電気通信大学情報理工学域
I 類 (情報系) 情報数理工学プログラム卒業論文

マルコフ決定過程における SeqBDD を用いた全列挙方策

2026 年 1 月 29 日

情報数理工学プログラム

学籍番号 2210042

石原 皐太郎

指導教員 植野 真臣

令和 8 年度

令和 4 年度　入学	**学籍番号**　2210042
指導教員　植野真臣	**氏名**　石原皐太郎
題目　マルコフ決定過程における SeqBDD を用いた全列挙方策	

概要

マルコフ決定過程 (MDP) とは，不確実な環境下における逐次的な意思決定問題を扱う数理モデルである．この MDP の枠組みにおいて，将来の累積報酬を最大化する最適な行動決定規則 (方策) を学習する手法は強化学習と呼ばれる．しかし，MDP における状態と行動の組み合わせは指数的に増加するため，すべての遷移を網羅的に計算して最適解を得ることは困難である．状態が高次元あるいは連続である時に列挙して計算することが困難な問題点に対し，近年では深層学習を用いた近似手法が数多く提案されている．一方，モンテカルロ木探索 (MCTS) をはじめとするサンプリングベースの近似手法が広く用いられ高いパフォーマンスを発揮しているが，これらは有望と推定される分岐への過集中や，類似した経路 (ロールアウト) の重複評価により，計算資源に対するサンプル効率の低下や探索性能の劣化を招くという課題がある．

そこで本研究では，文字列集合を効率的に扱うデータ構造である SeqBDD(Sequence Binary Decision Diagram) を用い，到達可能な全ての行動列集合を圧縮して列挙し，シミュレーションを行う手法を提案する．SeqBDD とは列集合を要素の選択・非選択に基づく二分決定グラフとして表現し、共通部分を共有させることで、大規模な列集合の効率的な格納と集合演算を可能にするデータ構造である．本手法では，MDP において h ステップ先までの全ての行動列集合を SeqBDD で構築する．これにより，共通する部分構造を共有して冗長な探索や評価を排除しつつ，指数的に増加する探索空間をメモリ効率よく管理することが可能となる．提案手法は，従来の深層強化学習手法を改善し，限られた計算資源内でより多くの遷移を考慮に入れた，高精度な行動決定を実現する．具体的には，マルコフ決定過程に基づくシミュレーション実験を通じて，提案手法の有効性を評価し，従来手法と比較して優れた性能を確認した．さらに，SeqBDD を用いた全列挙において，そのメモリ使用量の削減が頂点数と圧縮量の観点からも確認された．

1　まえがき

私たちは日常生活や社会の様々な場面で多くの意思決定を行う．これらの意思決定の多くは逐次的であり，意思決定は将来の状況や結果に影響を及ぼす．最適な意思決定を学習する枠組みを強化学習と呼ぶ [13]．実際に強化学習は，ゲーム，ロボット制御，資源管理，推薦システム，などに用いられている [4, 6, 8, 10–12, 14]．

逐次的な意思決定は一般的にマルコフ決定過程 (Markov Decision Process: MDP) で定式化される．MDP は，状態空間 (意思決定者が直面し得る状況の集合)，行動空間 (選択可能な行動の集合)，遷移確率 (状態と行動に応じて次の状態が決まる確率)，報酬関数 (状態と行動，あるいは遷移に応じて得られる実数値)，割引率 (将来報酬の重み) によって定義される．MDP で逐次的な意思決定は，状態を観測して行動を選択し，報酬を得つつ次状態へ遷移する過程を繰り返す，と示される [9]．ここで，行動選択の規則を方策と呼ぶ [9, 13]．したがって MDP における強化学習の目的は，割引率を加味した将来報酬の期待値を最大化する最適方策を学習することとなる [9]．

方策の最適化は，到達可能なすべての状態遷移の集合から将来報酬の期待値を計算する必要があるが，状態と行動の組合せは指数的に増加するため困難である．そのため従来の MDP では，方策に従ったときの将来報酬の期待値を示す価値関数の近似 [8, 13, 13] や到達可能なすべての状態遷移の集合の一部をサンプリングして方策を改善する手法が提案されてきた [1, 3, 11–13]．しかし，価値関数は状態が高次元あるいは連続である時に列挙して計算することが困難なため，近年では深層学習を用いた近似手法が数多く提案されている [8]．

一方，サンプリングに基づいて方策を改善する代表的な手法としてロールアウトがある．ロールアウトとは，ある方策に従って将来の状態遷移と報酬をシミュレーションし，その結果を用いて現在の行動価値を推定する手法である．Monte Carlo Tree Search（MCTS）[2] は，このロールアウトの考え方を探索木構造に組み込んだ代表的な手法であり，探索木上での選択と，葉ノードからのシミュレーションを繰り返すことで行動価値を推定し，有望な行動系列を重点的に探索する．この MCTS を基盤とした深層強化学習手法として，囲碁 AI である AlphaGo [12]，汎用ボードゲーム AI である AlphaZero [11]，さらに Atari や連続制御問題にも適用された MuZero [10] などが提案されてきた．AlphaGo では，探索木の葉から終局までを高速な方策でプレイアウトするロールアウトによって勝敗をサンプリングし，これを価値推定に用いていた．一方 AlphaZero では，ニューラルネットワークによる価値推定を用いた MCTS を自己対戦学習に組み込んだ．これにより AlphaZero は，終局までのロールアウトに依存しない探索を実現している．さらに MuZero では，環境の遷移規則

