2048 の研究

山下修平

2023 年 11 月 28 日

目次

第 1 章

序論

1.1　はじめに

第 2 章

2048

2.1　2048 のルールと用語説明

2048 は， Gabrirle Cirulli によって公開された 1 人用のパズルゲームである [1]．ゲームは 16 マスからランダムに選ばれた 2 マスに 2 か 4 の数字タイルが置かれた盤面から始まる．プレイヤが行うことは上下左右いずれかの方向を選択することである．プレイヤがある 1 つの方向を選ぶと， 盤面上のすべての数字タイルは選択した方向に向かってスライドして移動する．スライドする数字タイルは空きマスを通過し， 異なる数字タイルの直前か盤面の端で停止する．スライドして移動する際に 2 つの同じ数字のタイルが衝突すると， これらは合体してその合計の数字の 1 つのタイルへ変化し， プレイヤはその数値を得点として獲得する．そのため， ゲームには 2 の累乗の数字タイルしか現れない．図 2.1 にある盤面から上下左右を選択したときの， 数字タイルのスライドの仕方の具体例を示す．

数字タイルのスライド後， 空きマスから等確率に選択されたある 1 マスに 90% の確率で 2 のタイルが， 10% の確率で 4 のタイルが置かれる．ゲームはプレイヤの行動による数字タイルのスライドと新たな数字タイルの出現を交互に繰り返して進行する．盤面上の数字タイルが市松模様のようになると， プレイヤが選択可能な行動がなくなったときにゲームは終了する (図 2.2 を参照)．

ここでプレイヤが行動を選択する盤面を**状態**， 行動を選択して新たな数字タイルが出現する直前の盤面を *afterstate* と呼ぶ．図 2.3 に状態 s から afterstate s' を経由して， 次の状態 s_{next} に遷移する例を示す．

プレイヤの一般的な目標はゲームのタイトルが示す $2^{11} = 2048$ のタイルを完成させることだが， それ以降もゲームを続けることができる．

2.2　ゲームの進行と時刻

2048 はゲームの性質上， 状態から afterstate への遷移において盤面上の数字タイルの合計値は不変である (図 2.1 を参照)．盤面上の数字タイルの合計値は afterstate から次の状態への遷移においてのみ変

