修士学位論文

グリッド世界を用いた階層型強化学習の評価

2019 年度

広域科学専攻 広域システム科学系

31-186816

高岡峻

目次

第 1 章

はじめに

現実世界の諸問題を効率良く解くために，コンピュータは必要不可欠である．コンピュータの性能が向上するにつれて，我々の生活はより便利で快適なものになっている．近年，ユーザーの発言を理解して要求を実行するスマートスピーカー，スマートフォンのカメラ内蔵の顔検出機能等，我々の生活に機械学習の技術が浸透し始めている．機械学習の進化によって我々の生活がさらに便利になることが期待できる．

機械学習の研究プラットフォームとして，ゲームが挙げられる．現実世界の諸問題の多くはゲームの枠組みで扱うことができ，ゲーム自体はそれらと比べてルールが明確で扱いやすいものが多い．また，ゲームを上手くプレイできるということは，状況を見て自身への効用が大きくなる行動を考えることができることなので，機械学習分野での研究題材として用いられてきた．近年では，全ての情報がプレイヤーに提示されている完全情報ゲームである囲碁や将棋等のゲームにおいて人間のトッププロを破り，プレイヤーに提示されない情報がある不完全情報ゲームとして知られるポーカーにおいて，一対一の勝負でトッププロに勝利するなど，目覚ましい成果が出ている．

ゲームを題材とした研究でよく用いられる機械学習の手法として，強化学習がある．強化学習は事前知識を必要としないため，現実世界のような事前知識の想定が困難な環境に対しても適用可能な自律エージェント用の学習方法として研究されている．しかし，強化学習は試行錯誤の結果偶然得られた報酬から学習を進めていくため，報酬を得る頻度が低い環境では学習速度が遅いという問題点が指摘されている．

この問題を解決するために，階層型強化学習が提案されている．階層型強化学習のシステムでは，タスクをより小さなサブゴールに分割し，システムの下層ではサブゴールを達成するためのスキルの習得，上層では習得したスキルを使ったタスクの達成を学習させる．

報酬が貰える頻度が低く，強化学習が難しい環境の例として，Minecraft が挙げられる．Minecraft は，3D 空間上をプレイヤーが自由に行動できるサンドボックス型のゲームであり，プレイヤーに対して直接的に目的や報酬が明示されることが少ないゲームである．Minecraft 上の機械学習を支援するために， Project Malmo[1] (以下 Malmo と書く) というプラットフォームが提供されている．Malmo を使うことでエージェントが学習するためのタスクや報酬を容易に実装できる．Malmo を利用した AI エージェント作成の大会が 2017 年から毎年開かれており， 研究対象として注目が集まっている．

Minecraft を対象に階層型強化学習を適用した例として， H-DRLN [2] がある．3 つの部屋を用意し，それぞれの部屋で別々のタスクをこなして次の部屋に進んで行き，最後

の部屋のタスクをクリアした時に初めて報酬が手に入るというゲームで実験をしている．H-DRLN はこのタスクに対して，サブゴールを人間の手で設定することによって対応した．それぞれの部屋で達成するべきタスクをサブゴールに設定することで，階層型ではない学習方法と比べて高い性能を発揮したと主張している．

人手によるサブゴールの分割などを行わずに，階層ごとの学習を自律的におこなう強化学習の手法もいくつか提案されている．これらの手法は環境に関する経験的な知識を必要としないため，多くの問題に適用可能であると考えられる．

本研究では，Minecraft の世界を模した自作のグリッド世界を用いて，階層型強化学習の評価を行う．シンプルなグリッド世界を使うことで学習効率を早めるのと同時に，グリッド世界での階層型強化学習の可能性を探る．階層型強化学習の手法として近年注目されている，Option の概念を使った強化学習である Option-Critic アーキテクチャを用いて実験を行う．

本論文は以下のように構成する．まず第 2 章において強化学習についての概要をまとめる．次に第 3 章にて Minecraft における強化学習の概要についての紹介を行い，強化学習が難しい環境に対応する方法として，第 4 章にて階層型強化学習と本研究で使用する Option-Critic アーキテクチャについて紹介する．第 5 章にて本研究で使用する自作のグリッド世界について紹介し，第 6 章にてグリッド世界に対してハイパーパラメータ自動最適化ツールである Optuna の適用を行い，学習に適したモデルの探索を行う．第 7 章にて Option-Critic アーキテクチャを適用し，実験結果を記す．最後に第 8 章にて実験結果から得られた課題を考察し，今後の課題を議論する．

第 2 章

強化学習

本章では強化学習の概要について紹介した後，本研究での実験のベースラインとした A3C，LSTM の紹介を行う．強化学習 [3] は機械学習の一手法である．強化学習においては，行動の主体であるエージェントが与えられた環境の中で試行を繰り返し，より大きな報酬が貰えるように行動を最適化する．エージェントは環境の状態に基づいて行動を選択し，行動により変化した環境の状態と報酬を獲得する．エージェントは環境についての事前知識を持たないため，学習初期は何も分からずランダムに行動することになるが，何回も試すうちに偶然報酬を得ることがある．多くの報酬が得られるように試行を繰り返し行動を学習し，一連の行動の結果から得られる報酬の和が最も大きくなるように，行動を最適化する．事前知識を必要としないという特徴から，未知の環境でも適切な報酬を与えることでより大きな報酬が貰えるように学習する可能性がある．

2.1　マルコフ決定過程

エージェントが環境の全てを観測可能である場合において，その環境のダイナミクスをモデル化したものをマルコフ決定過程 (MDP) と呼ぶ．ここでは環境が取り得る状態 (state) の集合を $S = \{s_0, s_1, s_2, \ldots, s_n\}$，エージェントが取り得る行動 (action) の集合を $A = \{a_0, a_1, a_2, \ldots, a_k\}$ とする．ある状態 $s \in S$ においてエージェントが行動 $a \in A$ を選択したとき，次の状態 $s' \in S$ に状態が遷移する確率を $P(s'|s, a)$ と表し，そのときに環境から与えられる報酬 r の期待値を $R(s, a, s')$ と表す．このように，状態 s' への遷移が直前の状態 s とそのときの行動 a にのみ依存することをマルコフ性と呼ぶ．以上のように，MDP は $\{S, A, P, R\}$ の 4 要素で表される．

MDP における基本的な問題設定として，できるだけ多くの報酬をもらえるエージェントを作ることがある．将来得られる報酬の総和が最大になるように方策 π を学習させたい．そのために用いられる目的関数は $\sum_{t=0}^{\infty} \gamma^t r_{t+1}$ の期待値となる．t は時間ステップ，r は即時報酬を表す．未来の報酬をその分だけ割引するのが割引率 γ である．これは関数が発散しないためにも必要である．

エージェントが環境の全てを観測できるわけではない場合，そのような環境のダイナミクスを部分観測マルコフ決定過程 (POMDP) と呼ぶ．これは MDP を拡張したもので，エージェントの状態観測が不完全，不確実である場合の数理モデルである．

MDP は $\{S, A, P, R\}$ の 4 要素で表されるが，POMDP はさらに 2 つの要素を加えて表す．エージェントの状態観測の集合を Ω とし，エージェントが行動 $a \in A$ を選択し

て次の状態 $s' \in S$ に状態が遷移した場合に，エージェントが $\omega \in \Omega$ を観測する確率を $O(\omega|s', a)$ と表す．以上のように，POMDP は $\{S, A, P, R, \Omega, O\}$ の 6 要素で表される．

2.2　Value-Based の強化学習と Policy-Based の強化学習

ある状態の価値 $V(s)$，あるいはある状態である行動をとる価値を表す行動価値関数 $Q(s, a)$ を学習させて，それに基づいて最適な行動をおこなわせるという考え方に基づく強化学習を Value-based の強化学習と呼ぶ．特に行動価値関数 $Q(s, a)$ を学習させる手法を Q 学習と呼び，Q 学習では多数の行動を行い s, a, r, s' のサンプルを集め，それをもとに Q を更新する．

それに対して，方策 (policy) 関数 $\pi(s)$ を直接学習させる強化学習を Policy-Based の強化学習と呼ぶ．方策関数を π として，そのパラメータを θ とする．ある方策 π_θ における累積報酬和 の期待値 $E^{\pi_\theta}[R]$ を目的関数とし，これを方策のパラメータ θ で微分し勾配を得る．この θ を勾配に沿って更新する．目的関数を 累積報酬和の期待値とするため，状態遷移確率 P が分かっていない状態では目的関数を上手く表すことができないが，強化学習を進めていく中で多くの試行回数を重ねるので，確率的に勾配が分かってきて学習を進めることができる．

2.3　A3C

本研究のベースラインとして使用した強化学習アルゴリズムは Asynchronous Advantage Actor-Critic(A3C)[4] である．A3C には 3 つの考え方 (Asynchronous, Advantage, Actor-Critic) が組み込まれている．

A3C では複数のエージェントを別々のスレッドで動かして個々の経験を集め，その経験を利用して共有ネットワークを更新する．それぞれのスレッドが非同期 (Asynchronous) に共有ネットワークを更新する．マルチエージェントであるため，全体としてはランダムに経験を集めることができる．同時に多くの経験が得られるためマルチコア環境での学習に要する実時間が短いという利点がある．

Advantage とは，価値関数の推定値の更新に使われる値である．k ステップ先まで使うとしたとき， Advantage は以下のように定義される．

$$advantage = \sum_{i=0}^{k-1} \gamma^i r_{t+i} + \gamma^k V(s_{t+k}) - V(s_t) \tag{2.1}$$

γ は割引率，r は報酬，t は時間ステップ，V は価値関数を表している．価値関数の推定値 V を計算する際に，1 ステップ先だけを見る方法よりも それ以上の k ステップ先までを見たほうが収束が速くなるという考えである．Advantage の値が小さくなるように，つまり方策関数 π についての収益の予測が正確になるように， V を更新する．Advantage という量は，k ステップ先の報酬まで考慮した，より確からしい価値の推定誤差と言える．

Actor-Critic は Value-based と Policy-based を組み合わせた考え方で，Actor とは各行動を起こす確率を求める方策関数 π であり，Critic とは状態の価値を推定する価値関数 V のことである．これら 2 つの関数を独立して定義し，同時に学習させる．この 2 つが独立していることによって，連続的な行動でも学習させやすいという利点がある．