や報酬関数が既知であることを仮定せず，観測から学習された潜在状態上でのモデルを用いている．これにより MuZero は，終局までのロールアウトに依存しない探索を維持しつつ，明示的な環境シミュレータを必要とせずに，将来の行動系列に対する価値推定と方策改善を可能としている．これらの手法はいずれも，有望な行動に探索を集中させた多数のサンプリングを通じて方策の精度を向上させる点に特徴がある．しかし，MCTS では到達可能な状態遷移を十分に見ないまま，類似したロールアウト (あるいは近い葉評価) が繰り返される．これは，一見有望に見えた分岐への過集中，後に有望さが反転する分岐の見落としを招く．これらの問題点は限られた計算資源の中で，サンプル効率の低下や探索性能の劣化を起こし得る．

本研究では SeqBDD(Sequence Binary Decision Diagram) [5] を用いて MDP におけるすべての到達可能な状態遷移の集合を圧縮して列挙することを提案する．ここで，SeqBDD とは Loekito らによって提案された文字列集合を効率的に表現・操作するデータ構造である [5]．SeqBDD は，場合分け二分木 (Binary Decision Tree) の圧縮表現である ZDD(Zero-suppressed Binary Decision Diagram) [7] の派生構造であり，列集合に対し共通部分構造の共有，冗長構造の排除を行うことで重複した表現・操作を削減している．状態遷移を SeqBDD で表現することで，指数的に増加する状態遷移の集合を効率的に管理でき，従来より多くの遷移を考慮した行動決定を実現する．本手法では現実的な実装を考慮し，決定論的な MDP において h ステップ先までにおける，全ての行動列集合を SeqBDD で列挙し，重複する部分列を共有することで探索・評価の重複を削減する．これにより，指数的に増加する状態遷移の集合を効率的に管理でき，従来より多くの遷移を用いて行動を評価・決定する．

2　マルコフ決定過程と強化学習

2.1　マルコフ決定過程

マルコフ決定過程とは逐次的な意思決定とその相互作用を記述するための数理モデルであり, マルコフ決定過程は以下の状態空間 S, 行動空間 $A(s)$, 状態遷移確率 $P(s'|s,a)$, 報酬関数 $r(s,a,s')$, という四つの要素によって記述される確率過程である [9].

- 状態集合 S を, すべての状態からなる集合とする. この集合の要素を s で表す.
- 行動集合 $A(s)$ を, 状態 s においてエージェントが選択可能な行動からなる集合とする. この集合の要素を a で表す.
- 状態遷移確率 $P(s'|s,a)$ を, 状態 s において行動 a を選択したときに次状態 s' に遷移する確率とする.
- 報酬関数 $r(s,a,s')$ を, 状態 s において行動 a を選択し次状態 s' に遷移したときに得られる報酬とする.

なお, 行動する主体をエージェント, エージェントを取り巻くエージェント以外を環境という.

各時刻 t において, エージェントは状態 $s_t \in S$ に存在し, 行動 $a_t \in A(s_t)$ を選択する. その後, 確率 $P(s_{t+1}|s_t,a_t)$ で次状態 $s_{t+1} \in S$ に遷移し, 報酬 $r_t(s_t,a_t,s_{t+1})$ を得る. この一連の流れを繰り返すことで, エージェントは環境と相互作用を行う.

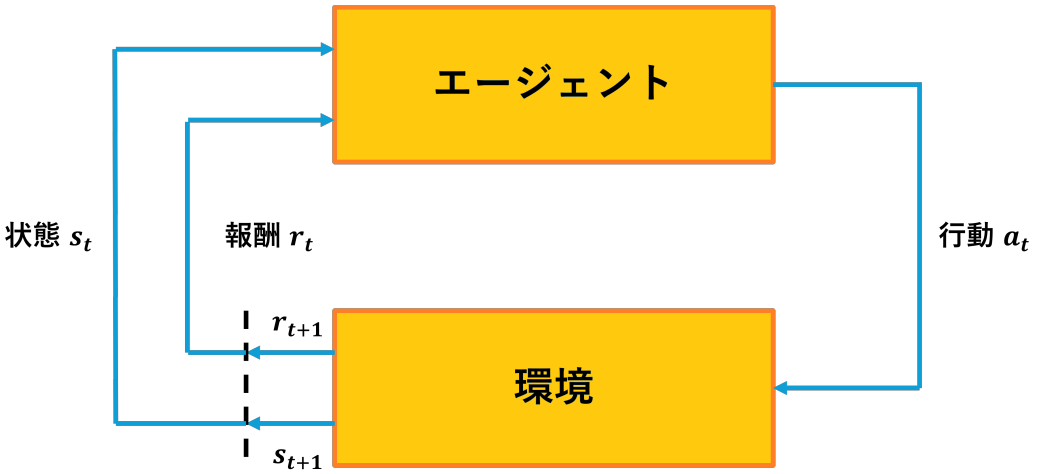

図 1　マルコフ決定過程におけるエージェントと環境の相互作用

マルコフ決定過程では, 各時刻 t における状態 s_t がマルコフ性を満たす. すなわち,

$$P(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, \ldots, s_0, a_0) = P(s_{t+1}|s_t, a_t) \tag{1}$$

が成り立つ. つまり, 次状態 s_{t+1} の確率分布は, 現在の状態 s_t と行動 a_t のみに依存し, それ以前の状態や行動には依存しない.