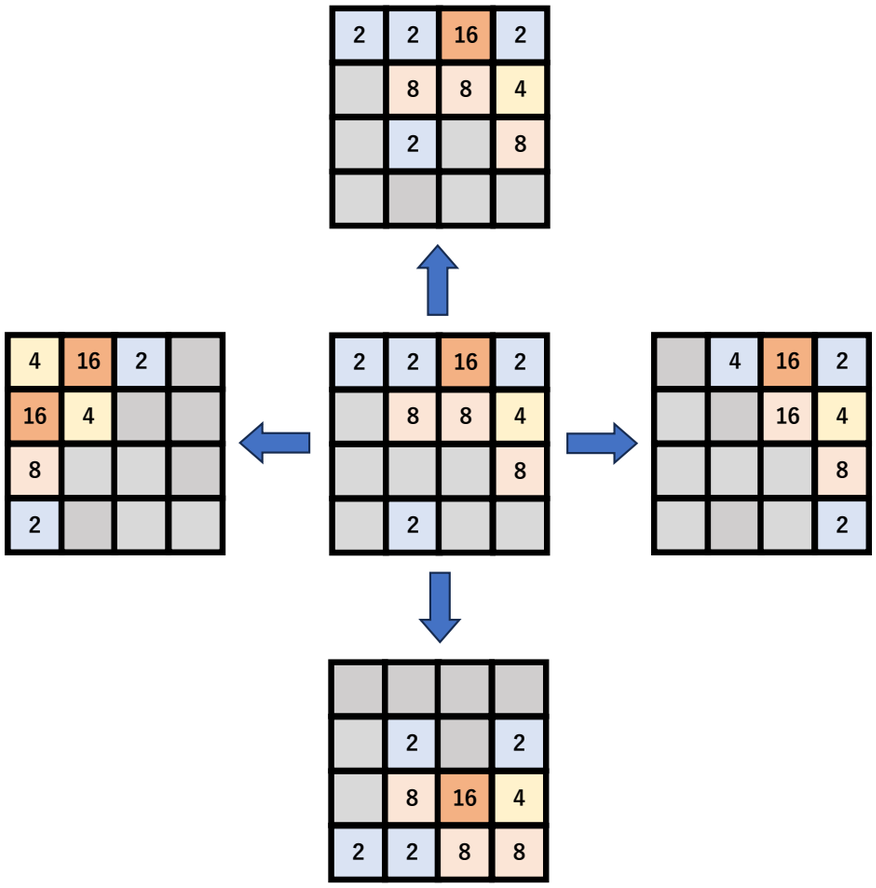

図 2.1: 上下左右それぞれへのスライドの例

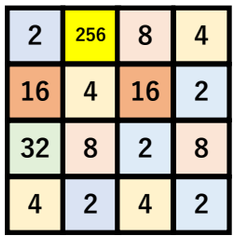

図 2.2: 終了状態の例

化する．新しい数字タイルとして 2 か 4 のタイルが出現することで，数字タイルの合計値はその値の分だけ必ず増加する．すなわちプレイヤが 1 回行動するたびに，盤面上の数字タイルの合計値は 2 か 4 ずつ単調に増加する．

よって盤面上の数字タイルの合計値をゲームの進行度合いとして用いることができる．以降これを**時刻**と呼ぶ．例えば図 2.3 では時刻 $2 \times 4 + 4 + 8 \times 3 + 16 = 52$ の状態 s が時刻 $52 + 2 = 54$ の状態 s_{next} に

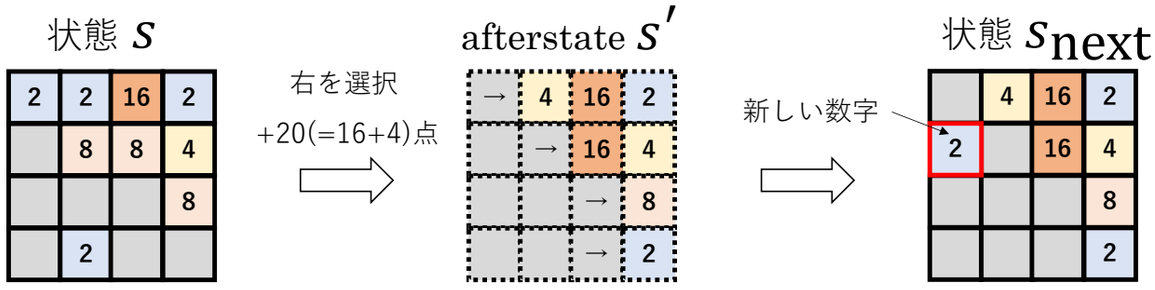

図 2.3: 状態遷移の例

遷移している．ゲームの時刻はプレイヤが行動するたびに必ず増加するため， 2048 はサイクルの出現しないゲームであることがわかる．

2.2.1 ゲームの終了状態

またゲームの開始盤面をまとめて初期状態と呼ぶことにする．

第 3 章

2048 と完全解析

3.1　2048 の完全解析とは

2048 は 1 人用のゲームであるため，勝敗のようなプレイヤの明確な目標は存在しない．そのためプレイヤが何を目標とするかによって，プレイヤの最善手の定義は変化する．また 2.1 節で述べたようにゲームはランダム性を伴うため，同じ状態から毎回同じ手を選んでも結果は確率的に変動する．

そこで本稿ではある状態 s における最善手を「s から獲得できる得点の合計の期待値が最も高くなるような手」と定義する．これは多くの上手な人間や AI が 2048 のタイルを完成させることに留まらず，より多くの得点を獲得することを目標としていることから妥当な定義であると考えられる．状態 s から最善手を選び続けて獲得できる得点の合計の期待値を状態 s の**価値**と呼び，$V(s)$ で表すことにする．このとき $V(s)$ は式 3.1 のように再帰的な形式で書くことができる．ただし $r(s,a)$ は状態 s から行動 a をとって獲得する得点，$s_{\text{next}} \in \mathcal{T}(s,a)$ は状態 s から行動 a をとって遷移しうる次の状態の集合を表す (図 3.1 を参照)．

$$V(s) = \begin{cases} 0 & (s \text{ が終了状態}) \\ \max_a \left(r(s,a) + \mathbb{E}_{s_{\text{next}} \in \mathcal{T}(s,a)} V(s_{\text{next}}) \right) & (\text{otherwise}) \end{cases} \tag{3.1}$$

