卒業論文

ゲーム「2048」のプレイヤについて

08-152021　**金澤望生**

指導教員　山口和紀 教授

2018 **年** 1 **月**

東京大学教養学部学際科学科総合情報学コース

Copyright © 2018, Kanazawa Nozomu.

概要

インターネットブラウザやスマートフォン上で遊ぶことのできるパズルゲーム「2048」をプレイする AI の改良を行った．改良には盤面上で最も大きな数のタイルが隅にあることを重視する独自のヒューリスティック「corner bonus」を使用した．(仮)

キーワード　ゲーム AI，機械学習

目次

第 1 章　導入 1

第 2 章　先行研究の紹介 2

2.1　Szubert & Jaskowski (2014) 2

2.2　Wu et al. (2014) 3

第 3 章　本研究のアイディア 7

第 4 章　提案と実装 8

第 5 章　実験 9

第 6 章　考察と結論 10

謝辞 11

参考文献 12

付録 A 13

第 1 章

導入

モチベーションや 2048 の基本ルール・指標について説明します.

第 2 章

先行研究の紹介

既存研究が使用している手法とプレイヤの成績について説明します.

2.1 Szubert & Jaskowski (2014)

Szubert & Jaskowski は，TD 学習を用いたプレイヤの訓練と n タプルネットワークを用いた価値関数の表現を組み合わせることによって，人間の知識やゲーム木探索を使用しないで十分強い 2048 プレイヤを実装することに成功した。

2.1.1 TD 学習

TD 学習の「TD」とは temporal difference の略であり，すなわち状態間における価値の差分を学習することによって学習器の訓練を行う手法である。2048 にあてはめると，とある盤面 s' の価値と，その盤面の 1 プレイ後の盤面 s'_{next} の価値の差分を取り，これを現状定まっている s' に足し込んでいくことで訓練を行うことになる。TD 学習にはさまざまな派生があるが，Szubert & Jaskowski が使用している TD(0) 学習は以下の式によって表現される：

$$V(s) \leftarrow V(s) + \alpha(r + V(s'') - V(s))$$

この式において，V は価値関数，α は学習率，r は報酬である。学習率は計算された差分を価値関数の更新にどれほど反映するかを決定するパラメータである。

TD 学習は Tesauro によるバックギャモンへの適用でよく知られるようになり，碁やオセロ，チェスにおけるゲーム AI の方策決定の手法として用いられるようになった。

2.1.2 n タプルネットワーク

TD 学習によって盤面の評価とその学習を行うことができるが，盤面と評価値をどのように結びつけるかが問題になる。まず，2048 で有り得るすべての盤面に対して評価値を与える 1 対 1 対応のルックアップテーブル（LUT）を作成することを考えると，2048 で有り得る盤面の数は $(4 \times 4)^{18} \approx 4.7 \times 10^{21}$ と膨大な数になり，このような LUT を計算機上で実装するこ

とは現実的に不可能である。

そこで，一部のマスの組み合わせによる「タプル」というクラスターを作成し，さらに複数のタプルを組み合わせることで盤面を表現する手法「n タプルネットワーク」を 2048 に導入することが，Szubert & Jaskowski によって提案された。たとえば，下記のような n タプルネットワークを実装した場合，1 つのゲーム内で保持すべき重みの数は 860625 であり，全ての有り得る盤面に対する LUT を保持するのに対して非常に少なくて済む。

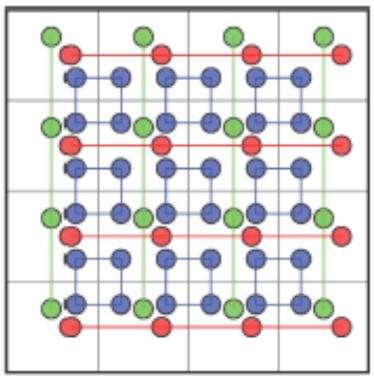

図 2.1. n タプルネットワークの例

n タプルネットワークは Bledsoe & Browning (1959) によりパターン認識に用いられたのが最初の採用例である。ゲーム AI の分野では Jaskowski (2014) によってオセロに適用され，一定の成果が得られた。