2.4　様々な環境における強化学習

強化学習の研究で題材としてよく使われる環境として，1977 年に米国で発売された atari 2600 というゲーム，物理エンジン環境である MuJoCo[5] 等が挙げられる．A3C[4] の研究では，アルゴリズムの評価に atari ゲームを利用しており，多くの atari ゲームで良い成果を残している．

A3C で上手く学習ができるゲームの例として，以下の atari ゲームの Breakout(ブロック崩し，図 2.1)，SpaceInvaders(スペースインベーダー，図 2.2) がある．[*1]．Breakout はブロックを崩すたび，SpaceInvaders は敵を撃破するたびに報酬が貰える．ランダムに動いても報酬を貰えることが多い環境であるため，A3C に限らず強化学習に向いている環境だと言える．

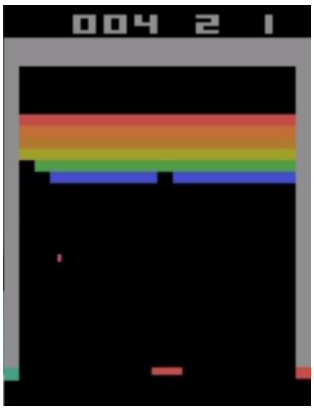

図 2.1　Breakout

図 2.2　SpaceInvaders

A3C で上手く学習ができないゲームの例として，atari ゲームでは図 2.3 の Montezuma's Revenge が挙げられる．図 2.3 の中央にいるエージェントは鍵を入手して上の左右のどちらかの扉に到達するのが目標で，鍵を入手したときと，それを持って扉に到達したときに報酬を貰える．ただそこまで行くのが大変であり，鍵を入手するには，はしごを降り，ロープにつかまり，右のはしごを降り，髑髏をかわして左に進み，はしごを登って手に入れる．その後は同じ道を逆に辿って，扉に到達する必要がある．報酬がなかなか貰えない上に死にやすい環境であるという点と，いきなり扉に行くのではなく鍵を拾ってからではないといけないという遅延報酬であるという点の 2 点の理由で学習が難しいことになる．

A3C に限らず，試行錯誤をして偶然得た報酬から学習を進めるという強化学習の考え方では，このような報酬が貰える頻度が低い環境 (報酬がスパースな環境) では学習が難しい．何も分からずにランダムに動く学習初期に滅多に報酬が貰えないようでは学習のしようが無いからである．Montezuma's Revenge に限らず，このような問題のある環境では強化学習が難しいという問題がある．

様々な研究で atari 環境が使われているが，本研究では POMDP における Option-Critic アーキテクチャを試すため，自作のグリッド世界を用いて実験を行った．

[*1] 図は https://gym.openai.com/ より引用

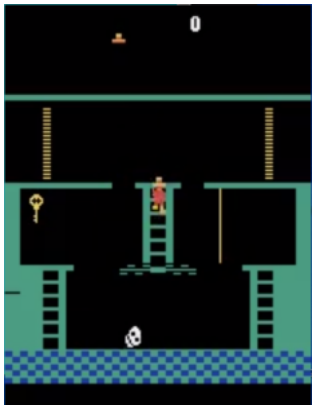

図 2.3　Montezuma's Revenge

2.5　LSTM

本研究の POMDP のタスクにおいて，利用する手法に Long short-term memory(LSTM)[6] がある．LSTM とは，時系列データを学習可能な RNN の拡張であり，長期的な記憶を可能にしたモデルである．通常の RNN(Recurrent Neural Network) ではニューラルネットワークのループにより過去のデータを次回の入力にすることで短期的な時系列データを扱うことができる．しかし，長期的な時系列データを扱おうとすると勾配の消失・発散が起こるため，扱うことが難しい．そこで，LSTM ではループ時の重みを 1 にすることでデータを記憶し続けることで，長期的な記憶を可能にした．これによって起きるデータの衝突等の問題を入力ゲートと出力ゲートを設置することで解決し，状態遷移等の理由で保持する価値が低くなったデータを記憶し続けてしまう問題を，忘却ゲート [7] を設置することで解決した．

A3C は多くの atari ゲームで高いスコアを記録したが，A3C に LSTM 層を加えたものがより高いスコアを記録した．

強化学習に LSTM を導入したアルゴリズムは，部分観測マルコフ仮定 (POMDP) の学習に向いている．POMDP の難しさはエージェントが環境の部分的な観測しかできず，現在の観測と過去の観測は同じとは限らないという点である．しかし，LSTM は長期的な記憶を扱うことができるため，過去の観測データを学習に使うことができるため，POMDP の学習に向いているといえる．

第 3 章

Minecraft の世界での強化学習

本章では, Minecraft 上で強化学習の研究をするためのプラットフォームである Project Malmo とその関連研究について紹介する．また，著者が参加した Project Malmo を用いた AI エージェント作成の大会について紹介する．

3.1 Minecraft

Minecraft は 2011 年に発売されたサンドボックス型のゲームであり，2019 年 5 月時点で 1 億 7,600 万本もの売り上げを達成したコンピューターゲームである．プレイヤーは一人称視点で，図 3.1 のような主に立方体のブロックで構成された世界を冒険する．プレイヤーは世界の構成要素であるブロックを自由に破壊してアイテムとして入手、また持っているアイテムをブロックとして設置することができる．世界には森，草原，海，砂漠等の様々な環境が存在し，プレイヤーは広大な世界の中で自由に行動する．また，持っているアイテムを組み合わせて新たなアイテムを作ることをクラフトと呼び，クラフトで便利な道具や装置等を作ることでより良い生活ができるようになっていく．1 つの世界を 2 人以上のプレイヤーで遊ぶマルチプレイに対応しており，サーバーの計算能力次第では数百人の同時接続も可能である．クラフトや建築等のものづくり，未知の場所への冒険，武器を手にして敵と戦うアクション，マルチプレイ等，様々な要素があるゲームであり，それら全てが詰まった世界で好きに遊べる自由度の高さが Minecraft の面白さである．

3.2 Project Malmo

Minecraft には，ゲームに様々な要素を追加する拡張である mod を導入して遊ぶことができる．例えば，操作を便利にするもの，画質を綺麗にするもの，大量のアイテムを追加して科学や工業の概念を与えるもの等，多種多様の mod が存在する．Minecraft は java で書かれたゲームであり，mod も基本的に java で開発される．個人が開発した mod を自由に公開することができる．

Project Malmo[1] (以下 Malmo) は Microsoft Research が提供している，Minecraft 上で強化学習の研究をするための OpneAI ベースの python モジュールであり，mod として提供されている．Minecraft はサンドボックス型のゲームであり，基本的に決まった目標や報酬が存在しないが，Malmo を使うことでそれらを設定することができるようになる． つまり，Minecraft 上で強化学習の題材となるタスクを簡単に作成できる．

図 3.1　Minecraft の世界 (Minecraft 公式 https://www.minecraft.net/ja-jp/ より引用)

Minecraft の特性として，多種多様なブロックを自由に設置できること，また一人称視点のプレイヤーが自然界を模したオープンワールドで行動すること，マルチプレイが可能であることがある．これらの特性から，3D 空間で多様なタスクを設定できること，現実世界に寄せた環境で実験ができること，マルチエージェントの実験ができることが Project Malmo の主な特徴として挙げられる．

3.3　MARLO 2018

Malmo を利用した強化学習エージェント作成の大会が 2017 年から毎年開かれており，研究対象としての Minecraft に注目が集まっている．MARLO 2018[8] (以下 MARLO) は 2018 年度に開催されたものである．この大会では，Minecraft の世界で作られた 3 つのマルチエージェント向けのタスクを与えられ、3 つのタスク全てに対応する学習済みのマルチタスクエージェントを提出する．以下で 3 つの環境を図と共に紹介する．

Build Battle

図　3.2　Build　Battle　(MARLO　2018　の　ド　キ　ュ　メ　ン　ト https://marlo.readthedocs.io/en/latest/available_envs.html より引用)

1 つの部屋の中で 2 人のエージェントが行動する．部屋には完成された直方体と，作りかけの直方体の 2 つの構造物が存在する．エージェントの目標は作りかけの直方体にブロックを設置して完成された直方体と同じ構造物を作ることである．エージェントは前後に移動，左右に回転，ブロックの設置または破壊の行動を取ることができる．左右の回転は 1 度の行動で 45°回転する．正しい場所にブロックを置く，または間違えた場所にあるブロックを破壊することで +0.2, 間違えた場所にブロックを置く，または正しい場所のブロックを破壊することで-0.2 の報酬を得る．また，直方体を完成させることで +1 の報酬を得る．2 人のエージェントが協力して直方体を完成させることが求められる．

Mob Chase

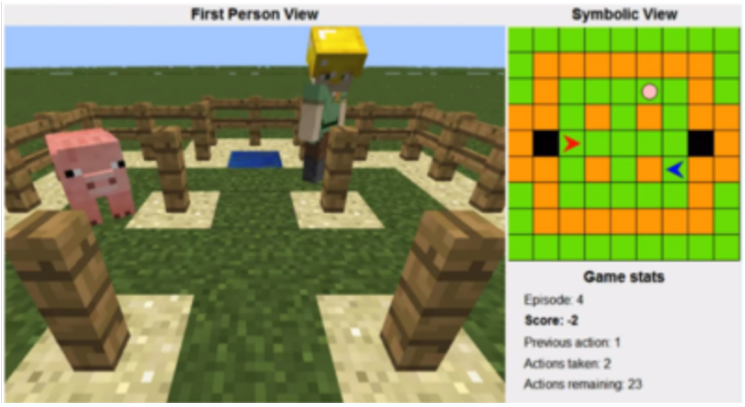

図　3.3 Mob Chase (MARLO 2018 のドキュメント https://www.crowdai.org/challenges/marlo-2018 より引用)

柵で囲まれた狭い区画の中で 2 人のエージェントが行動する．区画にはエージェントの他に Mob が存在し，動き回る Mob を 2 人のエージェントで挟み込んで動けなくするのが目標である．マップ両端にゴールが存在し，そこに触れることでギブアップして微量の報酬を獲得することもできる．エージェントは前後に移動，左右に回転をすることができる．Mob を挟み込めば +1 の報酬，ゴールにたどり着けば +0.2 の報酬を獲得する．