ゲームに現れうるすべての状態の価値を計算すれば，任意の状態において最善手を選ぶことができる．本稿ではこれを 2048 の完全解析ということにする．

完全解析によって，最善手を選び続けるプレイヤの戦略やゲームの初期状態の価値を知りたい人は少なくないだろう．また 2048 を対象とした強化学習の研究は数多くあり，完全解析によってそれらの良し悪しを定量的に評価することができる．一方で，2048 を完全解析することはそのゲーム木の大きさから現状難しいと考えられる．そこで以降では本来 4×4 盤面上で行われる 2048 のミニゲームとして，盤面サイズを縮小した 2048 を完全解析することを考える．

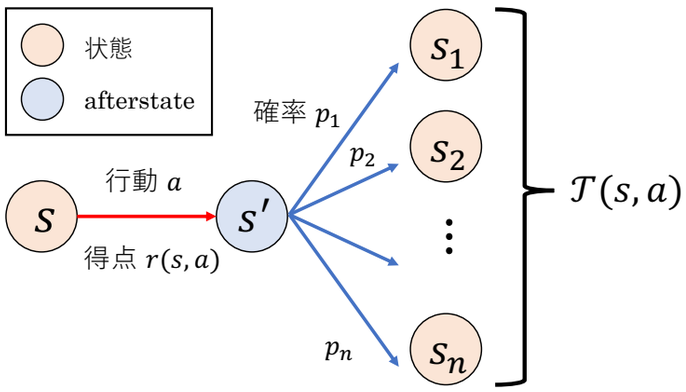

図 3.1: 式 3.1 の補足図

3.2　2048 のミニゲームの完全解析

基本的なルールは 2048 と同じで盤面サイズを 4×4 から縮小したゲームを完全解析することを考える．盤面サイズに関わらず，以下の 2 つのステップを順番に行うことで 2048 を完全解析することができる．

1. ゲームに現れうるすべての状態の列挙
2. 列挙した状態の価値の計算

3.2.1 節と 3.2.2 節で具体的な方法について述べる．なお本節の内容は文献 [2] および文献 [3] を元に執筆された．

3.2.1　幅優先探索によるすべての状態の列挙

完全解析の第 1 ステップとして幅優先探索によってゲームに現れうるすべての状態を列挙する．まず初期状態をキューに詰めて探索を開始する．キューの先頭の状態 s を取り出し，s から遷移可能な次の状態 $s_{\text{next}} \in \mathcal{T}(s)$ をキューに追加する．これをキューが空になるまで繰り返すことで，すべての状態を列挙することができる．

ここで $s_{\text{next}} \in \mathcal{T}(s)$ がすでに発見済みであるか確認するために，これまでに発見した状態を管理する集合が必要である．素朴な方法ではメモリでこれまでに発見した全状態を管理することで行えるが，状態数が非常に大きな場合にはメモリの容量を超えてしまう．

そこで 2.2 節で説明した時刻によってゲーム木を整理する．時刻 t の状態は時刻 $t+2$ か $t+4$ の状態にしか遷移しないため，時刻 $t+2$ と $t+4$ の発見した状態をメモリで管理すれば十分である．よって時刻が最小の 4 の状態から時刻 2 刻みで順番に列挙を行うことで，ディスクを効率的に活用することがで

Algorithm 1 幅優先探索によるすべての状態の列挙

```
1: function ENUMERATION(t)
2:     for all s_t ∈ queue_t do
3:         for all s_{t+2} ← s_t do
4:             queue_{t+2}.push(s_{t+2})
5:         end for
6:         if element > max then
7:             max ← element
8:         end if
9:     end for
10:    return max
11: end function
```

きる．以上を踏まえた疑似コードを Algorithm 1 に示す．

3.2.2　後退解析による状態の価値の計算

3.2.1 節で列挙した状態の価値を式 3.1 に従って計算する．時刻 t の状態の価値は，時刻 $t+2$ と $t+4$ の状態の価値が計算済みであれば計算できる．よって時刻が最大の状態から順番に走査することで，効率的にすべての状態の価値を計算できる．

