対戦型2048の網羅的解析とモンテカルロ木探索プレイヤ

岡　和人[1]　松崎　公紀[2*]　原口　和也[3]

(受領日：2015 年 5 月 11 日)

[1] 高知工科大学大学院工学研究科
〒 782–8502　高知県香美市土佐山田町宮ノ口 185

[2] 高知工科大学情報学群
〒 782–8502　高知県香美市土佐山田町宮ノ口 185

[3] 小樽商科大学商学部
〒 047–8501 北海道小樽市緑 3 丁目 5 番 21 号

[*] E-mail: matsuzaki.kiminori@kochi-tech.ac.jp

要約：「対戦型 2048」は、2014 年に公開された一人ゲーム「2048」を、二人ゲームに拡張したものである。本稿では、「対戦型 2048」に対して網羅的解析とモンテカルロ木探索プレイヤの作成を行う。網羅的解析によって、手数が増えると局面増加比が減少することが明らかになり、また、ゲーム終了までの最小手数とその手順を発見した。モンテカルロ木探索プレイヤの実装では、単純なモンテカルロ木探索プレイヤに加えて、木の合流を考慮するモンテカルロ木探索プレイヤを作成する。プレイアウト回数 10000 回のモンテカルロ木探索プレイヤと、既存の評価関数を用いる MiniMax プレイヤとを対戦させることにより強さの評価を行う。モンテカルロ木探索プレイヤ同士の対戦において、木の合流を考慮したプレイヤの得点は平均で 2036.2 点多く、プレイヤの強さが向上した。また、木の深さとプレイアウトにおいて展開したノード数との関係からその理由について考察する。

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)]。ゲームは、縦 4 マス横 4 マスの盤面で行われる。ゲームの進行とともに 2 または 4 の書かれたタイルが出現し、プレイヤはルールに従ってタイルを動かし、より大きな数を持つタイルへと結合させる（図 1）。ゲームの目標は 2048 もしくはそれよりも大きな数のタイルを作ることである。本稿では「2048」を二人ゲームに拡張した「対戦型 2048」を扱う。「対戦型 2048」は 2015 年の GPCC[1] の問題になっている。

「対戦型 2048」は、二人完全情報確定ゲームの一つである。二人完全情報確定ゲームには、チェス、将棋、囲碁などがあり、それらはこれまでにも多くの研究が行われている。しかし、「対戦型 2048」は、これらのゲームと比べると探索空間は小さいものの、ゲーム木がより深いという性質がある。そのため、これまでに研究されてきた手法が必ずしも有効であるとは限らない。

4	2	2	8
2	1024	1024	16
8		8	2
		2	

図 1. 2048 の局面の例

[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。http://hp.vector.co.jp/authors/VA003988/gpcc/gpcc.htm

そこで本研究では、まず、「対戦型2048」に対して網羅的解析を適用して、そのゲーム木の特徴を調べる。次に、囲碁で大きな成果を上げた着手決定アルゴリズムであるモンテカルロ木探索[2)]を用いて、「対戦型2048」のプレイヤを作成する。

本稿における貢献は以下の2点である。

- 「対戦型2048」に網羅的解析を適用し、そのゲーム木の局面数について調べた。その結果、局面の合流が非常に多く、局面数の増加の割合は手数とともに減少することが分かった。また、最短で詰みに至る手順を発見した。
- モンテカルロ木探索プレイヤを作成し、その強さや木探索の深さや幅を調査した。作成したモンテカルロ木探索プレイヤは既存プレイヤに対してより多くの得点を取ることを確認した。また、モンテカルロ木探索プレイヤが探索する深さと幅について調査した。

本稿の残りの構成は以下のとおりである。第2節では、「対戦型2048」のルールについて説明する。第3節では、網羅的解析を適用し、その結果について考察する。第4節では、「対戦型2048」にモンテカルロ木探索を適用する手法について説明する。第5節では、作成したモンテカルロ木探索プレイヤの強さについて評価実験を行い、結果を考察する。関連研究について第6節で述べた後、第7節で本稿のまとめを述べる。