Treasure Hunt

1 つの部屋の中で 2 人のエージェントが行動する．部屋には宝物，ゴール，障害となる溶岩，敵性 Mob が配置されている．敵性 Mob の攻撃を数回受けるか，溶岩に落ちることでエージェントは死んでしまう．このタスクでは 2 人のエージェントに別々の役割があり，宝物を集めてゴールする Collector と　Collector を守るために敵性 Mob と戦う Protector の 2 つがある．Collector のみが宝物を拾うことが可能で，Protector のみが敵と戦うための装備を持っている．また，どちらのエージェントも溶岩に落ちることを避けつつ行動しなければならない．宝物を持った Collector がゴールにたどり着くことが目標である．エージェントは前後に移動，左右に回転することができ，Collector はそれに加えてアイテムを拾う，Protector は前方に攻撃することができる．宝物を拾うと +0.25，その後ゴールにたどり着くと +0.5，死ぬと-1 の報酬を得る．

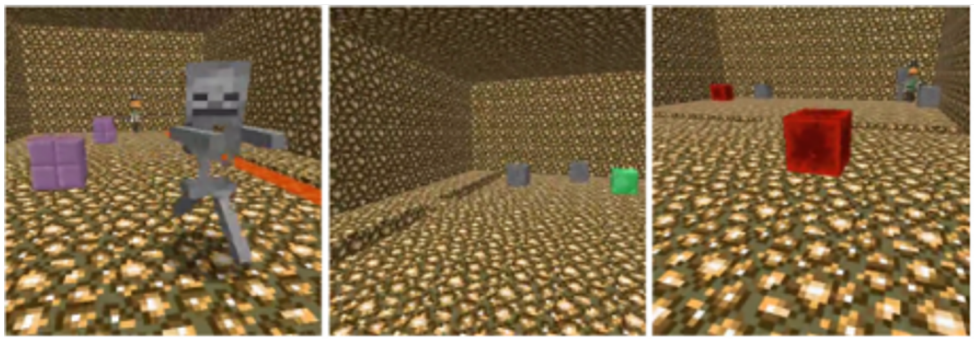

図　3.4 Treasure Hunt ((MARLO 2018 のドキュメント https://marlo.readthedocs.io/en/latest/available_envs.html より引用))

どのタスクでも 1 ステップごとに-0.02 の微量の負の報酬を与え，1 エピソードは最大 50 ステップとしている．タスクを少し変化させるパラメータがいくつか用意されており，様々な環境での学習が可能になっている．例えば Build Battle の直方体の大きさ変更，Mob Chase の Mob の見た目変更等ができる．この大会ではこれらのタスクを用いてエージェントを学習させ，パラメータの詳細が知らされていない本番環境での 3 つのタスクの合計スコアを競う．

3.3.1　著者のアプローチと結果

それぞれのタスクで 選択可能な action 数が違うことから全てのタスクを同じネットワークで学習させるのは難しいと判断し，action 数ごとに別々のネットワークを用意するような DQN[9] を試し，その時点でオンラインの一次予選で 1 位になった．しかし，実際に選ばれている action を分析したところ，全て「何もしない」action が選択されていた．そこで，どんな状況でも「何もしない」エージェントを提出してみた結果，結果を更新し，オンラインの一次予選が 1 位，その後の途中段階で行われた評価では 3 位で入賞し，2500 $ の travel grant を獲得した．このエージェントでは決勝には進めなかったため，決勝での入選はできなかった．

3.3.2　入賞者のアプローチ

決勝で 2 位に入選した Xu ら [10] は，著者と同じようにタスクごとに別々のネットワークを用意して学習を行った．著者との違いは，アルゴリズムに PPO[11] を用いたこと，タスクごとに Xu らの知識をもとにした外部報酬を設定して学習を進みやすくしたこと，タスクの判別を専用のネットワークで学習させたことである．ゲームが始まってすぐにどのタスクなのかを画像から判別し，それぞれのネットワークでどの行動を取るべきかを出力する．このような階層的なモデルで 2 位に入賞している．

3.4　H-DRQN

MARLO 2018 以外でも，Minecraft の環境で階層的なアプローチが有効であることを示したのが H-DRQN[2] である．

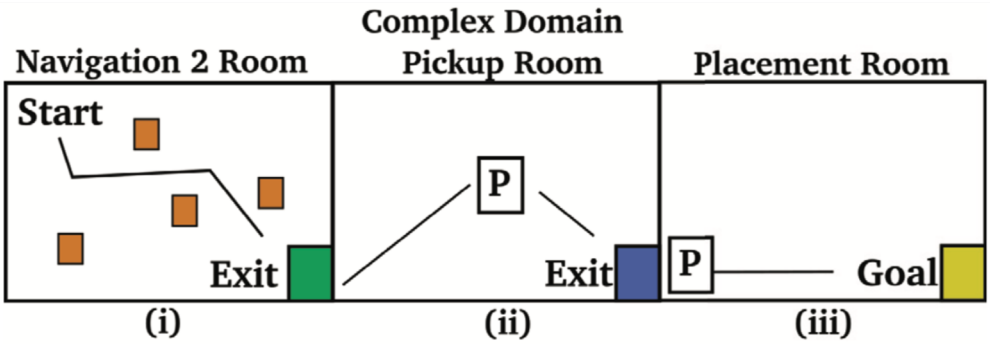

図 3.5　H-DRQN でのタスク ([2] より引用)

図 3.5 は H-DRQN で使用した Minecraft のタスクを上から見た図である．エージェントは 3 つの部屋全てのタスクをクリアする必要がある．それぞれの部屋では別々のタスクが設定されている．最初の部屋では障害物を避けて出口に向かう．次の部屋では中央に置かれたブロックを破壊して入手，その後出口を塞ぐブロックを破壊しなければならない．最後の部屋では，前の部屋で入手したブロックをゴールに設置する．エージェントはこれら全てを達成して初めて正の報酬を獲得する．

このような報酬が滅多に貰えないタスクに対して，Tessler らはそれぞれの部屋ごとのタスクを個別のネットワークで予め学習させ，全体タスクの学習において学習済みネットワークからの出力と通常の action の両方から行動選択をするようにした．学習アルゴリズムには DQN を用いており，この階層化を行ったモデルを特に H-DRQN と呼んだ．その結果，階層化を行わない DQN では一切学習が進まなかったのに対して，階層化を行った H-DRQN ではタスクをクリアすることができるまで学習した．

このことから報酬が獲得しづらい環境においても，人間の知識を用いて正しく階層化をすることができれば，強化学習が可能だと言える．

第 4 章

階層型強化学習

本章では，報酬が獲得しづらい環境における強化学習の手法として前章で触れた，階層型強化学習について紹介する．その後，本研究で使用した Option-Critic アーキテクチャについて紹介する．

4.1　階層型強化学習の概要

報酬がスパースな環境では学習が遅いという問題を解決するために，階層型強化学習が提案されている．まず通常の強化学習では，図 4.1 のように与えられたタスクを直接行動に分割する．ランダムに動いても高頻度で報酬が与えられるようなシンプルな環境ではこれで学習が可能かもしれないが，報酬がスパースな環境では意味のある行動の並びを学習させる必要があり，偶然そのような並びになることは少ないため，なかなか報酬が貰えないことになる．報酬が貰える頻度が少ないということは，その分学習が遅くなることを意味する．

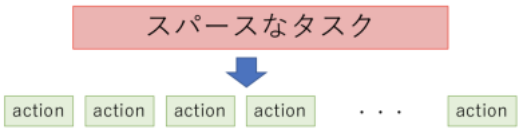

図 4.1　強化学習のイメージ

階層型強化学習のシステムは図 4.2 のような構造をしている．タスクをサブゴールに分割し，サブゴールを達成することを目標にプリミティブな行動の並びを学習させる．この行動の並びをスキルと呼ぶ．そして，事前に学習済みのスキルを使って全体のタスクを学習させる．前節の Montezuma's Revenge の例では，鍵を取ることを全体タスクとして，はしごを降りる，はしごを登る，ロープにつかまる，左に進みながら髑髏を避ける等の，十分学習できそうな目標をサブゴールとして設定することが考えられる．

階層型強化学習ではシステムの上層と下層で異なるレベルで学習を進めることになる．下層は従来通りの強化学習だが，上層では下層で学習した行動の並びを使って学習させるため，よりハイレベルな学習ができる．そのため，プリミティブな行動だけでは達成が難しい，または達成に時間がかかるような目標を効率的に達成できる可能性がある．また，下層で学習したスキルを他のタスクにも使いまわせる可能性を考えると，マルチタスクに対応できる汎用性も期待できる．例えば，Montezuma's Revenge の第 1 ステージの下層

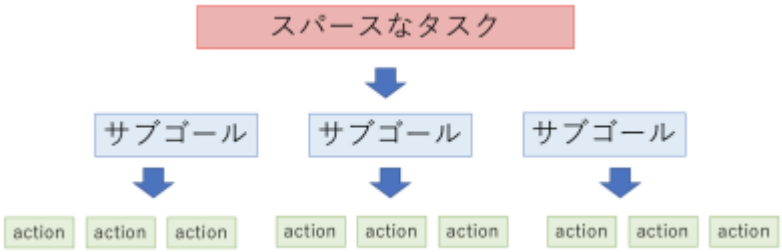

図 4.2　階層型強化学習のイメージ

の学習で”はしごを降りる”というスキルを獲得したとすると，別のステージの上層の学習でも使いまわせる可能性があるということである．

階層型強化学習に成功した例として，前章で紹介した H-DRQN[2] が挙げられる．これは Minecraft の世界を題材にスパースな環境で実験をしている．タスクを人の手で階層化し，それぞれのサブゴールを独立して学習させ，それを用いて全体のタスクを学習させることに成功した．

強化学習の特徴として，事前知識を必要としないため多くの問題に適用できるという強みがある．しかし，人の手でサブゴールを設定する方法はそのタスクの事前知識を用いているため，狭い範囲の問題にしか適用できない．そこで，本研究では人の手によるタスクの階層化ではなく，階層ごとの学習を自律的におこなう強化学習の手法である Option-Critic の実験を行う．

