第57回 プログラミング･シンポジウム 2016.1.8-10

「2048」プレイヤの評価関数 1人プレイと対戦プレイでの評価

岡 和人[1,a)]　松崎 公紀[2,b)]

概要：「2048」のプレイヤとして，TD 学習を用いて作成した盤面評価関数を用いるプレイヤが有効であることが示されている．盤面評価関数には，盤面から特徴点の情報を抽出して部分評価値を計算する，部分評価関数の組が用いられている．これまでの研究では，盤面評価関数の性能は，数通りしか検討されていない．十分な学習が行えると仮定すれば，部分評価関数の特徴点の個数を増やし，部分評価関数の個数を増やすことで，より高性能なプレイヤを作ることが出来ると予想されるが，記憶容量をより多く消費する．また，盤面評価関数は部分評価関数の組み合わせによって得られるため，全ての盤面評価関数を調べ上げることは計算時間や記憶容量の問題から困難である．よって，記憶容量と性能とのバランスに優れる盤面評価関数を効率よく調べ，得られた盤面評価関数を用いるプレイヤがより強くなるかを調査する必要がある．本稿では，考えられる部分評価関数を列挙し，特徴点数が 6 の場合で部分評価関数の性能を調査した．この結果，特徴点集合が連結成分数 1 の形をとる部分評価関数の性能が高いとが見込まれることを示した．得られた高性能な部分評価関数を用いて，盤面評価関数を作成した．作成した盤面評価関数を用いたプレイヤの得点は，500 万ゲームを学習することで平均 203,769 点となった．また，性能の高い部分評価関数から，盤面からバランス良く特徴点を抽出するような盤面評価関数を作成した．個の盤面評価関数を用いるプレイヤの得点は，500 万ゲームを学習することで平均 224,562 点となり，これまで報告されている TD 学習のみを用いるプレイヤの中で最も高い得点となった．性能の高い部分評価関数を用いて，盤面からバランス良く特徴点を抽出するような盤面評価関数を用いて，GPCC（Games and Puzzles Competitions on Computers, プログラミング・シンポジウムの分科会）で提案されている「対戦型 2048」で，提案する盤面評価関数を TD 学習によって作成したプレイヤが，既存のプレイヤよりも強いことを示した．

キーワード：2048, TD 学習, 評価関数

1. はじめに

「2048」は 2014 年に G. Cirulli が公開した一人ゲームである [1]．「2048」では，4×4 の盤面が用いられ，ゲームの各ステップにおいてプレイヤが 2 のベキの書かれたタイルを動かすと空いたマスのいずれかに新しいタイルが出現する．ゲームの目標は 2048 の書かれたタイルを作ること，もしくは，タイルの併合によって得られる得点の合計を高くすることである．

「2048」のコンピュータプレイヤについて，これ

[1] 高知工科大学大学院
[2] 高知工科大学
[a)] 195061f@gs.kochi-tech.ac.jp
[b)] matsuzaki.kiminori@kochi-tech.ac.jp

までに，Minimax アルゴリズム [8]，Expectimax アルゴリズム [8]，モンテカルロ木探索 [9]，TD 学習（Temporal Difference 学習）の応用 [3][6] などが検討されている．特にその中でも，TD 学習を用いて作成した盤面評価関数を用いたプログラムは，高得点を上げている．Szubert ら [3] および Wu ら [6] は，盤面上の n マスの組（これを n タプルと呼ぶ）を複数指定し，それらに対応する部分盤面評価関数の線形和を盤面評価関数として用いている．Szubert らおよび Wu らの研究では，主に学習の方法とそのときのプレイヤの強さに着目していた．一方，n タプルの選択方法が盤面評価関数にどのように影響するかについては，これまで詳細に検討されていなかった．

そこで本研究では，n タプルを網羅的に調べることで，より高性能な盤面評価関数を作成することを目的とする．しかしながら，「2048」において取りうる n タプルの個数は多いため，それらのすべての組み合わせを調べることは現実的ではない．そこで，個々の n タプルから得られる部分盤面評価関数の性能や性質について調査し，その性能や性質をもとにより適切に n タプルを組み合わせる方法をとる．このような方法によって得られた盤面評価関数に対して TD 学習を行い，その TD 学習プレイヤの強さについて調査する．

本稿ではさらに，「2048」を二人ゲームに拡張した「対戦型 2048」についても扱う．「対戦型 2048」は 2015 年の GPCC[*1] で提案された問題の 1 つである．本研究では，一人ゲーム「2048」に対して学習させた盤面評価関数を用いて，「対戦型 2048」の攻撃側･守備側の両方のプレイヤを実装した．その結果，TD 学習によるプレイヤは，既存の Minimax プレイヤとモンテカルロ木探索プレイヤに対して大きく勝ち越した．

本研究の主な貢献は以下の 3 つである．

(1) 6 タプルにおいて，部分評価関数が持つ部分評価値の分散と部分評価関数の性能に強い相関があることを示す．

(2) 6 タプルにおいて，提案する盤面評価関数を用いた TD 学習プレイヤは，50 万ゲーム時点で平均得点 133567 点，500 万ゲーム時点で平均得点 203,769 点となることを示す．