第 4 章

2048 と強化学習

完全解析はゲームの”答え”を知ることであり， 多くのプレイヤの関心の対象である．一方で 4×4 盤面の 2048 を完全解析することは難しく，本章では強化学習について説明する．

4.1 強化学習の概要

まず本節では 2048 との関係を踏まえつつ， 一般的な強化学習の概要について記述する．なお全体に文献 [4] を参照して書かれた．

4.1.1 マルコフ決定過程

強化学習はエージェントが与えられた環境において獲得する報酬を最大化するための手法である．強化学習が扱う問題はマルコフ決定過程 (MDP) という枠組みによって抽象化できる．MDP は以下の 4 つの要素で構成される．状態集合と行動集合が有限であるとき有限 MDP と呼ぶ．

- 状態集合 S
- 行動集合 A
- 状態遷移関数 $p : S \times A \times S \to [0, 1]$
- 報酬関数 $r : S \times A \times S \to \mathbb{R}$

エージェントは時刻 t で状態 $s_t \in S$ から行動 $a_t \in A$ を選択する．そして確率 $p(s_{t+1}|s_t, a_t)$ で次の状態 s_{t+1} に遷移し, 即時報酬 $R_{t+1} = r(s_t, a_t, s_{t+1})$ を獲得する．エージェントが状態から行動を選択する際の確率分布 $\pi : S \times A \to [0, 1]$ を方策という．また状態遷移関数と報酬関数は環境のダイナミクスと呼ばれることがある．図 4.1 に MDP の模式図を示す．

2048 は有限 MDP にそのまま当てはまるゲームである．すなわち行動集合 A は上下左右に対応し， 報酬はプレイヤが獲得する得点に対応する．

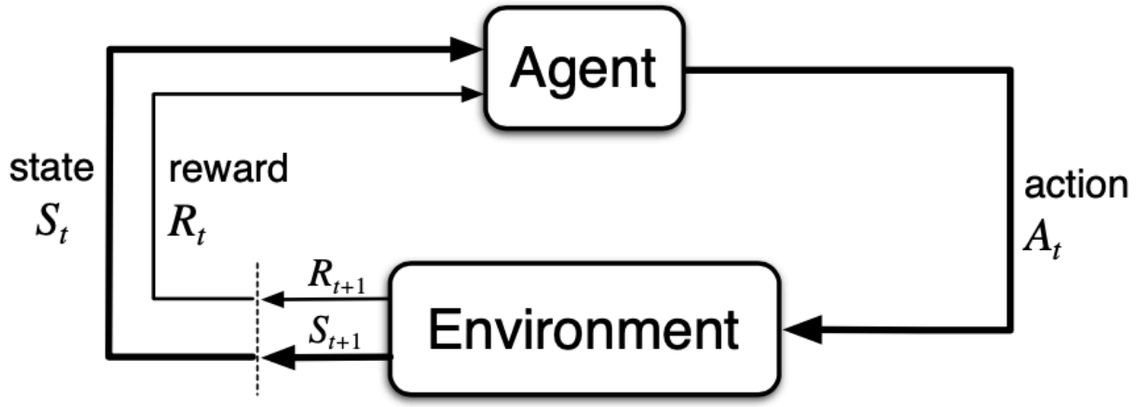

図 4.1: MDP の模式図 (文献 [4] より引用)

4.1.2 強化学習の目標と価値関数

一般に強化学習で扱う問題は, エージェントと環境のやり取りが終わる終了状態が存在する episodic task と終了状態が存在しない continuing task が存在する. episodic task ではエージェントと環境のやり取りを初期状態から終了状態までのエピソードと呼ばれる単位で分割することができる. 2.2 節で説明したように 2048 は必ず終了するゲームであるため, 以降 episodic task での定義を確認する.

強化学習の目標は $G_t = \Sigma_{k=0}^{T} \gamma^k R_{t+k+1}$ とすると 1 エピソードの累積報酬, すなわち G_0 の期待値を最大化するような方策 (最適方策) を学習によって見つけることである. 以降 G をリターンと呼ぶ. ここで割引率 γ は将来獲得する報酬が現在の状態にどれだけの影響を与えるかを決定する値であり, γ が 0 に近づくほどエージェントは現在獲得できる即時報酬を最大化し, 1 に近づくほど将来の報酬も考慮するようになる.