4.2　Option-Critic アーキテクチャ

1999 年に sutton らによって Option[12] という概念が提示された．Option とは前節で述べた階層型強化学習のサブゴール達成を学習して得られるような”行動の並び”，”スキル”，に対応する．近年，Option を利用した強化学習のモデルである Option-Critic アーキテクチャが提案された．Option-Critic アーキテクチャ [13] とは Actor-Critic をベースにして Option を利用する階層型強化学習の手法である．

Option-Critic アーキテクチャにおいて，Option $\omega \in \Omega$ は $(I_\omega, \pi_{\omega,\theta}, \beta_{\omega,\vartheta})$ の 3 つ組に対応する．$I_\omega \subseteq S$ は Option 開始時の状態 s，$\pi_{\omega,\theta}$ は Option の方策，$\beta_{\omega,\vartheta}$ は終了条件の関数である．θ はどの Option を選ぶべきかを出力する Option ポリシー π_Ω のパラメータであり，ϑ は Option の終了条件関数のパラメータである．これらのパラメータを勾配によって最適化する．

Option-Critic アーキテクチャは，固有数の Option をスイッチして行動を選択するアルゴリズムである．図 4.3 は Option-Critic アーキテクチャの構造図である．状態 s_t から Option ポリシー π_Ω により Option ω_t が選択され，ω_t の方策 π_ω により行動 a_t を決定する．ここで，状態 s で Option ω を選ぶ価値を Option の行動価値関数と呼び，$Q_\Omega(s,\omega)$ と表す．この値が大きい Option を選択する．Option の行動価値関数については以下のような式で表される．

$$Q_\Omega(s,\omega) = \sum_a \pi_{\omega,\theta}(a|s) Q_U(s,\omega,a) \tag{4.1}$$

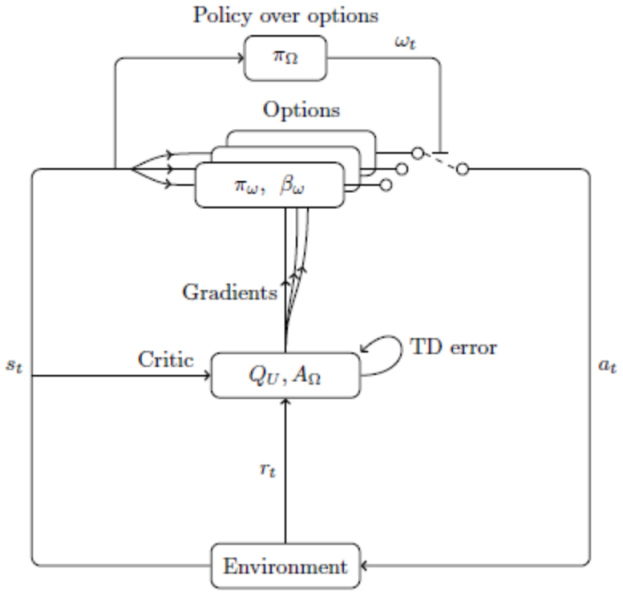

図 4.3　Option-Critic アーキテクチャの構造 ([13] より引用)

$$Q_U(s,\omega,a) = r(s,a) + \gamma \sum_{s'} P(s'\ |s,a) U(\omega, s') \tag{4.2}$$

$$U(\omega, s') = (1 - \beta_{\omega,\vartheta}(s'))Q_\Omega(s', \omega) + \beta_{\omega,\vartheta}(s')V_\Omega(s') \tag{4.3}$$

$U(\omega, s')$ は Option ω を実行したときに s' に入る価値である．$P(s'\ |s,a)$ は a によって状態が s から s' へ移される確率である．$V_Omega(s')$ とは，パラメータ θ，ϑ の勾配は，行動価値関数 4.1 式より，それぞれを偏微分することによって求める．つまり Option-Critic アーキテクチャでは，状態 s_t から求まる Critic である行動価値関数 4.1 式から，Option ポリシー π_Ω と Option の終了条件関数である β_ω それぞれの勾配を求めて学習を進める．

Option の終了条件である β_ω の理想的な値とは，人間が見てサブゴールが切り替わるべきタイミングで Option が切り替わるような β_ω である．そのような β_ω を学習することを期待している．

本研究では，既存研究 [14] で紹介されている A3C ベースの Option-Critic アーキテクチャである Asynchronous Advantage Option-Critic (A2OC) を実装し，実験を行った．図 4.2 が A2OC のアルゴリズムである[*2]．ここで，η は Option が切り替わったときに受け取る微量の報酬であり，本研究では全て 0 で実験を行った．t は時間ステップであり，t-max ステップ 分の勾配を一度に更新する．現在の Option の方策に基づいて行動を行い報酬を蓄積し，もし t-max ステップ分に達するか Option が終了したときに，蓄積された報酬を用いて価値関数を推定し，パラメータの勾配を求めて学習を進める．

[*2] [14] より引用

A2OC は atari ゲームを用いて評価された．その結果，いくつかの atari ゲームにおいて，A3C を超えるスコアを出している．

Algorithm 1: Asynchronous Advantage Option-Critic

Initialize global counter $T = 1$
Initialize thread counter $t = 1$
$c = 0$
repeat
  $t_{start} = t$
  $s_t = s_0$
  Reset gradients: $dw = 0$, $d\theta_\beta = 0$ and $d\theta_\pi = 0$
  Choose o_t with an ϵ-soft policy over options $\mu(s_t)$
  repeat
    Choose a_t according to $\pi_\theta(\cdot|s_t)$
    Take action a_t in s_t, observe r_t, s_{t+1}
    $\widetilde{r_t} = r_t + c_t$
    if *the current option* o_t *terminates in* s_{t+1} **then**
      choose new o_{t+1} with ϵ-soft$(\mu(s_{t+1}))$
      $c_t = \eta$
    else
      $c_t = 0$
    end
    $t = t + 1$
    $T = T + 1$
  until *episode ends or* $t - t_{start} == t_{max}$ *or* *(*$t \quad t_{start} > t_{min}$ *and* o_t *terminated)*
  $G = V_\theta(s_t)$
  for $k \in t-1, ..., t_{start}$ **do**
    $G = \widetilde{r}_k + \gamma G$
    Accumulate thread specific gradients:
    $dw \mathrel{-}= \alpha_w \frac{\partial (G - Q_\theta(s_k, o_k))^2}{\partial w}$
    $d\theta_\pi \mathrel{+}= \alpha_{\theta_\pi} \frac{\partial \log \pi_\theta(a_k|s_k)}{\partial \theta_\pi} (G - Q_\theta(s_k, o_k))$
    $d\theta_\beta \mathrel{-}=$
    $\alpha_{\theta_\beta} \frac{\partial \beta_\theta(s_{k+1}, o_k)}{\partial \theta_\beta} (Q_\theta(s_{k+1}, o_k) - V_\theta(s_{k+1}) + \eta)$
  end
  Update global parameters with thread gradients
until $t > t_{\max}$

第 5 章

グリッド世界の作成

本章では自作のグリッド世界の紹介を行う．既存研究 [2] より，Minecraft 上での階層型強化学習は効果があることが知られている．そこで，Minecraft で機械学習をするためのプラットフォームである Malmo[1] のタスクを模したグリッド世界を作成した．シンプルな世界を使うことで学習効率を早めることが期待できる．グリッド世界での Option-Critic アーキテクチャの可能性を探る．

グリッド世界は強化学習の環境として広く使われている OpenAI Gym[15] のフレームワークで実装する．以下では作成した 2 種類のグリッド世界の環境 one-room タスク，four-rooms タスク について説明する．

5.1　one-room タスク

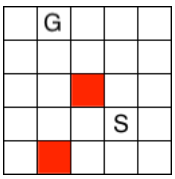

図 5.1　one-room

図 5.1 のような $N \times N$(図では N=5) の正方形のグリッド世界を用意し， 指定した確率 (図では 20%) で床に穴を開ける．穴を開けた結果，上下左右の 4 近傍について連結でない床ができてしまったら，穴の生成をやり直す．続いて，残っている床からランダムに異なる位置をスタート位置とゴール位置として選択する．スタートとゴールの位置，穴の位置についてはエピソード毎にランダムに決定する．エージェントは環境から自身の位置とゴールの位置，穴の有無を環境から観測として受け取る．行動は上下左右への移動の 4 通りで，ゴールにたどり着くと +100 の報酬を得る．step 毎に −1 の報酬を獲得し，穴に落ちる，または場外に出ることで床から外れると −10 の報酬を獲得する．エピソードはゴールにたどり着く，床から外れる，400 ステップ行動する のいずれかで終了する．図 5.1 では S がスタート，G がゴール，赤く塗りつぶされたマスが穴を表している． この例では 5 ステップでゴールにたどり着くのが最短であるので，報酬の最大値は 95 となる．

この環境ではエピソードごとにスタート，ゴール，穴の位置がランダムにそれぞれ決まるため，テーブルベースの強化学習は使えない．関数ベースの強化学習が必要となるが，

その中でも特に A3C 等のニューラルネットを用いたモデルを用いて方策関数と価値関数を構成する強化学習の手法を用いることを想定している．

作成したグリッド世界の環境は，パラメータにより環境を少し変化させた実験が可能であり，本研究で使用したものにも，グリッドサイズの変更，穴を開けない等の様々な変更が可能であり，多様な実験ができるように実装した．エージェントが観測する情報としては $N \times N$ のグリッドに 3 プレーン (床の有無，プレイヤーの有無，ゴールの有無) が与えられる．

5.2　既存研究における four-rooms タスク

four-rooms タスクは，option を使ったプランニング (planning) のために 1999 年に Sutton らが導入したタスクである [12]．図 5.2 が Sutton らが紹介した four-rooms である．

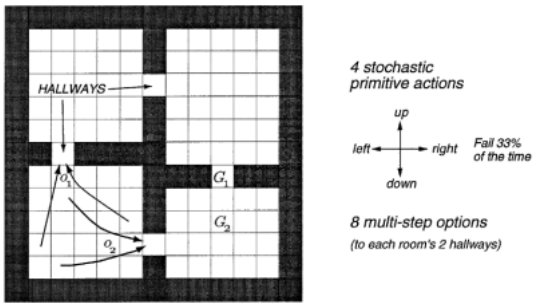

図 5.2　オリジナルの four rooms タスク ([12] より引用)

