令和 4 年度

修士学位論文

対戦型 2048 における評価関数の強化学習と性能の比較に関する研究

Reinforcement Learning of Evaluation Functions for Two-player 2048 Game and Their Performance Comparison

1255101　　小 田　駿 斗

指導教員　　松 崎　公 紀

2023 年 2 月 28 日

高知工科大学大学院 工学研究科 基盤工学専攻

情報学コース

要 旨

対戦型 2048 における評価関数の強化学習と性能の比較に関する研究

小 田 駿 斗

対戦型 2048 は非対称な 2 人ゲームであり，2 人のプレイヤ（攻撃側と防御側）が選択できる手と目標が全く異なる．対戦型 2048 のプレイヤの作成について，岡と松崎 [2] は N タプルネットワークに基づくプレイヤを提案し，横山と松崎 [4] は，ニューラルネットワークによる評価関数と TD 誤差学習の組み合わせにより対戦型 2048 プレイヤを作成した．しかし，これら二つの研究では異なるルールを採用しており，結果を直接比較できなかった．本研究では，横山と松崎 [4] の研究と同様のニューラルネットワークと，岡と松崎 [2] の研究に基づいて再学習を行った N タプルネットワークを作成し，さらにそれぞれに $\alpha\beta$ 探索を組み合わせたプレイヤを作成した．得られた 2 種類のプレイヤを相互に対戦することによって性能の評価を行った．実験結果として，探索を深くするごとにプレイヤが強くなること，攻撃側プレイヤの探索を深くすると徐々に得点の減少幅が小さくなることが共通していた．一方，同じ探索深さのプレイヤ同士の対戦ではニューラルネットワークプレイヤと N タプルネットワークプレイヤで平均得点が異なっていた．異なるプレイヤ間の対戦では，多くの条件でニューラルネットワークプレイヤの方が優れていた．

キーワード　対戦型 2048　非対称ゲーム　ニューラルネットワーク　N タプルネットワーク

Abstract

Reinforcement Learning of Evaluation Functions for Two-player 2048 Game and Their Performance Comparison

ODA, Hayato

Two-player 2048 is an asymmetric two-player game, in which moves and goals of the two players (attacker and defender) are completely different. Oka and Matsuzaki developed players based on N-tuple networks, and Yokoyama and Matsuzaki developed players by combining neural network evaluation functions and TD learning method. However, these two studies employed different rules, and the results could not be directly compared. In this study, we created neural networks in a similar way to Yokoyama and Matsuzaki and retrained N-tuple networks based on the work of Oka and Matsuzaki, and also created players combining $\alpha\beta$ search. We evaluated the performance of the two types of players by competing against each other. The experimental results were in common that the players became stronger as the search depth increased, and that the score gradually decreased as the attacker's search depth increased. On the other hand, the average scores of neural network players and N-tuple network players were different with the same search depth. In games between different type of players, neural network players were superior in many conditions.

key words two-player 2048 asymmetric game neural network N-tuple network

目次

目次

図目次

表目次

第 1 章

はじめに

「対戦型 2048」は，寺田 [1] によって提案された，確率的 1 人ゲーム 2048 の 2 人プレイゲームへの拡張である．対戦型 2048 は非対称な 2 人ゲームであり，2 人のプレイヤ（攻撃側と防御側）が選択できる手と目標が全く異なる．防御側プレイヤは，通常の 2048 と同様に，盤面上のタイルを動かす方向を選択し，できるだけ得点が大きくなることを目標とする．攻撃側プレイヤは，空マスの中からタイルを置くマスを選び，できるだけ得点が小さくなることを目標とする．このほうな非対称なゲームは，対称なゲームに比べると研究が少ない．

対戦型 2048 のプレイヤの作成に関していくつかの取り組みがある．岡と松崎 [2] は 2048 において有効性が示されている N タプルネットワークに基づくプレイヤを作成した．一方，2048 において，近年ニューラルネットワークを用いたプレイヤが一定の成果を上げている [3]．これを受け，横山と松崎 [4] は，ニューラルネットワークによる評価関数と TD 誤差学習の組合せにより対戦型 2048 プレイヤを作成した．しかし，これら二つの研究では，攻撃側が置くタイルの数について異なるルールを採用しており，結果を直接比較することができなかった．

そこで，著者らは岡と松崎 [2] が用いたものと同じルールのもとで横山と松崎 [4] の研究と同様のニューラルネットワークを作成し，岡と松崎 [2] の研究に基づいて再学習を行った N タプルネットワークを作成した．さらに，それぞれに $\alpha\beta$ 探索を組み合わせたプレイヤを作成し，得られたニューラルネットワークプレイヤと N タプルネットワークプレイヤを相互に対戦することにより性能の評価を行った．

本研究で得られた知見を以下に示す．

- ニューラルネットワークプレイヤと N タプルネットワークプレイヤはどちらも探索を深くすることでプレイヤが強くなる．
- ニューラルネットワークプレイヤと N タプルネットワークプレイヤの両方で，攻撃側プレイヤの探索を深くすると，平均得点の減少幅が小さくなる．
- 同じ深さの探索を行うプレイヤ同士の対戦では，ニューラルネットワークプレイヤと N タプルネットワークプレイヤで平均得点が異なっていた．
- ニューラルネットワークプレイヤと N タプルネットワークプレイヤ間の対戦では，多くの条件でニューラルネットワークプレイヤの方が優れていた．

第 2 章

対戦型 2048 のルール

「対戦型 2048」は確率的 1 人ゲーム「2048」を 2 人ゲームに拡張したものである．対戦型 2048 は完全情報非対称 2 人ゲームである．ある 1 ゲームにおいて，プレイヤは攻撃側と防御側のいずれかをプレイする．（寺田 [1] の提案では，攻撃側と防御側を入れ替えて 2 度対戦して勝敗を決める形を取っている．）対戦型 2048 には，攻撃側が置くタイルの数についていくつかのルールが存在する．