(3) 「対戦型 2048」に，本稿で提案する盤面評価関数を用いた TD 学習プレイヤを適用し，深さ 3 まで探索するミニマックスプレイヤや，プレイアウト回数 1000 回のモンテカルロ木探索プレイヤより強いことを示す．

2. 「2048」･「対戦型 2048」のルール

2.1 「2048」のルール

「2048」は，縦 4 マス横 4 マスの 16 マスで行われる．ゲームを通じて，各マスは空きマスであるか 2 のベキの書かれたタイルを持つ．ゲームの開始状態では，2 または 4 のタイルがランダムな位置に 2 つ置かれる．ゲームの各ステップでは，以下に示すとおり，プレイヤの動作の後で新しいタイルが出現する．

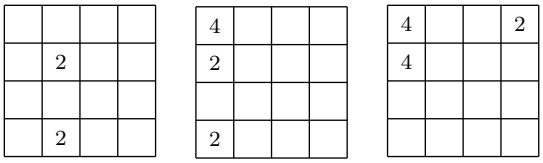
プレイヤの動作 プレイヤは，盤面全体のタイルを上下左右いずれかの方向に動かす．移動方向の先に空マスがあれば，タイルがそこへと順に移動する．同じ数を持つタイルが移動する方向に連続して 2 枚あるとき，それらのタイルが併合されて 2 倍の数を持つタイル 1 枚に変わる．ただし，同じ数を持つタイルが 3 枚並んでいる場合には移動する先からの 2 枚が併合され，4 枚並んでいる場合にはそれぞれ 2 枚ずつが併合される．プレイヤは，移動もしくは併合するようなタイルが少なくとも 1 枚存在する方向を選ばなければならない．

新しいタイルの出現 盤面の空マスのいずれかに，2 または 4 の値を持つタイルが 1 枚置かれる．タイルの置かれる位置は，全ての空マスから等確率に選ばれる．タイルの持つ数は，90%の確率で 2 であり，10%の確率で 4 となる．

プレイヤが動かせる方向が無いときゲームが終了する．

プレイヤが動作を行った際，タイルが併合され新しいタイルができると，その新しいタイルの持つ値を得点として得る．プレイヤの 1 動作によっ

[*1] GPCC（Games and Puzzles Competitions on Computers）は情報処理学会プログラミング・シンポジウムの分科会である．URL=http://hp.vector.co.jp/authors/VA003988/gpcc/gpcc.htm



ゲームの初期状態．2 のタイルが 2 箇所に置かれている．

左に動かし，左上に 4 のタイルが出現．

上に動かし 4 のタイルができ，4 点を得る．右上に 2 のタイルが出現．

図 1 初期状態からの盤面の遷移の例

て複数のタイルが同時にできたときは，新しいタイルの値の和が得点となる．もとの「2048」における目標は 2048 を値として持つタイルを作ることであるが，本研究ではゲーム中に獲得する総得点をできるだけ大きくすることとする．

初期状態からプレイヤが 2 ステップの盤面の遷移の例を図 1 に示す．

2.2 「対戦型 2048」のルール

「対戦型 2048」は，「2048」を二人ゲームに拡張したものである．二人のプレイヤは，攻撃側と守備側をそれぞれ 1 回ずつ行い，（守備側で得た）得点の多いほうが勝ちとなる．各ゲームは全てのマスが空いている状態で攻撃側から始め，ゲーム終了まで以下の手順を交互に繰り返す．

攻撃側 盤面の空いているマスに 2 のタイルを 1 つ置く．「対戦型 2048」は「2048」と異なり，4 のタイルを置くことは出来ない．[*2]

守備側 「2048」におけるプレイヤと同じ動作を行い，「2048」の場合と同様に得点を得る．

守備側が動かせる方向が無くなったとき，ゲームの終了とする．

3. n タプルに基づく盤面評価関数と TD 学習プレイヤ

3.1 盤面の対称性

「2048」では，ある盤面を 90°，180°，270° だけ回転させた盤面，またそれらの鏡像である盤面は，元の盤面と本質的に等価である．例えば，図 2 に

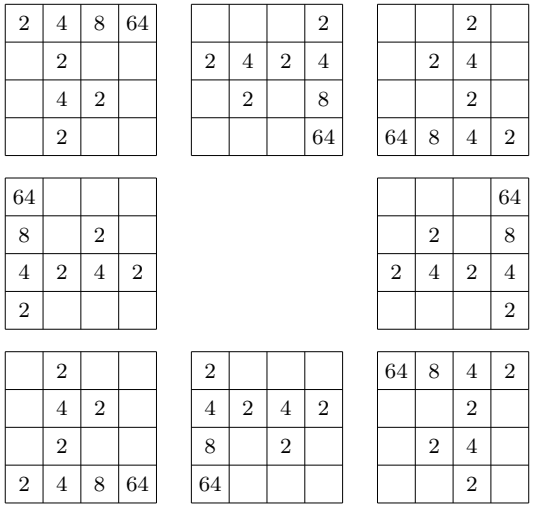


図 2 回転，鏡像変換によって得られる盤面の例

示される 8 つの盤面は互いに等価である．以降，本論文においては，ある盤面 S からこれらの 8 つの盤面を作る関数を $f_i(S)$ （ただし，$1 \leq i \leq 8$）と記述する．