2. 「対戦型2048」のルール

「対戦型2048」では、2つのプレイヤが攻撃側と守備側に分かれ、攻守を交代して合計2ゲームを行う。各ゲームの終了時に守備側が得点を得て、2ゲーム終了時に得点が多い方が勝ちとなる（得点は下で定義される）。以下に「対戦型2048」のルールの詳細を示す。

盤面の大きさ ゲームは、縦4マス横4マスの16マスで行われる。

ゲームの開始 全てのマスが空いている状態で、攻撃側からゲームを始める。

1ターンの流れ 以下の2手順をゲームが終了するまで交互に繰り返す。パスはできない。

攻撃側 盤面の空いているマスに2と書かれたタイルを1つ置く（X列Y行に2と書かれたタイルを置く手を“(X, Y)”と表す）。[2]

守備側 盤面全体の数字を上下左右いずれかの方向に動かす（上下左右へ移動する手それぞれを“N”,“S”,“W”,“E”と表す）。ただし、盤面上のタイルがまったく移動・変化しないような方向は選べない。

ゲームの終了 守備側が動かせる方向が無いとき詰みと言い、そこでゲームが終わる。

タイルの併合 同じ数の書かれたタイルが移動する方向に沿って2個並んでいる場合には、2倍の数のタイル1個に置き換えられ、空いたマスはつめられる。このとき、連鎖はしない。3個並んでいる場合には、移動する先から2個が併合される。4個並んでいる場合は、それぞれ2個ずつが併合される。

得点 タイルの併合によって新しくタイルが出来たとき、新しいタイルに書かれた数だけ得点が増加する。例えば、局面に2048が1つあることによる得点は20480点である[3]。

3. 網羅的解析

8192以下の数の書かれたタイルを並べてできる局面の数は、容易に分かる回転・鏡像の対称性を除くと $14^{16}/8 \sim 2.7 \times 10^{27}$ 程度となる。これは2007年に解かれたチェッカー[3)]の局面数 5×10^{20} より多いが、非常にうまくやれば完全解析が可能であることが示唆される。そこで本研究では最初のステップとして、ゲームの開始局面から始めて各手数ごとのとりうる局面数や現実的な分岐度について調べる。

実験では、与えられた初期局面から始めて、そこから任意の手によって推移できる局面を幅優先探索にて列挙する。その際、回転・鏡像によって同一視される局面は一つとする。またプログラム開発を容易にするため、攻撃側と守備側の一対の手を組にして2手ごとに考え、詰んでいない局面数、最大得点、最小得点を調べる。

ゲームの開始局面から実行した場合の局面数を図2に、最大得点と最小得点を図3にそれぞれ示す。また、盤面数の増加の割合を図4に示す。図2と図4より、局面数は開始後指数関数的に増加するものの、その増加比は徐々に減少することが分かる。「対戦型2048」において、攻撃側は空いているマスにタイルを置け、また守備側は最大で4方向に動かせるため、各局面の分岐度は大きい。しかし、100手以上の場合には局面数の増加の割合が1.2より小さいことを見ると、全体として見ると重複する局面が多いすなわち合流が非常に多いことが分かる。最大