このタスクの目標はスタートからゴールに到達することである．エージェントは上下左右の 4 通りの 行動 を選択するが，選択したとおりの方向に進むのは 2/3 の確率で，残りの 1/3 の確率でそれ以外の 3 つの方向にランダムに進む．図中で黒く表されている外周および 4 つの部屋を仕切っているマスは壁であり，壁に進む行動を選択するとその場にとどまる．

Sutton らはこの four-rooms タスクを option を使ったプランニングの題材として紹介しており，4 部屋それぞれに 2 つの通路に向かうための最短経路を辿る built-in option を手動で用意することで，プランニングの実験をしている．それぞれの Option の終了条件 $\beta(s)$ は s がその部屋の内側にいる状態の時に 0，その部屋の外側にいる状態の時に 1 になるように設定する．結果，ゴールが通路にあるときは option のみで効率の良いプランニングができ，部屋にあるときは Option とプリミティブな 行動 を組み合わせることで効率の良いプランニングができたと主張している．

以降，four-rooms タスク は Option 等の階層的なアイデアに関する研究題材として利用されてきた．元々の four-rooms タスクの環境に少し手を加えたものを ”four-rooms” として実験の題材に使うことが多い．例えば階層型強化学習の既存研究 [16] では単純にスタートからゴールに行くのではなく，中間ゴールを経由しなければならず，さらにぶつかると負の報酬を獲得する障害物を置いて複雑な環境にした four-rooms タスクを題材としている．

5.3　本研究で扱う four-rooms タスク

本研究では前節の four-rooms タスクをベースに深層階層型強化学習の評価を目的としたグリッド世界環境を提案する．

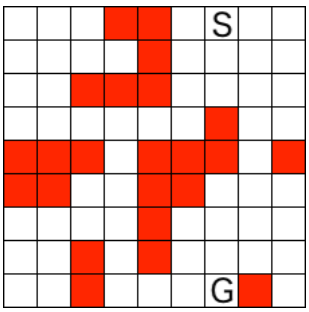

図 5.3　four-rooms-easy

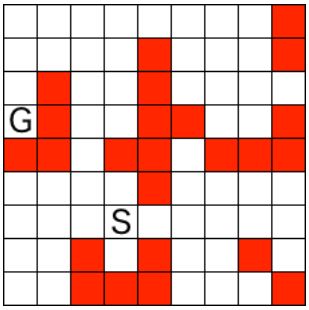

図 5.4　通路が封鎖された four-rooms

グリッド世界は図 5.3 のように 4 つの $N \times N$ (図では $N = 4$) の部屋で構成され，$(2N + 1) \times (2N + 1)$ のサイズである．それぞれの部屋は壁の代わりに穴で十字に仕切られていて，部屋を移動するためには幅 1 マスの通路を通る必要がある．この通路の場所はランダムな位置に配置される．また，十字に仕切られる場所は常に同じであるので，それぞれの部屋の大きさは変わらない．スタートとゴールの場所，十字以外の穴が空く確率，行動，報酬の与え方等の条件は one-room タスクと同様である．通路自体が穴になることは無いが，図 5.4 のように通路付近のマスが穴になることによって事実上通路が塞がれることはある．しかし，one-room タスクと同様にすべての床が上下左右の 4 近傍について連結であるような穴しか許容しないので，ゴールにたどり着くことができない世界は生成されない．

本研究で提案する four-rooms タスクはエージェントが環境の全てを観測できない POMDP 環境である．エージェントは現在自分がいる部屋の $(N + 1) \times (N + 1)$ の部分の情報だけを観測できるので，スタート地点からゴールが観測可能であるとは限らない．そのため，ゴールが観測できない場合は通路に向かう行動を学習する必要があるが，報酬が貰えるのはゴールに到達したときのみであり，通路に到達しても報酬は無い．エージェントの観測情報としては，例えば図 7.6 の場合では，それぞれの部屋にエージェントがいる場合，それぞれ図 5.5 5.6 5.7 5.8 のようになっており，通路のマスに到達したときに

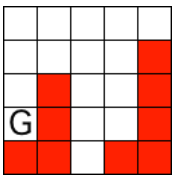

図 5.5　左上にいるときの観測情報

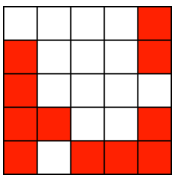

図 5.6　右上にいるときの観測情報

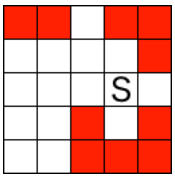

図 5.7　左下にいるときの観測情報

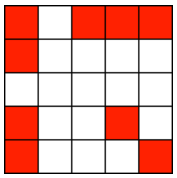

図 5.8　右下にいるときの観測情報

観測する部屋が変わる．左上から右上の部屋に向かう場合，通路のマスに到達したときに観測が左上から右上に変化する (前のステップでいた部屋でない部屋が観測される) ．逆に右上から左上に向かう場合，通路のマスに到達したときに観測が右上から左上に変化する．エージェントは，観測される情報から穴の位置を見て自分がどの部屋にいるのか特定することができる．Option-Critic エージェントが，部屋ごとの戦略となる Option を選択することで上手く学習することを期待してこのような設定にしている．

第 6 章

パラメータの探索

Optuna[17] というハイパーパラメータ自動最適化ツールを用いることで，学習モデルを決定する．前章で提案したグリッド世界は，深層ニューラルネットワークで扱うことを想定している．深層強化学習では学習対象の深層ニューラルネットワークのモデルの選択と，強化学習の手法の両方が重要であるが，本研究は後者を主な対象にしているため，前者については強化学習ではなく，教師あり学習で適切に学習できるようなモデルをあらかじめ求めることにする．

モデルはネットワークの種類も含めたハイパーパラメータで選択されるが，このハイパーパラメータの選択にハイパーパラメータ自動最適化ツールである Optuna[17] を用いる．

また，ニューラルネットワークの実装には Chainer[18] を用いる．これは次章以降で深層強化学習のフレームワークとして，ChainerRL[19] を用いるためである．

6.1 Optuna の適用

ハイパーパラメータとは機械学習の挙動を制御するパラメータであり，機械学習を実行する際に人間が一般に手動で決める．ハイパーパラメータの調整は機械学習アルゴリズムの性能を大きく左右するため，性能の良いパラメータを得るために手動で試行錯誤をするのが一般的である．

Optuna とは，[17] で紹介されたハイパーパラメータ自動最適化ツールである．Optuna を用いて，教師あり学習を何回も実行しながらハイパーパラメータの試行錯誤を自動で行う．試行錯誤をするハイパーパラメータとその探索範囲を指定することで，モデルが良い結果を出したときのパラメータを自動で得ることができる．

6.2 実験

どのようなモデルが良いかを確認するために，強化学習の前の準備として，方策関数のみに対して教師あり学習を行った．環境は one-room タスクのみを使用した．入力としては 5×5 のグリッドに 3 プレーン (床の有無，プレイヤーの有無，ゴールの有無) を与え，方策関数の出力の教師データは最短となる方向への動きを 1 で残りを 0 とする行動数の次元のベクトルとしている．Optuna を用いて, 隠れ層の数，ユニット数，学習率などのパラメータの範囲や， 最適化アルゴリズムを何種類か選択肢として入力し，Define by run

で教師あり学習を実行し，最適なパラメータやアルゴリズムの種類を入力された範囲内から導き出した． これを用いてモデルを構築した．

表 6.1　Optuna での最適化

パラメータの種類	探索範囲	結果
layer_type	MLP or CNN	CNN
n_layers	1 〜 3	3
n_output_channels	16 〜 256	136
n_channels_cnn	16 〜 128	128
optimizer	sgd or Adam or momentumSGD	sgd
sgd_lr	1e-3 〜 1e-1	0.09315239055341833
momentun_lr	1e-5 〜 1e-1	
adam_final_lr	1e-2 〜 0.2	

表 6.2　Optuna を使って得られたモデル

層の種類	パラメータ
Conv	入力 3, 出力 128, カーネルサイズ 3, ストライド 1
ReLU	
Conv	入力 128, 出力 128, カーネルサイズ 3, ストライド 1
ReLU	
Conv	入力 128, 出力 128, カーネルサイズ 3, ストライド 1
ReLU	
FC	出力サイズ 136
FC	出力サイズ 4

表 6.1 は最適化したパラメータの種類，探索範囲，結果を示しており[*3]， 表 6.2 はその結果を用いて構築されたモデルを表している． 教師であるサンプル (状態と最適行動の組み合わせ) の数が 50 万で，このパラメータでの accuracy は 99.7% にまで到達した．

[*3] 学習率，最適化アルゴリズムは次章以降使わない．

第 7 章

階層型強化学習のグリッド世界への適用

本章では，作成した環境を用いて強化学習の実験をする．A3C，A3C+LSTM， A2OC の 3 通りのアルゴリズムで実験を行い，グリッド世界における Option-Critic アーキテクチャの可能性を探る．

7.1 one-room タスクでの実験

7.1.1 実験条件

まずは，前章の教師あり学習で得たモデルを使って強化学習でも適切に学習できることを示すために one-room タスクで実験を行う．本研究では，A3C，A3C+LSTM，A2OC の 3 通りのアルゴリズムで実験を行う．A3C と LSTM は ChainerRL にエージェントが含まれているが，A2OC については含まれていないため，論文 [14] および，著者自身による Theano 用の実装[*4]を参考に ChainerRL 用に移植した．

本研究で使うニューラルネットワークのモデルは，前章で Optuna を利用して 5×5 の one-room タスクで作成したヘッドを使う．それぞれのアルゴリズムについて，図 7.1，図 7.2 のようなモデルとパラメータを用いて実験を行った．

最初の実験では，他のサイズのグリッド世界でも機能するかどうかを確かめるため，A3C を 5×5 と 8×8 の one-room タスクで動かして結果を比較する．5×5 での実験は全て 100 万ステップまで学習させるが，8×8 は 500 万ステップまで学習させる．1 万ステップごとに 10 エピソードの評価を行う．

次の実験では，5×5 の one-room タスクで A3C， A3C+LSTM， A2OC の 3 通りのアルゴリズムを適用して比較する．

7.1.2 実験結果

A3C の 5×5 の環境と 8×8 の環境での実験結果を図 7.3 に示す．横軸がステップ 数で縦軸がスコアを表している．ゴールにたどり着くと $+100$ で 1 ステップ ごとに -1 の報酬が与えられるので，最適な行動を選択した時に得られるスコアは穴がない場合は 5×5