状態価値関数 $v_\pi(s)$ は状態 s から方策 π に従って行動を選択し続けた場合のリターンの期待値であり, 次のように定義される.

$$v_\pi(s) \stackrel{\text{def}}{=} \mathbb{E}_\pi \left[G_t | S_t = s \right] = \mathbb{E}_\pi \left[\sum_{k=0}^{T} \gamma^k R_{t+k+1} | S_t = s \right] \tag{4.1}$$

同様に状態 s から行動 a を選択し, その後方策 π に従って行動を選択し続けた場合のリターンの期待値である状態行動価値関数の定義は以下のようになる.

$$q_\pi(s, a) \stackrel{\text{def}}{=} \mathbb{E}_\pi [G_t | S_t = s, A_t = a] = \mathbb{E}_\pi \left[\sum_{k=0}^{T} \gamma^k R_{t+k+1} | S_t = s, A_t = a \right] \tag{4.2}$$

この定義の下で，ある状態 s の価値とその次の状態 s' の価値の間の関係は次のベルマン方程式によって記述される．

$$v_\pi(s) = \sum_{a \in A} \pi(a|s) \sum_{s' \in S} p(s'|s,a)[r(s,a,s') + \gamma v_\pi(s')] \tag{4.3}$$

$$q_\pi(s,a) = \sum_{s' \in S} p(s'|s,a) \left[r(s,a,s') + \gamma \sum_{a' \in A} \pi(a'|s') q_\pi(s',a') \right] \tag{4.4}$$

すなわち方策 π に従った際のある状態の価値は，次の状態の価値と即時報酬の合計を環境のダイナミクスと方策の確率分布で期待値を取ったものである．状態行動価値についても同様である．

すでに述べたように，良い方策とは獲得するリターンの期待値が大きいような方策である．また状態価値 $v_\pi(s)$ とは状態 s から方策 π に従って行動を選択し続けた場合のリターンの期待値である．

よって 2 つの方策 π と π' があるとすると，すべての状態 $s \in S$ について $v_\pi(s) \geq v_{\pi'}(s)$ が成り立つならば π は π' よりも良い方策だと言える．ここから最適方策 π_* はすべての方策の中で最も状態価値関数および状態行動価値関数が大きいような方策であると定義できる．

$$v_{\pi_*}(s) = \max_\pi v_\pi(s) \quad \text{for all } s \in S \tag{4.5}$$

$$q_{\pi_*}(s,a) = \max_\pi q_\pi(s,a) \quad \text{for all } s \in S \text{ and } a \in A(s) \tag{4.6}$$

さらに式 4.5，4.6 から次のベルマン最適方程式を導出することができる．

$$v_{\pi_*}(s) = \max_a \sum_{s'} p(s'|s,a)[r(s,a,s') + \gamma v_{\pi_*}(s')] \tag{4.7}$$

$$q_{\pi_*}(s,a) = \sum_{s'} p(s'|s,a)[r(s,a,s') + \gamma \max_{a'} q_{\pi_*}(s',a')] \tag{4.8}$$

式 4.7 は最適方策 π_* の下での状態 s の価値は，すべての行動 $a \in A(s)$ について a を選択した場合の遷移後の状態 s' の価値と即時報酬の合計を環境のダイナミクスについて期待値を取ったものの最大値であるということを示している．式 4.8 も同様に解釈できる．

最適方策の状態価値関数 $v_{\pi_*}(s)$, 状態価値関数 $q_{\pi_*}(s,a)$ をそれぞれ最適状態価値関数，最適状態行動価値関数という．

4.2　2048 に対する強化学習の先行研究

付録 A

実装の詳細

A.1　ゲーム環境の実装

A.2　完全解析の実装

A.3　強化学習の実装

参考文献

[1] G Cirulli. 2048, available from ⟨http://gabrielecirulli.github.io/2048/⟩, 2014.

[2] 山下修平, 金子知適, 中屋敷太一. 3 × 3 盤面の 2048 の完全解析と強化学習の研究. ゲームプログラミングワークショップ 2022 論文集, 第 2022 巻, pp. 1–8, nov 2022.

[3] 山下修平, 金子知適. 4 × 3 盤面の 2048 の完全解析. ゲームプログラミングワークショップ 2023 論文集, 第 2023 巻, pp. 1–5, nov 2023.

[4] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.