マルコフ決定過程の目的は, エージェントが得られる累積報酬を最大化することである. 累積報酬は, 割引率 $\gamma \in [0, 1)$ を用いて以下のように定義される.

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{2}$$

ここで, G_t は時刻 t における累積報酬であり, r_{t+k+1} は時刻 $t+k+1$ における報酬である.

エージェントの行動方策 π は, 各状態 s において選択する行動 a の確率分布である. すなわち,

$$\pi(a|s) = P(a_t = a|s_t = s) \tag{3}$$

が成り立つ.

2.2 強化学習と価値関数

強化学習とは，環境との相互作用を通じて，将来得られる報酬の総和が大きくなるような行動の取り方 (方策) を学習する機械学習の一分野である．前節で述べたマルコフ決定過程 $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$ を環境のモデルとみなし，この上でエージェントが方策を更新していく．

各時刻 t において，エージェントは状態 $s_t \in \mathcal{S}$ を観測し，行動 $a_t \in \mathcal{A}$ を選択する．環境はその結果として報酬 $r_t \in \mathbb{R}$ を与え，次の状態 s_{t+1} に遷移する．この相互作用

$$s_t \xrightarrow{a_t} (r_t, s_{t+1})$$

が繰り返されることで 1 エピソードが生成される．状態から行動を選ぶ確率分布

$$\pi(a \mid s) = \Pr(a_t = a \mid s_t = s) \tag{4}$$

をエージェントの行動方策（方策）と呼ぶ．強化学習の目的は，この方策 π を，得られる報酬の期待値が最大になるように改善していくことである．

時刻 t から見た将来の報酬の割引和

$$G_t := \sum_{k=0}^{\infty} \gamma^k r_{t+k} \tag{5}$$

をリターンと呼ぶ（$0 \le \gamma < 1$ は割引率）．ある方策 π の良さを定量化するために，以下の

2 種類の価値関数が用いられる．

- **状態価値関数（state-value function）**

$$V^{\pi}(s) := \mathbb{E}_{\pi}[G_t \mid s_t = s], \quad s \in \mathcal{S}, \tag{6}$$

すなわち「状態 s から方策 π に従って行動したときに得られるリターンの期待値」を表す．

- **行動価値関数（action-value function）**

$$Q^{\pi}(s,a) := \mathbb{E}_{\pi}[G_t \mid s_t = s, a_t = a], \quad (s,a) \in \mathcal{S} \times \mathcal{A}, \tag{7}$$

すなわち「状態 s で行動 a をとり，その後は方策 π に従ったときに得られるリターンの期待値」を表す．

最適制御の観点からは，行動価値関数に対して

$$Q^{*}(s,a) := \sup_{\pi} Q^{\pi}(s,a) \tag{8}$$

を満たす最適行動価値関数 Q^{*} を求め，

$$\pi^{*}(s) \in \arg\max_{a \in \mathcal{A}} Q^{*}(s,a) \tag{9}$$

で定まる貪欲方策 π^{*} を得ることが目標となる．Q^{*} が既知であれば，(9) は最適方策の 1 つを与える．

実際の強化学習では，遷移確率 P や真の Q^{π} をあらかじめ知ることはできないため，サンプルされた経験 (s_t, a_t, r_t, s_{t+1}) に基づいて価値関数を逐次的に推定する．行動価値関数を直接推定する代表的な手法が Q 学習（Q-learning）である．

2.3 Q 学習

Q 学習は行動価値関数 (Q 関数) を用いて行動方策を学習する強化学習の手法である．Q 学習の行動方策 π は, 各状態において Q 関数が最大となる行動を選択する．

$$\pi(s) = \arg\max_{a} Q(s,a) \tag{10}$$

Q 学習では, Q 関数を更新することで行動方策を学習する．Q 関数の更新は以下の式で行われる．

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\left(r_{t+1} + \gamma \max_{a} Q(s_{t+1},a) - Q(s_t,a_t)\right) \tag{11}$$

ここで, α は学習率である．

Q 学習では, Q 関数をテーブルで管理する方法と, ニューラルネットワークで近似する方法がある. ニューラルネットワークで近似する方法は Deep Q-Network と呼ばれ, 近年多くの研究が行われている.

2.4 Deep Q-Network

Deep Q-Network は Q 関数をニューラルネットワークで近似し, ニューラルネットワークのパラメータを更新することで Q 関数を学習する強化学習の手法である. 行動方策 π は Q 学習と同様に各状態において Q 関数が最大となる行動を選択することで定義される.

$$\pi(s) = \arg\max_a Q(s, a; \theta) \tag{12}$$

ここで, θ はニューラルネットワークのパラメータである.