[*4] https://github.com/jeanharb/a2oc_delib

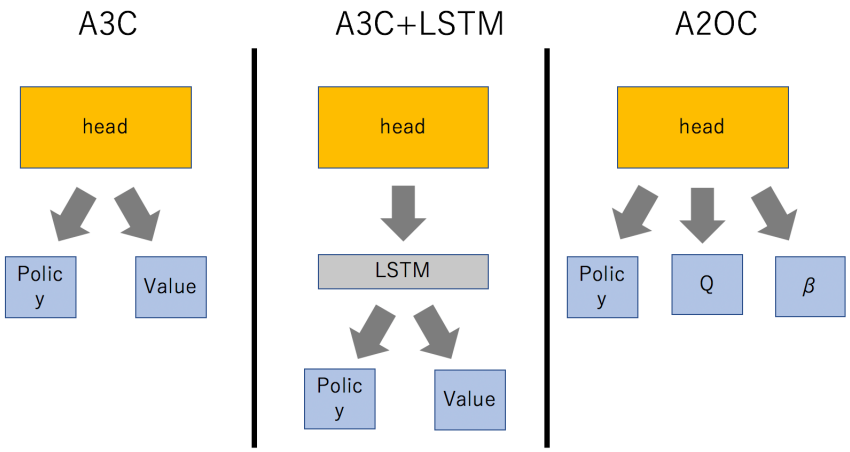

図 7.1　実験に使用したモデル

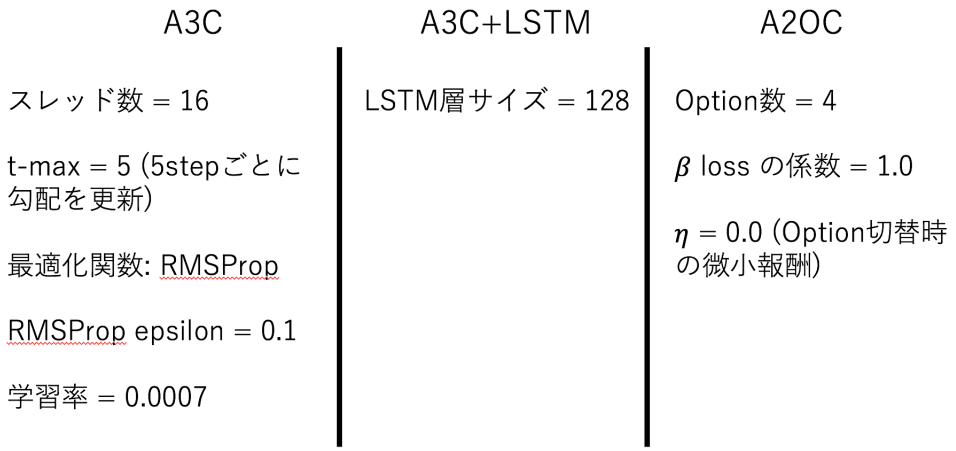

図 7.2　実験に使用したパラメータ

の環境で $92-99$, 8×8 の環境で $86-99$ である．5×5 でも 8×8 でも理想のスコア付近で安定したので，上手く学習ができたと言っていいだろう．8×8 はグリッド世界が広いので，その分学習に時間がかかっている．

次に，5×5 の one-room タスクにおける A3C，A3C+LSTM，A2OC の 3 通りのアルゴリズムの実験結果が図 7.4 である．収束するまでのステップ数が A3C で小さい傾向があるものの，最終的なスコアはどのアルゴリズムも同じような結果を得た．LSTM による記憶や Option による戦略の選択が有利に働く環境では無いため，同じような結果になったと考えられる．

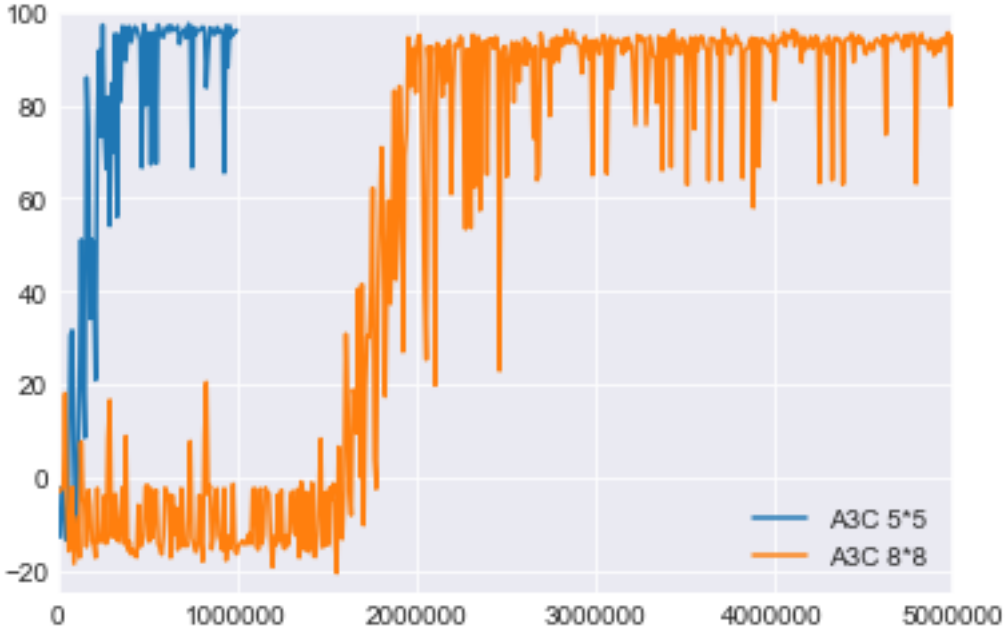

図 7.3　one-room のグリッドサイズを変更した実験

図 7.4　one-room におけるアルゴリズム比較

7.2　four-rooms タスクでの実験

7.2.1　実験条件

続いて four-rooms タスクで実験を行う．one-room タスクでの実験と同じく，A3C，A3C+LSTM，A2OC の 3 通りのアルゴリズムで実験する．エージェントは現在自分がいる部屋の 5×5 の部分しか観測できない部分観測環境である．そのため，LSTM による記憶や Option による戦略の選択なしには適切な行動を選択するよう学習できないと予想される．観測情報から自分がどの部屋にいるのか判別可能な環境であるため，A2OC では部屋ごとに Option を変化させることを期待している．

最初の実験として，A3C，A3C+LSTM，A2OC (Option 数 $= 4$) で実験を行い，結果を比較する．four-rooms タスクでの実験は全て 1000 万ステップまで学習を行う．10 万ステップごとに 10 エピソードの評価を行う．Option-Critic アーキテクチャの論文 [13] や A2OC の論文 [14] において，実験では Option 数 $= 4$ という値がよく使われているため，4 に設定した．POMDP の環境であるため，A3C だけは上手く学習できないと推察される．

次に，A2OC の Option 数を変えた実験を行う．Option 数 $= 4$ と部屋ごとに 2 つの Option を学習することを期待した Option 数 $= 8$ で比較実験を行う． これは，four-rooms タスクの元論文 [12] において four-rooms タスクにおける Option を用いたプランニングが紹介されているが，ここで Option 数 $= 8$ という数が用いられている．部屋ごとに通路は 2 つずつあるため，8 通りの戦略があっても良いという考えである．

次に，t-max というパラメータを変化させた実験を行う．t-max とは，A3C のアルゴリズムにおいて，何ステップごとにモデルを update をするかというハイパーパラメータである．本研究での今までの実験では全て ChainerRL の examples にある A3C の実装と同じ，5 に設定されている．これは，勾配の更新をするときに 5 ステップ以上古い情報は使われないことを意味するので，この環境では足りない可能性があると考えた．そこで，t-max を変化させた実験をする．A2OC (option 数 $= 4$) において，t-max=5 と t-max=15 で比較実験を行う．

最後に，A2OC に LSTM 層を加えたアルゴリズムで実験を行う．2 つのアルゴリズムが合わせることで A2OC 単体，または LSTM 単体と比べて，更に学習速度が上がらないだろうかという期待をしている．

7.2.2　実験結果

four-rooms タスクにおける A3C， A3C+LSTM， A2OC の 3 通りのアルゴリズムの実験結果を図 7.5 に示す．横軸がステップ 数で縦軸がスコアを表している．A3C と A2OC はどちらも学習が進まず，同じ程度の結果になった．A3C+LSTM は高いスコアをで比較的安定して獲得するまで学習できたように見える．

A3C の学習が難しいのは想定していたが，A2OC が上手くいかない理由が分からなかったので，A2OC で学習済のモデルを動かしてエージェントの挙動を調べてみた．すると，常に反時計回りに部屋を探索しようとすることが判明した．その影響で，通路が塞がれていて時計回りに探索しなければならない環境では失敗することが分かった．また，