- タイルの値は 2 で固定（寺田 [1] で提案されたルール）
- タイルの値はタイルを置く位置を決定した後に，2 と 4 でランダムに選択
- タイルの値はタイルを置く位置を決定する前に，2 と 4 でランダムに選択（横山と松崎 [4] で採用したルール）
- タイルの値は 2 と 4 で任意に選択（岡と松崎 [2] で採用したルール）

本研究では，岡と松崎 [2] で採用された，タイルの値は 2 と 4 で任意に選択できるルールを採用する．このルールは，攻撃側プレイヤにとって最も有利なものである．以下に本研究で用いたルールを示す．

ゲームの各局面は，4×4 の大きさの盤面に，2 のべき乗の値をもつタイルが置かれたものとなっている．初期局面において，攻撃側プレイヤは以下のルールに従って 2 つのタイルを置く．続けて，防御側プレイヤと攻撃側プレイヤが交互に，以下のルールに従って手を選択していく．防御側プレイヤが選択できる手がなくなったときゲームは終了する．攻撃側プレイヤの目標はゲーム終了時の得点を小さくすることであり，防御側プレイヤの目標はゲーム終了時の得点を大きくすることである．

■攻撃側プレイヤ　本研究における攻撃側プレイヤは，盤面上の空きマスを一つ選び，そこに 2 または 4 のタイルを置く．

■防御側プレイヤ　防御側プレイヤは，タイルを動かす方向を上下左右の中から一つ選択する．方向を選択すると，タイルは以下の規則で移動・併合する．いずれのタイルも移動・併合しない方向は選択できない．

- タイルは，選択した方向にできるだけ移動する（「Threes!」のように，1 マスだけ移動するわけではない）．たとえば，右を選択した際に，2,_,4,_ という列は _,_,2,4 へと変化する．
- 選択した方向に，同じ値をもつタイルが連続している場合，併合が起こる．併合により，タイルの値の和をもつ一つのタイルができ，その和が得点に加算される．たとえば，右を選択した際に，2,2,8,8 という列は _,_,4,16 に変化し，$4 + 16 = 20$ 点が得点に加算される．
- 併合によってできたタイルは，その一回の移動の中では再度併合されることはない．たとえば，右を選択した際に，_,4,2,2 という列は _,_,4,4 に変化する．
- 選択した方向に対して同じ値を持ったタイルが連続して 3 枚以上置かれている場合，より移動先に近い 2 枚が併合される．たとえば，右を選択した際に，_,2,2,2 という列は _,_,2,4 に変化する．

第 3 章

関連研究

確率的 1 人ゲーム 2048 のプレイヤにおいて現在最も成功しているアプローチは N タプルネットワークによる評価関数（局所パターンの線形和）と Expectimax 探索を組み合わせる手法である．N タプルネットワークと強化学習を組み合わせる手法は，Szubert と Jaśkowski [9] により最初にゲーム 2048 に対して導入された．その後，Expectimax 探索の導入とマルチステージ化 [10, 11]，時間コヒーレンス学習 [12]，リスタート戦略 [13] などの改良が提案された．Guei らによる最先端のプレイヤ [14] は，楽観的初期化法を用いて平均得点 625 377 を達成している．

一方，ニューラルネットワークによる評価関数を用いたプレイヤの研究も進められており，強化学習により学習したバリューネットワークと Expectimax 探索を組み合わせたもの [3] では，平均得点 406 927 を達成している．

また，対戦型 2048 についても 2048 プレイヤと同様のアプローチを適用したプレイヤが研究されている．

岡と松崎 [2] は，N タプルネットワークによる評価関数と minimax 探索を併用したプレイヤの対戦結果について報告している．横山と松崎 [4] は，主に [5] で用いられたニューラルネットワーク構造を用いて対戦型 2048 プレイヤを作成し，攻撃側と防御側を交互に学習する学習方法が最も性能がよかったことなどを報告している．

第 4 章

ニューラルネットワークプレイヤ

4.1 評価関数の設計

本研究では，ニューラルネットワークを用いて防御側評価関数と攻撃側評価関数を作成した．本研究では，松崎 [3] が用いたバリューネットワーク（盤面を入力にとり，評価値を 1 つ返す関数）を用いた．本研究で用いたニューラルネットワークの構成を図 4.1 に示す．ニューラルネットワークは，畳み込み層 2 層と全結合層 3 層で構成されている．畳み込み層では 2×2 のフィルタを用いている．第 1 層から第 4 層では，活性化関数として ReLU を用い，フィルタ／ニューロン関数を順に 256, 512, 1024, 256, 1 とした．最終層で計算される 1 つの値を盤面の評価値とする．各フィルタの重みの初期値は，$\pm 2\sigma$ の切断正規分布（平均 $\mu = 0$，分散 $\sigma^2 = (0.1)^2$）からランダムに与えた．

損失関数には平均二乗誤差を用いた．最適化アルゴリズムには，Adam を TensorFlow のデフォルトのハイパーパラメータ（$\alpha = 0.001$，$\beta_1 = 0.9$，$\beta_2 = 0.999$，$\varepsilon = 10^{-7}$）のまま使用した．

ニューラルネットワークの入力は，盤面を $4 \times 4 \times 16$ の 3 次元の 2 値配列に変換したものである．図 4.2 に盤面を入力に変換する方法を示す．1 番目の 4×4 は空きマスの位置を値 1 で示し，2 番目は 2 のタイルの位置を値 1 で示し，以下同様にして，16 番目の 4×4 は 32768 のタイルの位置を値 1 で示す．

■攻撃側評価関数　攻撃側評価関数は，与えられた局面から任意の空きマスに 2 または 4 のタイルを 1 つ配置した盤面を入力として（最大 $(16 - 1) \times 2 = 30$ 通り），それらの盤面を