ニューラルネットワークのパラメータの更新は, 損失関数を最小化することで行われる. 損失関数は以下の式で定義される.

$$L(\theta) = \mathbb{E}_{(s_t, a_t, r_{t+1}, s_{t+1}) \sim D} \left[\left(r_{t+1} + \gamma \max_a Q(s_{t+1}, a; \theta^-) - Q(s_t, a_t; \theta) \right)^2 \right] \tag{13}$$

ここで, θ^- はターゲットネットワークのパラメータ, D は経験再生バッファである. 経験再生バッファは, エージェントが経験した状態, 行動, 報酬, 次状態の組を保存するバッファであり, ミニバッチ学習に用いられる. ターゲットネットワークは, Q 関数の安定性を向上させるために用いられる. ターゲットネットワークのパラメータは, 一定の間隔でニューラルネットワークのパラメータに更新される.

Deep Q-Network は, テーブルで管理する Q 学習に比べ, 大規模な状態空間や連続的な状態空間に対しても適用可能であるという長所があり, 多くの応用例が存在する.

3　先読み (look-ahead) や rollout による方策改善

3.1　h ステップ先読み方策

h ステップ先読み方策 (h-step look-ahead policy) を定義する．これは, 1 ステップ貪欲方策 (greedy policy)

$$\pi^{(1)}(s) \in \arg\max_{a \in \mathcal{A}} Q(s,a) \tag{14}$$

を h ステップへと一般化したものであり, h ステップ先までの報酬と末端の価値を用いて方策改善を行うものである．価値関数 V が与えられているとする．状態 s_0 における h ステップ先読みによる行動列 $a_{0:h-1} = (a_0, \ldots, a_{h-1})$ の評価は

$$J_h(s_0, a_{0:h-1}) := \mathbb{E}\left[\sum_{t=0}^{h-1} \gamma^t r_t + \gamma^h V(s_h) \,\middle|\, s_0, a_{0:h-1} \right] \tag{15}$$

で定義される．ここで (s_{t+1}, r_t) は遷移モデル P と報酬関数 r に従って生成される．

このとき, h ステップ先読み方策 $\pi^{(h)}$ は

$$\pi^{(h)}(s_0) \in \arg\max_{a_0 \in \mathcal{A}} G_h(s_0, a_0), \quad G_h(s_0, a_0) := \max_{a_{1:h-1} \in \mathcal{A}^{h-1}} J_h(s_0, a_0, \ldots, a_{h-1}) \tag{16}$$

として定義される．すなわち, 先頭行動 a_0 を固定したときに取りうる $h-1$ ステップ分の行動列の中でもっとも期待リターンが大きいものを選び, その値 $G_h(s_0, a_0)$ に基づいて最良の a_0 を選択する方策である．$h = 1$ とおくと (16) は通常の 1 ステップ貪欲方策に一致する．

3.2　rollout アルゴリズムと方策反復

h-step 先読み方策の定義を用いて (16) を厳密に計算しようとすると, 行動空間が大きい場合や h が大きい場合には計算量が指数的に増大する．rollout アルゴリズムは, この h ステップ先読み方策をベース方策を用いてサンプルベースに近似する実装として広く知られている．ベース方策 μ とその性能指標 J^μ が与えられているとする．rollout アルゴリズムでは, 状態 s_0 において任意の候補行動 a_0 を選択した後, 残りのステップ $t \geq 1$ ではベース方策 μ に従って行動する．具体的には, K 本のロールアウトを用いて

$$\hat{Q}^{\mathrm{roll}}(s_0, a_0) := \frac{1}{K} \sum_{k=1}^{K} \left(\sum_{t=0}^{h-1} \gamma^t r_t^{(k)} + \gamma^h J^\mu(s_h^{(k)}) \right) \tag{17}$$

を推定し,

$$\pi^{\mathrm{roll}}(s_0) \in \arg\max_{a_0 \in \mathcal{A}} \hat{Q}^{\mathrm{roll}}(s_0, a_0) \tag{18}$$

によりベース方策 μ を改善した新たな方策を得る．さらに, 改善後の方策を新たなベース方策として繰り返し適用することで, rollout に基づく方策反復（rollout-based policy iteration）を構成できる.

3.3 Monte Carlo Tree Search とその応用

MCTS は，MDP を基盤とする探索アルゴリズムの一つであり，現在の状態から将来の行動結果をシミュレーションすることで行動を選択する手法である．MCTS は，膨大な状態空間を持つ問題に対しても，全ての状態を網羅的に探索することなく，近似的な最適行動を効率的に求められる点に特徴があり，囲碁や将棋，チェスなどの複雑な完全情報ゲームにおいて顕著な成果を挙げてきた．MCTS は，探索木を根ノードから構築し，ランダムプレイアウトによるシミュレーションを繰り返すことで報酬を推定し，期待値の高い行動経路を重点的に探索する．典型的な MCTS アルゴリズムは,

(1) UCT（UpperConfidence Bound for Trees）を用いて既存の探索木から次に展開するノードを選択する選択
(2) 未探索の行動を展開して木構造を拡張する展開
(3) ランダムまたはヒューリスティックなプレイアウトを実行して報酬を評価するシミュレーション
(4) 得られた結果を親ノードに逆伝播して価値を更新する逆伝播

