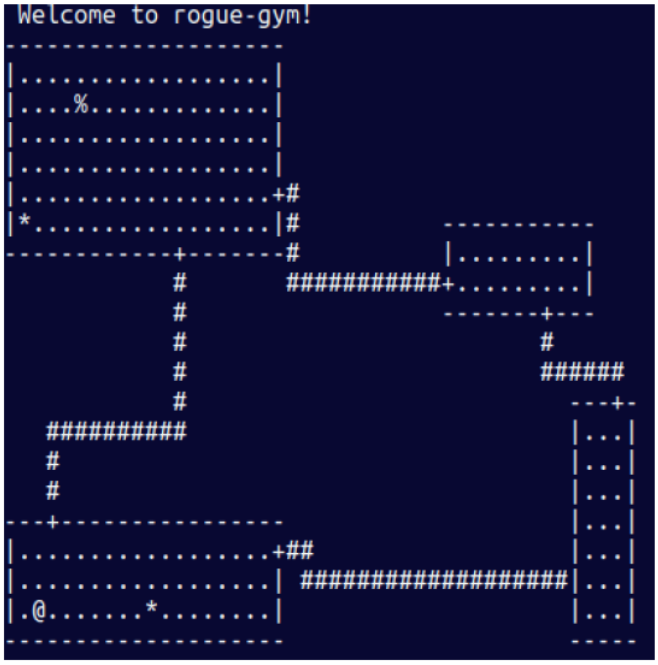



**図 2.4:** Rogue Gym のスクリーンショット．“@” がプレイヤ，“.” が床，“#” が通路，“+” がドア，“-|” が壁，“%” が階段，“*” が金貨．

## 2.3 多様なゲームジャンルにおける研究用プラットフォーム

ゲーム AI 研究および開発を行う際に，対象とするゲームジャンルの研究用プラットフォームの有無は懸念すべき点である．研究用プラットフォームが存在することのメリットとして (1) 実装コストの削減 (2) 実験再現性の向上（比較が容易），の 2 つがあると考える．プラットフォームを作成するためには，まずそのゲームの本質や基盤となる部分がどこにあるのか分析する必要があり，そのうえでゲーム AI を開発するためのゲーム環境を実装する必要がある．そのため，研究用プラットフォームの存在は大きく，メリット (1) 実装コストの削減へとつながる．またゲーム AI を開発した際には，従来の手法の再現やその性能比較が必要になることもあるだろう．そのためにも共通の研究用プラットフォームの存在は重要であり，再現や性能比較は “ゲーム AI のアルゴリズム部分を変更するのみで容易” ということから，メリット (2) につながる．

現在，研究用プラットフォームは様々なゲームジャンルにおいて存在し，ゲーム AI 開発やその AI コンペティションなどで活用されている．本節ではそれら研究用プラットフォームの一例を紹介する．

【2D 横スクロールアクション】
任天堂のスーパーマリオを基に開発された，MarioAI Benchmark という Java 環境のプラットフォームがある．これを用いて，ゲーム AI の国際会議 IEEE-CIG[20] では過去に Mario AI Competition が開催されていた．コンペティションにはいくつかのトラックがあり，作成したゲーム AI のステージクリアまでの成績を競う．ゲームクリアを目的としたトラックや，チューリングテストを行うもの等がある．

【2D 格闘】
格闘ゲームにおいても，コンペティションが開催されている．FightingICE がその一例として挙げられる．これは，格闘ゲームの AI キャラクタを開発するためのプラットフォームである．Java 言語による開発環境が用意されている．

【TBS（Turn-Based Strategy）】
ターン制ストラテジーゲームのプラットフォームとして，JAIST 池田研から TUB-STAP が公開されている．多くの既存戦略ゲームから，ゲームの持つ要素を分析したうえでプラットフォームのルールを提案・構築している．

【RTS（Real-Time Strategy）】
StarCraft は，ブリザード･エンターテインメント社が 1988 年に発売した RTS ゲームである．IEEE-CIG で頻繁に，ゲーム AI の競技会が行われており，思考ルーチンの開発のための BWAPI（The Broad War Application Programming interface）[21]

が公開されている．これは，C++言語で作られたプログラムである．また後続作品である StarCraft II[22] においても AI 開発環境として，DeepMind 社と Blizzard 社から SC2LE（StarCraft II Learning Environment）[23] という機械学習 API やデータセット等が提供されている．

【FPS（First Person Shooting）】
FPS では，チューリングテストの派生形の一つである BotPrize というコンペティションが有名である．チューリングテストを行うことでゲーム AI の人間らしさを評価するための大会が頻繁に開催されている．強さに焦点を置いた研究ではないが，2012 年の The2KBotPrize[24] において，大会史上初で人間よりも人間らしいと評価されるゲーム AI の開発に成功した．

【カーレーシング】
TORCS（The Open Racing Car Simulator）はマルチプラットフォームドライビングシミュレータである [10]．Linux，Windows をはじめとする様々な環境に対応している．Google の DeepMind 社は，3D グラフィックスゲームの 1 つとしてこれを選択し，強化学習によるゲーム AI の作成を行った．

【落ちものパズル】
研究用の代表的なプラットフォームに，JAIST 池田研究室から Web 上に崩珠というプラットフォームが公開されている [13]．このプログラムはぷよぷよのルールをターン制にしたもので，これを土台に落下型パズルゲームの学術研究が行われている．

【カードゲーム】
ハースストーンを題材とした研究用プラットフォームとして MetaStone がある．Java 言語で記述されており，実際のゲームと同様の仕組み・ルールにて構築されている．ゲーム AI の作成に限らず，デッキ作成も研究の対象となっている．

これらのほとんどは，何かしらの有名なゲームをそのまま，あるいは簡略化したうえで，AI 部分の改良を容易にしたり，実験の高速化や複数回試行などを補助する機能が付いたものであり，学術研究の推進に役立っている．

# 第3章　研究用プラットフォームの提案

本章ではプラットフォームを構築するにあたり，まずその必要性について述べる．その後，ローグライクゲームを構成する要素を列挙・解説し，それらと Berlin Interpretation と呼ばれるローグライクの定義を加味し，研究基盤用のルールの提案を行う．また，それを用いたプラットフォームの設計・構築について述べる．

## 3.1　プラットフォームの必要性

ローグライクゲームを直接扱った研究としては，2.2 節にて述べたように田中らによる提案をはじめ，Vojtech ら，加納ら，金川らによるものがある．田中らの提案は現在では追試・実装が困難であり，また Vojtech らは Desktop Dungeon というゲームを対象としているが，これには複数階層性がなく，長期的なスケジューリングが必要ないという不満がある．加納らは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ゲーム AI の作成を目指し実験環境を用意しているものの，敵やアイテムをはじめとするいくつかの要素が存在しない．同様に，金川らはローグライクゲームのベンチマーク環境 Rogue Gym を提案しているが，こちらも現段階では敵やアイテムが存在しない．以上から，新規プラットフォームを作成公開することには一定の価値があると考える．

またそもそもローグライクゲームに絞って AI を研究することに価値があるのかという議論もあり得るだろう．近年は IEEE-CIG などの国際会議で General Game Playing (GGP) および General Video Game Playing (GVGP) といった汎用ゲーム AI の競技会が行われ，さまざまなモンテカルロ探索の拡張 [25] や Deep Q-Network[4] などによりその性能も向上しているためである．しかし，いずれはそういった問題群の 1 つになるにせよ，課題を明確にしてそれを解決していくことには価値があると考える．

## 3.2　ローグライクゲームの構成要素

本節ではローグライクゲームに登場する要素・概念・システムの一部を列挙する．これらの要素すべてがローグライクゲームに採用されているわけではなく，作品毎に様々である．以下に示す構成要素の F1-9 はゲームのステージやゲームの進行に関する部分，F10-20 はステージに配置されるオブジェクトやイベントに関する部分であり，順に説明する．

【構成要素】

F1 プレイ人数

基本は 1 人プレイである．タイトルの中には，人間プレイヤ 2 人での協力プレイや，仲間 AI（1 人〜複数）と協力プレイできるものも存在する．しかしこの仲間 AI の場合，しばしば人間プレイヤの期待と異なる行動を選択し，人間プレイヤのストレスとなっていることもある．

F2 クリア条件

“ダンジョン” と呼ばれるゲームステージの踏破，最下層のボスを倒す，など様々．

F3 ダンジョン

複数の部屋と複数の通路からなるフロアがあり，そのフロアが複数層重なったものをダンジョンと呼ぶ．フロア同士は階段でつながっており，プレイヤはそこから次の階層へと移動する（図 3.1）．この階段は，一方通行であったり行き来が可能であったりする．

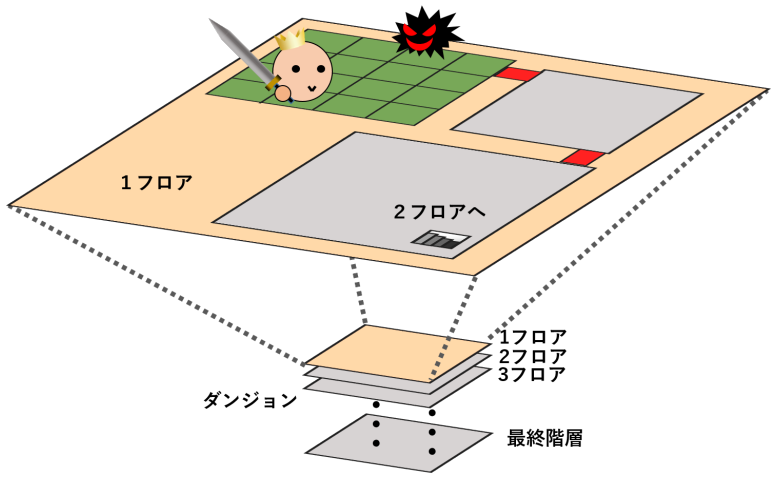



**図 3.1:** ダンジョンの構造例．
複数の層（フロア）が重なった形状をとる．

F4 地形

基本的には，すべてのキャラクタが移動できる “床”，移動することができな

い “壁” により構成されている．“水路” や “谷” など，通常は移動できないが，特別な条件（装備やアイテムなど）で移動することが可能になる特殊な地形が存在したり，壁を破壊して進むことができたりする場合もある．

F5 ランダムな環境生成
ダンジョンや敵・アイテムの配置はランダムで，ゲームをするたびに変更される．

F6 マス目状
フロアはマス目状に表現され，基本的に 1 マスに 1 つのオブジェクト（キャラクタ，アイテムなど）を格納できる（図**??**）．

F7 探索
ゲームの開始直後は，プレイヤから観測できる部分は限られる（図 3.2a）．そのため，プレイヤはフロアを探索し，アイテムを拾い，敵と戦い自身を強め，階段を見つけ次の階へと移動しなければならない（図 3.2b）．

F8 ターン制
プレイヤと敵が交互に行動するターン制である．1 マス移動，1 回攻撃，アイテム使用などで 1 ターン消費する．

F9 シームレス
移動・戦闘で場面が切り替わらない．

F10 プレイヤ
操作できるキャラクタ．体力が 0 になるとゲームオーバーとなる．

F11 敵
プレイヤを狙い，攻撃してくるキャラクタ．倒すことでプレイヤは経験値を得る．

F12 NPC
プレイヤが操作することのできない，敵とは異なるキャラクタ．ダンジョンを徘徊するだけのものもいれば，仲間となりプレイヤと共に戦うこともある．

F13 ステータス
強さを示す “レベル” や，敵から攻撃を受けることで減少する “体力”，相手に与えるダメージに影響する “攻撃力” や，移動・攻撃などにより減少する “満腹度” など．

F14 特殊状態
睡眠・混乱など，敵の攻撃や罠などから受ける．数ターン行動できない，意図した方向に移動・攻撃できない，などの影響が出る．

F15 視界
キャラクタの持つ，ダンジョンを観測できる範囲．

F16 アイテム

拾う，店から買う，敵からのドロップなどで入手できる．プレイヤを回復するものや敵を攻撃するもの，お金や装備など様々．“使う”“投げる”“その場に置く” などを選択できる．

F17 アイテムの鑑定

アイテムの中には，使用するまでどの様なアイテムか識別できないものが存在する．

F18 罠

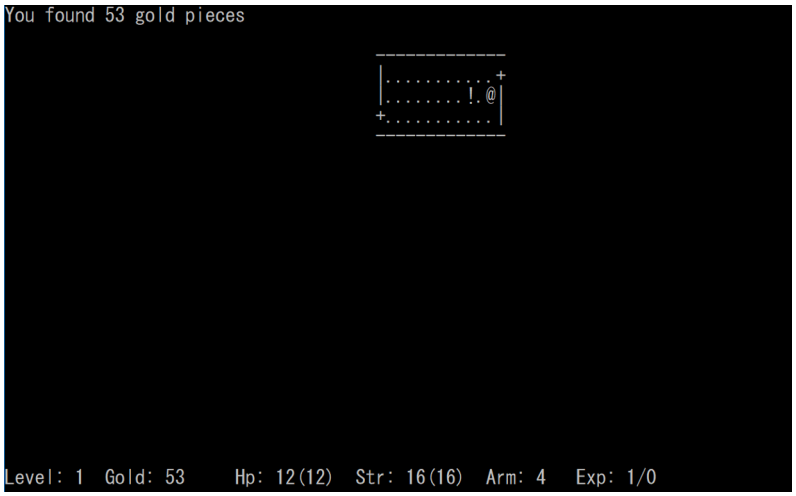
通常は設置されていることがわからないが，一度踏むと認識できるようになる．ダメージを負うもの，状態異常を食らうものなど様々．