[2]「対戦型2048」は「2048」と異なり、4を置くことは出来ない。

[3]これは、1024個の2を併合した得点、得られた512個の4を併合した得点、（中略）得られた2個の1024を併合した得点の和である。

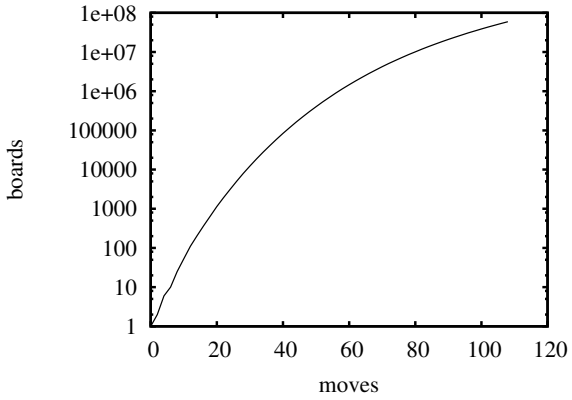

図 2. 局面数の推移

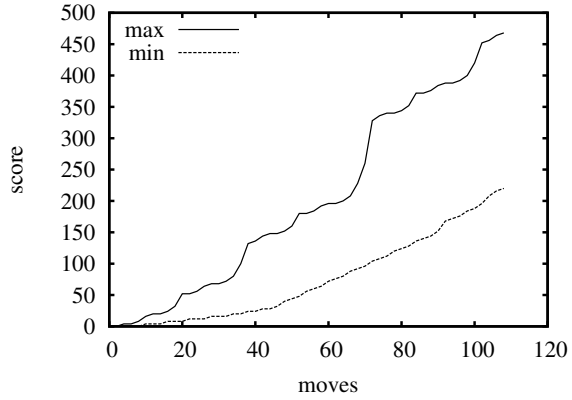

図 3. 最大得点と最小得点の推移

得点のグラフが波打っているのは、タイルの値の最大値が変わる際に (それまでよりも) 大きな得点が得られるためである。

このプログラムを動かす過程で、(双方が協力した場合に) 詰みに致る最小手数が 51 手であることが分かった。この最小手数 51 手で詰んだ局面は図 5 に示すものであり、この局面のみである。その詰み局面に至る手順は複数あるが、そのうちの一つを以下に示す。

(4,4) → W → (4,4) → W → (4,1) → W → (4,1) → W → (4,4) → E → (1,1) → E → (1,3) → E → (1,3) → E → (1,3) → E → (1,1) → W → (2,2) → S → (4,4) → N → (4,4) → N → (4,4) → S → (4,1) → S → (4,1) → S → (4,1) → N → (4,4) → N → (3,2) → N → (3,2) → N → (3,2) → S → (4,1) → S → (4,1) → N → (3,4) → W → (4,3) → N → (4,1) [詰み]

現状のプログラムでは、100 手を越えて 10^7 程度の局面数となると非常に時間がかかる。これよりも大きな局面数に対して解析を行うには手の絞り込みに加えてプログラムの並列化などの工夫が必要である。

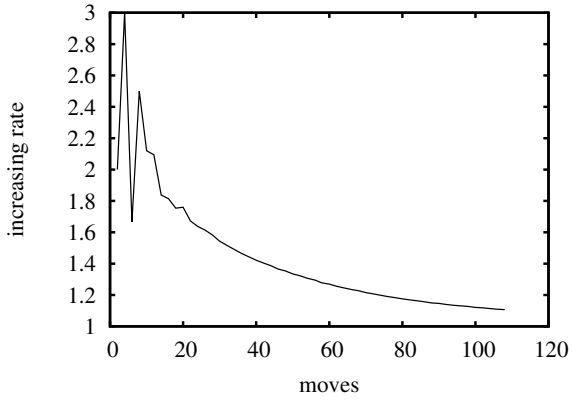

図 4. 局面の増加比の推移

4	2	4	2
2	4	2	4
4	2	8	2
2	4	2	4

図 5. 最小手数で詰んだ局面

4. モンテカルロ木探索プレイヤ

本研究では、モンテカルロ木探索アルゴリズムを適用したプレイヤを 2 種類作成した。ひとつは単純に探索木を作るもの（MCTS[4] と呼ぶ）であり、もうひとつは同一の局面を合流させた有向非巡回グラフを作るもの（MCTS2 と呼ぶ）である。

4.1 モンテカルロ木探索アルゴリズムの基礎

ゲームプレイヤに対するモンテカルロ法では、ランダムに選んだ合法手によってある局面からゲーム終了までプレイするシミュレーションを用いる。このシミュレーションをプレイアウトと呼ぶ。