という 4 つのステップから構成される．この手順を反復することで探索木が成長し，エージェントは将来報酬が最大となると推定される行動を選択することができる．MCTS の強みは，状態空間が極めて大きい問題でも限られた計算資源で先読みを実現できる点にあり，さらに事前に完全な環境モデルやドメイン知識を必要としないため，幅広い領域への適用が可能である．実際，AlphaGo では深層ニューラルネットワークによる状態評価と MCTS を組み合わせることで，人間のトップ棋士を凌駕する性能を実現している．しかしながら，MCTS には探索が偏りやすいという課題も存在する．UCT に基づく探索では，将来報酬が高いと推定される経路が優先的に選択されるため，他の可能性のある経路が十分に探索されないまま探索が一本道に集中する傾向がある．このような偏りは，局所最適な行動方策への収束や，環境変化への適応力低下を引き起こす可能性があり，特に複雑な報酬構造を持つ問題や限られた探索時間の下では深刻な問題となる．本研究では，MCTS のように将来の行動報酬を用いながら，探索の偏りを軽減する新たな手法を提案することで，より汎用的かつ

柔軟な意思決定アルゴリズムの実現を目指す.

3.4 深層強化学習における木探索

近年の深層強化学習では, ニューラルネットワークによる表現学習と, MCTS による先読み方策改善を組み合わせることで卓越した性能が得られている. AlphaGo [12] は, 手作業の特徴量と深層ポリシーネットを用い，MCTS による探索で方策を改善することで囲碁における世界レベルの性能を実現した. AlphaZero [11] では, ニューラルネットワークが方策と価値を同時に出力し, 自己対戦における探索結果を用いて方策ネットワークを強化する枠組みが採用されている. MuZero [10] では, 環境モデルもニューラルネットワークで近似し, 学習されたモデル上で MCTS を行うことで, Atari2600 やチェス, 将棋など多様な環境において高い性能を達成している. これらの手法はいずれも,

- ニューラルネットワークは価値関数・方策・（場合によってはモデル）を近似する
- MCTS は, それらを用いた先読みによる方策改善オペレータ

として機能している, という二層構造を明示的に持っている. 本研究は, ニューラルネットワークの有無に依らず, この「先読みによる方策改善」の部分の構造に着目する. すなわち, 深層学習は価値関数やモデルの近似手段として利用可能である一方, rollout 方策改善そのものは, より抽象的な MDP レベルの構造の問題として捉えることができる.

4　マルコフ決定過程におけるSeqBDDを用いた全列挙方策の提案

従来手法の問題点を解決するために, 本研究では Sequence Binary Decision Diagram(以下 SeqBDD) を行動方策に用いた強化学習手法を提案する.

4.1　SeqBDD

SeqBDD (Sequence Binary Decision Diagram) とは Loekito らによって提案された文字列集合を効率的に表現・操作するデータ構造である [5]. SeqBDD は, 場合分け二分木 (Binary Decision Tree) の圧縮表現である ZDD(Zero- suppressed Binary Decision Diagram) [7] の派生構造であり, 列集合に対し共通部分構造の共有, 冗長構造の排除を行うことで重複した表現・操作を削減している.

SeqBDD は, 列 (sequence) の集合をコンパクトに表現するために提案された BDD 系データ構造である. ZBDD は集合操作に有効な圧縮規則を持つ一方で, 1 本のパス上に同一変数を複数回出現させられないため, 同じアイテムが繰り返し現れ得る列集合をそのまま表現できない. この制約に対し SeqBDD では, 親子の変数順序を 0-枝にのみ課し, 1-枝には順序制約を課さないという構造を採る. これにより, 同じ変数が 1 つのパスに複数回現れることが許され, 列の表現に必要な反復を直接扱える. 以下は SeqBDD の構造の例である. 図 2

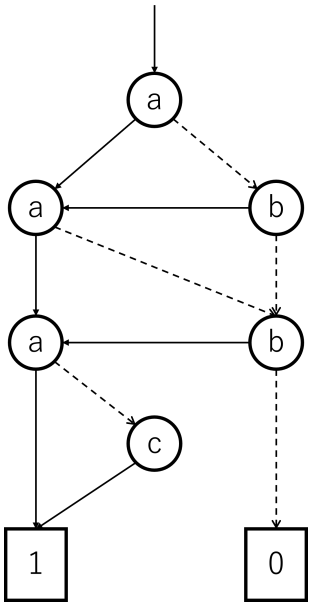

図 2　$\{aaa, aac, baa, bac, ba, bc\}$ を表現する SeqBDD

は，文字列集合 $\{aaa, aac, baa, bac, ba, bc\}$ を表現する SeqBDD である．SeqBDD の各節点は変数を表し，枝はその変数の選択を示す．0-枝はその変数を選択しない場合，1-枝は変数を選択する場合を表す．SeqBDD では，共通部分構造が共有されるため，冗長な表現が排除され，効率的な列集合の表現が可能となる．

4.2 SeqBDD を用いた全列挙方策

本研究では，SeqBDD を用いて MDP におけるすべての到達可能な状態遷移の集合を圧縮して列挙することを提案する．決定論的な MDP において行動列集合を SeqBDD で列挙し，重複する部分列を共有することで探索・評価の重複を削減する．これにより，指数的に増加するすべての到達可能な状態遷移の集合を効率的に管理でき，従来は困難であった全遷移を考慮した行動決定を実現する．SeqBDD における節点は各時刻における行動を表し，枝は次の時刻に遷移する行動を示す．すなわち，0-枝はその時刻における行動を選択しない場合，1-枝は行動を選択する場合を表す．現在の状態から到達可能な最終状態に至るまでの行動列を SeqBDD で表現し，各行動列に対応する累積報酬を計算することで，最適な行動列を選択する．具体的には，以下の手順で行う．