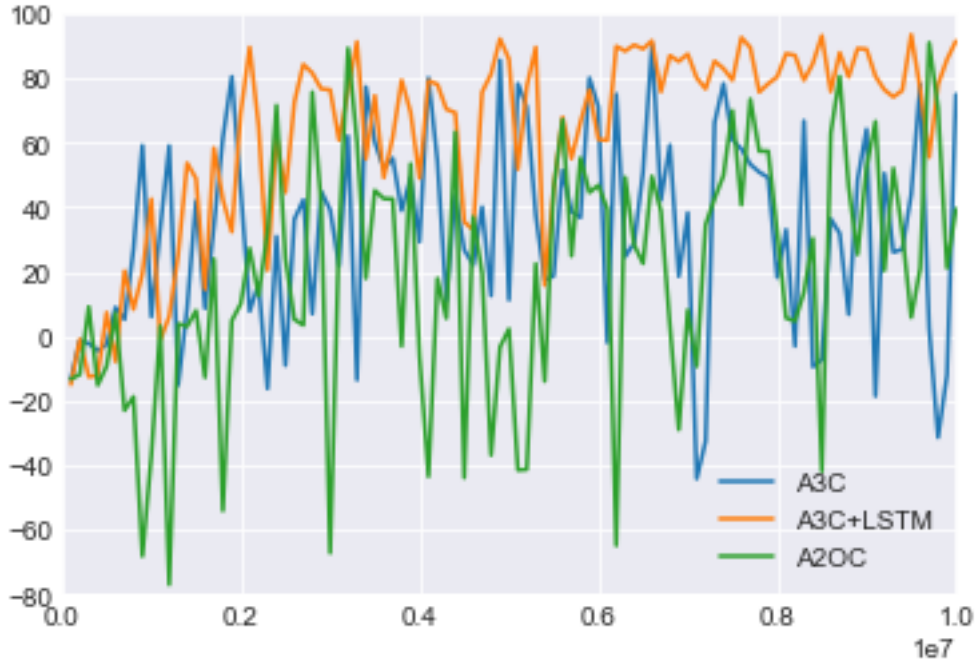

図 7.5　four-rooms におけるアルゴリズム比較

Option は常に一定になっており，変化することは無かった．

例えば，図 7.6 のように通路が塞がれている場合，理想的な動きとしては時計回りに探索すればゴールすることができる．もし右下の部屋に移動してしまっても，右上の部屋に移動したときに通行不能であることを理解して戻れば良いのである．しかし，A2OC のエージェントは右下の部屋から右上の部屋に移動してそのまま上の穴に突っ込んでしまった．

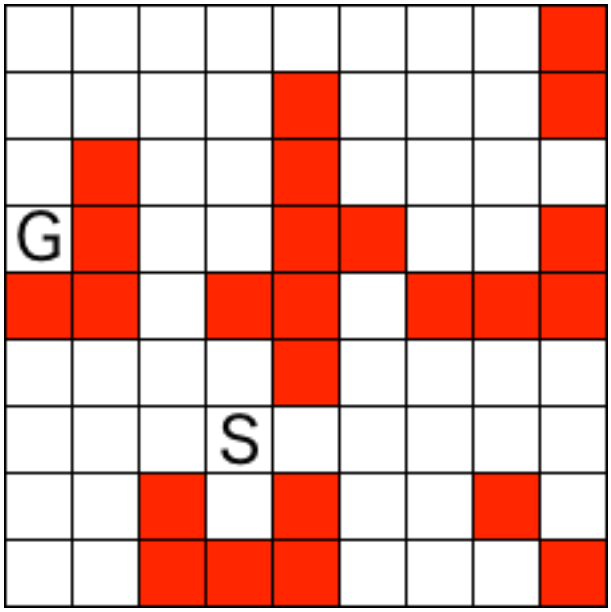

図 7.6　通路が塞がれていて失敗する環境

図 7.7 のように通路が塞がれている場合，反時計回りに探索すればゴールすることができるため，このような環境ではゴールに到達することができた．

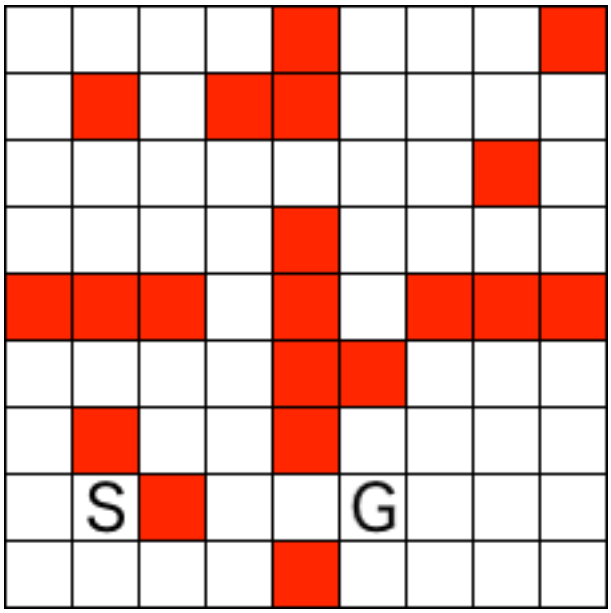

図 7.7　通路が塞がれていても成功する環境

図 7.8 のように通路が塞がれていない場合，時計回りでも反時計回りでもゴールに到達可能である．1 ステップごとに −1 の報酬が与えられるため，最短ルートとしては時計周りでゴールするのが正解であるが，A2OC のエージェントは反時計回りに全ての部屋を探索してゴールに到達した．

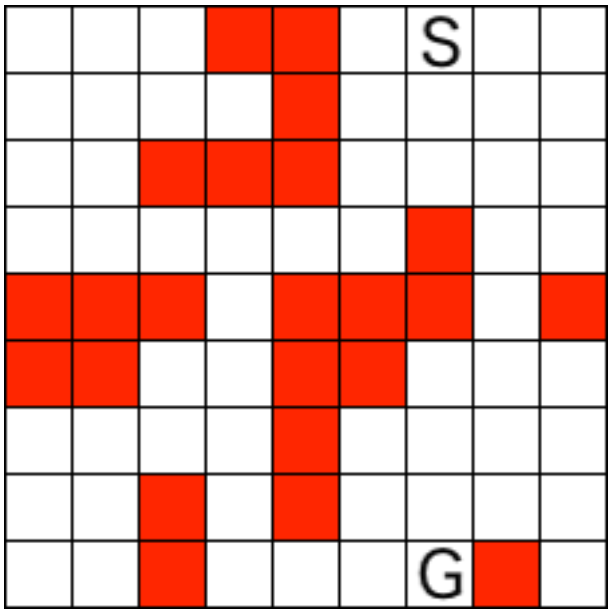

図 7.8　遠回りをして成功する環境

最後に，Option 数を変えた実験，t-max を変えた実験，A2OC に LSTM 層を追加した実験結果はそれぞれ図 7.9 7.10 7.11 のようになった．Option 数，t-max の変化，LSTM