F19 店

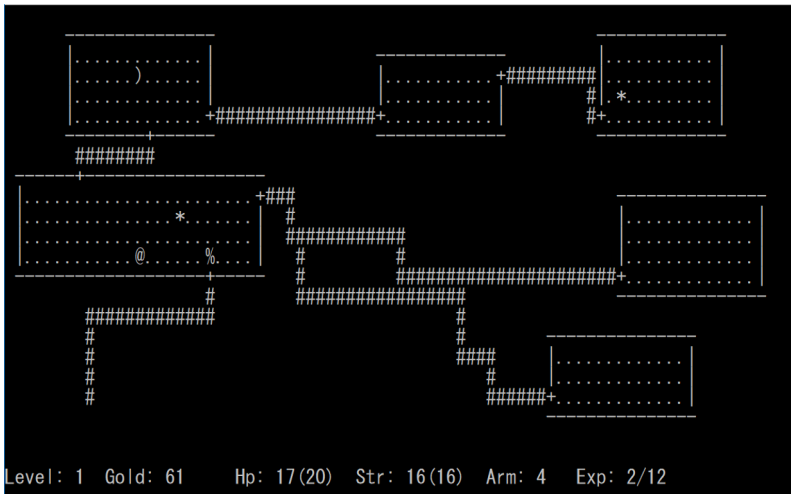
お金を支払うことで，アイテムを購入することができる．お金を支払わずにアイテムを持ち去ることもできるが，店主（NPC）が敵対状態となる・強力な NPC が敵対状態で出現するなどのペナルティがある．

F20 モンスターハウス

多くの敵・罠・アイテムが存在する部屋のことを指す．場合によっては非常に危険だが，レベル・アイテム稼ぎができるため，ハイリスク・ハイリターン．




**(a)** ダンジョン探索前．ゲーム開始直後のためフロアの大部分が観測できない．唯一観測できるのは現在プレイヤのいる一部屋のみ．




**(b)** ダンジョン探索後．探索前には観測できなかった 5 部屋があらわになり，このフロアには 6 部屋存在していることがわかった．

**図 3.2:** ダンジョン探索前後の観測できる範囲の比較．プレイヤは “@”．

## 3.3 ローグライクゲームの定義

ローグライクゲームは，3.2 節にて列挙したような様々な要素により構成されている．本研究ではその中からゲームの本質となる要素を抽出し，研究基盤用ルールを策定するために，“Berlin Interpretation”[26] と呼ばれるローグライクゲームの持つ要素の定義に着目した．本節ではこの定義を紹介する．この定義は，2008 年の International Roguelike Development Conference において，Roguelike はどのような要素を持つか定義する試みが行われ，まとめられたものである．High value factors と Low value factors の 2 部から構成されており，“この全てが必須という意味ではなく，開発者やゲームに対して制約を与えるものではない” と断った上で，以下のような要素が提唱されている．

【High value factors】

- Random environment generation.
  マップ，アイテムの配置や出現，モンスターの配置（種類は固定）が毎回ランダムに決定されること．
- Permadeath.
  ゲームキャラクタは，アイテムや経験を溜めても，死ねばそれらを失って最初からプレイすること．
- Turn-based.
  リアルタイムの判断や操作を要求せず，熟考できること．通常，自分のキャラクタが 1 つ移動または行動をすれば，その後にモンスターも 1 つ移動または行動する（註：倍速などの敵もいる）．
- Grid-based.
  マップは正方形のマス目で構成され，敵味方のキャラクタは 1 つのマス目をふさぐこと．ピクセル単位ではない．
- Non-modal.
  移動と戦闘のシーンは別個でなく，全ての行動はゲームのどの時点でも利用できること．ただしいくつかのゲームでの “店” はこれにあてはまらない．
- Complexity.
  ゲームの目標は共通でも，それを解決する方法や道筋は複数あること．敵とアイテムの相性，アイテム同士のシナジーなどがこれを生みだすこと．
- Resource management.
  食料や傷薬といったアイテムは有限で，利用が制限されること．
- Hack’n’slash.
  モンスターを倒す必要とメリットがあること．
- Exploration and discovery.
  プレイヤはマップやアイテムの位置を知らされず，毎回慎重に探索してアイテムを探し，時には未鑑定アイテムの使用法を考える必要があること．

【Low value factors】

- Single player character.
  プレイヤは1キャラクタを操作し，そのキャラクタが死を迎えるとゲームは終了となる．
- Monster are similar to players.
  プレイヤに適応されるルールは，モンスターにも同様に適応される．
- Tactical challenge.
  戦術について学ぶ必要がある．
- ASCII display.
  伝統的な表示方法は，ASCII 文字による表示である．
- Dungeons.
  部屋や廊下からなる階層など，ダンジョンがある．
- Numbers.
  キャラクタのステータス（hp など）は意図的に表示される．

上記の High Value Factors に着目し，他のジャンルの有名ゲームと比較してみる．

StarCraft（RTS）はリアルタイムで進行するが，ローグライクは Turn-based である．そのため，シミュレーションに時間を要するモンテカルロ木探索等のアプローチの適用が可能である．また，マップが Grid-based でないこともローグライクとの大きな違いである．Grid-based であるものと，そうでないものを比較すると，探索空間に大きな差が生まれるため，ゲーム自体の難易度やそのゲーム AI の作成難易度に大きくかかわってくる．

ドラゴンクエストシリーズ（RPG，以下ドラクエ）は敵と遭遇すると戦闘用の画面に移行するため，戦闘中には戦闘に集中，戦闘外では探索や進行に集中できる．一方，ローグライクは Non-modal であるため移動や戦闘において画面が切り替わることはない．戦闘をしながら探索・逃亡という選択ができる．またドラクエと比べるとランダム性が高く，Random environment generation にあるように，マップや敵配置をはじめとしたゲーム環境が毎回ランダムに生成される．そのため，毎回状況に応じた異なる戦略を要求される．Permadeath という点も異なり，ドラクエではプレイヤが死んだからと言ってアイテムや経験値のすべてを失い，ゲームが最初からになることはない．一方ローグライクでは，死ぬとそれらのすべてを失い，最初からゲームを始める必要がある．その中で，唯一積みあがっていくものはプレイヤの経験のみである．

大戦略（TBS）とは共通する項目も多い．どちらも Turn-based であり，Non-modal，Grid-based である．異なる点として，ローグライクにはゲーム中にマップの遷移がある．大戦略は1ゲームに1マップ固定であることが多い．その一方で，ローグライクはマップが毎回ランダムに生成され，複数階層からなるダンジョンの場合には1ゲーム中にマップが次々と遷移していく．他にも，「1チーム vs1 チーム の対等な戦闘」なのか「1人 vs 多数 の冒険」なのかといった違いもある．またこの大戦略をはじめとする TBS には，「複数の駒（キャラクタ）を1ターンに操作

でき，その行動順も自由」というような，ローグライクゲームの持たない特徴を持つ．

以上の様に，High Value Factors のみに着目し StarCraft やドラクエ，大戦略などの他ジャンルの有名なゲームと比較しても，ローグライクゲームは異なる特徴を持つことがわかる．

## 3.4　研究基盤用ルールの提案

本研究では 3.3 節の Berlin Interpretation の High value factors をベースに，多くのローグライクゲームに共通しているかどうか，また良い AI を作ることが「面倒」ではなく「挑戦的」になるかどうか，という基準に基づき，第一段のプラットフォームのルールを策定した．本節ではその詳細を以下に述べる．

- マップは四角マスで，50 × 30 とする．
- マップは 3〜4 部屋で構成され，1 つの階段があり，階段を進むと新しい階層となる．4 階層から構成され，最後の階層で階段に辿りつけば勝ちである．
- 部屋のサイズや通路は 5 パターンのみを用意した（図 3.3）．

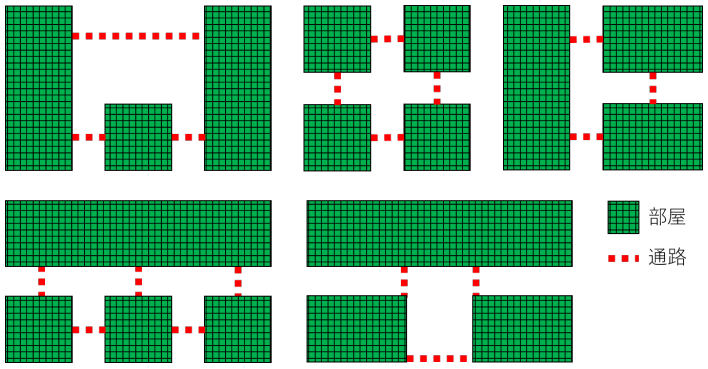



**図 3.3:** 部屋と通路の構成．5 パターンを用意．

- プレイヤは，9 × 9 視野を持つ．さらに，部屋の内部にいる場合はその内部の全ての敵やアイテムの位置が分かる．
- ターン制で，自分の移動または攻撃後，敵の移動または攻撃が処理される．
- 移動や攻撃は上下左右斜めの 8 方向 1 マスである．ただし，上または右に壁がある場合，右上には移動や攻撃ができない（図 3.4）．他の場合も同じ．

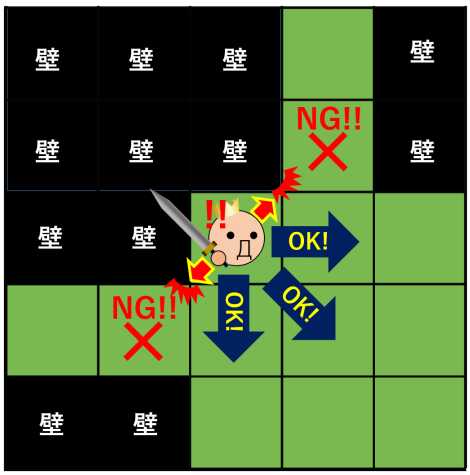



**図 3.4:** 移動・攻撃できない例．壁があることにより，右上・左下に移動・攻撃ができない

- プレイヤキャラクタにはレベル・経験値・体力・満腹度・攻撃力のパラメータがある
  - レベルは 1 を初期値とし，経験値は 0 を初期値とする．経験値は敵を倒すと入手でき，$10 \times 1.1^{(\text{レベル}-1)}$ 溜まるとレベルが 1 上昇し，経験値はその分減る．
  - 攻撃力は 35 を初期値とし，レベルごとに 1 上昇する．
  - 体力は 100 を初期値とし，レベルごとに上限が 1 上昇する．また，毎ターン，上限の 1/200 だけ回復する．
  - 満腹度は 100 を初期値および上限とし，10 ターンに 1 減り，0 になると体力が毎ターン 1 減る．
- 飢餓または敵の攻撃で体力が 0 になると負けである．
- 通常攻撃は周囲 8 マスのいずれかに対して行え，攻撃力分のダメージを与える．
- アイテムは食料，回復薬，矢，杖がある．
  - アイテムは各階層に合計 3〜5 つ，数も種類も位置もランダムに配置される．通路には配置されない．アイテム数は 50 %の確率で 4 つ配置され，3 つまたは 5 つの場合が 25 %ずつである．
  - 食料は満腹度を 20 回復する．
  - 回復薬は体力を 50 回復する．
  - 杖と矢は消耗品で，上下左右斜めの 8 方向，プレイヤから 5 マス以内に見えている敵に遠隔で利用することができる．
  - 杖は敵を別の部屋のどこかに飛ばす（一時しのぎ）ことができる．
  - 矢は敵に 15 ダメージ与えることができる．矢は 3 本まとめて拾える．
  - アイテムは 10 個まで所持できる．
  - アイテムを置いたり投げたりはできない．
- 敵キャラクタは各階層 1 種類である．
  - 敵の体力は階層ごとに 60,80,90,95，攻撃力は 20,25,30,35, 得られる経験値は 15,20,25,30 とした．
  - 敵キャラクタは階段を下りた時点で 4 匹ランダムな位置に配置され，さらに HP が 0 になってから 64 ターン後にランダムな位置に再配置される．
  - 敵はプレイヤキャラクタを見つけると最短距離で接近して攻撃し，そうでなければランダムに移動する．

上記のルールの下で，プレイヤは短期的には目前の敵を倒したり，強い敵や複数の敵にはアイテムを消費して凌ぐ，階段から次階層に逃げるなどの選択を行わなければならない．一方で長期的には将来を見据えアイテムの収集と節約，食料に余裕があれば経験値稼ぎなどを考慮する必要がある．「突発死を恐れるかジリ貧を恐れるか」「満腹度と経験値とアイテムのバランス」などはローグライクゲームの中心的な課題であり，簡単に判断することはできない．本ルールは第一段階と

して多くの要素を排除したが，それでも十分に困難で面白い課題になっていると考える．

例えば，3.2 節における以下の要素は今回敢えて含めていない．

- F11：個性豊かな敵キャラクタ．
- F12：NPC．自分と敵以外のキャラクタ．
- F14：特殊状態．睡眠・混乱など．
- F16：多様なアイテム．お金・装備など．
- F17：使うまで効果の分からない未鑑定アイテム．
- F18：罠．踏むとペナルティのある隠れタイル．
- F19：店，泥棒．
- F20：モンスターハウス．敵が大量出現する部屋．

これらは多くのゲームに登場し，面白い要素ではあるが，AI 作成をむやみに煩雑にする可能性が高いと判断し，今後その導入を検討する予定である．

【作成したプラットフォーム】

上記の提案ルールを採用し，作成したプラットフォームのスクリーンショットを図 3.5 に示す．図の左がプレイヤの視界になっており，その中の左上にはプレイヤのステータス情報等（体力，満腹度，レベル，現在の階層）を載せている．図の右上にはマップ，右下には行動のログを出力している．