1. 現在の状態から開始し，SeqBDD を用いて可能な行動列を全て列挙する．
2. 各行動列に対して，対応する累積報酬を計算する．
3. 最も高い累積報酬を持つ行動列を選択し，その最初の行動を実行する．
4. 環境からのフィードバックを受け取り，次の状態に遷移する．
5. 必要に応じて SeqBDD を更新し，次の行動決定に備える．

この手法により，重複する行動列の部分構造を共有することで，計算資源の節約と効率的な探索が可能となる．ここで，SeqBDD の圧縮効果により，指数的に増加する行動列の管理が現実的な計算時間内で可能となることが期待される．また，この時点で，SeqBDD を用いた行動列の全列挙は，従来のロールアウト手法に比べて，探索の偏りを軽減し，多様な行動選択肢を評価できる点で優れていると考えられる．全行同列列挙の方策は以下の式で表される．

$$\pi(s) = \arg \max_{a_{0:H-1} \in \mathcal{A}^H} \sum_{t=0}^{H-1} r(s_t, a_t) \tag{19}$$

ここで，H は計画のホライズン長，s_t は時刻 t における状態，a_t は時刻 t における行動，$r(s_t, a_t)$ は状態 s_t で行動 a_t を取ったときの報酬を表す．この方策は，SeqBDD を用いて全ての行動列 $a_{0:H-1}$ を効率的に列挙し，各行動列に対応する累積報酬を計算することで最適

な行動列を選択する．深層強化学習における提案手法は，従来の木探索同様に，ニューラルネットワークによる価値関数近似と組み合わせて利用可能である．具体的には，SeqBDD を用いた全列挙方策において，各行動列の評価にニューラルネットワークによる価値関数を用いることで，より精度の高い行動選択が可能となる．

$$\pi(s) = \arg \max_{a_{0:H-1} \in \mathcal{A}^H} \left(\sum_{t=0}^{H-1} r(s_t, a_t) + \gamma^H V(s_H) \right) \tag{20}$$

また，AlphaZero [11] や MuZero [10] のように，SeqBDD を用いた全列挙方策をニューラルネットワークによる方策近似と組み合わせることも可能である．

$$\pi(s) = \arg \max_{a_{0:H-1} \in \mathcal{A}^H} \left(\sum_{t=0}^{H-1} r(s_t, a_t) + \gamma^H Q(s_H, a_H) \right) \tag{21}$$

ここで，$V(s_H)$ は状態 s_H における価値関数，$Q(s_H, a_H)$ は状態 s_H で行動 a_H を取ったときの行動価値関数を表す．これらの近似手法と組合わせた際も，SeqBDD を用いた全列挙方策は，重複する行動列の部分構造を共有することで，計算資源の節約と効率的な探索が可能となるだけでなく，探索の偏りを軽減し，多様な行動選択肢を評価できる点で優れていると考えられる．

4.3 SeqBDD を用いた全列挙方策の計算量

SeqBDD を用いた全列挙方策の計算量について述べる．同一量のシミュレーションを行う場合の空間計算量を比較する．従来のロールアウト手法では，K 本のロールアウトを行う場合，それぞれのロールアウトで H ステップ分の状態と行動を保存する必要があるため，空間計算量は $O(K \cdot H)$ となる．ここでロールアウトの本数 K は，行動集合の大きさ $|\mathcal{A}|$ に対し指数関数的に増加する．全列挙する場合，$K = |\mathcal{A}|^H$ となるため，空間計算量は $O(|\mathcal{A}|^H \cdot H)$ となる．これを順序付き二分木で表現した場合，頂点数は $|\mathcal{A}|^H$ となる．一方，SeqBDD を用いた全列挙方策では，重複する行動列の部分構造を共有することで，行動の保存に必要な空間を削減できるため，全列挙を SeqBDD で表現した場合の頂点数を N_{SeqBDD} とすると，その頂点数は $N_{SeqBDD} \leq |\mathcal{A}|^H \cdot H$ である．また，SeqBDD を用いた全列挙方策の空間計算量は $O(N_{SeqBDD})$ となる．

5　評価実験

本研究では，提案手法の有効性を評価するために，以下の実験を行う．

- 全列挙の有効性評価: SeqBDD を用いた全列挙方策が，従来のロールアウト手法に比べて探索の偏りを軽減し，多様な行動選択肢を評価したことで性能の向上が見られるかをを検証する．
- SeqBDD の圧縮効果の評価: 提案手法における SeqBDD のサイズを評価し，従来の全列挙手法と比較してどれだけ圧縮されているかを検証する．
- 提案手法と従来手法の性能比較: 提案手法を用いた強化学習エージェントと，従来の Deep Q-Network やその改良手法を用いたエージェントの性能を比較する．
- 学習速度の評価: 各手法における学習速度を評価し，提案手法がより迅速に最適方策に収束するかどうかを検証する．