3.2 n タプルに基づく盤面評価関数とプレイヤの行動選択

Szubert ら [3] は，複数の n タプルに基づく盤面評価関数を用いている．本研究でも，それと同じ考え方を用いて盤面評価関数を設計する．その基本的な考え方は，各 n タプルに対応するタイルの値の組から部分評価値を求め，それらの和をとって盤面評価関数を求めるというものである．

第 3.1 節で述べたとおり，「2048」におけるある盤面に対して，他に等価な盤面が 7 つある．逆に考えると，1 つの n タプルは，ある盤面の 8 つの位置に対応付けられる．これにより，少ない n タプルでより多くの盤面の特徴を捉えることができ，実際，対称性を考慮しない盤面評価関数を用いた場合よりも強いプレイヤの作成ができることが確認されている [3]．

ある盤面を S，および，n タプルで示された位置のタイルの値から部分評価値を返す関数を V_j（ただし，$1 \leq j \leq m$，m は用いる n タプルの個数）とする．このとき，全体の盤面評価関数 W は，部分評価値の和

[*2] 本研究では，2015 年の GPCC の当初の仕様に従う．その後，攻撃側の置くタイルに 4 がある場合について検討されている．

(1) 2	(2) 4	8	64
(3)	2		
	4	2	
	2		

(1)	(2)	(3)	評価値
0	0	0	419.7
0	0	1	477.3
⋮	⋮	⋮	⋮
1	**2**	**0**	**854.0**
⋮	⋮	⋮	⋮

図 3 3 タプルに基づく部分評価関数の例．対応するタイルの値 2^k の指数を取ってテーブルを引くことで，部分評価値 854.0 を得る．

$$W(S) = \sum_{j=1}^{m} \sum_{i=1}^{8} V_j(f_i(S))$$

によって与えられる．「2048」に表れるタイルの値は 2 のベキである．本研究では，その上限が $2^{15} = 37768$ であると仮定し，さらに空マスが 2^0 であると捉える．すると，部分評価値を返す関数 V_j は，n タプルに対応するタイルの数 2^k の指数 k を並べて得られる 16 進数をインデックスとするテーブル参照によって取得することができる．

図 3 に，ある 3 タプルに対応する部分評価関数の例を示す．n タプルの要素間の順序は本質ではないが，実装の便宜のため番号を振っている．

このように定義した盤面評価関数を用いて，プレイヤは，動作によって得られる得点と動作直後の盤面の盤面評価関数の値との和が最大となるような動作を選択する．すなわち，ある盤面 S において，可能な動きの集合が $A(S) \subset \{\mathsf{N}, \mathsf{E}, \mathsf{S}, \mathsf{W}\}$，そのうちの動作 a によって得られる得点が $R(S, a)$，動作直後の盤面を $N(S, a)$ とすると，プレイヤの選択する動作は次の式によって与えるられる．

$$\arg\max_{a \in A(S)} \left(R(S, a) + W(N(S, a))\right)$$

「対戦型 2048」においても，上で定義した盤面評価関数を用いてプレイヤの動作を選択する．「対戦型 2048」の防御側においては，プレイヤは先に述べた「2048」における行動選択と同じアルゴリズムを用いて行動を決定する．一方，「対戦型 2048」の攻撃側は，防御側の総得点を最小化することが目的であるため，盤面評価関数の値が最も小さくなる位置に値が 2 のタイルを置く．

3.3 TD 学習による部分盤面評価値の決定

TD 学習は，強化学習として一般的に用いられる手法の 1 つである．TD 学習では，現在の状態と，実際に行動した際の状態と報酬との和を比較し，その誤差を 0 に近づけていく方法で学習を進める．ゲームの言葉では，ある時刻での状態の評価値と，実際にプレイした後の状態の評価値とそこで得られる報酬（もしくは勝ち負け）の和を比較することになる．TD 学習はこれまで，バックギャモン [4] やオセロ [5] などの評価関数の作成に適用され，有用であることが示されている．

前節の n タプルに基づく盤面評価関数において，n タプルに対応する部分評価値を TD 学習によって決定する．ゲームの各ステップにおいて得られる得点を報酬として TD 学習すると，ある盤面に対する盤面評価関数の値が，その盤面から進めて（適切に行動して）得られる得点の和を近似すると期待される．その仮定のもとでは，前節のプレイヤの行動選択は，行動によって得られる得点とその結果の盤面から得られる得点の和が最大となるものを選ぶことに等しく，合理的である．