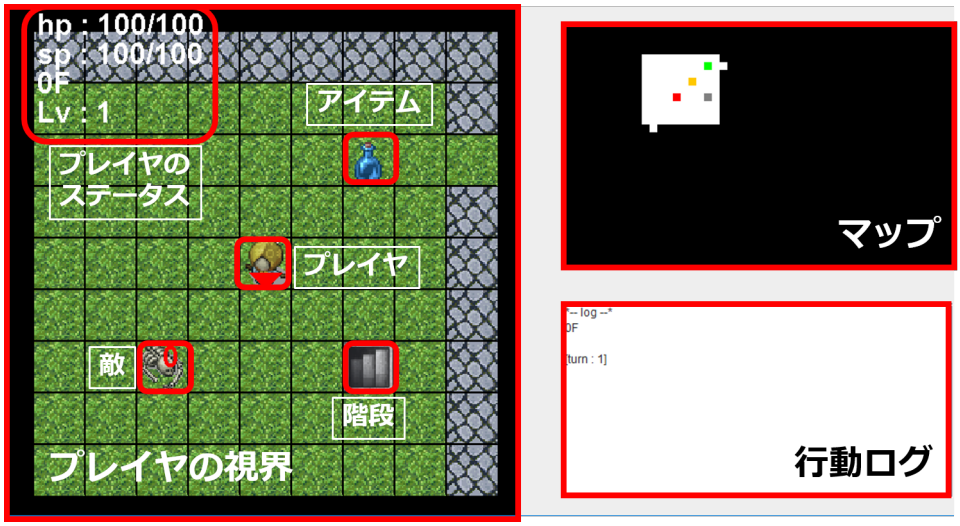



**図 3.5:** 作成したプラットフォームのスクリーンショット．左：プレイヤの視界とステータス情報，右上：マップ，右下：行動ログ．

# 第4章　アプローチ

本章では以降の第5-7章で紹介する，プラットフォームを用いて実装したゲームAIのアプローチの概要について述べる．

ゲームAIに用いられる手法は教師あり学習，強化学習，木探索，Direct Policy Searchなど様々であるが，我々はモンテカルロ法が有望であると考えている．囲碁ではモンテカルロ法登場初期から「形勢が悪い場合は勝負手を，良ければ安全な手を」といった人間的判断が得意であることが知られており，それは前述のローグライクゲームの中心的課題とも良く似ているからである．直近の死亡リスク回避と，中長期的なジリ貧の回避，またたまたま階段が近くにあってクリアできるような幸運，これらを自然にバランスした意思決定ができる可能性が高いと期待している．

一方でモンテカルロ法をこのようなゲームに用いるには，囲碁などとは違う点に注意する必要がある．一つには行動に疎密がある点である．敵に囲まれているような状況では1ターン1ターンに細心の意思決定をしなければならない一方で，単なる移動や経験値稼ぎの待機状態は，数十ターン程度をひとかたまりにした意思決定が求められる．毎ターンモンテカルロ法で行動選択をしていては，時間もかかるうえに行動の一貫性が損なわれる可能性がある．もう一つはゲーム終了までに多くのターンが必要な点である．囲碁でも数百手で終わるのに対し，これらゲームは1マップに数百ターン，それが数十階層あることも珍しくない．

これらの課題を解決する手段は単純なものから複雑なものまでさまざまに考えられる．図4.1に本研究の模式図を示す．まず第5章において他手法のベースラインとして，ルールベースで動作するゲームAIについて述べる（図4.1上段）．第6章では，通常はルールベースで動作し，敵が存在する場合のみモンテカルロ法を用いることで行動の疎密に対処する手法を紹介する（図4.1中段）．ゲーム終了までに多くのターンが必要な点については，ランダムシミュレーションをゲームの終了まで行わず，途中で打ち切り状態評価関数を用いて勝率を推定するDepth Limitedなアプローチを試みる．さらに第7章では，第6章にて手作業で調節していた状態評価関数を，教師あり学習を用いて求めた場合について比較を行う（図4.1下段）．

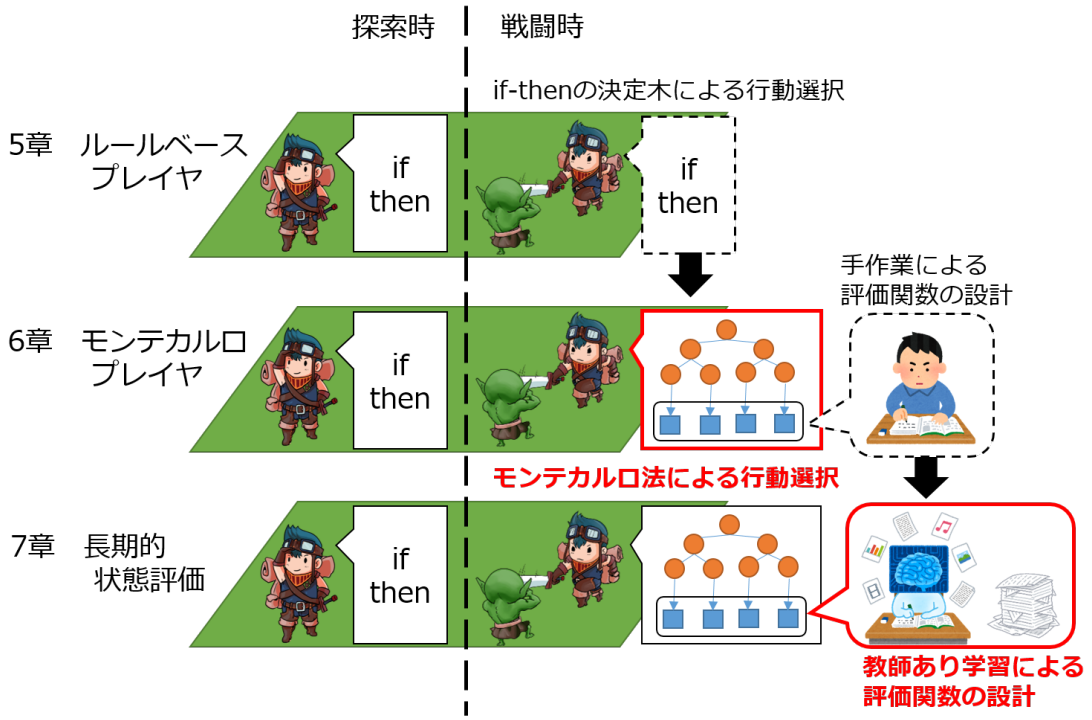



図 4.1: アプローチの模式図．5 章ルールベースプレイヤ（上段）では探索時・戦闘時どちらも if-then の決定木により行動を選択する．6 章モンテカルロプレイヤ（中段）では探索時はそのまま，戦闘時のみ深さ限定モンテカルロ法による行動選択を行う．用いる評価関数は手作業により作成する．7 章長期的状態評価（下段）では，6 章において手作業により作成していた評価関数を教師あり学習により求める．

# 第5章　ルールベースプレイヤ

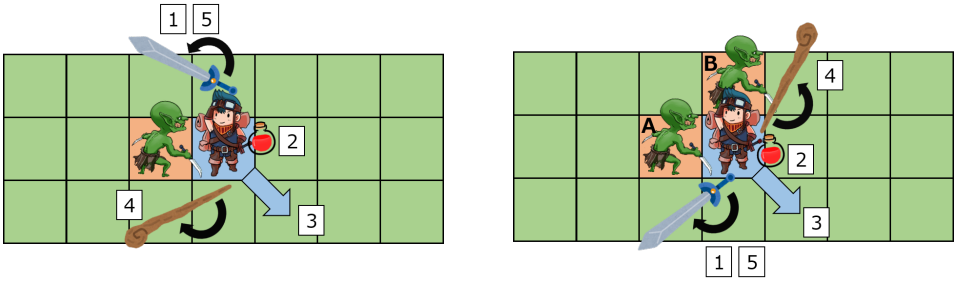
本章では他手法のベースラインとして，if-then の決定木からなるルールベースプレイヤについて述べる．

## 5.1　アルゴリズム

本節にて示す if-then の決定木は，我々がこのゲームをプレイする際に考えている内容や優先順の一部を書き起こしたものであり，賢いとは言い難い．より複雑な分岐を加えることでこれをより高性能することはできるだろうが，手間の割に性能は伸びにくいと考えるため，これをベースに汎用的な手法を構築していく．

【敵が隣接している場合】

1. 隣接する敵の攻撃力の合計を計算（敵の中で体力の最も低い 1 体がプレイヤキャラクタの攻撃力以下ならば，加算しない）し，その値がプレイヤキャラクタの体力未満ならば，体力の最も低い 1 体を攻撃する．
2. （そうでなければ：以下同じ）1 の条件に当てはまらない場合，危険な状態である．そのため，回復薬を所持しており，その回復量が敵の攻撃力の合計を上回るならば，回復薬を使う．
3. 回復量が間に合わない／回復薬が存在しない場合，左下，下，右下，左，右，左上，上，右上の順に見ていき，敵からの攻撃を一旦避けられるマスがあるならば，移動する．
4. 逃げ場がないとき，杖があるならば最も体力の多い敵に使用する．
5. 1 の条件に当てはまらなかったということは倒せないので，死ぬことになる．




**(a)** 敵が単体の時．図中の数字の小さい順に行動が優先される．

**(b)** 敵が複数の時．敵 B が敵 A より体力が多いとき，杖は敵 B に使用する．

**図 5.1:** 敵が隣接している場合の行動．

【敵が 2 マス離れたところにいる場合】

1. 最下層で敵よりも階段に近いならば，階段に向かう（到達すればゲームクリアであるため）.
2. 敵の攻撃をくらうと体力が上限の 25 %以下になり，回復薬を持っているならば，使う.
3. 矢を持っていて当たるなら，使う.
4. その場で待機する．敵はこちらに向かって移動してくるため隣接した状態となり，次のターンにはプレイヤキャラクタが先制できる.

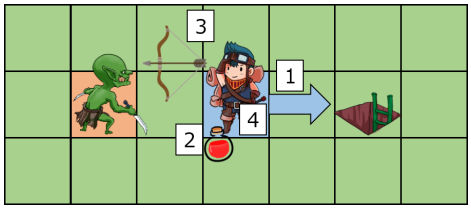



**図 5.2:** 敵が 2 マス離れている場合の行動.

【敵が 3 マス以上離れたところにいる場合】

1. 2 マスの場合の 1 と同じ.
2. 2 マスの場合の 3 と同じ.
3. 敵に近づく．敵と 3 マス離れた状態であれば，敵もこちらに向かって移動してくるため隣接した状態となり，次のターンにはプレイヤキャラクタが先制できる.

【敵が視野内にいない場合】

1. 最下層で階段が見えていれば，それに近づく.
2. アイテムが落ちていて，所持アイテム数が上限に達していないなら，それに近づく.
3. 体力が上限の 70 %未満，満腹度が 40 以上の場合，待機して体力を回復させる.
4. 満腹度が 70 未満で，食料を持っているならば食べる.
5. 階段を発見しており，満腹度が 70 未満かつ食料を持っていない状態，または全ての部屋を探索済み，またはアイテムを 4 つ以上拾った状態であるならば，階段に近づく.
6. 階段を発見していない，もしくは満腹度に余裕があり，未探索の部屋が存在し，充分にアイテムを収集できていない状態のため，探索したことのない通路を移動することで別の部屋に向かう.

以上の if-then の決定木に基づき，ゲームをプレイする.