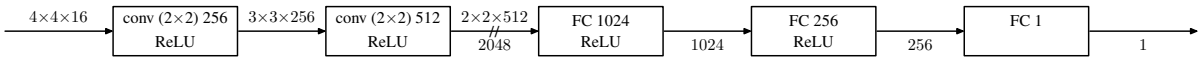


図 4.1　本研究で用いたニューラルネットワークの構成．“conv (2×2) n” は大きさ 2×2 の畳み込みフィルタを n 個もつ畳み込み層を表し，“FC n” はニューロン数 n の全結合層を表す．矢印のラベルは中間データの形を表す．第 2 畳み込み層の後に，3 階テンソルを 1 階テンソルにする平坦化処理が行われている．

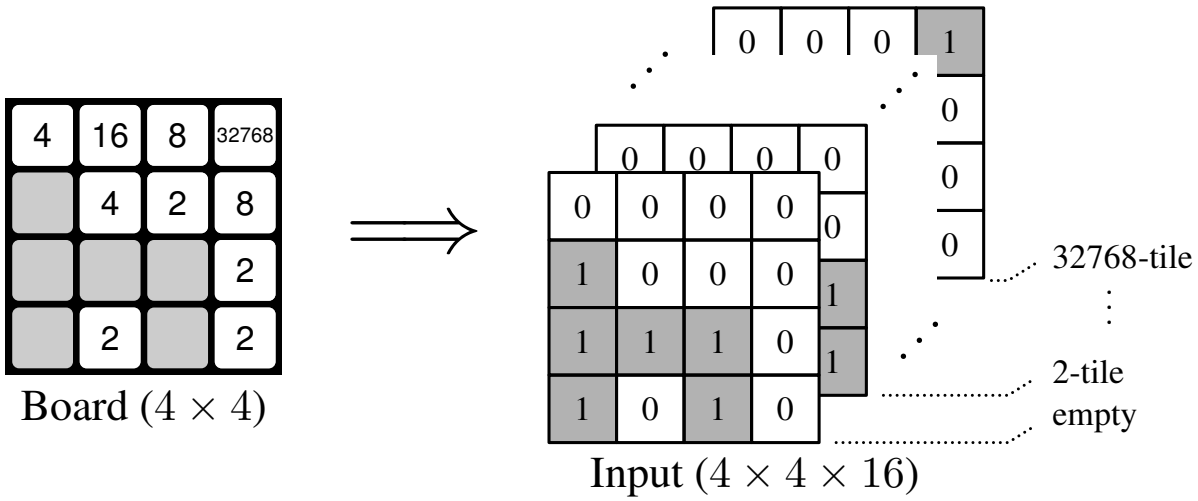


図 4.2　入力データの成形方法 [3]

バリューネットワークにより評価した値を出力する．

■防御側評価関数　防御側評価関数は，与えられた局面に対して，4 方向に移動・併合を行った後の盤面を入力として，バリューネットワークにより評価値を計算する．

4.2　評価関数の学習

本研究では，先行研究 [5, 4] と同様に TD 誤差学習の手法を用いて，ニューラルネットワークの重みを調整した．学習に用いた計算機環境は表 4.1 のとおりである．

本研究では，まず通常の 2048 における評価関数を作り，その後，攻撃側評価関数と防御側評価関数を交互に学習するようにした．先行研究 [4] において，攻撃側の学習がすぐに収束するという結果が得られていたので，本研究では攻撃側の学習を 12 時間/ステップ，防御側の学習を 24 時間/ステップとした．したがって，学習は以下の 7 ステップからなる．

ステップ 0　まず，1 人ゲームのルールのもとで，防御側評価関数を学習する（24 時間）.

ステップ $n = 2k - 1$（$k = 1, 2, 3$）　ステップ $n - 1$ の結果得られた防御側評価関数を固定して，攻撃側評価関数を学習する（12 時間/ステップ）.

ステップ $n = 2k$（$k = 1, 2, 3$）　ステップ $n - 1$ の結果得られた攻撃側評価関数を固定して，防御評価関数を学習する（24 時間/ステップ）.

以降では，ステップ 0 の学習が終わった時点を時刻 0 とし，それ以降のステップ 1〜6 の学習時間を累積して示す．すなわち，12，36，48，72，84，108 がそれぞれステップ 1〜6 の終了時刻である.

学習に用いたプログラムでは，その時点での評価関数を用いて対戦を行うデータ生成スレッド 5 つと，得られたデータをもとに評価関数を更新する学習スレッド 1 つを同時に動かした．学習は，1 ゲーム分のデータが得られるごとに，そのゲームに出現した局面を一度に学習させた.

プレイヤを学習するにあたって，本研究では以下の点を考慮した.

対称性　対称性を考慮する学習（SYM）では，局面ごとにランダムに回転・鏡映した盤面を入力として与えるようにした．対称性を考慮しない学習（NOS）では，ゲームの盤面をそのまま入力として与えた．なお，1 人ゲームに対する先行研究 [5] では，対称性を考慮しない学習のほうが良い結果が得られている.

ランダムな手　本研究で用いたルールでは確率的要素が含まれていない．そのため，強化学習の過程が特定の盤面に偏ってしまう可能性が考えられる．そこで，学習データの生成において，ε 貪欲法により攻撃側プレイヤの手を選択した．具体的には，p を 0，25，50，75 の 4 通りの値として，$p\,\%$ の確率でランダムな手を，$(100 - p)\,\%$ の確率で評価値の最も低い手を選択するようにした．学習にあたっては，ランダムに選んだ手については学習対象としない（したがって，p が大きいほど単位時間に学習できる局面数が減る）．防御側プレイヤがランダムな手を選択するとゲームがすぐに終わってしまう可能性が高いので，防御側は常に評価値の最も高い手を選択する.