TD 学習を用いた「2048」プレイヤ（以下，TD 学習プレイヤと呼ぶ）のアルゴリズムを示す．ある時刻 t における盤面を S_t とする．TD 学習プレイヤは，前節のアルゴリズムによって，動作後の盤面に対する盤面評価関数の値と得点の和が最大となるような動作 a を選択する．そのときの得点を $r = R(S_t, a)$，動作直後の盤面を S'_t とする [*3]．このとき，1 時刻前の動作直後の盤面評価関数に対する TD 誤差 d は

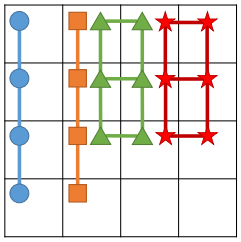
$$d = r + W(S'_t) - W(S'_{t-1})$$

で与えられる．この TD 誤差をもとに，n タプルに対応する部分盤面評価値 $V_j(S_{t-1})$ を同じ割合で更新し，$V'_j(S_{t-1})$ とする．

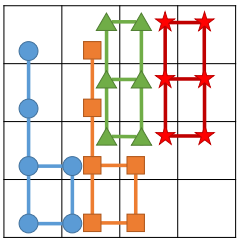
$$V'_j(S_{t-1}) = V_j(S_{t-1}) + \alpha \frac{d}{8m}$$

ただし，α は学習率であり，本研究では 0.08 とした．

[*3] S'_t と S_{t+1} は異なるので注意．S'_t の空マスにタイルを 1 つ追加されたものが S_{t+1} となる．



Szubert らが用いた 2 つの 4 タプルと 2 つの 6 タプル [3]



Wu らが用いた 4 つの 6 タプル [6]

図 4 先行研究で用いられた盤面評価関数を構成する n タプル

4. n タプルの数え上げ

Szubert ら [3] および Wu ら [6] の研究では，図 4 に示す 4 タプルまたは 6 タプルに基づく盤面評価関数が用いられている [*4]．これらの n タプルは，各著者らによって人手で選ばれたものであるが，以下の観点から確かに有望そうである．

- 「2048」においては，大きな数は端のほうに集めておくのがよい．その性質は，図 4 の青（丸）や赤（星形）で示される n タプルによって特徴付けられる．
- 「2048」では，タイルは縦または横方向に動き，連続したタイルのみが併合する．したがって，縦または横方向に連続してどのような数のタイルがあるかは重要な性質である．

しかしながら，n タプルについて網羅的に調べることにより，これらのヒューリスティックで見逃してしまいそうな，より有望な n タプルを発見できるかもしれない．また機械的に n タプルを調べる方法であれば，他の n に対する n タプルについても同様にして選択することができることができる．

そこで，本研究ではまず，16 個のマスから任意の n 個のマスを選んだ際にできる n タプルを網羅することとした．ただし，回転や鏡像のもとで等価な n タプルは重複して数えないこととした．一方で，縦方向もしくは横方向に連続したマスが連結であるとみなしたときに，n 個のマスすべてが連結であることは要求せず，すべての可能性を列挙することとした．

回転や鏡像の対称性を法として全ての n タプルを列挙し，タプルの特徴点の個数 n と連結成分数とで分類してその個数を示したのが表 1 である．全ての n タプルの個数は 8547 個であり，連結成分数が 1 であるようなものに限っても 1524 個ある．また，6 タプルの総数は 1051 と多いが，連結成分数が 1 であるような 6 タプルに限ると 68 とかなり小さいことが分かる．

本稿では最終的に，10 個の 6 タプルを組み合わせた盤面評価関数を作成する [*5]．それでも，連結成分数が 1 であるような 6 タプル 10 個を選ぶ全ての組み合わせはおよそ 3000 億通りあり，すべてを試すのは現実的ではない．そこで，まず各 n タプルについて，それらが有望そうであるかどうかをそれほど多くない数の実験から評価し，その後，有望そうな n タプルを選んで盤面評価関数を作成することとする．

5. 部分評価関数の性能調査

連結成分数 1 に限らないすべての 6 タプルについて，対応する部分評価関数の性能・性質について調査する．ここでは，部分評価関数の性能と評価値の分散という 2 つの観点で調査を行う．

部分評価関数の性能 ある盤面評価関数が m 個の n タプルによって与えらえているとする．そのとき，その盤面評価関数を用いた TD 学習プレイヤが得た得点 P が，各 n タプルに対応する値 P_j の和となる，すなわち，$P = \sum_{j=1}^{m} P_j$ であると仮定する．このとき，P_j を対応する部分評価関数の性能と定義する．

評価値の分散 各 n タプルには，最大で 16^n 個の評価値が対応する．正しく学習が行われたならば，より有用な n タプルほど，それらの評価値の値の分散が大きくなるはずである．ただし，評価値の中には出現しえないような数に対応するものもあるため，学習が一度も行

[*4] Szubert ら [3] は，回転や鏡像を考慮しない定義のもとでの 17 個の 4 タプルによる実験結果も示している．

[*5] 1 つのタイル（と空白）の取りうる値が 16 通りであるとき，そのような盤面評価関数を実現するには $10 \times 16^6 =$ 約 1.6 億 (160M) 個の値を格納することが必要となる．これは，各評価値に倍精度浮動小数点数を用いて約 1.3 GB である．