## 5.2　実験結果と例

ルールベースプレイヤによりゲームを 100000 試行したところ，クリア率は 70.8 %であった．各階における死亡回数は表 5.1 に示すとおりである．また，ゲームを通しての餓死（満腹度が 0 になり体力が徐々に削られ，ゲームオーバーになった）回数は 185 回，1 ゲーム当たりのアイテムの使用個数は，食料：2.2 個，回復薬：1.3 個，矢：7.5 個，杖：0.4 個，となった．1 ゲーム当たりの実行時間は 1 秒程度であった．

**表 5.1:** ルールベースプレイヤの各階における死亡回数．

| 階層 | 死亡回数 |
|---|---|
| 1 階 | 4128 |
| 2 階 | 8284 |
| 3 階 | 9593 |
| 4 階 | 7243 |

挙動を見ると，広い部屋の中で複数の敵を相手にしてしまっていることが，直接の死因に繋がっていることが特徴的であった．例えば図 5.3 左の場合，プレイヤは右側にある通路に逃げ込み，一対一の状況を作ることが有効な対処法である．しかしルールベースプレイヤの場合，そのまま部屋の中で複数の敵から攻撃を受ける状況で敵を攻撃していた．これにより多くのダメージを食らい（図 5.3 右），前述した様に直接の死因へと繋がっていた．

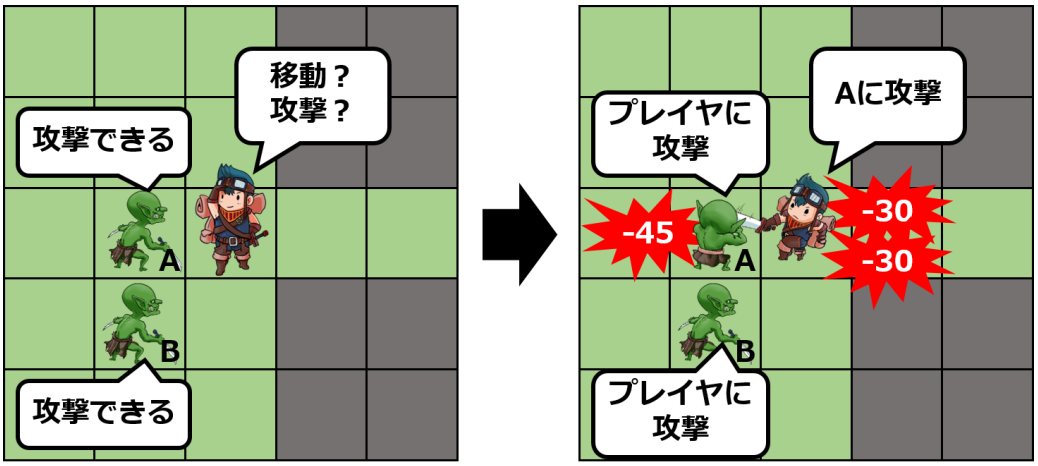



**図 5.3:** ルールベースプレイヤの行動例．部屋の中で戦闘してしまっているため，2 体から攻撃を受けてしまう．左図のような状況ならば，通路に逃げ込み一対一の状況で迎え撃つことが有効．

## 5.3 考察

5.1 節にて示した if-then の決定木では図 5.3 のような “通路に逃げた後に敵を順に処理することが良い” 状況に対処することは難しいことがわかった．他にも，if-then の決定木の構成から，杖や回復薬を使うタイミングが非常に限られているということも問題として挙げられる．ぎりぎりの状況でなくとも，リソースに余裕があるならば「沢山持っているし，ちょっとでも危なくなったら使ってしまおう」と考えることは充分にありえるだろう．もちろん新しいルールを追加すれば対処が可能かもしれないが，そのような考え方に基づく行動選択を if-then の決定木で実現することは，体力や所持アイテム，レベルなど考慮しなければならない要素が多く，実装には大きなコストがかかり，容易ではない．

そこで次章ではこの戦術面を改善すべく，戦闘場面においてモンテカルロ法による探索とシミュレーションを活用することで，短期的な意思決定の改善を図る．

# 第6章　モンテカルロプレイヤ

本章ではルールベースプレイヤの戦術面を改善すべく，戦闘において探索の深さを限定したモンテカルロ法を活用することで，短期的な意思決定の改善を図る．

## 6.1　原始モンテカルロ法と深さ限定モンテカルロ法

モンテカルロ法を用いたゲーム AI の作成に先立ち，本節では原始モンテカルロ法と深さ限定モンテカルロ法について紹介する．

モンテカルロ法とは，候補手をランダムに終局までシミュレーションを行い，有望な手を探索する手法である．モンテカルロ木探索や UCT（UCB applied to Tree）など様々な実装法が存在するが，その中で最も単純な手法が原始モンテカルロ法 [28] である．原始モンテカルロ法とは，局面の合法手に対して均等にシミュレーションを行い，その中で期待される報酬が最大となるような手を選択する手法である（図 6.1）．シミュレーションを終局まで行うため，途中局面の状態評価関数の表現が難しいゲームや関連研究に乏しいゲームに対しても容易に AI 開発ができるという利点がある．

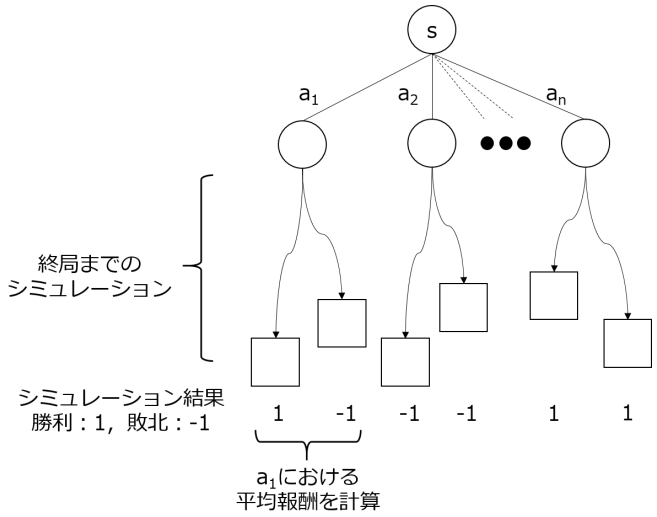



**図 6.1:** 原始モンテカルロ法の概要図．局面 $s$ から候補手に応じてゲーム木を展開，展開したノードから終局までランダムシミュレーションを行い，勝敗を得る．これを 1 ノードに対して複数回行う．この結果から候補手の平均報酬が得られるため，局面 $s$ における候補手の平均報酬を比較し，最も高い手を選択する．

原始モンテカルロ法ではシミュレーションを終局まで行う必要があり，終局まで時間の要するゲームにおいては計算コストが大きくなる．この問題への対処法の 1 つとして，深さ限定モンテカルロ法（Depth-Limited Monte-Carlo，DLMC）が挙げられる．DLMC とは，シミュレーションを一定の深さで打ち切り，局面の評価関数を用いて評価し，この数値をシミュレーション結果の代わりとする手法である（図 6.2）[27]．原始モンテカルロ法が終局までシミュレーションするのに対して，DLMC は一定の深さで打ち切るためにシミュレーションにかかる計算コストが小さく，高速な動作が期待できる．この手法はボードゲーム Amazon やターンベーストラテジーゲーム TUBSTAP において優れた結果を残している [27][29][30]．原始モンテカルロ法は状態評価関数を要しない点が利点であり，DLMC はその点を放棄しているように見えるかもしれない．しかし将棋の駒の取り合いやローグライクゲームの戦闘の後など,「落ち着いた局面」でならば状態評価関数を定義しやすいことはしばしばあるため，$\alpha\beta$ 法などの普通のゲーム木探索と原始モンテカルロ法の中間的な存在である DLMC が有望なことも多い．

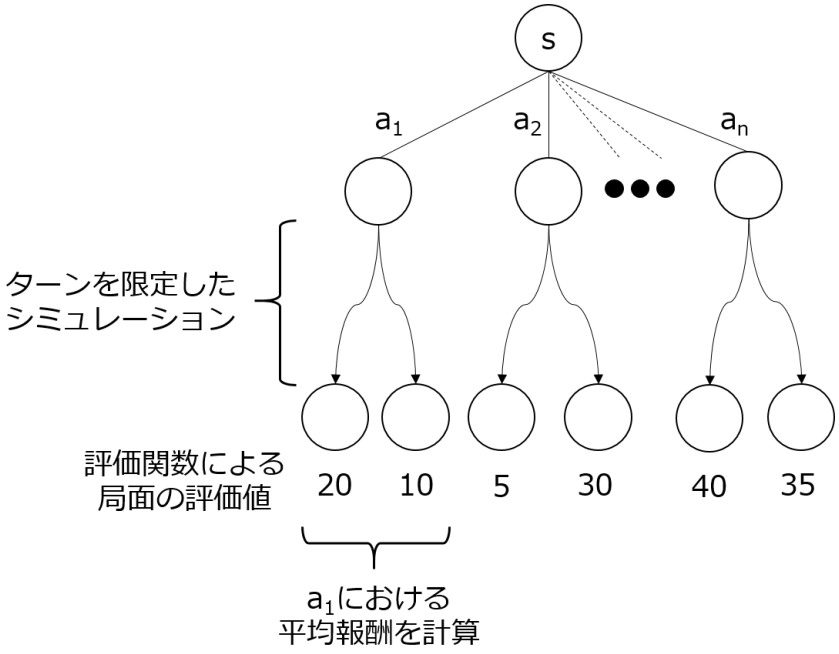



**図 6.2:** 深さ限定モンテカルロ法の概要図．シミュレーションは一定の深さで打ち切り，リーフノードから評価関数により評価値を得る．この結果から候補手の評価を得ることができるため，最も高い手を選択する．

## 6.2 アルゴリズム

本ゲームを知的にプレイするためには，短期的長期的さまざまな種類の能力と判断が求められる．対峙した敵にどう対処するか，マップをどのように巡回するか，経験値稼ぎを行うかどうか，アイテムの良い保持バランス，など考えるべきことは多い．5.2 節の観測から，まず我々はプレイヤの戦術面の弱さに着目した．

モンテカルロ法を用いる場合，直近行動の長期的な評価を行うことは短期的な評価を行うことに比べて一般に難しい．本ゲームにおいても,「ゲームをクリアできたかどうか」で評価を行うとすると，シミュレーション部分は非常に長くなる．これでエージェントをランダムに動かしすぎると，一手目の良し悪しが結果に反映されにくくなってしまう．そこで一旦，“一般的で長期的な” 意思決定を改善することは諦め，“特定の短期的な” 意思決定についてのみ改善を図った．

具体的には，提案 AI は，敵が視野内にいる場合のみモンテカルロ法を行う．また，ゲーム終了までではなく,「10 ターン経過，自分の死亡，階段への到達，または敵が視野内にいなくなった時点で」シミュレーションを打ち切り，結果の評価を行う．その評価関数は以下の式とした．アイテムの回復薬・杖・食料と矢の重みが異なるのは，矢が 3 本まとめて 1 アイテムとして拾うことができるからである．3 本まとめて拾うことで，約 70 程度となるよう 1 本あたりの重みを 23 とした．

$$
\begin{aligned}
\text{評価値} = &(\text{敵のダメージ}) \\
&- (\text{自分のダメージ}) \\
&+ (\text{どれだけ早く敵を倒せたか}, \\
&\qquad 50 \text{ から } 1 \text{ ターン毎に } 5 \text{ 減少し}, \\
&\qquad \text{倒せなかったら } 0) \\
&+ (\text{自分が生きているならば } 1000) \\
&+ (\text{クリアならば } 1000) \\
&+ 70(\text{回復薬の数} + \text{杖の数} + \text{食料の数}) \\
&+ 23(\text{矢の数})
\end{aligned}
$$

モンテカルロ法には UCT など様々な実装法があるが，本研究の実験では深さ 2 の全幅均等探索を行った．すなわち，現在の状況で取れる全ての行動を列挙し，敵行動を予測し，さらにそこから自分が取れる行動を列挙する．なお本ゲームでは視野内の状態遷移についてランダム性はないので（敵の行動パターンは既知で，攻撃が外れるようなこともない），min-max 探索等は不要である．その上で，各遷移先状態について同じ回数ランダムシミュレーションを行い，前述の評価関数値の平均値が最も高い行動対の初手を選択する．

## 6.3　実験結果と例

シミュレーション回数を 100 とし，ゲームを 1000 試行したところ，クリア率は 82.3%となった．各階における死亡回数は表 6.1 に示すとおりである．また，ゲームを通しての餓死回数は 21 回，1 ゲーム当たりのアイテムの使用個数は，食料：2.5 個，回復薬：0.4 個，矢：1.2 個，杖：1.6 個，となった．実行時間は手元の PC で 1 ゲームあたり平均 54 秒であった．これはルールベースプレイヤの場合の平均 1 秒に比べれば長いが，想定しているような用途では十分な短さである．

**表 6.1:** モンテカルロプレイヤの各階における死亡回数．

| 階層 | 死亡回数 |
|---|---|
| 1 階 | 16 |
| 2 階 | 45 |
| 3 階 | 57 |
| 4 階 | 59 |

戦闘中の行動を改善したのみで 11.6 ポイントのクリア率向上が見られた．ただし，試行回数が異なるため，統計的に有意であるのは 8.9 ポイント程度である．改善された点として，具体的には，部屋に入って敵を近くに発見した場合や，部屋内で複数の敵に囲まれそうなとき（図 6.3 左），通路側に戻り敵を単体で迎え撃つことができるようになったことが大きい（図 6.3 右）．

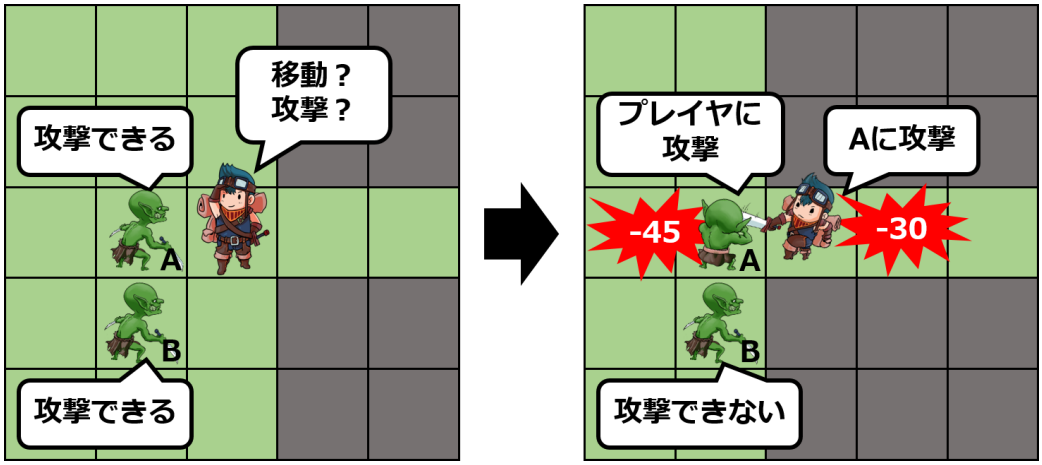



**図 6.3:** モンテカルロプレイヤの行動例 1．左図の状態から 1 マス右に移動し，それから攻撃している．このように，通路に逃げ込み一対一の状況を作り出すことができた．ルールベースプレイヤでは 2 匹分のダメージを同時に受けていたところを 1 匹分に抑えることができている．