モンテカルロ木探索アルゴリズムは、このモンテカルロ法を木探索に応用したものである。そのうち、UCT（UCB applied to Trees）[4] は、多腕バンディット問題の解法の UCB アルゴリズムを応用したものである。UCB アルゴリズムのうち本研究で用いる UCB1 値 [5] は以下のように定義される。それまでのプレイアウト回数の合計を n、選択肢 j に対して行ったプレイアウト回数を n_j、選択肢 j によって得られた報酬の平均を $\overline{X_j}$ とする。このとき選択肢 j の UCB1 値は、適当なバランスパラメータ c を用いて

$$\overline{X_j} + c\sqrt{\frac{2\log n}{n_j}} \quad (1)$$

によって与えられる。UCT では、この UCB アルゴリズムを用いて以下の処理を繰り返す。ステップ 2’

[4] MCTS: Monte Carlo Tree Search

表 1. 対戦結果：得点の平均と標準偏差、2048 のタイルが出来た割合

対戦の組合せ	得点の平均と標準偏差	平均得点の差	2048 が出来た割合
MCTS / evMiniMax	27806.1 (SD = 10182.2) / 18267.9 (SD = 9975.3)	9538.2	67.5% / 40.0%
MCTS2 / evMiniMax	26702.6 (SD = 9919.5) / 16696.2 (SD = 9183.8)	10006.4	63.8% / 31.2%
MCTS2 / MCTS	22190.1 (SD = 7558.1) / 20153.9 (SD = 7102.4)	2036.2	43.8% / 33.8%

の木のノードを展開する処理がモンテカルロ木探索とモンテカルロ法との主たる違いである。

アルゴリズム（**UCT の繰り返し処理**）

ステップ 1 探索木の根ノードから始めてある葉ノードに至るまで、子ノードのうち UCB1 値の最も高いノードを再帰的に選択する。

ステップ 2 ステップ 1 で得られた葉ノードの局面からプレイアウトを行う。

ステップ 2' ステップ 1 で得られた葉ノードにおいて、ある閾値回数以上のプレイアウトを行っていた場合には、1 手先の盤面を求めてノードを展開する。

ステップ 3 ステップ 2 のプレイアウトによって得られた報酬を、その葉ノードから根ノードまでの経路上のノードに反映する。

UCT では、上記の 4 ステップを任意の回数行ったのち、木の深さ 1 のノード（自分が選択する手）のうち報酬の平均値が最大となるものを選択する。

4.2 単純なモンテカルロ木探索プレイヤ

単純なモンテカルロ木探索プレイヤ MCTS は、第 4.1 節のモンテカルロ木探索を「対戦型 2048」プレイヤにそのまま適用したものである。

プレイアウトにおける報酬は、ゲーム終了時に得られる得点とする。また、プレイアウトを行う回数は 10000 回とした。UCB1 値（ステップ 1）について、守備側は得点を最大化することが目標なのでそのまま用いる。バランスパラメータは $c = 1000$ とした。一方、攻撃側は相手の得る得点を最小化することが目標なので、バランスパラメータを $c = -1000$ とし、UCB1 値が最小となるものを選択するようにした。ノードを展開するための閾値（ステップ 3）は 5 とした。

4.3 合流を考慮したモンテカルロ木探索プレイヤ

合流を考慮したモンテカルロ木探索プレイヤでは、ノードを展開する際に現れた局面が既に存在するノードと同じ局面であった場合に、すでに存在するそのノードを子ノードとするようにしたものである。したがって、探索木は有向非巡回グラフとなる。

モンテカルロ木探索アルゴリズムのうち、ステップ 1、ステップ 2、ステップ 2' には変更はない。ステップ 3 において、ある葉ノードから根ノードへプレイアウト結果を反映する際に、祖先ノードにはそれぞれ 1 回ずつしか反映しないようにしている。