5.1　実験環境

今回の評価実験では，以下の 4 つ環境を適宜使用する．

- Acrobot: OpenAI Gym の Acrobot-v1 環境を使用．エージェントは 2 つのリンクからなるアクロバットを制御し，目標位置まで到達させる．
- CartPole: OpenAI Gym の CartPole-v1 環境を使用．エージェントはカートを左右に動かし，ポールを倒さないようにバランスを取る．
- MountainCar: OpenAI Gym の MountainCar-v0 環境を使用．エージェントは谷底にいる車を制御し，山頂まで到達させる．
- FrozenLake: OpenAI Gym の FrozenLake-v1 環境を使用．エージェントは氷の湖を渡り，ゴールに到達する．

5.2　評価指標

本研究では，以下の評価指標を用いて提案手法の性能を評価する．

- 累積報酬: 各エピソードにおける累積報酬の平均値を計算し，提案手法と従来手法の性能を比較する．
- 学習速度: 目標累積報酬に到達するまでのエピソード数を計測し，提案手法と従来手法の学習速度を比較する．

- SeqBDD のサイズ: 提案手法における SeqBDD のノード数を計測し，順序付き二分木と比較して圧縮効果を評価する．
- 使用メモリ量: 全列挙方策において SeqBDD と SeqBDD を用いない場合のメモリ使用量を比較し，提案手法の圧縮効果を評価する．

5.3 比較手法

本実験では，以下の従来手法と提案手法を組合わせたものを比較する．

- Deep Q-Network: ニューラルネットワークで Q 関数を近似し，Deep Q-Network アルゴリズムを用いて行動方策を学習する．
- Double DQN: Deep Q-Network の改良版であり，過大評価を抑制するために 2 つの Q ネットワークを用いる．
- Dueling DQN: Deep Q-Network の改良版であり，状態価値関数とアドバンテージ関数を分離して学習する．
- Prioritized Experience Replay DQN(PER DQN): 経験再生バッファに優先度を導入し，重要な経験を優先的に学習する．
- NoisyNet DQN: ニューラルネットワークの重みにノイズを加えることで探索を促進する．
- Categorical DQN: Q 関数の分布を近似し，分布に基づく行動選択を行う．
- N-step DQN: 複数ステップの報酬を考慮して Q 関数を更新する．

5.4 実験設定

本研究では，提案手法と従来手法を比較するために，以下の実験設定を用いる．

表 1　各環境におけるハイパーパラメータ設定

Hyperparameter	Acrobot	CartPole	MountainCar	FrozenLake
訓練ステップ数	200,000	50,000	100,000	300,000
メモリサイズ	50,000	20,000	20,000	100,000
バッチサイズ	32	32	32	32
ターゲットネットワーク更新頻度（ステップ）	1,000	1,000	1,000	1,000
割引率 γ	0.99	0.99	0.99	0.99
学習率	0.001	0.001	0.001	0.001
ε-greedy 探索	初期値 1.0→0.1（線形減衰），その後 0.01 まで線形減衰			
ε 減衰期間（ステップ）	150,000	30,000	80,000	250,000
ウォームアップ期間（ステップ）	10,000	5,000	5,000	10,000
PER: α	0.5	0.5	0.5	0.5
PER: β	初期値 0.4→1.0（線形増加）			
PER: ε	1×10^{-6}	1×10^{-6}	1×10^{-6}	1×10^{-6}
Categorical DQN: $v_{\min}$	-500	-10	-1	-300
Categorical DQN: $v_{\max}$	0	10	1	0
Categorical DQN: ビン数 N_{atom}	51	51	51	51
N-step DQN: n	3	3	3	3
提案手法の先読み数（ステップ）	3	3	3	3

なお各実験題材において同じ五つの環境を用意し，それぞれ 5 回ずつ学習を行い平均値を算出する．また，評価に用いるモデルは各訓練ステップ数における最良モデルを採用する．

5.5　実験結果

実験結果は以下の通りである．表 2 に各環境における各手法の評価結果を示す．提案手法は従来手法の多くを改善した．書く環境で最も大きい割引累積報酬の和を太字で示す．Acrobot 環境では，Categorical DQN + 提案手法が-69 で最も良い結果を示した．CartPole 環境では，Categorical + 提案手法が 337 で最も良い結果を示した．MountainCar 環境では，Dueling + 提案手法が-162.6 で最も良い結果を示した．FrozenLake 環境では，全ての提案手法が 1 で最も良い結果を示した．また，提案手法は全体的に従来手法よりも高い性能を示し，SeqBDD を用いた全列挙方策の有効性が確認された．

表 2　各環境における各手法の評価結果

Method	Acrobot	CartPole	MountainCar	FrozenLake
DQN	-500	190	-200	0.2
Double DQN	-500	152	-200	0.6
Dueling	-418.2	196.2	-200	0.4
PER DQN	-430.2	212.8	-200	0.2
NoisyNet DQN	-500	147.8	-200	0
Categorical DQN	-473.4	184.6	-200	0.4
N-step DQN	-422.8	174.4	-163.5	0.6
DQN + 提案	-266	195.6	-165.4	**1**
Double DQN + 提案	-315	138.2	-186	**1**
Dueling + 提案	-133	299.6	**-162.6**	**1**
PER DQN + 提案	-94.4	255.2	-200	**1**
NoisyNet DQN + 提案	-500	163.8	-200	**1**
Categorical DQN + 提案	**-69**	**337**	-200	**1**
N-step DQN + 提案	-109.2	203.2	-165.6	**1**