その一方で，長期的な観点では，アイテムを無駄遣いしてしまっている状況が見受けられる．使わなくとも良い状況でアイテムを使えば，それ以降で本当にアイテムが必要になったときに窮することになりかねない．具体的には，アイテムを使わずとも対処可能な状況に対して，杖を使って戦闘を回避してしまうことがある．杖の使用を数値で見ると，先ほど示した 1 ゲーム当たりの杖の使用回数は 1.6 回であり，ルールベースプレイヤの 0.4 回を上回っている．

例えば，図 6.4 最上段のような状況があったとする（敵から攻撃を 4 発食らうとゲームオーバー，所持アイテムは杖 1 個のみ，敵を倒すには 2 回の攻撃が必要）．このとき，モンテカルロプレイヤは杖をすぐに使ってしまい一対一の状況を作ってから（図 6.4[結果 1]），敵 B を対処している．しかし実際には，敵 A を普通に攻撃して倒してから（図 6.4[結果 2]）敵 B を倒すことで，敵から 2 回攻撃を食らうものの，杖を使用せずに状況を乗り切ることができる．これは，手作業で作成した状態評価関数の重みのバランスが良くないということや，ランダムシミュレーション中に自分が悪い行動をして死ぬリスクを避けてしまっていることが理由であると考える．

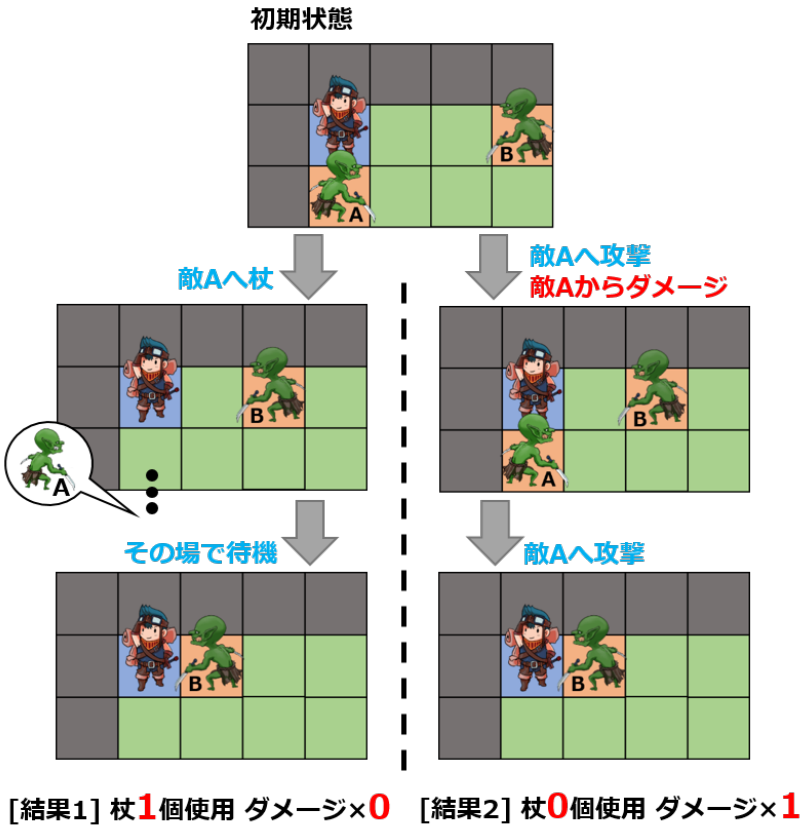



**図 6.4:** モンテカルロプレイヤの行動例 2．敵から攻撃を 4 発食らうとゲームオーバー，所持アイテムは杖 1 個のみ，敵を倒すには 2 回の攻撃が必要．杖を使用しなくとも乗り切ることは可能だが，使用して一対一の状況を作る．

## 6.4 考察

6.3 節で示したように，ルールベースプレイヤと比較すると，短期的な戦術の改善が見られたものの，その一方でアイテム等の無駄遣いが見られた．このゲームはランダム性が高く，不完全情報ゲームであることから「(万が一の場合を考えた)アイテムを温存した立ち回り」「満腹度を考慮した行動」など，ゲーム後半を見越した長期的な計画が非常に重要である．そのためにも，戦闘はただ乗り切るのではなく，どのような状態を目指して乗り切るか，ということを考慮しなければならない．そのためには評価関数の調整が必要だが，6.2 節にて行ったような手作業での作成・調節には限界がある．

そこで次章では状態評価関数の教師あり学習を試みる．

# 第7章　長期的な状態評価

キャラクタは体力・満腹度・レベルといったパラメータと，戦闘用のアイテムを持つ．これらは全て多いほうが良いが，どんな場合に何がどの程度有効なのかは自明ではない．前章では短期的な戦闘を乗り切るためのモンテカルロ法を提示したが，より強いプレイヤを作成するためには，戦闘をただ乗り切るだけでなく，どうやって乗り切るかを判断させなければならない．すなわち，複数の状態（体力やアイテム数）を評価して最善の状態を目指せるようにならなければならない．

この目的のために行えることは様々あるが，本章では，盤面と勝敗のペアを学習データとして集め，前章 DLMC における状態評価関数を，平均化パーセプトロンを用いた教師あり学習により求めるアプローチについて紹介する．

## 7.1　平均化パーセプトロン

単純パーセプトロンとは，二値クラス分類に適用できる教師あり学習のモデルである．学習データが線形分離可能であれば，それらを正しく二値クラスに分類することができる．単純パーセプトロンの学習アルゴリズムは以下の流れとなる．

1. 重みベクトル $\boldsymbol{w}$ を初期化
2. 学習データからランダムにサンプルを選択
3. $\phi(\boldsymbol{x})$ をサンプル $\boldsymbol{x}$ の特徴ベクトルとするとき，
   判別式 $f(\boldsymbol{x}) = \boldsymbol{w} \cdot \phi(\boldsymbol{x})$ が 0 以上ならば正例と判定．
4. 分類が間違っていたら以下の式で重みベクトルを更新
   - 正例の場合 $\boldsymbol{w} \leftarrow \boldsymbol{w} + \phi(\boldsymbol{x})$
   - 負例の場合 $\boldsymbol{w} \leftarrow \boldsymbol{w} - \phi(\boldsymbol{x})$
5. すべてのサンプルを正しく分類できるまで 2 に戻り繰り返す

学習アルゴリズムが単純であり実装が容易というメリットがある一方で，学習データにノイズが含まれる場合，振動して重みが収束しない点や，学習の終盤で選択されたサンプルに影響されやすいというデメリットがある．

そのデメリットを補うための手法が，平均化パーセプトロンである．単純パーセプトロンでは「学習事例から重みベクトルを更新」という処理を反復した後に，重みベクトルを出力した．平均化パーセプトロンでは過去の反復で学習した重みベクトルの平均を出力する．このため，単純パーセプトロンと比較してノイズに強く，終盤の学習サンプルからの影響が小さい．本研究ではこの平均化パーセプトロンを用いて学習を行う．

## 7.2　アルゴリズムの概要とオプション

本章では，6.1 節で述べた DLMC を引き続き用いるが，6.2 節で述べた状態評価関数の代わりに，それを教師あり学習することを試みる．この技術が確立できれば，ダンジョンの設定や構成要素が変わっても評価関数を手作業で作成することなく DLMC を用いることができるようになる．基本的な手順は以下の通りである．

1. 何度もゲームを行い，その途中状態 $s$ と，クリアできたかどうか $r$ をペア $(s, r)$ で保存する．
2. 状態 $s$ は，$n$ 次元の特徴量ベクトル $\boldsymbol{x} \in \mathbf{R}^n$ に変換して保存する．
3. 十分なデータが集まったら，$\{(\boldsymbol{x}_i, r_i)\}_i$ を学習データとして，平均化パーセプトロンを行う．
4. 平均化パーセプトロンの判別式 $f(\boldsymbol{x})$ を，状態評価関数として DLMC のリーフノードで用いる．

このとき，データ数や学習パラメータなどの他に，上記手続きにも様々なオプションが存在することに注意しなければならない．

**A. 政策**
1. で，どんな政策に従ってゲームを行うか．固定のもの（5,6 章で提案したもの）か，自分自身を使うオンライン学習にするか．

**B. 収集タイミング**
1. で，どんなタイミングで途中状態を保存するか．毎ステップか，毎戦闘後か，毎階段を降りる際かなど．

**C. データ補完の有無**
1. ではまんべんなく多様な状態を収集しにくい場合があるので，レアな状態を作成して与えるかどうか．

**D. 状態の表現**
2. でどのようなベクトル表現を行うか．特に，アイテムの個数について線形表現を行うかワンホット表現を行うか．

**E. リーフノードでの処理**
リーフノードで，判別式をそのまま使う（平均化する）か，勝ち負けに変換するか，別のものに変換するか．

**F. DLMC の適用範囲**
DLMC を戦闘のみに使うか，階段を下りるかの判断などにも使うか．

これらは本来全てが関連しているものであり，そのすべての組み合わせを網羅して実験することは事実上不可能である．本研究では数十通り程度の実験を行ったが，本論文ではその中の特徴的な組み合わせについて，比較を行うことにする．

まず 7.3 節では，データの収集タイミングを階段降下時のみ（設定 1）と戦闘終了ごと（設定 2）の比較を行う．結果，2 つの間に有意な差はみられなかったため，タイミングの自然さ・収集できるデータ数の多さから，データ収集は戦闘終了ごととした．続いて 7.4 節では，線形表現を用いてきた特徴量をワンホット表現にする試み（設定 3）と，学習データの補完の試み（設定 4）の 2 つをおこない，最終的に複数の特徴量をワンホットに表現し，データを補完する実験（設定 5）を行った．その結果，特定の階層に限ってだが，手作業で作成した状態評価関数を用いた DLMC と同程度の性能を得ることができた．

**表 7.1:** 設定の比較.

| 設定 | A．政策 | B．収集 | C．レアデータの補完 | D．表現 |
|---|---|---|---|---|
| 1 | ルールベース固定 | 階段降下時のみ | なし | 線形 |
| 2 | ルールベース固定 | 戦闘終了ごと | なし | 線形 |
| 3 | ルールベース固定 | 戦闘終了ごと | なし | ワンホット |
| 4 | ルールベース固定 | 戦闘終了ごと | あり | ワンホット |
| 5 | ルールベース固定 | 戦闘終了ごと | あり | ワンホット（複数） |

## 7.3　データ収集タイミングでの比較

### 7.3.1　概要

本来，任意の状態からその状態の勝率を推定できれば，探索等にとっては最も都合がよい．しかし，未探索部分があったり，敵がいたりするような状況を正しく学習させるのは，入力の次元数が高すぎて非常に多くの学習データや複雑な入出力モデルが必要になると考える．

そこで本節では,「(設定1) 階段を下りたタイミングでのみデータ収集」「(設定2) 戦闘終了ごとにデータ収集」を比較実験する．どちらも,「全ての（任意の）状態でのデータ収集」に比べれば頻度は少なく，かつ，敵の配置を考えなくて良いという点で特徴量設計が楽な設定である．

DLMC を用いた行動選択では戦闘終了のタイミングで評価関数を用いるので，その意味ではデータ収集のタイミングをそれに合わせた設定2が自然に思える．一方で，複数階層のあるローグライクゲームでは階層を移ると新しいマップ（敵・アイテムを含む）がランダムに与えられるため，その移ったタイミングでの「それ以降の勝率」を推測するためには，その時点でのパラメータのみを考慮すればよいという特徴がある．すなわち，設定1を用いる場合にはマップ情報や自分の位置情報などを完全に無視することができるという利点がある．

**表 7.2:** 設定の比較（設定1・2）．A 政策はルールベース固定，C レアデータの補完はなし，D 表現は線形型で共通である．

| 設定 | B．収集 | 次元数 | 特徴量 |
|---|---|---|---|
| 1 | 階段降下時のみ | 6 | 体力，レベル，満腹度（+食料），回復薬，杖，矢 |
| 2 | 戦闘終了ごと | 8 | 設定1の6特徴量に加え，未知領域割合，階段 |

### 7.3.2　データ収集

設定1・2どちらもルールベースプレイヤによりデータを収集する．このとき，設定1ではある階層から階段を下りた時の（体力，レベル，満腹度+食料による回復見込みの量，回復薬の数，矢の数，杖の数）の6次元を入力，その試行がクリアできたかどうかをバイナリの出力とするような学習データを作成した．

満腹度と食料を1入力に統合した理由としては，戦闘面での寄与が小さく，戦闘時以外の任意のタイミングでの回復が一般的だと考えたためである．一方，体力と回復薬は戦闘時の危機的状況での使用など，戦闘面に大きくかかわると考えたため別入力とした．

設定2では，設定1の6次元に加え（階段を発見しているか，フロアの未探索領域の割合は何%か）という2つのマップ情報に関連した特徴量を加えた8次元を入力，出力は設定1と同様である．

追加した2つの特徴量は，ローグライクゲームをプレイする上で重要な要素であるうえ，表現が容易である．例えば，戦闘時において階段を発見していると，単純な移動・戦闘のほかにも，次の階へ逃げるという選択も考えることができる．また未知の領域からは,「フロアほぼ探索済みの戦闘」と「ほぼ未探索での戦闘」ではプレイヤの強さ・アイテムの豊富さが異なるという点を考慮することができる．