2.1.3 結果と課題

本手法をもとに行った実験のうち，最も良い勝率を達成したプレイヤを用いて 10 万ゲーム中の成績を検証したところ，勝率は 0.9781 であり，平均スコアは 100,178 であった．1 ゲーム中に達成されたスコアで最も良かったのは 261,526 であった．Szubert & Jaskowski による新たな手法は探索ベースの手法よりも大幅に高速で，かつ成績が良かった．

しかしながら，この手法では「常に 2048-tile を生成すること」よりも「時々 16384-tile を生成すること」を重視しているため，勝率は必ずしも 100% を達成できていない．また，人間の知識を一切導入していないため，最も大きな数のタイルが盤面上の端に配置されないなど，人間の直感的な戦略とは反しているといったデメリットがあった．

2.2 Wu et al. (2014)

Wu は，Szubert & Jaskowski の手法を改良し，木探索を用いた先読みと組み合わせることによってさらに良いプレイヤを実装することに成功した．

2.2.1　n タプルネットワークの配置の改善

Wu は，Szubert & Jaskowski が考案した n タプルネットワークのうち，直線型で 4 タプルとして配置していたタプルを，図 2.2.（b）のように柄杓型の 6 タプルに変更した．これによって増える重みの数は約 2 倍程度であったが，この変更によって Szubert & Jaskowski のものよりも飛躍的に良い成績を得ることができた．なお，なぜこのようなタプルの配置が最善だと判断したのかについて，Wu は論文において言及していない．

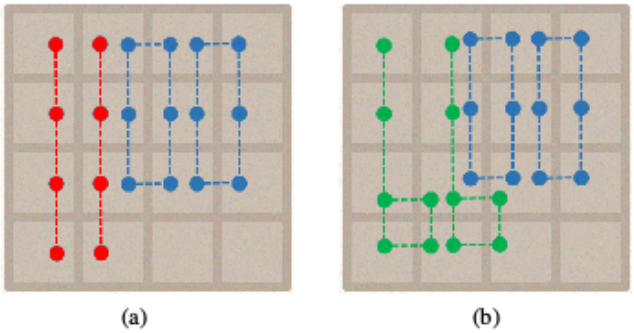

図 2.2. Szubert & Jaskowski（a）と Wu（b）が提唱した n タプルネットワーク

2.2.2　Multi-Stage TD 学習の導入

Multi-Stage TD 学習（MS-TD 学習）とは，ゲームの局面に応じて異なる価値関数を保持することによって，よりそれぞれの局面に対して適切な重みを学習させることを目的とした手法である．Wu は学習のプロセスを 3 つのステージに分割し，ゲームプレイも同様に 3 つのステージに分割して行うようにした．Wu の提唱した 2048 における MS-TD 学習は，以下のような手順で学習を行う．なお，「T_{16k}」とは「そのゲーム中で初めて 16384-tile を生成することに成功した時」，「T_{16+8k}」とは「そのゲーム中で初めて 16384-tile を生成した後に，初めて 8192-tile を生成することに成功した時」のことを示す．

1. 第 1 ステージにおいては，初期盤面からゲームを始めて，価値関数が十分飽和するまで学習を行う．このステージで学習された価値関数の重みのことを「Stage-1 価値関数」と呼ぶことにする．また，学習ゲーム中に T_{16k} を達成したなら，その時の盤面を全て保存しておく．
2. 第 2 ステージにおいては，第 1 ステージで保存した盤面からゲームを始めて，TD 学習を行う．このステージで学習された価値関数の重みのことを「Stage-2 価値関数」と呼ぶことにする．また，学習ゲーム中に T_{16+8k} を達成したなら，その時の盤面を全て保存しておく．
3. 第 3 ステージにおいては，第 2 ステージで保存した盤面からゲームを始めて，TD 学習

を行う．このステージで学習された価値関数の重みのことを「Stage-3 価値関数」と呼ぶことにする．

その後，以下のような手順でゲームプレイを行う．

1. 盤面が T_{16k} を達成するまでは，Stage-1 価値関数を用いてゲームプレイを行う．
2. 盤面が T_{16k} を達成してから T_{16+8k} を達成するまでは，Stage-2 価値関数を用いてゲームプレイを行う．
3. 盤面が T_{16+8k} を達成してからは，Stage-3 価値関数を用いてゲームプレイを行う．