層の導入をしてもほぼ変わらず，A3C+LSTM が一番良いという結果になった．

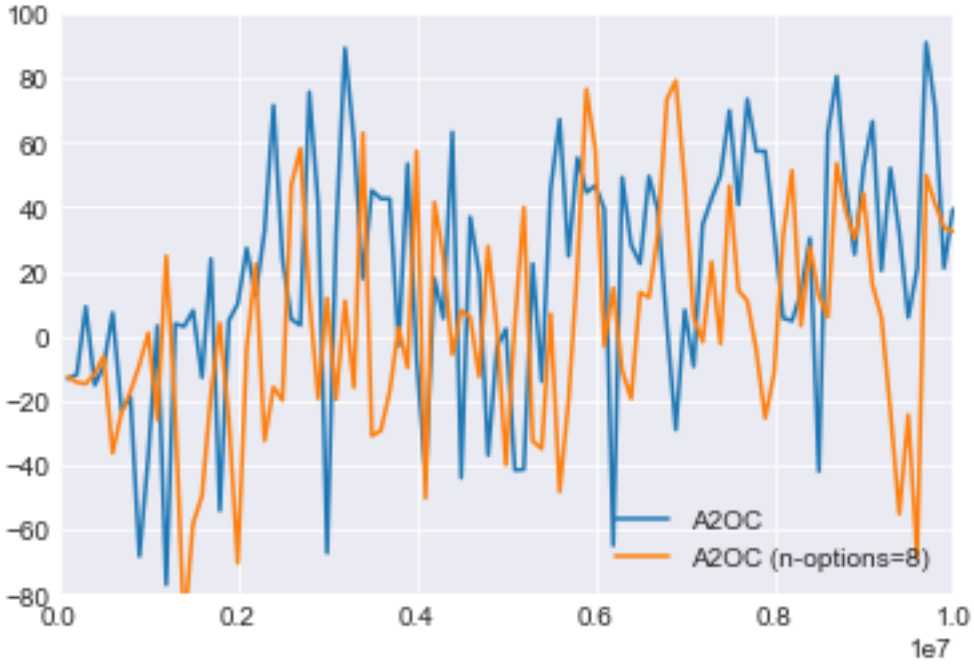

図 7.9　option 数を変更した実験

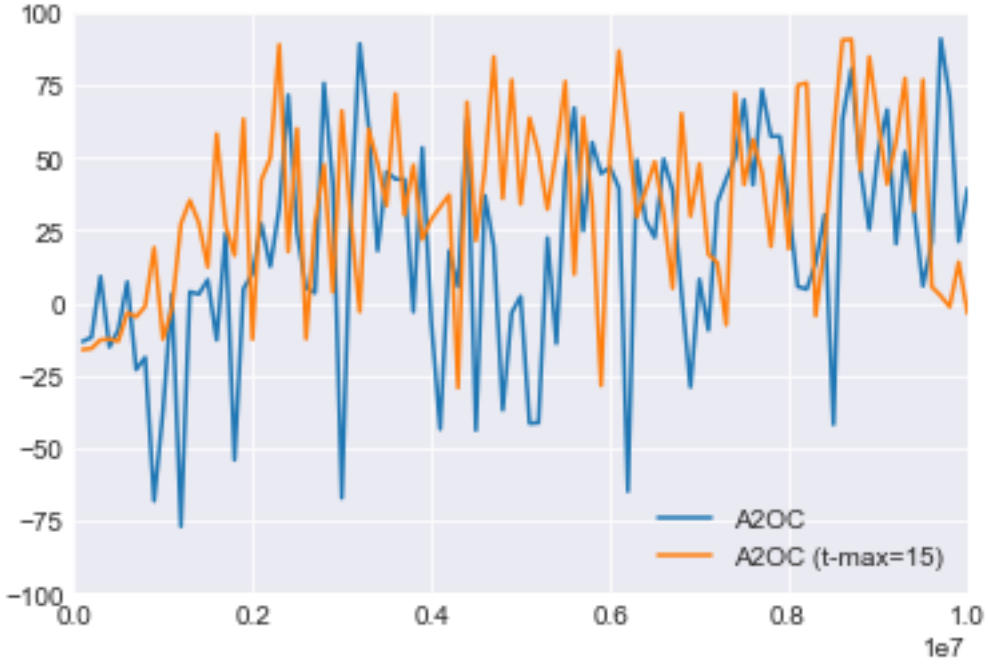

図 7.10　t-max を変更した実験

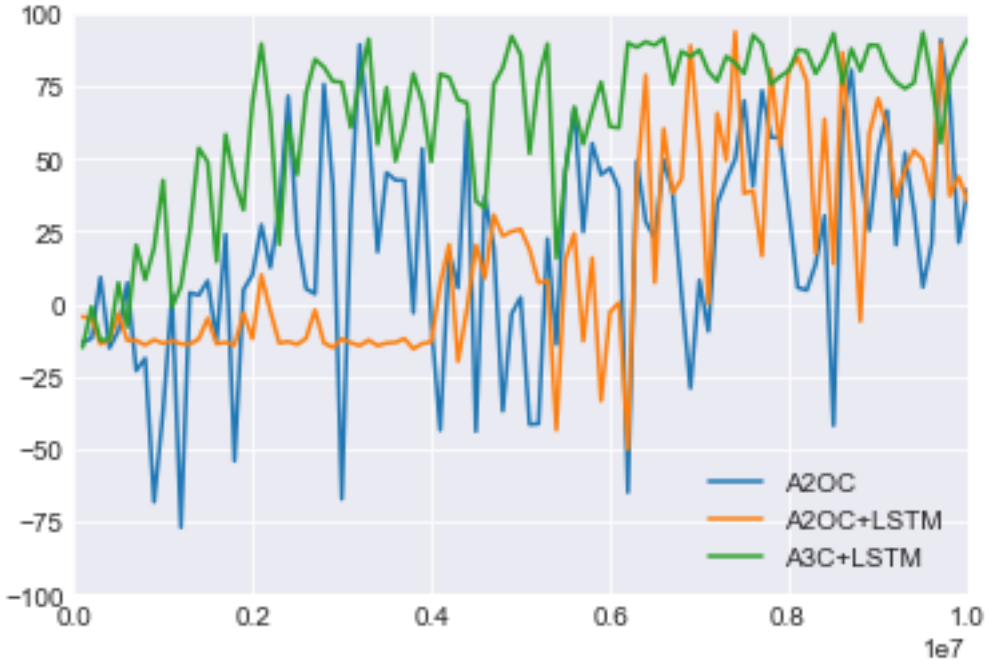

図 7.11　A2OC に LSTM を導入する実験

7.3　全体を観測できる four-rooms タスクでの実験

最後に, 追加実験として常に全体を観測できる four-rooms タスクで実験を行う．A2OC で部分観測的な four-rooms タスクの学習ができなかったが，その原因が部分観測的な問題なのか，それ以外の four-rooms タスクに問題があるのかを確かめるためである．

部分観測的な four-rooms タスクでの実験と同じように，A3C，A3C+LSTM，A2OC で実験を行い，結果を比較する．タスクでの実験は全て 1000 万ステップまで学習を行い，10 万ステップごとに 50 エピソードの評価を行う．

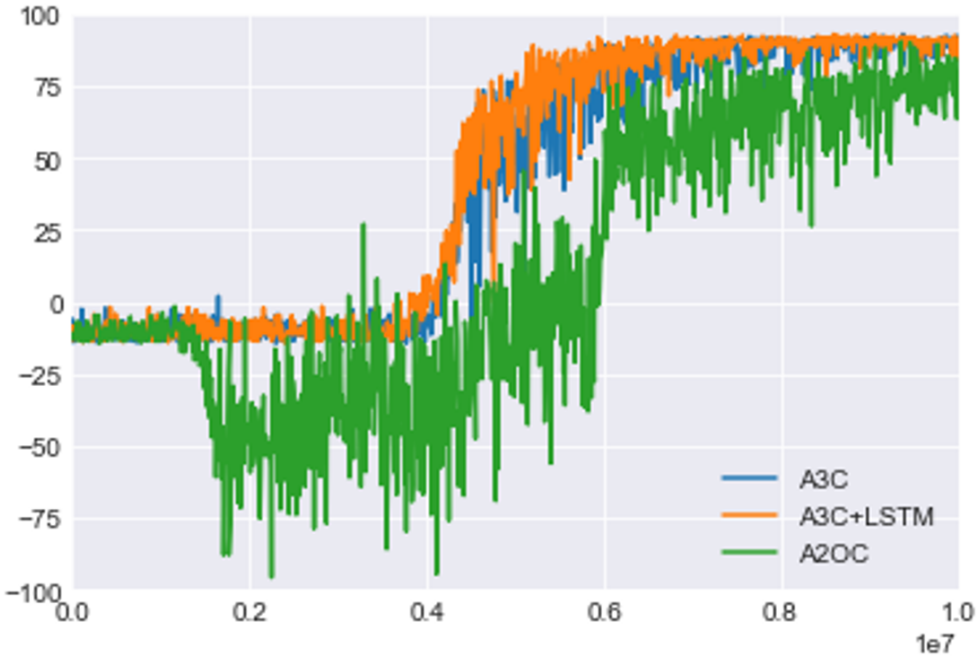

図 7.12　全体を観測できる four-rooms タスクでの実験

この結果から，全体観測的な four-rooms タスクであれば A2OC は学習できることが分かった．このことから，four-rooms の部分観測的な側面が A2OC の学習を難しくしていることが分かった．しかし，この環境でも A2OC エージェントの Option は常に一定であったため，A3C で学習可能な環境であったから A2OC でも学習ができただけに過ぎないと考えられる．

第 8 章

結論と課題

実験の結果から，one-room タスクでは階層型の有無を問わず高い結果を出すことに成功した．しかし，four-rooms タスクでは， A2OC より LSTM を導入した A3C の方が高い結果を出すことが分かった．

本研究で使用した four-rooms タスクは POMDP であり，階層型強化学習をすることによって POMDP 問題を非マルコフ性を解消するようなサブゴールに自動的に分解することで解決することを期待していた．しかし前章での結果から，four-rooms タスクにおいて A2OC はあまり効果が無く，LSTM を導入したほうが良い成績を残すということが分かった．Option-Critic が LSTM 以上の性能を出せる環境としては，下位方策と上位方策のレベルがもっと離れている環境でないと難しいかもしれない．

階層型強化学習と言っても，少なくとも Option-Critic アーキテクチャはヒストリのある世界に向いていない可能性がある．Option の概念を紹介している [12] を見ると，Option の集合を与えられた MDP は Semi-MDP になることを示しており，Semi-MDP での Option についての理論が述べられている．しかし，POMDP での Option については特に述べられていない．

また実験結果から，1 つの決まった Option から変化しないという問題があった．これは終了条件 β の学習が上手くいかなかったことを意味する．Option を変化させたときに貰える微量の報酬 η を 0 で学習させていたので，η を正の値に設定する，または β loss のハイパーパラメータを変更する等の工夫で結果が変わる可能性はある．

意味のある報酬を $t - max = 5$ ステップ 以内に獲得するのが難しいことを考えて 15 に変化させた実験を行ったが，結果は変わらなかった．Option 数を 8 にした実験でも，Option が変化することは無かった．本来であれば行き先が 2 通りなので Option 数は 2 で十分な問題である．

既存研究 [20] では Option が変化しない，もしくは細分化され過ぎることによって変化し続ける問題を提唱している．β が上手く学習できない理由はゴールが遠いときの時間が影響しているため，終了条件分布から時間の概念を分離して終了条件を学習させるという主張である．また，[21] では β を行動価値関数 Q を微分して学習させるのではなく，Option 遷移モデルと β によって Q から独立して学習させることで，有効な Option 切り替えができる場合があることを主張している．

今後の課題としては，有効な Option 切り替えの方法を探るとともに，Minecraft 等の複雑な環境についても実験をして，Option-Critic アーキテクチャを始めとした階層型強化学習の有効性を検証していきたいと考えている．

謝辞

本研究を進めるにあたり，様々なご指導を頂きました田中哲朗先生をはじめ，普段から多くの意見を頂いた GPS の皆さま，Graco ゼミの皆さまに深く感謝いたします．

参考文献

[1] Matthew Johnson, Katja Hofmann, Tim Hutton, and David Bignell. The malmo platform for artificial intelligence experimentation. In *IJCAI*, pp. 4246–4247, 2016.

[2] Chen Tessler, Shahar Givony, Tom Zahavy, Daniel J Mankowitz, and Shie Mannor. A deep hierarchical approach to lifelong learning in minecraft. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[3] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[4] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937, 2016.

[5] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012.

[6] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, Vol. 9, No. 8, pp. 1735–1780, 1997.

[7] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *9th International Conference on Artificial Neural Networks: ICANN '99*, 1999.

[8] Diego Perez-Liebana, Katja Hofmann, Sharada Prasanna Mohanty, Noburu Kuno, Andre Kramer, Sam Devlin, Raluca D Gaina, and Daniel Ionita. The multi-agent reinforcement learning in malmö competition. *arXiv preprint arXiv:1901.08129*, 2019.

[9] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, Vol. 518, No. 7540, pp. 529–533, 2015.

[10] Linjie Xu and Yihong Chen. A hierarchical approach for marlö challenge. In *2019 IEEE Conference on Games (CoG)*, pp. 1–4. IEEE, 2019.

[11] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[12] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-

mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, Vol. 112, No. 1-2, pp. 181–211, 1999.

[13] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[14] Jean Harb, Pierre-Luc Bacon, Martin Klissarov, and Doina Precup. When waiting is not an option: Learning options with a deliberation cost. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[15] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[16] Tuyen P Le, Ngo Anh Vien, and TaeChoong Chung. A deep hierarchical reinforcement learning algorithm in partially observable markov decision processes. *IEEE Access*, Vol. 6, pp. 49089–49102, 2018.

[17] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2623–2631. ACM, 2019.

[18] Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. Chainer: a next-generation open source framework for deep learning. In *Proceedings of workshop on machine learning systems (LearningSys) in the twenty-ninth annual conference on neural information processing systems (NIPS)*, Vol. 5, pp. 1–6, 2015.

[19] Yasuhiro Fujita, Toshiki Kataoka, Prabhat Nagarajan, and Takahiro Ishikawa. Chainerrl: A deep reinforcement learning library. *arXiv preprint arXiv:1912.03905*, 2019.

[20] Anna Harutyunyan, Peter Vrancx, Pierre-Luc Bacon, Doina Precup, and Ann Nowe. Learning with options that terminate off-policy. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[21] Anna Harutyunyan, Will Dabney, Diana Borsa, Nicolas Heess, Remi Munos, and Doina Precup. The termination critic. *arXiv preprint arXiv:1902.09996*, 2019.