表 4.1　学習に用いた計算機環境

CPU	Intel(R) Core(TM) i7-9800X (8 コア, 16 スレッド, 3.80GHz)
メモリ	128 GB
GPU	2 × NVIDIA GeForce RTX 2080Ti (GPU メモリ 11 GB) *
OS	Ubuntu 20.04
Python	3.8.8 (conda 4.10.3)
TensorFlow	tensorflow-gpu 2.4.1 **

* 実験では，1 つの GPU を占有するプロセスを 2 つずつ実行した．

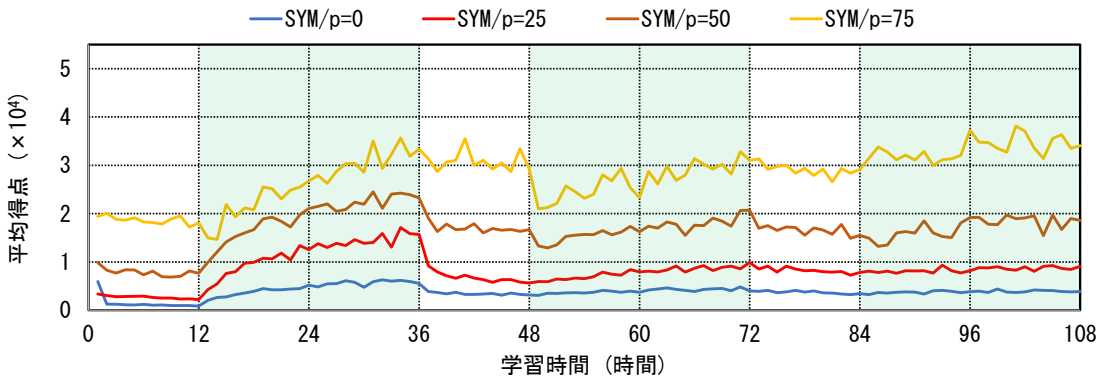
**TensorFlow version 1 の互換 API のみを用いた．

これらの組み合わせ 8 通りのそれぞれについて，108 時間ずつ学習を行った．

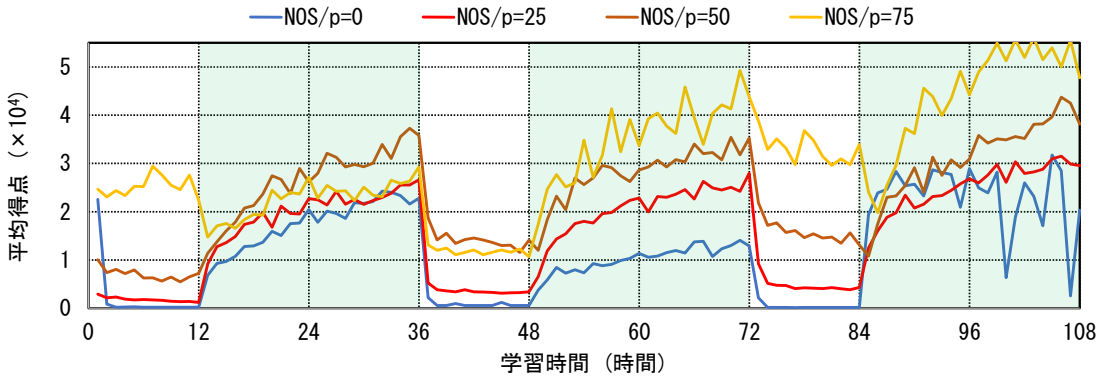
4.3　学習結果

学習の進み方について，ステップ 0 の結果，対称性ありで学習したプレイヤの平均得点は 103,832 点，対称性なしで学習した評価関数の平均得点は 155,336 点であった．続くステップ 1〜6 について，学習の 1 時間ごとに，学習データとして用いた直前 100 ゲーム分の平均得点を求めてプロットした．図 4.3 に，対称性を考慮する学習と対称性を考慮しない学習のそれぞれにおける，学習に用いたデータの平均得点を示す．また，学習データの生成において，一定の確率で攻撃側がランダムに手を選択している．そこで，攻撃側がランダムに手を選択する影響を除いた評価として，1 時間ごとにとった重みのスナップショットを用いて対戦した結果についてもプロットした．（評価実験における対戦では，攻撃側は最初の 20 手に