2.2.3 Expectimax 木探索

Szubert & Jaskowski (2014) によって，木探索ベースの 2048 プレイヤは強化学習で訓練したプレイヤに劣ることが示されたが，Wu は強化学習によるプレイヤに対して Expectimax 木探索を補助的に組み合わせることによって，さらにプレイヤの性能を高めようと試みた．

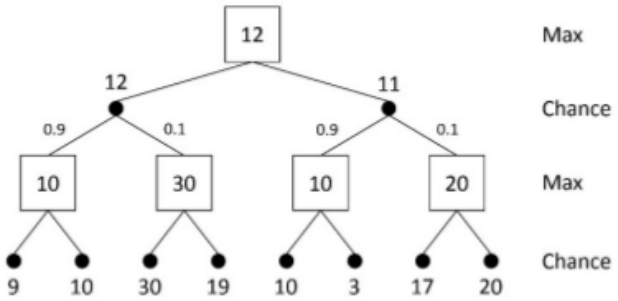

図 2.3. Expectimax 木の例 (Wu et al. (2015))

Expectimax 木探索には，Max ノードと Chance ノードという 2 種類のノードがあり，それぞれのノードの値は子ノードから決定される．Max ノードの値は，子ノードのうち最も大きな値のノードの値となる．例えば図 2.3. の根ノードの値は 12 であるが，これは子ノードの「12」と「11」のうち最大の値である 12 を取ったものである．一方で Chance ノードの値は子ノードの期待値となる．例えば図 2.3. の根ノードの子ノードの 1 つである「12」というノードは，0.9 の確率で 10 となるノードと 0.1 の確率で 3 となるノードの期待値，すなわち $0.9 \times 10 + 0.1 \times 3 = 12$ によって 12 という値が決定する．

Wu の提案においては，Max ノードの値はプレイヤがアクションを選択して遷移を行った後の盤面，Chance ノードは遷移を行った後にランダムタイルを発生させた後の盤面が与える評価値となる．例えば，ある盤面 s（アクション選択と遷移が終わった直後の盤面とする）の評価値を深さ 3 の Expectimax 木探索を用いて求めたい時，図 2.4. のような探索木が考えられる．s の盤面が分かれば，その子ノードであるランダムタイル生成後の盤面，さらにその子ノードである遷移後の盤面を求めることができる．さらに，葉ノードにあたる Chance ノード

の値は価値関数が与える評価値とすることで，各ノードの値を求めることができ，最終的に根ノード，すなわち評価値を求めたい盤面 s の評価値も求められるということになる．

（図 2.4. 何かいい感じの 2048 の探索木を自分で描画して貼る）

2.2.4　結果と課題

Wu によるプレイヤは Szubert & Jaskowski のものに比べて著しく良い成績を達成した．まず Szubert & Jaskowski が達成できなかった 32768-tile の生成に成功し，10.9% の確率で 32768-tile を生成できるようになった．勝率に関しては 1 を達成，すなわち 2048-tile は 100% の確率で生成できるようになり，平均スコアは 328,946，最大スコアは 605,752 を記録した．これは当時としては 1 つの例外[*1]を除き，計算機による 2048 プレイヤの中で最も優れた成績であった．

一方で，n タプルネットワークの形状変更や MS-TD 学習のステージングについては，この研究で行なわれた調整についてこれといった根拠が述べられておらず，依然として改良の余地は残していた．特に n タプルネットワークの形状変更については，次の Oka & Matsuzaki の研究で詳しく検討されることとなった．

[*1] Xiao による深深度先読みと人間による調整を行った評価関数を用いたプレイヤがこれにあたるが，Wu のものよりも 100 倍遅い：https://www.youtube.com/watch?v=JQut67u8LIg

第 3 章

本研究のアイディア

本研究で導入しようとしている手法のアイディアについて説明します.

第 4 章

提案と実装

前章で説明したアイディアの具体的な提案とその実装方法を説明します.

第 5 章

実験

提案したアイディアの実験結果と既存研究の実験結果を比較します.

第 6 章

考察と結論

実験結果をもとに，結果の考察を行い，本研究をまとめます.

謝辞

謝辞を書きます.

参考文献

[1]
[2]

付録 A

表やプログラムリストの掲載が必要になったらここに掲載します.