表 1 存在する n タプルの個数

n	全て	特徴点の集合の連結成分数							
		1	2	3	4	5	6	7	8
全て	8547	1524	2744	2606	1254	351	61	6	1
1	3	3							
2	21	4	17						
3	77	8	29	40					
4	252	17	74	99	62				
5	567	33	136	220	133	45			
6	1051	68	254	372	257	80	20		
7	1465	119	383	524	309	107	19	4	
8	1674	195	522	573	290	74	18	1	1
9	1465	261	544	469	149	38	3	1	
10	1051	300	456	238	49	7	1		
11	567	257	241	64	5				
12	252	169	76	7					
13	77	66	11						
14	21	20	1						
15	3	3							
16	1	1							

われなかった評価値については，分散の計算から除く．

以下，調査の方法について詳しく述べる．1051 個の 6 タプルから乱数を用いて重複なく 8 つを選び，それらを用いた TD 学習プレイヤで 500,000 ゲームについて学習を行う．誤差の影響を小さくするため，最終的に TD 学習プレイヤが得た得点 P は，最後の 1000 ゲームの平均得点とする．これを 1 セットの実験として，全体で 2000 セットの実験を行う．部分評価関数の性能については，2000 セットの実験で得られた得点 P に対して，最小二乗法によって計算する．評価値の分散については，各セットの実験で，8 つの 6 タプルそれぞれについて分散を求め，6 タプルごとに実験全体でその分散の平均を求める．実験に用いた計算機は次のものである．

CPU	Intel Xeon E5645 (2.4GHz, 6 コア) × 2
OS	Linux 64bit 2.6.18–194.el5
メモリ	12 GB

本実験では，実験に必要な時間を短縮するため，部分評価値のテーブルを共有する 12 個のスレッドで並列にゲームのプレイと学習を行った [*6]．

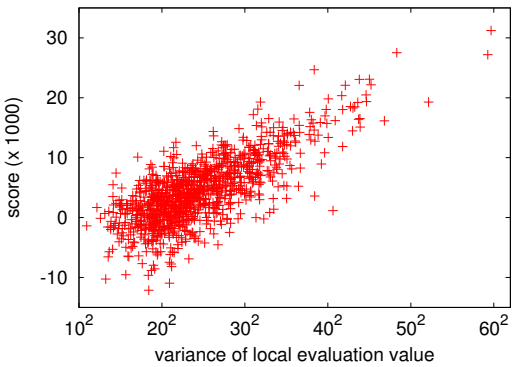
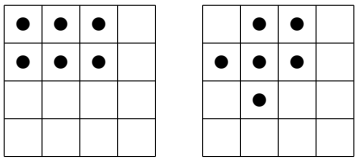


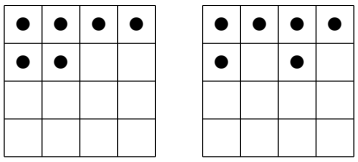
図 5 6 タプルの場合の，部分評価関数の性能と部分評価値の分散

実験で得られた部分評価関数の性能と部分評価値の分散を，図 5 に散布図にて示す．部分評価関数の性能と部分評価値の標準偏差との相関係数は 0.75 となり，強い正の相関がみられる．したがって，部分評価値の分散を計算することでも，部分評価関数の性能が評価できることが期待される．

[*6] 著者の調査で，並列にゲームのプレイと学習を行うことにより，並列に行わない場合に比較して数%程度平均得点が低下することが判明している．

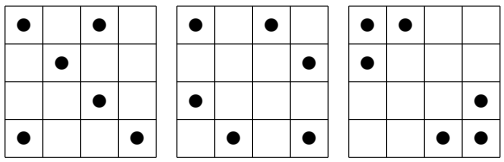


性能：31214.03　性能：27572.41
分散：3560.70　分散：2333.92
連結成分数：1　連結成分数：1



性能：27187.94　性能：19285.69
分散：3513.63　分散：2718.82
連結成分数：1　連結成分数：1

図 6 性能の高い，または部分評価値の分散が大きい部分評価関数



平均：-12135.40　平均：-10264.40　性能：-1382.03
分散：339.11　分散：174.83　分散：119.54
連結成分数：6　連結成分数：6　連結成分数：2

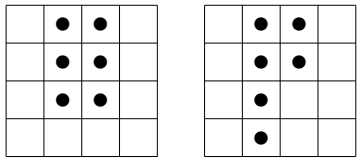
図 7 性能が低いまたは部分評価値の分散が小さい部分評価関数

(比較的少ない実験回数のもとでは，最小二乗法によってフィッティングした部分評価関数の性能のほうが誤差が大きく，部分評価値の分散を n タプルに対応する部分評価関数の性能の指標とすることが有用である可能性がある.)