4.3 学習結果



(a) 対称性を考慮した学習の場合

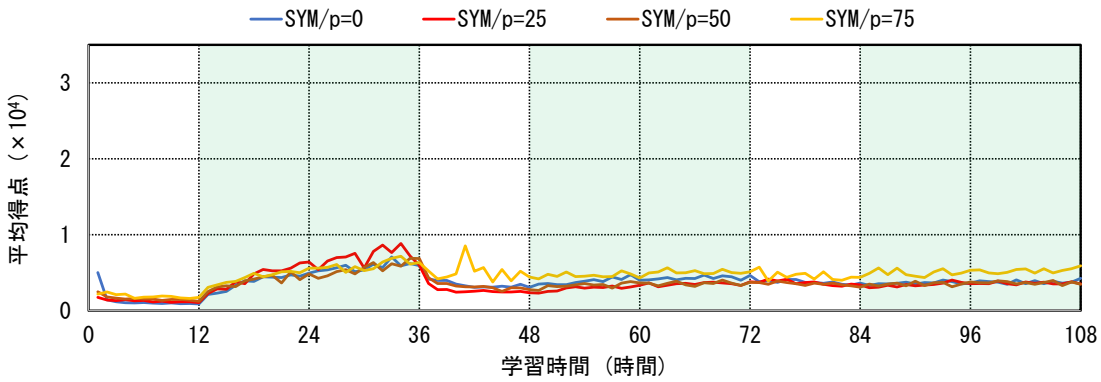


(b) 対称性を考慮しない学習の場合

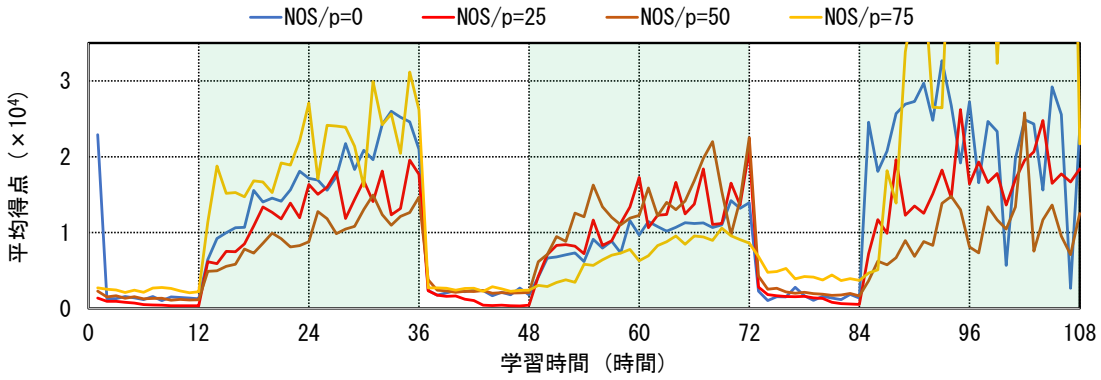
図 4.3 (a) 対称性を考慮する学習と (b) 対称性を考慮しない学習のそれぞれにおける，学習に用いたデータ（直前 100 ゲーム分）の平均得点の推移.

ついてランダムに選択することとした.）図 4.4 に，対称性を考慮する学習と対称性を考慮しない学習のそれぞれにおける，スナップショットを用いた対戦の平均得点を示す.

図 4.3，図 4.4 から，対称性を考慮しない場合に比べて，対称性を考慮する場合は学習がおおよそ収束していることが分かる．また，ランダムに手を選択する確率 p を大きくすると，ゲームがより長く続くようになり，学習データにおける平均得点も大きくなっている．とくに，対称性ありの学習において，この影響がはっきりと現れている．しかし，対戦評価における平均得点のグラフでは，ランダムに手を選択する確率 p による影響はほとんど見られない.



(a) 対称性を考慮した学習の場合



(b) 対称性を考慮しない学習の場合

図 4.4　(a) 対称性を考慮する学習と (b) 対称性を考慮しない学習のそれぞれにおける，スナップショットを用いた対戦の平均得点の推移.

4.4　相互対戦

8 通りの学習方法のそれぞれについて，6 つの学習ステップでの最終的なスナップショット（時刻 12, 36, 48, 72, 84, 108）をとった．それにより，全体として $8 \times 3 = 24$ 通りの攻撃側プレイヤと $8 \times 3 = 24$ 通りの防御側プレイヤを得た．これらの攻撃側プレイヤと防御側プレイヤのそれぞれの対について 100 ゲーム対戦させ，得られたプレイヤを評価した．この対戦評価においても，攻撃側は最初の 20 手についてランダムに選択することとした．

攻撃側プレイヤと防御側プレイヤのそれぞれについて，全対戦相手に対する平均得点，最大得点，および 2048 のタイルの達成率を表 4.2 に示す.

表 4.2　得られたプレイヤ間の相互対戦の結果．全対戦相手に対する平均得点，最大得点，および 2048 のタイルの達成率．

(a) 攻撃側プレイヤの対戦結果

		ステップ1			ステップ3			ステップ5		
		平均	最大	2048	平均	最大	2048	平均	最大	2048
SYM	0	3458	25240	0.3%	2132	11252	0.0%	1896	10780	0.0%
	25	4238	26076	0.6%	2344	13176	0.0%	2219	11144	0.0%
	50	5024	29176	1.3%	3247	13596	0.0%	3033	14712	0.0%
	75	7707	33760	4.6%	5968	25576	1.1%	5533	28428	0.5%
NOS	0	8252	42784	6.2%	6368	30140	4.3%	15783	63456	33.2%
	25	7813	33324	4.4%	6185	34376	3.0%	6842	33452	2.1%
	50	8538	34172	5.8%	9778	48992	12.4%	7491	34296	4.9%
	75	17092	57536	37.5%	9080	51540	9.0%	14204	73524	26.3%

(b) 防御側プレイヤの対戦結果

		ステップ2			ステップ4			ステップ6		
		平均	最大	2048	平均	最大	2048	平均	最大	2048
SYM	0	8097	34172	4.8%	9937	35092	6.0%	9083	34188	1.8%
	25	7613	55928	7.4%	9128	34160	8.7%	10756	53136	13.9%
	50	6388	56740	5.1%	7929	53560	7.5%	9085	56368	11.2%
	75	5330	34736	4.2%	5776	50996	4.4%	6724	57976	5.9%
NOS	0	5831	42784	7.5%	6112	31044	4.0%	5098	33852	5.3%
	25	5737	36096	5.8%	8791	73524	15.0%	7280	63456	9.5%
	50	5599	52808	5.5%	5892	51520	7.7%	5472	56084	7.8%
	75	3558	34928	3.0%	5104	34068	3.2%	3904	67452	2.7%

4.5　対戦結果の考察

対称性ありの学習により得られた攻撃側評価関数では，ステップを進むごとに平均得点が少しずつ下がっていっている．学習時にランダムに手を選択する確率 p について見ると，より小さい確率 p で学習した場合に平均得点がより小さい（より強い）．したがって，対称性ありで学習した場合については，ランダムに手を選択することによる局面の多様性の効果が小さいことが分かる．

対称性ありの学習により得られた防御側評価関数では，全体的にはステップを進むごとに平均得点が上がっていっている．また，学習時にランダムに手を選択する確率 p が小さいほうが，平均得点がより大きい（より強い）．よって，防御側の学習においても，ランダムに手を選択することによる局面の多様性の効果は小さい．