5. 強さの調査

モンテカルロ木探索プレイヤの強さを対戦によって調査する。比較対象として、MiniMax アルゴリズムによる以下のプレイヤを用いる。

プレイヤ evMiniMax タイル i に書かれた数を T_i と書く。ある局面の評価値を、

$$2 \times \text{得点} - \sum_{\text{隣合うタイル } (i,j)} |T_i - T_j| \qquad (2)$$

によって計算する。このプレイヤは、守備側で数えて 3 手先の全ての手について評価値を計算し、Mini-Max 法を用いて自分に最も有利な評価値となる手を選ぶ。

5.1 プレイヤ evMiniMax との対戦

モンテカルロ木探索プレイヤ MCTS と MCTS2 をプレイヤ evMiniMax と対戦させる実験を 80 回行った。80 回の対戦によって得られた得点の平均と標準偏差、および、2048 のタイルが出来た割合を表 1 の上 2 段に示す。

プレイヤ evMiniMax との対戦では、いずれのモンテカルロ木探索プレイヤも多くの場合で 2048 のタイルを作ることができ、また相手には 2048 のタイルを作らせていない。プレイヤ MCTS と MCTS2 の得点を比較すると、守備側では 1103.5 だけ MCTS の得点が良く、攻撃側では 1571.7 だけ MCTS2 が良い得点であった。しかしこの得点差は標準偏差と比べて小さいため、続いてプレイヤ MCTS2 と MCTS とを直接対戦させ、その性能差を調査する。