部分評価関数の性能が高い，または部分評価値の分散が大きい 6 タプル 4 個を図 6 に示す.

部分評価関数の性能が低いまたは部分評価値の分散が小さい 6 タプル 3 個を図 7 に示す.

最後に，図 4 に示されている Wu らが用いた 6 タプルのうち，すでに示されていない 2 つの 6 タプルについて図 8 に示す．これらの結果より，Wu らの研究で用いられた 4 つの 6 タプルは，特に性能の高い 2 個の 6 タプルと，比較的性能の高い 1 個の 6 タプル，中程度の性能をもつ 1 個の 6 タプルからなっていることが分かる.



性能：23072.30　性能：13714.11
分散：1917.31　分散：1293.37
連結成分数：1　連結成分数：1

図 8 Szubert らによる研究 [3] と Wu らによる研究 [6] で用いられた他の部分評価関数

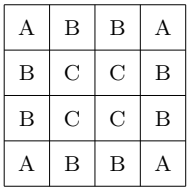


図 9 盤面の分割

6. 提案する盤面評価関数の性能

高い性能の盤面評価関数を作成するため，連結成分数が 1 である部分評価関数の性能を調査し，高性能な部分評価関数を用いて盤面評価関数を作成する．連結成分数 1 である部分評価関数の性能調査は，300 セット行うこととし，残りの実験条件は節 5 と同じとした．高性能となった部分評価関数を用いて，盤面評価関数を作成する．作成した盤面評価関数の性能を調査する.

特徴点数が 6 の部分評価関数を 10 個用いて 2 通りの盤面評価関数を作成する．盤面評価関数 (a) は，部分評価値の分散の大きい順に部分評価関数 10 個を選択する．盤面評価関数 (b) は，図 9 に示すように盤面を A, B, C に分割し，部分評価関数の特徴点が A, B, C に何個あるかによって部分評価関数をグループに分け，グループ内で重複して選ばない前提で，部分評価値の分散の大きい順に部分評価関数を選択する．作成した盤面評価関数 (a)，(b) それぞれの性能を調査する．1 セットとして 5,000,000 ゲームを学習し，1000 ゲーム毎に平均得点を調査する．実験は，それぞれの盤面評価関数で 1 セットづつ行う.

作成した盤面評価関数とその性能を示す．盤面評価関数 (a) を図 10 に示す．線によってつなが

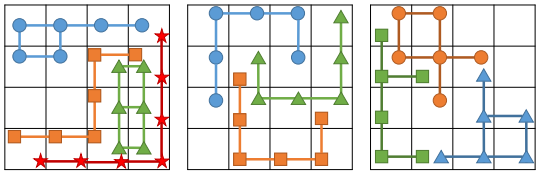


図 10 盤面評価関数（a）に用いた部分評価関数

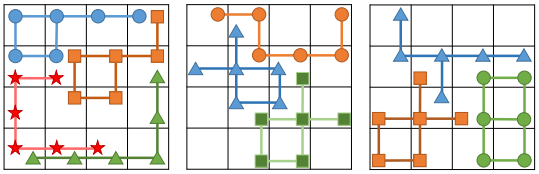


図 11 盤面評価関数（b）に用いた部分評価関数

れた 6 個の記号が 1 個の n タプルを意味する．盤面評価関数（b）を図に示す．

得られた得点を図 12 に示す．盤面評価関数（a）は，500,000 ゲーム時点で，平均得点 120,310.68 となり，5,000,000 ゲーム時点で平均得点 203,768.58 となった．盤面評価関数（b）は，500,000 ゲーム時点で，平均得点 133,566.55 となり，5,000,000 ゲーム時点で平均得点 224,561.84 となった．また，盤面評価関数に用いる部分評価関数の個数が異なるため，単純な比較を行うことはできないが，盤面評価関数（b）は，これまで TD 学習で用いられてきた盤面評価関数 [3], [6] では，最も高い平均得点を記録している．

盤面評価関数（b）は，盤面評価関数（a）に比較して，500,000 ゲーム時点と 5,000,000 ゲーム時点の両方で，高い性能である．このことは，盤面評価関数の性能は，用いる部分評価関数の性能の和となるという，著者らの仮説を否定している．これは，盤面評価関数に用いる部分評価関数同士には，盤面評価関数（b）に用いる部分評価関数を選択する際に用いた n タプルのバランスなど，相性のような何らかの関係があり，これが盤面評価関数の性能に影響していることを示していると著者らは考える．

7. 対戦型 2048 への適用

これまでに，「対戦型 2048」のプレイヤとして，ミニマックスプレイヤや，著者らが作成したモンテカルロ木探索プレイヤがある [9]．本節では，TD 学習プレイヤを「対戦型 2048」に適用し，モンテカルロ木探索プレイヤと対戦させることでその強さを調べる．

ミニマックスプレイヤは，ミニマックス法を用いて行動を決定するプレイヤである．既定の深さまで探索を行い，根からそのノードまでの得点を評価値としてミニマックス法によって行動を決定する．防御側のノードでは，可能な移動から最も評価値の高い移動を選択し，攻撃側のノードでは，タイルを置くことが可能なマスから最も評価値の低いマスを選択する．本実験では，1–3 の深さで実験を行う．