対称性ありの学習によって得られた評価関数は，学習対象の評価関数を使用したプレイヤのみでなく，他の評価関数を使用したプレイヤに対しても適切にプレイできていると言える．

一方で，対称性なしの学習により得られた攻撃側評価関数では，ステップ 3 で平均得点が下がったものの，ステップ 5 では平均得点が上がっている．また，防御側評価関数では，ステップ 4 で上がった平均得点が，ステップ 6 では下がっている．このことから，対称性なしの学習では，学習対象に合わせて学習を進めた結果，他の評価関数を使用したプレイヤに対して適切でないプレイヤとなってしまっている．

学習時にランダムに手を選択する確率 p に関しては，一部例外があるものの，全体として確率 p が小さい場合により強いプレイヤが得られており，この点については対称性ありの学習の結果と同じ傾向にあった．

以上のことから攻撃側，防御側の両方で，対称性を考慮して，ランダムな手を入れないという条件で学習した評価関数が最も強いことが分かった [6]．

4.6　$\alpha\beta$ 探索の追加

1 人用の 2048 プレイヤにおいて，既存研究 [3] ではニューラルネットワークを用いた強化学習に，Expectimax 探索を組み合わせることで性能が向上するという結果が得られている．対戦型 2048 についても，岡と松崎 [2] により作成された N タプルネットワークを用いるプレイヤでは，強化学習と組み合わせる minimax 探索の深さが深いほど性能が向上している．

これらのことから，対戦型 2048 におけるニューラルネットワークプレイヤについても，強化学習に探索を組み合わせることで性能の向上が期待できる．そこで，前節で得られた最も強い評価関数を使用するプレイヤに $\alpha\beta$ 探索を組み合わせ，性能の向上が見られるかを確認する．

本研究における $\alpha\beta$ 探索プレイヤは，前節で作成した攻撃側評価関数または防御側評価関

数を用いて $\alpha\beta$ 枝刈りありの Minimax 探索を行う．計算時間の制約から，探索の深さ d は最大 $d = 7$ とした．具体的には，攻撃側プレイヤでは，深さ d が奇数のときには攻撃側評価関数を用い，d が偶数のときには防御側評価関数を用いる．一方，防御側プレイヤでは，深さ d が奇数のときには防御側評価関数を用い，d が偶数のときには攻撃側評価関数を用いる．以降では，探索深さ d の攻撃側プレイヤを atk_d と表記し，探索深さ d の防御側プレイヤを def_d と表記する．

なお，$\alpha\beta$ 探索の実装にあたっては，探索速度の向上のため，同一親ノードの子ノードを評価値でソートするムーブオーダリングを実装した．最も時間のかかった $d = 7$ の場合，atk_7 で平均約 3.2 秒，def_7 で平均約 6.9 秒であった．

4.7　対戦実験

本研究では，以下に示す 2 つの実験を行った．実験に用いた環境は，実験 1，実験 2 のいずれも表 4.4 に示すとおりである．

4.7.1　実験 1：攻撃側・防御側それぞれの評価関数を用いた探索の性能評価

本研究では，フェーズごとに攻撃側と防御側を学習している．そこで評価関数の一貫性を確保するため，攻撃側プレイヤと防御側プレイヤの両者について，探索深さを $d = 1, 3, 5, 7$ と奇数に限定した場合（すなわち，攻撃側プレイヤは攻撃側評価関数のみを用い，防御側プレイヤは防御側評価関数のみを用いる）の総当たりの実験を行う．各攻撃側プレイヤと防御側プレイヤの組合せについて，攻撃側が最初の 20 手をランダムにプレイした局面からスタートして 100 回ずつ対戦を行った（合計 1600 試合）．対戦実験の全体は 2 週間で終了した．

実験 1 の結果として，それぞれの組み合わせについて平均得点，最大得点，最大タイルの値を表 4.3 に示す．平均得点について，防御側プレイヤを横軸にプロットしたグラフを図 4.5 に，攻撃側プレイヤを横軸にプロットしたグラフを図 4.6 に示す．図 4.5 には追加で黒

表 4.3　各対戦の対戦結果（平均得点，最大得点，達成した最大タイル）

	def_1			def_3			def_5			def_7		
	平均	最大	タイル	平均	最大	タイル	平均	最大	タイル	平均	最大	タイル
atk_1	4051	11328	1024	7031	14388	1024	10588	25324	2048	11244	25404	2048
atk_3	1611	3732	256	3367	9740	1024	5200	13352	1024	7381	13772	1024
atk_5	1270	2868	256	1767	3720	256	3167	6660	512	4673	11236	1024
atk_7	1157	2380	256	1471	2796	256	2005	4896	512	3020	6008	512

表 4.4　実験に用いた計算機環境

CPU	Intel Core i3-8100 BOX
	(4 コア, 4 スレッド, 3.60GHz)
メモリ	16 GB
GPU	ZOTAC GeForce GTX 1080 Ti Mini
	ZT-P10810G-10P (GPU メモリ 11 GB)
OS	Linux version 4.15.0-139-generic
Python	3.6.9
TensorFlow	1.14.0

色の線が引かれている．これは，攻撃側プレイヤと防御側プレイヤで同じ探索深さの場合の結果を線で結んだものである．

4.7.2　実験 2：異なる評価関数により探索を行った場合の性能評価

次に，攻撃側プレイヤと防御側プレイヤの両者について，探索深さに $d = 2, 4, 6$ を追加して総当たりの対戦を行う．探索深さが偶数のとき，攻撃側プレイヤは防御側の評価関数を用いて計算を行い，防御側プレイヤは攻撃側の評価関数を用いる．各攻撃側プレイヤと防御側プレイヤの組合せについて，攻撃側が最初の 20 手をランダムにプレイした局面からスタートして 100 回ずつ対戦を行った（合計 3300 試合）．対戦実験の全体は 2 週間で終了した．