5.6　全列挙方策における SeqBDD の圧縮効果

本実験では，提案手法における SeqBDD の圧縮効果を評価するために，行動数 2,3,4 のそれぞれ深さ 3,5,7,9 の SeqBDD と順序付き二分木のノード数を計測し，SeqBDD の圧縮効果を計測する．表 3 に各行動数における SeqBDD と順序付き二分木のノード数である．SeqBDD は順序付き二分木に比べて大幅にノード数が削減されていることがわかる．例えば，行動数 4，深さ 9 の場合，SeqBDD のノード数は 36 であるのに対し，順序付き二分木のノード数は 262,144 であり，約 7,282 倍の圧縮効果が得られている．この結果から，SeqBDD を用いた全列挙方策は，重複する行動列の部分構造を共有することで，計算資源の節約と効率的な探索が可能となることが示された．

表 3　行動数 4 の時の全列挙方策におけるノード数 (SeqBDD / 順序付き二分木)

行動数＼深さ	3	5	7	9
2	6/8	10/32	14/128	18/512
3	9/27	15/243	21/2187	27/19683
4	12/64	20/1024	28/16384	36/262144

5.7　提案手法のメモリ使用量比較

各行動次元数における全列挙方策において，SeqBDD を用いた場合と用いない場合のメモリ使用量を比較する．表 4，5，6 に各行動数における SeqBDD と順序付き二分木のメモリ使用量を示す．SeqBDD を用いた場合，メモリ使用量が大幅に削減されていることがわかる．例えば，行動数 4，深さ 9 の場合，SeqBDD のメモリ使用量は 1.97KB であるのに対し，順序付き二分木のメモリ使用量は 16,384.02KB であり，約 8,320 倍の削減効果が得られている．この結果から，SeqBDD を用いた全列挙方策は，重複する行動列の部分構造を共有することで，計算資源の節約と効率的な探索が可能となることが示された．

表 4　行動数 2 の時の全列挙方策におけるメモリ量使用量の比較 (単位: KB)

深さ	3	5	7	9
順序付き二分木	0.52	2.02	8.02	32.02
SeqBDD	0.33	0.55	0.77	0.98

表 5　行動数 3 の時の全列挙方策におけるメモリ量使用量の比較 (単位: KB)

深さ	3	5	7	9
順序付き二分木	1.70	15.20	136.70	1230.20
SeqBDD	0.49	0.82	1.15	1.48

表 6　行動数 4 の時の全列挙方策におけるメモリ量使用量の比較 (単位: KB)

深さ	3	5	7	9
順序付き二分木	4.02	64.02	1024.02	16384.02
SeqBDD	0.66	1.09	1.53	1.97

6　むすび

本研究では，マルコフ決定過程における方策改善のために，将来の到達可能な状態遷移をより広く考慮する手法として，SeqBDD を用いた行動列集合の列挙と圧縮を提案した．従来のロールアウトに基づく手法では，有望と判断された一部の遷移のみをサンプリングすることで計算量を抑える一方，類似したロールアウトが繰り返し生成され，探索および評価の重複が生じるという問題があった．これに対し，本研究では決定論的な MDP を対象として，h ステップ先までのすべての行動列を SeqBDD により表現することで，共通する部分列を共有し，冗長な探索・評価を削減する枠組みを示した．現在多く用いられている深層強化学

習手法においてSeqBDDを用いた全列挙方策を組み合わせることで評価が向上した．これは，従来のサンプリングベースの方策改善手法とは異なる観点から，探索空間全体の構造を活用するアプローチである．一方で，本研究では決定論的なMDPに限定して議論を行っており，確率的遷移を含む一般的なMDPや行動が連続値の環境での実験は今後の課題である．また，hの増加に伴うSeqBDDのノード数や計算時間の増大については，さらなる理論的解析と実験的評価が必要である．今後は，サンプリング手法とのハイブリッド化や連続値などを含むより一般的かつ実用的な方策改善手法へと発展させること，本手法の有効性を示すためより多くの環境での実験を行う事が課題である．

参考文献

[1] Dimitri Bertsekas. *Dynamic programming and optimal control: Volume I*, volume 4. Athena scientific, 2012.

[2] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

[3] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

[4] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International journal of robotics research*, 37(4-5):421–436, 2018.

[5] Elsa Loekito, James Bailey, and Jian Pei. A binary decision diagram based approach for mining frequent subsequences. *Knowledge and Information Systems*, 24(2):235–268, 2010.

[6] Hongzi Mao, Mohammad Alizadeh, Ishai Menache, and Srikanth Kandula. Resource management with deep reinforcement learning. In *Proceedings of the 15th ACM workshop on hot topics in networks*, pages 50–56, 2016.

[7] Shin-ichi Minato. Zero-suppressed bdds for set manipulation in combinatorial problems. In *Proceedings of the 30th International Design Automation Conference*, pages 272–277, 1993.

[8] 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*, 518(7540):529–533, 2015.

[9] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[10] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model.

Nature, 588(7839):604–609, 2020.

[11] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[12] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[13] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[14] Xiangyu Zhao, Long Xia, Liang Zhang, Zhuoye Ding, Dawei Yin, and Jiliang Tang. Deep reinforcement learning for page-wise recommendations. In *Proceedings of the 12th ACM conference on recommender systems*, pages 95–103, 2018.