モンテカルロ木探索プレイヤは，モンテカルロ木探索をもちいて行動を決定するプレイヤである．モンテカルロ木探索は，モンテカルロ法を木探索に応用したアルゴリズムである．モンテカルロ木探索プレイヤは，プレイアウトと呼ばれる乱数を用いたシミュレーションによって行動を評価し，効率的に木探索を行う．モンテカルロ木探索プレイヤの着手決定アルゴリズムやパラメータは，基本的に著者らが過去に用いたモンテカルロ木探索プレイヤと同じとし，プレイアウト回数を 1000 回 [*7]，ノード展開の閾値は 10 回とし，UCT アルゴリズムを用いて，有望なノードを探索し，そのノードからプレイアウトを行う．

本実験では，TD 学習プレイヤ TDL は攻撃側，防御側でそれぞれ独立した盤面評価関数をもつ．用いる盤面評価関数は，6 で作成した盤面評価関数（b）とする．ランダムに着手を決定するプレイヤと対戦し，50,000 回対戦したを学習したのち，ミニマックスプレイヤ MiniMax，モンテカルロ木探索プレイヤ MCTS と対戦することを 10 回行った．実験の結果は 10 回の対戦で得られた得点の平均とする．実験の結果を図 13 に示す．TD 学習を用いるプレイヤは，他の既存のプレイヤに対して大きな差をつけて勝った．特に，TD 学習プレイヤが攻撃側の場合に，防御側のプレイヤの得点を低く抑えている．ただし，TD 学習プレイヤが防御側の場合では，ミニマックスプレイヤの探索

[*7] 計算時間を比較的公平にするために，プレイアウト回数を 1000 回とした

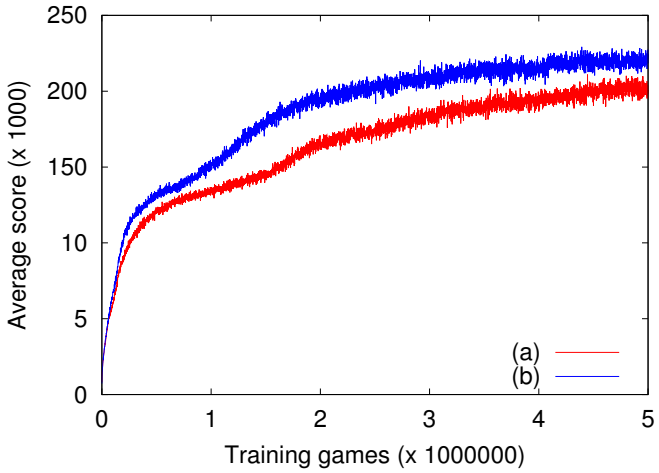


図 12 提案する盤面評価関数の性能

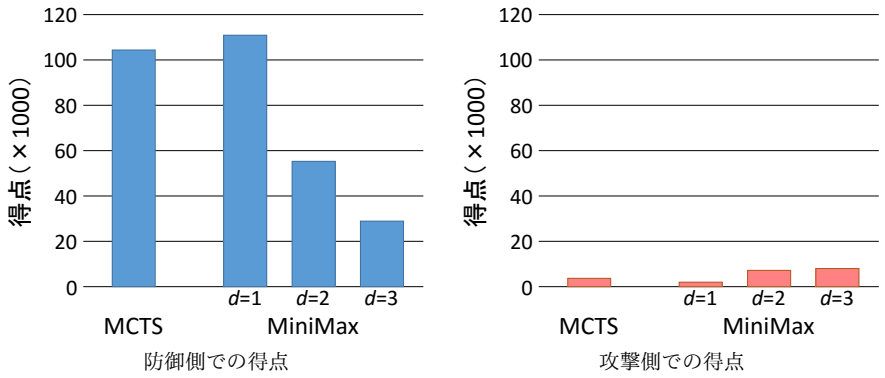


図 13 TD 学習を用いるプレイヤの性能

の深さが 3 になると，探索の深さが 1 の場合に比較して，得点が約 26%まで減少する．これは，TD 学習プレイヤが学習したランダムに 2 を置くプレイヤの戦略が，実際に戦っている攻撃側プレイヤの戦略と異なるためであると著者らは考える．

「対戦型 2048」で TD 学習プレイヤをを向上させるためには，TD 学習プレイヤが学習する対戦相手のプレイヤの戦略を強くすることや，複数の対戦相手から学習を行い，様々な盤面での盤面評価値を学習することが必要であると著者らは考える．