対戦実験 2 の結果として，図 4.5，図 4.6 に深さ $d = 2, 4, 6$ の $\alpha\beta$ 探索を組み合わせたプ

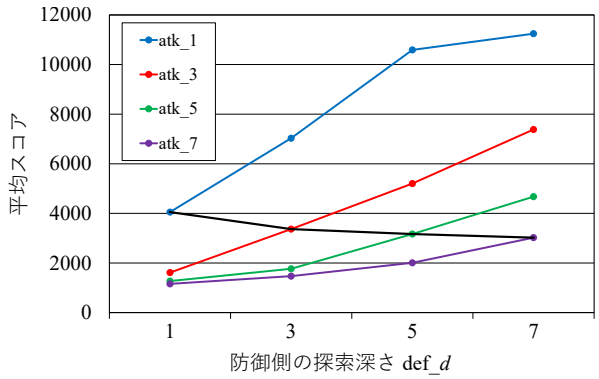


図 4.5　各攻撃側プレイヤの平均得点

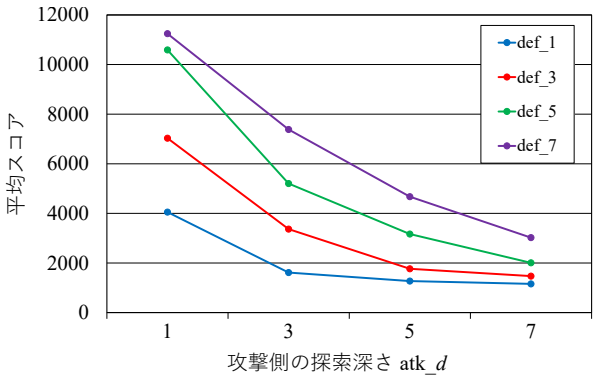


図 4.6　各防御側プレイヤの平均得点

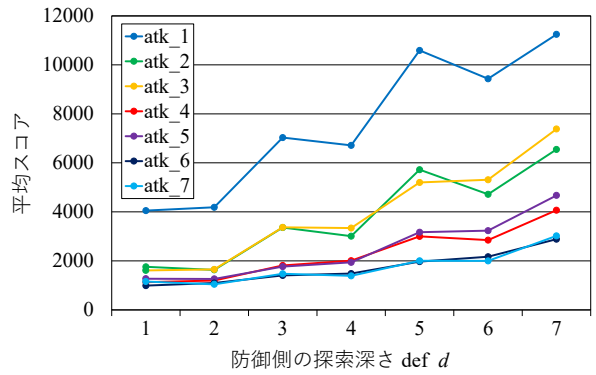


図 4.7　各攻撃側プレイヤの平均得点

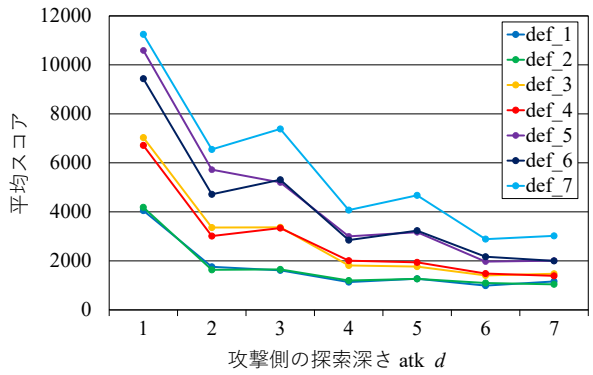


図 4.8　各防御側プレイヤの平均得点

レイヤを追加したものを図 4.7，図 4.8 に示す．

4.8　対戦結果の考察

図 4.5，図 4.6 より，攻撃側プレイヤ，防御側プレイヤの両方で探索を深くすると強くなるという結果が得られた．攻撃側プレイヤでは，探索を深くしていくと徐々に得点の減少幅が小さくなっているが，これはゲームの特性上，非常に小さな得点で終えさせることがより困難であるためだと考えられる．

また，図 4.5 に示した同じ深さの探索を行うプレイヤ同士の対戦において，探索を深くしていくと徐々に得点が下がるという結果が得られている．探索深さ $d = 1$ から $d = 7$ の結果からの推測では，双方が探索深さを増やしていった際に，平均得点 3000 点に近いところに収束するのではないかと考える．

さらに，図 4.7 より，防御側プレイヤの探索深さが奇数のときと比べて偶数のときに平均得点が下がっていることが分かる．防御側プレイヤは平均得点が高いほど性能が良いので，防御側の評価関数を用いた方が性能が高かったと言える．また，図 4.8 より，攻撃側プレイヤの探索深さが奇数のときと比べて偶数のときに平均得点が下がっていることが分かる．攻撃側プレイヤは平均得点が低いほど性能が良いので，防御側の評価関数を用いた方が性能が高かったと言える．これらの結果から，攻撃側の評価関数と防御側の評価関数を比較すると，防御側の評価関数の方が優れていることが分かる．つまり，タイルを動かした後の盤面を評価関数に入力した方が性能が良いということが示唆される．これは，Szubert と Jaśkowski による 2048 に対する考察 [8] と一致している．

以上のことから，探索を深くするごとにプレイヤが強くなること，同じ深さの探索を行うプレイヤ同士の対戦は探索を深くしていくと徐々に得点が下がること，タイルを動かした後の盤面を評価関数入力した方が性能が良いことなどが結果として得られた [7].