5.2 プレイヤ MCTS2 と MCTS の対戦

合流を考慮したモンテカルロ木探索プレイヤが単純なモンテカルロ木探索プレイヤに比較して強く

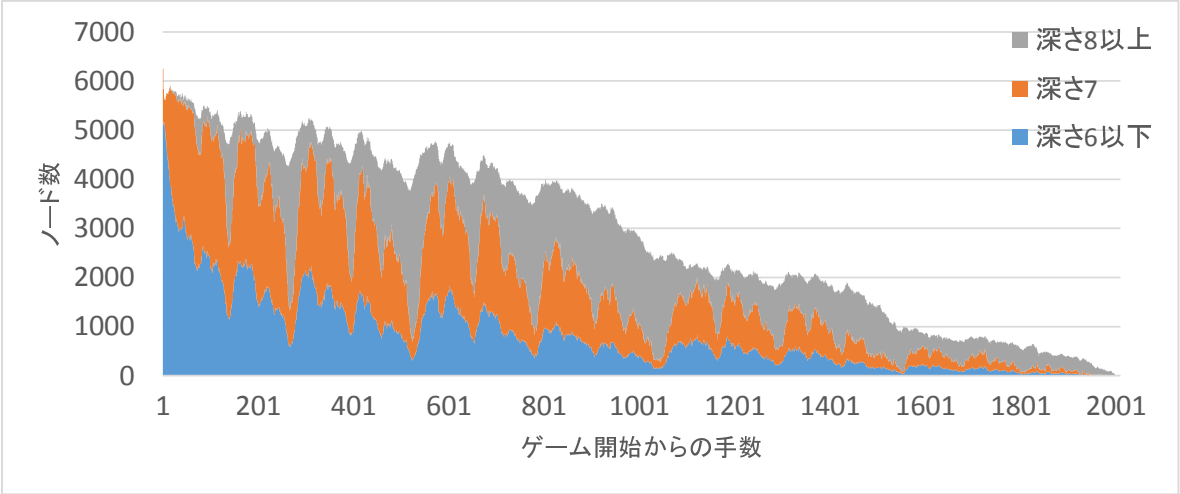

図 6. ノードの深さと幅（MCTS2, 防御側）

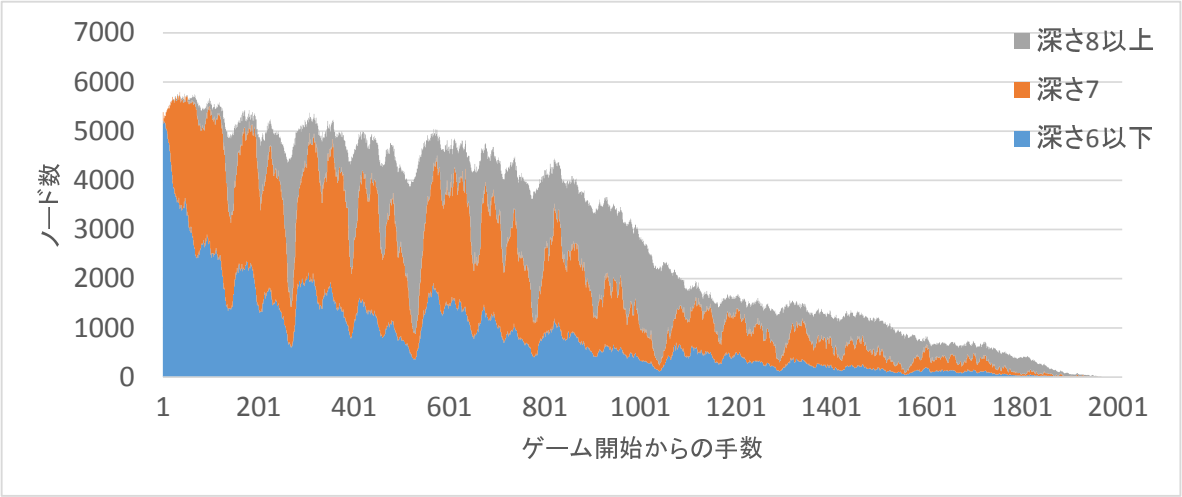

図 7. ノードの深さと幅（MCTS, 防御側）

なっているかを確認するため、MCTS2 と MCTS とを対戦させる。同様に対戦回数は 80 とする。結果を表 1 の下段に示す。この結果、合流を考慮したモンテカルロ木探索プレイヤ MCTS2 の得点が 2036.2 点高い。改良したモンテカルロ木探索プレイヤはより強くなっている。

さらに、MCTS2 と MCTS との対戦の途中で、プレイアウトのために展開したノードの深さと幅を防御側と攻撃側のそれぞれで考察する。そのため、80 回の対戦について、ゲーム開始からの手数と展開されたノード数の平均との関係を調べる。ただし、合法手が 1 つしかない場合にはモンテカルロ木探索を行わないため、母集団から除く。

図 6、7、8、9 は、MCTS2 ／ MCTS および攻撃側／守備側のそれぞれについて、横軸にゲーム開始からの手数、縦軸に展開されたノード数をとり、深さ 6 以下／深さ 7 ／深さ 8 以上に分けて累積度数をグラフにしたものである。深さ 7 のノードに注目すると、MCTS のノード数よりも MCTS2 のノード数の方が少ない。また、深さ 8 以上のノードの数は MCTS2 の方が多い。これが木の合流による効果である。MCTS2 は MCTS に比べ、より手の進んだ局面についてプレイアウトができ、そのため強さが向上していると考えられる。

6. 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。これらは、「対戦型 2048」では守備側プレイヤに相当する。

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 [6]。しかし、MiniMax アルゴリズムでは、2 のタイルが置かれる場所がプ