以上のような学習データをルールベースプレイヤにより100000試行分収集した結果，それぞれ表7.3のように得られた．1階から3階までの設定1と設定2の間には5倍前後のデータ数の差があり，1階層あたり約5回の戦闘を行っていることがわかる．設定2の4階のデータ数が他の階層と比較して少ないのは，フロアの探索が不十分でも，発見次第すぐに階段へ向かわせているからである．他の階層では，フロアが十分でない，もしくはアイテムを充分拾っていない場合には，発見しても階段へは向かわないようにしている．しかし最終層では，階段への到着＝ゲームクリアのため，先述したように発見次第すぐに向かわせている．そのため，フロアを移動するターン数が減り，敵と遭遇・戦闘する機会が減ったのだと考える．

正例数・負例数を見ると，同じ階層であっても設定1と2の間には差があることがわかる．設定1では，無事該当の階層を突破できた場合のデータのみを収集しているため，負例（突破後，別の階層で死んだデータのみ）は少なくなってしまう．一方設定2は，戦闘後のこれからやられるかもしれないデータについても収集しているため，設定1と比べて多くの負例を収集することができる．

**表 7.3:** 収集したデータ数の比較（設定1・2）.

| 階層 | 設定1（正例数／負例数） | 設定2（正例数／負例数） |
|---|---|---|
| 1階 | 95872（70752／25120） | 478563（351473／127090） |
| 2階 | 87588（70752／16836） | 469108（362859／106249） |
| 3階 | 77995（70752／7243） | 367151（307825／59326） |
| 4階 | 70752（70752／-） | 164176（149476／14700） |

### 7.3.3　学習結果と考察

【学習設定】

収集したデータは，学習データとテストデータを 9:1 に分割するホールドアウト法を用いた．また，パーセプトロンでの学習時に，ノイズ等により重みが振動し収束しないことを考慮し，1 データ当たりの学習回数は 1000 回を上限とした．

【得られた重み】

学習後のそれぞれの 3 階の平均重みを表 7.4 に示す．どちらも共通して，レベルや回復薬・杖など重要な部分に大きな重みが与えられていることがわかる．体力や満腹度は，その値が 100 前後をとることが多いため，重みも他と比較して低めに落ち着いたのだと考える．矢がほかのアイテムと比較し低いのは，1 アイテムで 3 個獲得することができ，回復薬や杖よりもその所持数が多くなりやすいからだと考える．設定 2 における $w_{unarea}$, $w_{stair}$ の値を見ると，探索をすればするほど評価が低くなるような重みが得られている．これは，敵との連属した戦闘による体力およびアイテムの消耗により，フロア突入直後よりもある程度探索した終盤にやられてしまうことが多いためだと考える．

**表 7.4:** 3 階の平均重み（設定 1・2）.

| 特徴量 | 重み $\boldsymbol{w}$ | 設定 1 | 設定 2 |
|---|---|---|---|
| 体力 | $w_{hp}$ | 35.2 | 90.2 |
| レベル | $w_{lv}$ | 160.7 | 329.6 |
| 満腹度 | $w_{sp}$ | 19.9 | 20.3 |
| 回復薬 | $w_{pt}$ | 718.1 | 968.4 |
| 矢 | $w_{ar}$ | 185.5 | 87.1 |
| 杖 | $w_{st}$ | 270.1 | 719.8 |
| 未知領域の割合 | $w_{unarea}$ | - | 6.3 |
| 階段発見の有無 | $w_{stair}$ | - | -220.1 |
| バイアス | $w_{bias}$ | -7245.6 | -12434.1 |

【予測精度】

表 7.5 に設定 1・2 における混同行列を示す．一致率では設定 1 が設定 2 よりも優れている．しかし実際には，設定 1 は 3 階を無事突破できた状況のみを集め，設定 2 は 3 階の中でこれから敵にやられるかもしれない状況も含めて集めているため，正例数の割合が異なる．そのため汎化性能のみで性能を判断することは難しい．

**表 7.5:** 設定 1・2 における混同行列．

**(a)** 設定 1.
一致率 88.9%，F 値 0.94.

| | | 予測値 | |
|---|---|---|---|
| | | 正 | 負 |
| 真値 | 正 | 6878 | 212 |
| | 負 | 657 | 52 |

**(b)** 設定 2.
一致率 80.4%，F 値 0.88.

| | | 予測値 | |
|---|---|---|---|
| | | 正 | 負 |
| 真値 | 正 | 27863 | 2932 |
| | 負 | 4246 | 1669 |

【DLMC で利用する際に予想される挙動】

実際に得られた重みを使用した場合について考えてみると，3 階では敵からは一度に 30 の攻撃を食らうため，評価値上ではそれぞれ $35.2 \times 30 = 1056$，$90.2 \times 30 = 2706$ ほどマイナスされる．一方，杖の重みを見ると 1 本あたり 270.1, 719.8 と，先ほどの被ダメージによる評価値マイナス分と比較すると小さく，敵から攻撃を食らうようならばどんどんアイテムを使うようなプレイが予想される．次節では実際にこの重みを用いてゲームを行う．

### 7.3.4 パーセプトロンの重みを用いたDLMCの実験

本節では，前節で学習したパーセプトロンの重みをDLMCのリーフノード評価に用いる実験（表7.1のDLMC+パーセプトロン 設定1・2）を行い，この結果を5章（ルールベース）と6章（手つくり評価関数によるDLMC）の結果と比較する．

1階2階はアイテムを所持していないもしくは少ない事が多く，運に左右される部分が多いと考え，今回は3階に限定した比較を行う．具体的には，1階2階をルールベースプレイヤによりプレイし，3階で上記の各プレイヤに切り替える．そして3階を突破できた割合を比較することで，特定の階層に限った状態で十分な性能を発揮することができるか確認する．

実験の結果を表7.6に示す．突破率を見ると手作り評価関数を用いたDLMCが最も高く，試行回数に違いがあるものの95%信頼区間において突破率は92.0 ± 1.7%であり，他の設定と比較して有意な差が見られた．設定1・2はルールベースプレイヤ以下の性能となった．原因として，前節で述べたようなアイテムの無駄遣いが挙げられる．実際に挙動を確認してみると，体力に余裕があり，杖を使わなくとも問題なく敵を倒すことのできる場面で，敵から攻撃を受ける前に杖を使用し戦闘を終了していた．

**表 7.6:** 3階の突破率の比較（設定1・2）．

| 設定 | ゲーム数 | 3階突破数 | 突破率（%） |
|---|---|---|---|
| ルールベース | 87588 | 77995 | 89.0 |
| DLMC + 手作り評価関数 | 10000 | 9204 | 92.0 |
| DLMC + パーセプトロン 設定1 | 1000 | 854 | 85.4 |
| DLMC + パーセプトロン 設定2 | 1000 | 855 | 85.5 |

設定1・2の間に有意な差が見られなかったことから，データの収集タイミングの変更や設定2において追加した2つの特徴量による影響は小さかったことがわかる．特に，追加した2つの特徴量は，パーセプトロン学習時の「ゲームをクリアできそうか／できなさそうか」という予測を行う分には有益だが，実際のゲームにおいて，DLMCを用いて行動選択をする際には無益（戦闘途中にこの2特徴量の値が変わることは少ない）であったためだと考える．

設定1・2で見られたアイテムを無駄遣いするような挙動の原因の1つは，特徴量におけるアイテムの表現方法にあると考える．現状の表現方法（線形表現）では，例えば杖0個と1個，3個と4個の差は等しいもの（杖1つあたりの重みが100であれば，どちらも杖使用で評価値100減少）となってしまう．しかし，実際には「1個しかないから使わないで乗り切ろう（1個→0個は例えば1000減少）」「4個もあるから使っても大丈夫だろう（4個→3個は例えば100減少）」といった様に，所持数によりその考え方（重み）を切り替えることが必要だと考える．そこで，次節から状態の表現方法の工夫を試みる．また，設定1・2の間に差が見られなかったことから，タイミングの自然さと収集できる数を考慮して，次節以降データ収集は戦闘終了ごとに行うこととする．

## 7.4　ワンホット表現とレアデータ補完の試み

### 7.4.1　概要

本節では，前節の設定 2 をベースにして，2 つの改善を試みる．

一つは，前節まで線形表現してきた特徴量を，0 と 1 のベクトルで表すワンホット表現に変更する．より豊かな状態表現を可能にすることで，アイテムの個数などに応じたそれぞれの重みの学習を行う．

もう一つは，学習に用いるデータが “学習したい状況” を網羅していないという状況を改善するための，レアデータの補完の試みである．これは，ワンホット表現を行うことで入力の次元数が増え，それまでのデータ数では十分な学習結果が得られず未知データに適切に対応できないということで発生した問題である．

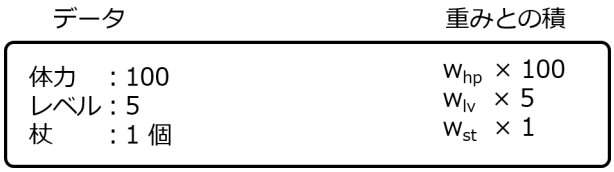
以上の 2 つの改善を試みた上で，複数の特徴量をワンホット表現に変更し，それに応じたデータ補完を行う．これにより，最終的に DLMC のリーフノード評価として用いた際の性能改善を図る．表 7.7 に設定 2・3・4・5 の関係をまとめる．

**表 7.7:** 設定の比較（設定 2・3・4・5）．A 政策はルールベースで固定，B 収集は戦闘終了ごとで共通である．

| 設定 | C．レアデータの補完 | データ数 | D．表現 |
|---|---|---|---|
| 2 | なし | 367151 | 線形 |
| 3 | なし | 367151 | ワンホット（矢） |
| 4 | あり | 367151+1699752 | ワンホット（矢） |
| 5 | あり | 367151+10083237 | ワンホット（体力，全アイテム） |

ワンホット表現を用いた具体的な表現方法の例を図 7.1 に示す．杖の場合は [4 個以上 3 個 2 個 1 個 0 個] の 5 つの要素を用いて表現する．杖 0 個は [0 0 0 0 1] と表される．同様に 1 個は [0 0 0 1 0]，4 個以上の場合にはすべて [1 0 0 0 0] と表現する．もちろん，表現方法はこれのみに限らず，例えば所持できる限界数まで一つ一つ細かく特徴量を用意したり，逆にいくつかまとめてしまうなどの工夫をすることもできるだろう．

このようなワンホット表現を用いることで，アイテムであればその所持数に応じて重みを使い分けることができるため，DLMC において利用した際に「1 個しかないから使わないで乗り切ろう」「4 個もあるから使っても大丈夫だろう」という柔軟な行動選択が期待できる．しかし，だからといって特徴量のすべてを細かくワンホットに表現する必要はない．例えば hp100, 99, 98, ... のように，極端にモデルをリッチにしてしまうと過学習が起きるからである．




**(a)** 線形表現．入力データの数値をそのまま用いて学習・評価値計算を行う．


**(b)** ワンホット表現．入力データの数値をベクトルに変換，杖 1 個を杖 [0 0 0 1 0] のように表現する．この変換後の特徴量を用いて学習・評価値計算を行う．

**図 7.1:** 特徴量の表現方法．

### 7.4.2　設定 3 における学習結果と考察

【学習データとその表現方法】

はじめに，設定 3 として矢についてのみをワンホット表現にした場合について学習を行う．矢は 3 個まとめて 1 アイテムとして拾うことができるため，杖・回復薬より所持数が多くなりがちである．そのため，杖は [4 個以上 3 個 2 個 1 個 0 個] と表現していたが，矢は [12 個以上 9 個 6 個 3 個 0 個] と表現する．0 と 1 で表現できない場合，例えば矢 4 個 [0 0 0.33 0.66 0]，矢 5 個 [0 0 0.66 0.33 0] のように表現する．

以上より，入力は（体力，レベル，満腹度+食料による回復見込みの量，回復薬の数，矢の数 12 本以上，矢の数 9 本，矢の数 6 本，矢の数 3 本，矢の数 0 本，杖の数）の 13 次元，出力は設定 1・2 と同様にクリアできたかどうかのバイナリとした．なお，学習データは設定 2 のもの（前節の表 7.3，戦闘終了ごとに収集したデータ）を用いる．

【結果と考察】

学習の結果，重みは表 7.8 に示す値となった．設定 2 と設定 3 の重みを比較すると，矢の部分を除いて似た値であることがわかる．矢の重みについても，設定 2 が 0〜12 個で差が $87.1 \times 12 = 1045.2$ なのに対し，設定 3 においても $w_{ar0}$ と $w_{arover12}$ の差は $2628.4 - 1509.7 = 1118.7$ であり，近い値が得られている．しかしその一方で，$w_{ar9}$ と $w_{ar6}$ において重みの逆転が起きていた．

重みの逆転は望ましいものではなく，この場合「矢 9 本持つよりも 6 本のほうが良い」というおかしい判断をしていることになる．これでは線形表現と同様に，必要のない状況で無駄遣いしてしまうと考える．原因として，学習途中であり重みが収束していないことなども考えられるが，まず矢の個数毎のデータ数（表 7.9）に着目した．この表から，矢の個数 9 個や 12 個付近のデータ数が 3 個や 0 個と比較して少ないことがわかる．ゲーム中，アイテムは 4 種類からランダムに配置されるため，矢が多く出現し，なおかつ戦闘中に使わなかったような場面でなければ，矢の個数 9 個や 12 個というデータは得られない．そのため，矢の個数が多くなればなるほど，なかなか出現しないようなレアなデータと言える．ワンホット部分における各重みの関係を正すためには，現状数少ないこのようなデータを収集できるような方法を検討することが必要だと考える．