8. 関連研究

「2048」に関する研究として，その着手決定アルゴリズムが複数研究されている．Zaky は，ゲームアルゴリズムとして初歩的な MiniMax アルゴリズムを採用したプレイヤで深さ 4 まで探索する場合，2048 のタイルが出来る可能性は 37%であったと報告している [8]．しかし，MiniMax アルゴリズムでは，自分の手番でないとき，つまり 2 のタイルが置かれる場所がプレイヤにとって最も不利な場所であるとして計算している．これを，各状態の生起確率によって重みづけしたアルゴリズムが Expectimax アルゴリズムであり，そのアルゴリズムを用いたプレイヤが Xiao によって作成されている [7]．深さ 3 まで探索する Expectimax アルゴリズムを採用したプレイヤが，2048 のタイルを作る可能性は 80%であると Zaky が報告している [8]．また，プレイアウトと呼ばれる乱数を用いたシミュレーションを，ゲーム木探索に応用したアルゴリズムとして，モンテカルロ木探索アルゴ

リズムがある．Rodgers らはモンテカルロ木探索アルゴリズムを採用したプレイヤの強さを調査した．また，Rodgers らは，Expectimax アルゴリズムを並列化した Averaged Depth-Limited Search（ADLS）アルゴリズムについても，そのプレイヤの強さを調査している [2]．Rodgers らの報告によると，1 手を 1 秒で選択する場合，モンテカルロ木探索プレイヤの平均得点は 11,000 点程度，ADLS プレイヤの平均得点は 36,000 点程度であった．また，1 手を 16 秒で選択する場合，モンテカルロ木探索プレイヤの平均得点は 40000 点程度，ADLS プレイヤの平均得点は 77,000 点程度であった．

「2048」において，現在までに最も成果を挙げているのが，機械学習アルゴリズムの一つである Temporal Difference Learning（TD 学習）を応用したプレイヤである．Szubert らは TD 学習によって調整した評価関数が出力する評価値を用いて，着手を決定するプレイヤを作成した [3]．1,000,000 ゲーム分のデータから学習を行った評価関数を用いることで，2048 のタイルが出来る確率は 97.81%となり，平均得点は 100,178 点となったと報告している．また，Wu らは，TD 学習を複数ステージ化する拡張を提案している [6]．ステージ化の深さを 3 とし，Expectimax アルゴリズムによって深さ 5 まで探索したとき，2048 のタイルが出来る確率を 100%，平均得点が 328,946 点であったと報告している．

9. まとめ

本稿では，1 人用ゲームである「2048」の TD 学習プレイヤに用いられる盤面評価関数を改善し，プレイヤの強さを向上させた．高性能な盤面評価関数を作成するため，部分評価関数の性能を調査した．部分評価関数を特徴点数と連結成分数で分類し数え上げ，性能を調査する部分評価関数を限定した．調査によって明らかにした高性能な部分評価関数を用いて，盤面評価関数を構成した．

提案した盤面評価関数を TD 学習によって作成することで，500,000 ゲームの学習で平均得点は 133,566.55 点となり，5,000,000 ゲーム学習することで，平均得点は 224,561.84 点となった．

作成した盤面評価関数を用いる「対戦型 2048」の TD 学習プレイヤを作成した．ランダムにプレイするプレイヤと対戦し，学習した TD 学習プレイヤが，ミニマックス法を用いて深さ 3 まで探索するプレイヤや，プレイアウト回数を 1000 回に設定したモンテカルロ木探索プレイヤよりも強いことを示した．

謝辞

本稿に掲載した研究成果は，その多くを高知工科大学の IACP クラスタ計算機を使用して得られたものである．

参考文献

[1] G. Cirulli, 2048. `http://gabrielecirulli.github.io/2048/`, 2015.

[2] P. Rodgers and J. Levine, An Investigation into 2048 AI Strategies. *2014 IEEE Conference on Computational Intelligence and Games*, pp. 1–2, 2014.

[3] M. Szubert and W. Jaśkowski, Temporal Difference Learning of N-Tuple Networks for the Game 2048. *2014 IEEE Conference on Computational Intelligence and Games*, pp. 1–8, 2014.

[4] G. Tesauro, TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural computation*, Vol. 6, No. 2, pp. 215–219, 1994.

[5] M. van der Ree and M. Wiering. Reinforcement learning in the game of Othello: Learning against a fixed opponent and learning from self-play. *IEEE Symposium on Adaptive Dynamic Programming And Reinforcement Learning (ADPRL)*, pp. 108–115, 2013.

[6] I.-C. Wu, K.-H. Yeh, C.-C. Liang, C.-C. Chang, H. Chiang, Multi-Stage Temporal Difference Learning for 2048. *Technologies and Applications of Artificial Intelligence*, LNCS 8916, pp. 366–378, 2014.

[7] R. Xiao, nneonneo/2048-ai — GitHub. `https://github.com/nneonneo/2048-ai`, 2015.

[8] A. Zaky, Minimax and Expectimax Algorithm to Solve 2048. `http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2013-2014-genap/Makalah2014/MakalahIF2211-2014-037.pdf`, 2014.

[9] 岡 和人, 松崎 公紀, 原口 和也, 対戦型 2048 の網羅的解析とモンテカルロ木探索プレイヤ. **高知工科大学紀要**, Vol. 12, No. 1, pp. 123–130, 2015.