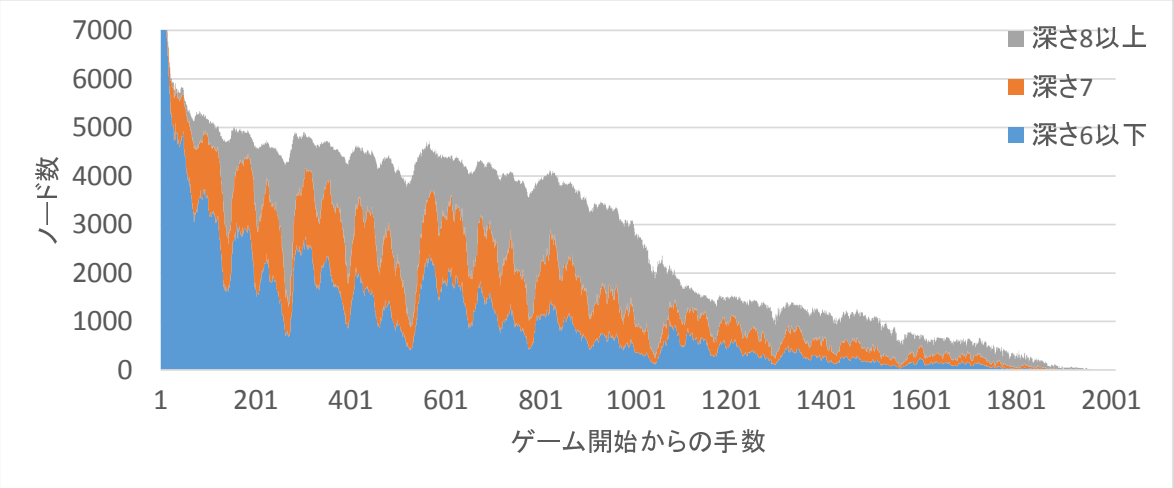

図 8. ノードの深さと幅（MCTS2, 攻撃側）

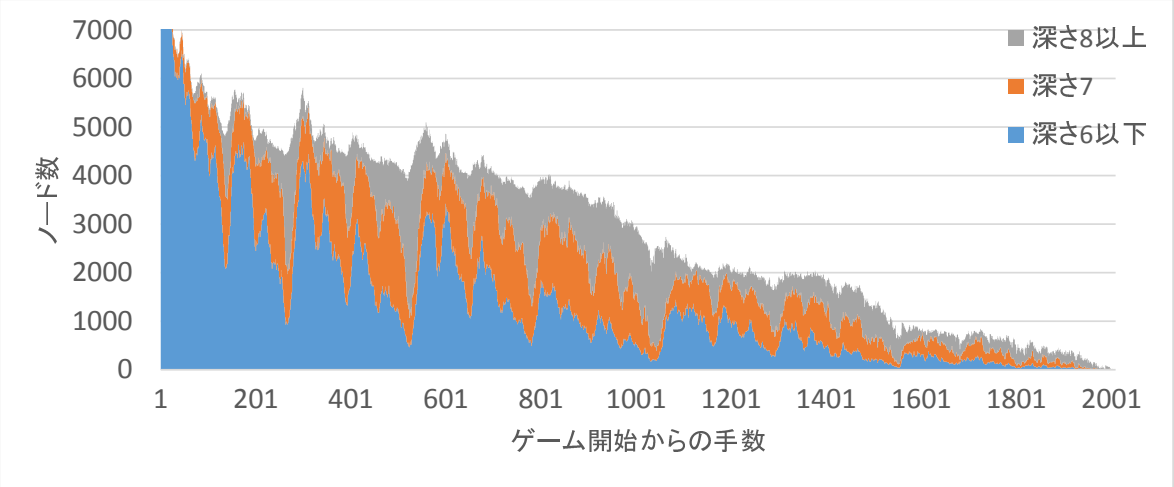

図 9. ノードの深さと幅（MCTS, 攻撃側）

レイヤにとって最も不利な場所であるとして計算を行っている。これを、各ノードの生起確率によってバイアスをかけたのが 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について、2048 のタイルがが出来る可能性は 80 %であると Zaky が報告している [6]。

本稿で扱ったモンテカルロ木探索については、Rodgers らによって「2048」のプレイヤの強さが調査されている。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 [8]。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 40,000 点程度、ADLS プレイヤの平均得点は 77,000 点程度であった。

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ひとつである Temporal Differencd 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 [9]。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 [10]。ステージ化の深さを 5 としたとき、2048 のタイルが出来る確率が 100 %、平均得点が 328,946 点であったと報告している。著者らが知る限り、この Wu らの結果が最も高い平均得点である。

7. まとめ

本稿では、1 人用ゲームである 2048 を 2 人対戦に拡張した「対戦型 2048」に対して、網羅的解析とモンテカルロ木探索プレイヤの作成を行った。網羅的解析によって、「対戦型 2048」では合流が多く手数が増えると全体の局面の増加比が減少することを明らかにし、また、ゲーム終了までの最短手順（51 手）を発見した。また、モンテカルロ木探索のプレイヤを作成し、プレイアウト回数を 10000 回とした場合、深さ 3 まで探索する評価関数 Mini-Max プレイヤと対戦して、平均 27806.1 点を得ることができ、相手の得点を平均 18267.9 点に抑えた。さらに、合流を考慮したモンテカルロ木探索プレイヤを作成し、プレイアウト回数を 10000 回とした場合、深さ 3 まで探索する評価関数 Mini-Max プレイヤと対戦して、平均 26702.6 点を得ることができ、相手の得点を平均 16696.2 点に抑えた。