**表 7.8:** 3 階の平均重み（設定 2・3）．$w_{arover12}$ から $w_{ar6}$ において重みの逆転が見られる．

| 特徴量 | 重み $\boldsymbol{w}$ | 設定 2 | 設定 3 |
|---|---|---|---|
| 体力 | $w_{hp}$ | 90.2 | 89.0 |
| レベル | $w_{lv}$ | 329.6 | 335.3 |
| 満腹度 | $w_{sp}$ | 20.3 | 20.5 |
| 回復薬 | $w_{pt}$ | 968.4 | 977.8 |
| 矢 | $w_{ar}$ | 87.1 | - |
| 矢 12 個以上 | $w_{arover12}$ | - | -1509.7 |
| 矢 9 個 | $w_{ar9}$ | - | -1623.8 |
| 矢 6 個 | $w_{ar6}$ | - | -1510.1 |
| 矢 3 個 | $w_{ar3}$ | - | -2593.4 |
| 矢 0 個 | $w_{ar0}$ | - | -2628.4 |
| 杖 | $w_{st}$ | 719.8 | 713.8 |
| 未知領域の割合 | $w_{unarea}$ | 6.3 | 6.2 |
| 階段発見の有無 | $w_{stair}$ | -220.1 | -246.5 |
| バイアス | $w_{bias}$ | -12434.1 | -9865.4 |

**表 7.9:** 矢の個数毎のデータ数（設定 3）．矢の個数 12 個は 40 データしかない．

| 矢の個数 | データ数 | 正例数 | 負例数 |
|---|---|---|---|
| 0 | 309752 | 260113 | 49639 |
| 1 | 16595 | 13805 | 2790 |
| 2 | 14503 | 11953 | 2550 |
| 3 | 15050 | 12561 | 2489 |
| 4 | 4030 | 3401 | 629 |
| 5 | 3080 | 2597 | 483 |
| 6 | 2307 | 1898 | 409 |
| 7 | 716 | 594 | 122 |
| 8 | 477 | 394 | 83 |
| 9 | 342 | 271 | 71 |
| 10 | 135 | 109 | 26 |
| 11 | 91 | 73 | 18 |
| 12 | 40 | 32 | 8 |
| 13 | 18 | 15 | 3 |
| 14 | 7 | 4 | 3 |
| 15 | 7 | 5 | 2 |
| 16 | 1 | 0 | 1 |

### 7.4.3　レアデータの補完

【収集方法の検討】

前節では，矢をワンホット表現にした結果，得られた重みにおいて「矢の個数が少ないほうが，評価は高い」ということが発生していた．これは，学習データに矢の所持数毎に大きな偏りがあり，“学習したい状況” を網羅できていないことが原因であると考えた．

そこで設定 4 では，不足している状況（レアな学習データ）を補うことを試みる．方法として，(1) 単純にゲームを繰り返すことでデータを集める (2) 経験が少ない状況をあたえデータを作る，の 2 つが考えられる．しかし (1) のようにやみくもにデータを作成するのは，本当に必要なデータを収集するまでに多くの時間を要する恐れがある．そのため，今回は (2) の方法によりデータを収集する．(2) のデータの収集方法を以下に示す．

1. ルールベースプレイヤの 2 階降下時（3 階突入時）の状態を収集
2. 1 の状態における矢の個数を 0, 3, 6, 9, 12 に変更
3. 2 を初期状態として 3 階突入時点からゲーム終了まで再びルールベースプレイヤによりプレイし，その戦闘終了ごとにデータを収集する

1 に関しては，7.3.2 節において収集した 87588 件のデータを用いる．

【収集結果】

実際に 87588 件について矢をそれぞれ 0, 3, 6, 9, 12 本に初期化し，ゲーム終了までプレイした結果を表 7.10 にまとめる．普通に考えれば，他の状態が全く同じであれば矢の本数が多いほど有利になるはずであり，ゲーム終了時の勝率も高くなってほしい．実際，表 7.10 を見ればその通りになっていることがわかる．また，矢が 0 個と 3 個の間に他と比べて大きな差があることもわかり，ワンホット表現にすることの有効性も期待できる．

**表 7.10:** 初期化時の矢の本数による，3 階からルールベースでクリアできた割合（勝率）．初期化時の矢の本数が増加するごとに，勝率の向上が見られる．

| 初期化数 | 勝利 | 敗北 | 勝率（%） |
|---|---|---|---|
| 0 | 69793 | 17795 | 79.7 |
| 3 | 71909 | 15679 | 82.1 |
| 6 | 73405 | 14183 | 83.8 |
| 9 | 75019 | 12569 | 85.6 |
| 12 | 75751 | 11837 | 86.5 |

矢の所持数ごとのデータ数を表 7.11 に示す．データ補完前（設定 3）では矢 12 個に該当するデータは 40 ほどしか存在しなかったが，(2）により新たにデータを収集した結果，3 万データ以上を収集することができた．また，(1）のような方法を用いてデータを収集した場合，補完前（設定 3）の様に矢の個数が増加するほどにデータ数が少なくなるような傾向がみられるが，補完後（設定 4）の場合は（2）で初期化する数を 0, 3, 6, 9, 12 としたため，多少データ数の増減の波が発生した．

**表 7.11:** 矢の個数毎のデータ数の比較（設定 3・4）．補完前には矢 12 個の場合 40 データしかなかったが，補完後には多くのデータが収集できている．13 以上のデータはすべて 12 として扱うため，なくともは問題ない．

| 矢の個数 | 補完前（設定 3） | 補完後（設定 4） |
|---|---|---|
| 0 | 309752 | 1150684 |
| 1 | 16595 | 111506 |
| 2 | 14503 | 120793 |
| 3 | 15050 | 133493 |
| 4 | 4030 | 78001 |
| 5 | 3080 | 86868 |
| 6 | 2307 | 92277 |
| 7 | 716 | 55297 |
| 8 | 477 | 61026 |
| 9 | 342 | 66030 |
| 10 | 135 | 32278 |
| 11 | 91 | 30060 |
| 12 | 40 | 32859 |
| 13 | 18 | 6120 |
| 14 | 7 | 4705 |
| 15 | 7 | 3005 |
| 16 | 1 | 900 |
| 17 | 0 | 558 |
| 18 | 0 | 308 |
| 19 | 0 | 58 |
| 20 | 0 | 59 |
| 21 | 0 | 14 |
| 22 | 0 | 1 |
| 23 | 0 | 2 |
| 24 | 0 | 1 |

### 7.4.4　設定 4 における学習結果と考察

平均化パーセプトロンを用いた学習の結果，得られた平均重みを表 7.12 に示す．設定 3 と比較して重みの傾向は似ているが，矢が多いほど高い値を得られるようになっており，ワンホット表現を使う上でやや満足のいくような結果となった．

**表 7.12:** 3 階の平均重み（設定 3・4）．全体的に重みの傾向が似ており，設定 3 の矢のワンホット表現部分にみられた重みの逆転が設定 4 では見られなくなった．

| 重み $\boldsymbol{w}$ | 設定 3 | 設定 4 |
|---|---|---|
| $w_{hp}$ | 89.0 | 96.5 |
| $w_{lv}$ | 335.3 | 334.3 |
| $w_{sp}$ | 20.5 | 21.0 |
| $w_{pt}$ | 977.8 | 907.9 |
| $w_{arover12}$ | -1509.7 | -1004.9 |
| $w_{ar9}$ | -1623.8 | -1270.7 |
| $w_{ar6}$ | -1510.1 | -1656.3 |
| $w_{ar3}$ | -2593.4 | -2670.4 |
| $w_{ar0}$ | -2628.4 | -3056.9 |
| $w_{st}$ | 713.8 | 620.8 |
| $w_{unarea}$ | 6.2 | 2.5 |
| $w_{stair}$ | -246.5 | -124.6 |
| $w_{bias}$ | -9685.4 | -9659.2 |

以上から，ワンホット表現を用いる際にはデータの補完がある程度有効であることがわかった．そこで，今まで設定 3・4 と矢に対してのみワンホットな表現を用いてきたが，次節の設定 5 では体力とすべてのアイテムに対して適用する．

### 7.4.5　設定5における学習結果と考察

【学習データとその表現方法】

本節では設定5として，体力・回復薬・矢・杖についてワンホット表現を適用する．ワンホット表現の方法は，矢は前節までと同様 [12個以上 9個 6個 3個 0個]，回復薬・杖については [4個以上 3個 2個 1個 0個]，体力は [120以上 90 60 30 1 0] とした．このため，入力は（体力 $\times 6$，レベル，満腹度，回復薬 $\times 5$，矢 $\times 5$，杖 $\times 5$，未知領域の割合，階段発見の有無）の25次元となる．

体力のみ1と0を分けるなど，他とは異なる表現を適用している．これは，ゲームにおいて体力が1残っている事と0であることには大きな違いがあるからである．体力が1でも残っていればそこからまだ挽回の余地はあるため，必ずしもゲームに負けるとは限らない．しかし，体力0は確実な敗北を意味しているため，2つの評価（重み）の間には大きな数値の差が存在するはずである．そのため，このような表現を採用した．

【レアデータの補完】

データの補完は7.4.3節にて行った矢のワンホット表現の際に用いた方法を体力・回復薬・杖に対して同様に行う．そのため，約200万ゲーム分（ルールベースプレイヤ，3階スタートのため1ゲーム当たり0.5秒として約278時間分）を収集した結果，データ数は10450388件と大きいものとなった．もちろんこのすべてが必要なわけではなく，例えばアイテム数が少ないデータは通常のプレイでも十分な数収集できていたため，データを間引くこともできるだろうし，そもそも7.4.3節のデータ収集において，アイテム数が少ない状況からゲームを行うことを省略することもできるだろう（杖0個を初期状態としたデータ収集は行わない，など）．

【得られた重み】

収集した学習データを用いて学習を行った結果，得られた重みを表7.13に示す．体力・回復薬・矢・杖についてワンホットな表現を適用したが，データの補完を行っているため，重みの逆転はみられなかった．また，$w_{hp0}$ の値が負の大きな値をとっており，$w_{hp1}$ と約24000の差がある．これは先述したように，体力が1でも残っていればそこからまだ挽回の余地はあるが，体力0は確実な敗北であることから，このような差が得られたのだと考える．

また，アイテムの個数毎の重みの間にも重要な差が見られた．矢は6個以上では1個あたりの差が約100なのに対して6個未満は1個あたり約250，杖は2個以上では1個あたり約150なのに対して2個未満は500，1300となっている．回復薬においても同様で，個数が少なくなるほどに400, 450, 900, 1300という差が見られる．これらは個数が少なくなるほどに重みの差が大きくなっており，「沢山持っているうちは使ってもそこまで評価は変わらないけど，少ないときに使うことは評価に大きく影響する」ということを表現できるような重みを学習できたと考える．

**表 7.13:** 3 階の平均重み（設定 5）.

| 特徴量 | 重み $\boldsymbol{w}$ | 設定 5 |
|---|---|---|
| 体力 120 以上 | $w_{hpover120}$ | 5729.7 |
| 体力 90 | $w_{hp90}$ | 4761.9 |
| 体力 60 | $w_{hp60}$ | 4138.6 |
| 体力 30 | $w_{hp30}$ | 3677.4 |
| 体力 1 | $w_{hp1}$ | 3107.9 |
| 体力 0 | $w_{hp0}$ | -27200.6 |
| レベル | $w_{lv}$ | 336.8 |
| 満腹度 | $w_{sp}$ | 15.8 |
| 回復薬 4 個以上 | $w_{ptover4}$ | 110.7 |
| 回復薬 3 個 | $w_{pt3}$ | -273.1 |
| 回復薬 2 個 | $w_{pt2}$ | -726.3 |
| 回復薬 1 個 | $w_{pt1}$ | -1634.1 |
| 回復薬 0 個 | $w_{pt0}$ | -3262.4 |
| 矢 12 個以上 | $w_{arover12}$ | -194.8 |
| 矢 9 個 | $w_{ar9}$ | -548.1 |
| 矢 6 個 | $w_{ar6}$ | -880.7 |
| 矢 3 個 | $w_{ar3}$ | -1735.8 |
| 矢 0 個 | $w_{ar0}$ | -2425.8 |
| 杖 4 個以上 | $w_{stover4}$ | -463.2 |
| 杖 3 個 | $w_{st3}$ | -593.5 |
| 杖 2 個 | $w_{st2}$ | -773.4 |
| 杖 1 個 | $w_{st1}$ | -1291.2 |
| 杖 0 個 | $w_{st0}$ | -2663.9 |
| 未知領域の割合 | $w_{unarea}$ | 25.9 |
| 階段発見の有無 | $w_{stair}$ | -155.3 |
| バイアス | $w_{bias}$ | -5785.2 |

### 7.4.6　パーセプトロンの重みを用いたDLMCの実験

本節では，設定3･4･5において学習したパーセプトロンの重みをDLMCのリーフノード評価に用いる実験を行い，この結果を設定1･2，5章（ルールベース）と6章（手つくり評価関数によるDLMC）の結果と比較する．

実験は7.3.4節と同様に，3階を対象として行う．1階2階をルールベースプレイヤによりプレイし，3階で上記の各プレイヤに切り替え，3階を突破できた割合を比較することで，特定の階層に限った状態で十分な性能を発揮することができるか確認する．

実験の結果を表7.14に示す．設定1･2と3･4はいずれも3階突破率が85%前後であり，有意な差は確認できなかった．設定3では矢をワンホット表現に変更し，設定4ではレアデータの補完を行うことで矢の本数が多いほど高い値をとるような重みを得ることができたが，実際にDLMCのリーフノード評価に用いても，性能の向上にはつながらなかった．

その一方で，体力･回復薬･杖についてもワンホットな表現を適用した設定5は，設定1〜4と比較して突破率は向上しており，手作り評価関数を用いたDLMCと同等の性能を得ることができた．これは，矢以外の要素をワンホットにした影響が大きいと考える．矢はこのゲーム（作成したプラットフォーム上）における唯一の遠距離攻撃手段であり，確かにその重要度は高い．しかし，実際にはピンチの場面での活躍が期待できる回復薬や杖の重要度は矢以上に高く，それらをワンホットに表現した設定5は性能が高かったのだと考える．

**表 7.14:** 3階の突破率の比較（設定1〜5）．

| 設定 | ゲーム数 | 3階突破数 | 突破率（%） |
|---|---|---|---|
| ルールベース | 87588 | 77995 | 89.0 |
| DLMC + 手作り評価関数 | 10000 | 9204 | 92.0 |
| DLMC + パーセプトロン 設定1 | 1000 | 854 | 85.4 |
| DLMC + パーセプトロン 設定2 | 1000 | 855 | 85.5 |
| DLMC + パーセプトロン 設定3 | 1000 | 854 | 85.4 |
| DLMC + パーセプトロン 設定4 | 1000 | 847 | 84.7 |
| DLMC + パーセプトロン 設定5 | 10000 | 9165 | 91.7 |

### 7.4.7　得られた挙動

実際に得られた挙動についてみてみる．図 7.2 最上段のような場面があったとする（敵から 4 回攻撃を受けるとゲームオーバー，所持アイテムは杖のみ，敵を倒すには 2 回の攻撃が必要）．

杖を 3 本所持しているとき，設定 1〜5 のすべての場合で敵 A に対して杖を使用し，一対一の状況を作ることで対処していた．設定 1〜4 は，その後も杖を敵 B に対して使ってしまっていた（図 7.2[結果 1]）が，設定 5 は杖を使うことなく敵 B を倒していた（図 7.2[結果 2]）．この場合，敵からは 1 回しか攻撃を食らわない．

杖を 1 本しか所持していないとき，設定 1〜4 が同様に杖を使用していたのに対し，設定 5 では杖を温存するような動きが見られた．その際は敵 A を先に倒し，順に敵 B を倒していた（図 7.2[結果 3]）．この場合，敵 AB から各 1 回ずつ，合計 2 回の攻撃を受けてしまうが，杖というとっておきを残したまま，この状況を突破することができる．

このことから設定 5 では狙い通り「1 個しかないから残しておこう」「いくつかあるから使っても大丈夫だろう」というような，アイテムの所持数による柔軟な行動選択が，重みをワンホット表現にすることにより得られたと考える．

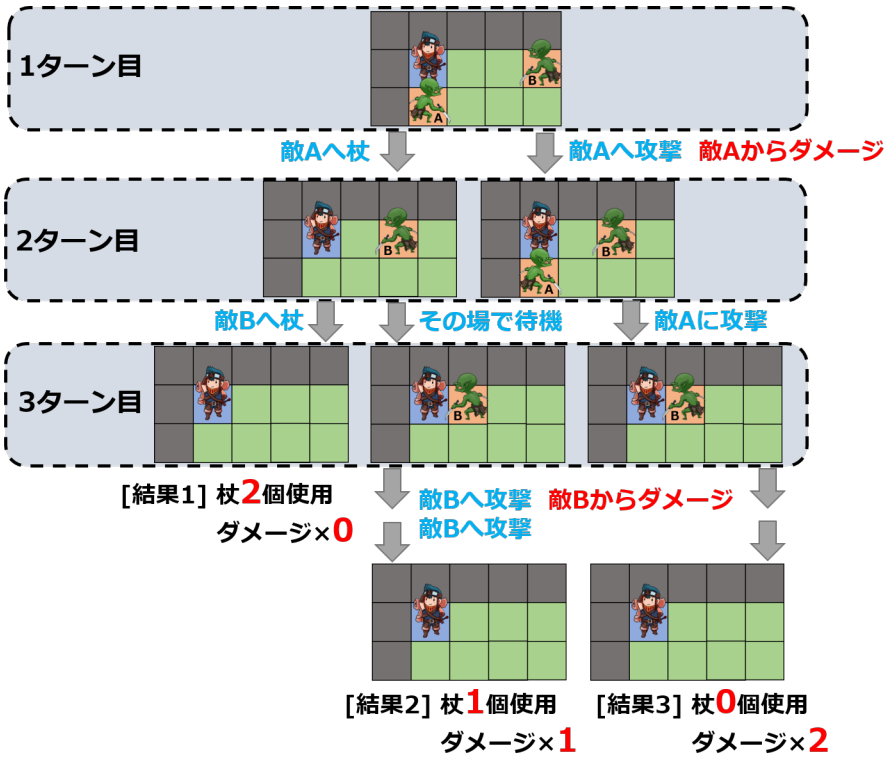



**図 7.2:** 得られた行動例．敵から攻撃を 4 発食らうとゲームオーバー，所持アイテムは杖のみ，敵を倒すには 2 回の攻撃が必要．杖を使ってダメージを食らわないように行動するか，ダメージを食らう代わりに杖を温存するか，選択の余地がある．

### 7.4.8　今後の課題

アイテムの所持数により挙動を変えることができ，手作り評価関数を用いたDLMC と同等の性能を得ることができた設定 5 であったが，不自然な挙動もみられた．それは，隣接する敵から逃げ回るような挙動である．このような動きは，逃げた結果敵に挟まれてしまうような恐れがあるため，うれしいものではない．

評価値を確認した結果，これはターン経過による体力の自動回復が原因であった．DLMC による探索・シミュレーションの中には「(1) 攻撃，攻撃」「(2) 移動，攻撃，攻撃」「(3) 移動，移動，攻撃，攻撃」などの選択肢がある．いずれも 2 回攻撃することで敵を倒しているため，どれも同じように思われるが，ターン経過による体力の自動回復があるために，(3) の評価値が最も高くなってしまう．そのため，設定 5 では敵から逃げるような行動が得られてしまった．手作り評価関数では「どれだけ早く敵を倒すことができたか」ということを考慮していたため，このような問題は発生しなかった．

対策として，学習データに予想する残りの敵の数（敵を倒したらその数を減らし，ターン経過によりポップするタイミングであれば数を増やす）という特徴量を含める，などが考えられるが，その試みは現在成功していない．

今回，設定 5 として特徴量のうち体力・回復薬・矢・杖をワンホットに表現し，足りないデータの補完を行った．これを平均化パーセプトロンにより学習し，得られた重みを用いて DLMC のリーフノード評価を行った結果，特定の階層において，手作り評価関数と同等の性能を得ることができた．しかし，先に述べたような「逃げ」への対策や，他の階層への適用は未だできていないため，今後これらを適用していく所存である．

# 第8章　おわりに

本研究では，ローグライクゲームというジャンルを新たなゲームAIの研究対象として取り上げ，何が面白くまた難しいのかをまとめた後，重要な課題を失わない程度には複雑さを保ちつつ，煩雑でない程度に単純化した研究基盤用ルールを提案した．作成したプラットフォームにおけるゲームのルールや設定パラメータ，ソースコードなどは，必ずしも現時点では洗練されたものとは言えない．ルールやコードを改良したうえで，多くの人に使ってもらえるようにプラットフォームの公開を行いたい．

さらに，このプラットフォームを使用し，ゲームAIの作成を行った．他手法の基盤としてif-thenの決定木からなるルールベースプレイヤを作成し，70.8%の勝率が得られた．これを基に作成したモンテカルロプレイヤは，戦闘場面において探索の深さを限定したモンテカルロ法（DLMC）を適用することで，勝率82.3%を得ることができた．さらに，手作業で作成していた状態評価関数を教師あり学習により求める試みとして，ルールベースプレイヤの戦闘後の状態を学習データとして収集し，体力・回復薬・矢・杖の特徴量をワンホットに表現，これにより不足するデータを改めて収集し，平均化パーセプトロンによる学習を行った．その結果，3階のみを対象としてではあるが，手作業で作成していた状態評価関数と同程度の性能が得られた．今後は他の階層への適用や，「敵から逃げる」ような挙動への対策を講じていきたい．

# 謝辞

本研究を進めるにあたり，多くの方々にご指導ご協力を賜わりました．ここに感謝の意を表します．なかなか研究の進まない自分に，研究の進め方から発表の基本まで，懇切丁寧なご指導をいただきました主指導教員の池田心准教授に深謝いたします．また，副指導教員の飯田弘之教授，研究をはじめ生活面に関しても支えていただきましたTemsiririrkkul Sila さん，そして池田研究室のメンバーに心より感謝いたします．最後に，これまで温かく見守ってくれた家族に感謝します．

# 参考文献


[1] Cambell Murray A. Joseph Hoane Jr, and Feng-hsiung Hsu. Deep blue. Artificial intelligence, 134.1 pp.57-83, 2002

[2] David Silver, Aja Huang, et al. Mastering the game of Go with deep neural networks and tree search, Nature 529, pp.484-489, 2016

[3] Volodymyr Mnih, et al. Playing atari with deep reinforcement learning, arXiv preprint arXiv:1312.5602, 2013

[4] Volodymyr Mnih, et al. Human-level control through deep reinforcement learning, Nature 518.7540, pp.529-533, 2015

[5] Max Jaderberg, et al. Human-level performance in first-person multiplayer games with population-based deep reinforcement learning, arXiv preprint arXiv:1807.01281, 2018

[6] 「StarCraft: Remastered」 https://starcraft.com/ja-jp/ (2019 年 1 月 22 日アクセス)

[7] 「MetaStone - Hearthstone Simulator」 http://www.demilich.net/metastone/index.html (2019 年 1 月 23 日アクセス)

[8] Feiyu Lu, Kaito Yamamoto, Luis H. Nomura, Syunsuke Mizuno, YoungMin Lee, and Ruck Thawonmas. Fighting Game Artificial Intelligence Competition Platform, Proc. of the 2013 IEEE 2nd Global Conference on Consumer Electronics, pp.320-323, 2013

[9] 「Mario AI BenchMark」 https://code.google.com/p/marioai/ (2019 年 1 月 23 日アクセス)

[10] 「torcs」 http://torcs.sourceforge.net/index.php (2017 年 7 月 24 日アクセス)

[11] 「FrontPage - デジタルカーリング」 http://minerva.cs.uec.ac.jp/curling/wiki.cgi?page=FrontPage (2019 年 1 月 23 日アクセス)

[12] 「aiwolf.org」 http://aiwolf.org/ (2019 年 1 月 23 日アクセス)

[13] 「Poje -Ikeda laboratory Project」 http://www.jaist.ac.jp/is/labs/ikeda-lab/poje/index.html (2019 年 1 月 23 日アクセス)

[14] 「TUBSTAP-プロジェクトページ」 http://www.jaist.ac.jp/is/labs/ikeda-lab/tbs/ (2019 年 1 月 23 日アクセス)

[15] 高橋 一幸，Temsiririrkkul Sila，池田 心．ローグライクゲームの研究用ルール提案とモンテカルロ法の適用，第 22 回ゲームプログラミングワークショップ (GPW-17)，2017

[16] 田中 成俊，橋山 智訓，市野 順子，田野 俊一．ローグライクの AI コンペティション，第 29 回ファジィシステムシンポジウム，pp.435-440, 2013

[17] Vojtech Cerny and Filip Dechterenko. Rogue-Like Games as a Playground for Artificial Intelligence Evolutionary Approach, International Conference on Entertainment Computing (ICEC), pp.261-271, 2015

[18] 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，第 23 回ゲームプログラミングワークショップ (GPW-18)（2018）

[19] 金川 裕司，金子 知適．ローグライクゲームによる強化学習用ベンチマーク環境 Rogue-Gym の提案，第 23 回ゲームプログラミングワークショップ (GPW-18)（2018）

[20] IEEE Conference on Computational Intelligence and Games — IEEE Computational Intelligence Society, http://www.ieee-cig.org/ (2019 年 1 月 23 日アクセス)

[21] 「StarCraft AI, the resource for custom StarCraft Brood War AIs」http://www.starcraftai.com/wiki/Main_Page (2019 年 1 月 23 日アクセス)

[22] 「StarCraft II Official Game Site」https://starcraft2.com/ (2019 年 1 月 22 日アクセス)

[23] 「GitHub - Blizzard/s2client-proto: StarCraft II Client - protocol definitions used to communicate with StarCraft II.」https://github.com/Blizzard/s2client-proto (2019 年 1 月 23 日アクセス)

[24] 「The2K BotPrize : Home」http://botprize.org/ (2019 年 1 月 23 日アクセス)

[25] D. J. N. J. Soemers, C. F. Sironi, T. Schuster and M. H. M. Winands. Enhancements for real-time Monte-Carlo Tree Search in General Video Game Playing, IEEE Conference on Computational Intelligence and Games (CIG), pp.1-8 (2016)

[26] 「Berlin Interpretation - RogueBasin」http://www.roguebasin.com/index.php?title=Berlin_Interpretation (2019 年 1 月 23 日アクセス)

[27] 藤木 翼，村山公志朗，池田 心．ターン制ストラテジーにおける状態評価関数を用いた深さ限定モンテカルロの適用，エンターテイメントと認知科学研究ステーション（E&C）第 8 回シンポジウム (2014)

[28] 美添一樹，山下宏．コンピュータ囲碁-モンテカルロ法の理論と実践，共立出版，2011

[29] Kloetzer, J., Iida, H., Bouzy, B., The Monte-Carlo Approach in Amazons, Computer Games Workshop, 2007

[30] Kloetzer, J. Monte-Carlo Techniques:Applications to the Game of the Amazons, Japan Advanced Institute of Science and Technology, 博情第 240 号 Includes bibliographical references pp.87-92, 2010