本研究で作成したモンテカルロ木探索プレイヤには、まだ多くの改善点がある。まず、局面の正規化を実装することで、MCTS2 の局面一致判定の精度を向上し、より深い局面でのプレイアウトを可能にする。また、プレイアウトの回数を増やして実験するため、プレイアウトを実行する部分を並列化する。また、これまでに「2048」に対して最も成果を挙げている機械学習の成果を適用することによって、プレイアウトにおいてある程度選択的にシミュレーションすることも必要であると考える。

文献

1) G. Cirulli, “2048,” URL = http://gabrielecirulli.github.io/2048/, 2015/5/13.
2) 松原 仁 (編), 美添 一樹, 山下 宏 (著). “コンピュータ囲碁—モンテカルロ法の理論と実践,” 共立出版, 2012.
3) J. Schaeffer, E. Al, J. Schaeffer, N. Burch, Y. Bjornsson, A. Kishimoto, M. M. R. Lake, “Checkers is solved,” *Science*, Vol. 317, pp. 1518–1522, 2007.
4) L. Kocsis and C. Szepesvári, “Bandit Based Monte-Carlo Planning,” *17th European Conference on Machine Learning (ECML 2006)*, Lecture Notes in Computer Science 4212, pp. 282–293, 2006.
5) P. Auer, N. Cesa-Bianchi and P. Fischer, “Finite-time Analysis of the Multi-armed Bandit Problem,” *Machine Learning*, Vol. 47, pp. 235–256, 2002.
6) A. Zaky, “Minimax and Expectimax Algorithm to Solve 2048,” URL = http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2013-2014-genap/Makalah2014/MakalahIF2211-2014-037.pdf, 2014.
7) R. Xiao, “nneonneo/2048-ai — GitHub,” URL = https://github.com/nneonneo/2048-ai, 2015/5/13.
8) P. Rodgers and J. Levine, “An Investigation into 2048 AI Strategies,” *2014 IEEE Conference on Computational Intelligence and Games*, pp. 1–2, 2014.
9) 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.
10) 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*, Lecture Notes in Computer Science 8916, pp. 366–378, 2014.

Exhaustive Analysis and Monte-Carlo Tree Search Player for Two-Player 2048

Kazuto Oka[1] **Kiminori Matsuzaki**[2*] **Kazuya Haraguchi**[3]

(Received: May 11th, 2015)

[1] Graduate School of Engineering, Kochi University of Technology
185 Tosayamadacho-Miyanokuchi, Kami, Kochi, 782–8502, JAPAN

[2] School of Information, Kochi University of Technology
185 Tosayamadacho-Miyanokuchi, Kami, Kochi, 782–8502, JAPAN

[3] Department of Information and Management Science, Otaru University of Commerce
3–5–21 Midori, Otaru, Hokkaido, 047–8501, JAPAN

[*] E-mail: matsuzaki.kiminori@kochi-tech.ac.jp

Abstract: "Two-player 2048" is a two-player extension of the one-player game "2048" developed in 2014. In this paper, we apply an exhaustive analysis for "two-player 2048" and develop players based on the monte-carlo tree search (MCTS) algorithm. By the exhaustive analysis, we found that the increasing ratio of the number of boards decreases as games proceed, and a shortest sequence of moves to the game end. We developed two MCTS players: a simple one and an involved one in which we consider the confluence of boards in a tree. We evaluated the strength of MCTS players using an existing MiniMax player with an evaluation function. The score of the involved MCTS player is higher by 2036.2 in average than that of the simple MCTS player. We investigate the reason in the relationship of depth-width of the trees generated by the playouts in MCTS algorithm.