第 5 章

N タプルネットワークプレイヤ

5.1 評価関数の設計

本研究では，岡と松崎の先行研究 [2] で用いられていたものと同じ N タプルネットワークを使用する．これは Matsuzaki [15] が提案した 8 つの 6 タプルで構成される（図 **??**）．N タプルネットワークは図 **??**に示す 6 タプルの各位置のタイルの値をサンプリングし，次節で説明する強化学習により得られたルックアップテーブルから部分評価値を得る．各 6 タプルについて，盤面の対称性から 8 通りのサンプリングを行い，これによって得られた $8 \times 8 = 64$ 個の部分評価値を和を全体の評価値とする．この全体の評価値が与えられて局面からゲーム終了までに加算される平均得点となるよう，強化学習によりルックアップテーブルの要素が更新される．

5.2 評価関数の学習

本研究では，先行研究 [2] に基づき，TD 誤差学習の手法を用いて評価関数を学習を行う．学習の条件は，先行研究 [2] で最も性能の良かった，8 つの 6 タプルを使用し，ランダムな手を入れる割合を 50 %とした．また，学習時間はニューラルネットワークの学習と統一するため，以下の 7 ステップで学習を行った．学習環境には表 5.1 を用いた．

ステップ 0　まず，1 人ゲームのルールのもとで，防御側評価関数を学習する（24 時間）．

ステップ $n = 2k - 1$（$k = 1, 2, 3$）　ステップ $n - 1$ の結果得られた防御側評価関数を固定して，攻撃側評価関数を学習する（12 時間/ステップ）．

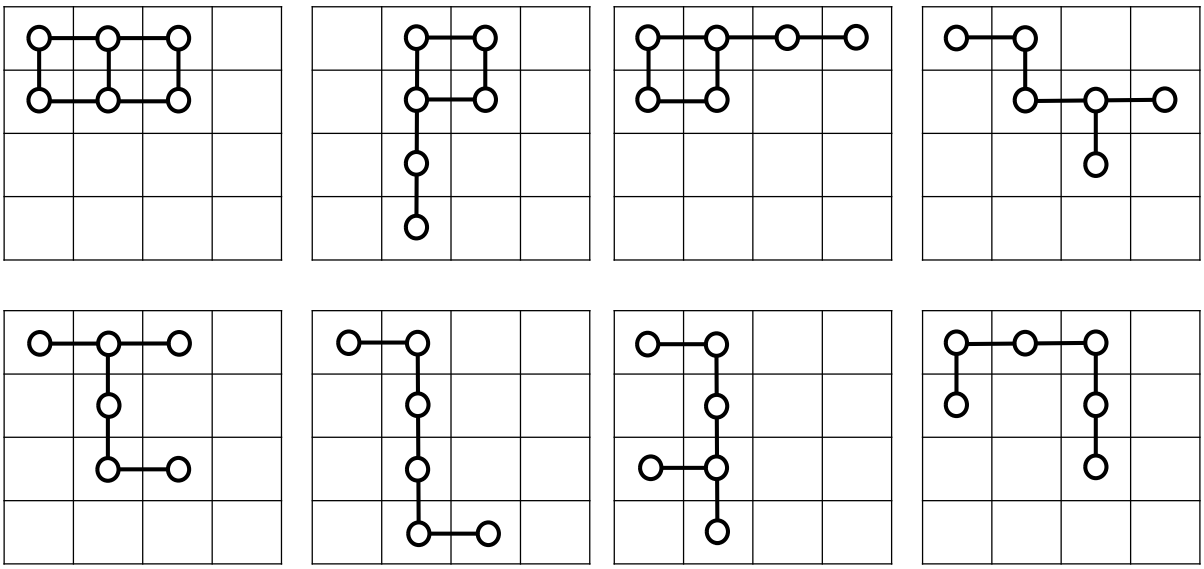


図 5.1　本研究の N タプルネットワークに用いた 8 つの 6 タプル

表 5.1　学習に用いた計算機環境

CPU	Intel Core i3-8100 BOX
	(4 コア, 4 スレッド, 3.60GHz)
メモリ	16 GB
GPU	ZOTAC GeForce GTX 1080 Ti Mini
	ZT-P10810G-10P (GPU メモリ 11 GB)
OS	Linux version 4.15.0-139-generic
g++	7.5.0

ステップ $n = 2k$（$k = 1, 2, 3$）　ステップ $n - 1$ の結果得られた攻撃側評価関数を固定して，防御評価関数を学習する（24 時間/ステップ）.

5.3　学習結果

ステップ 0 の結果，平均得点 235,924 点の評価関数となった．続くステップ 1〜6 について，学習 1 時間ごとに，学習データとして用いた直前 10,000 ゲーム分の平均得点を求めて

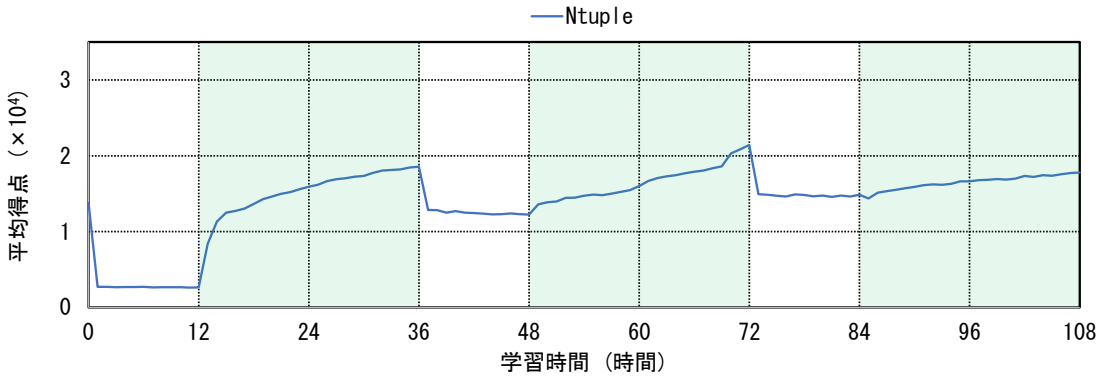


図 5.2 N タプルネットワークを用いた学習について，学習に用いたデータ（直前 10,000 ゲーム分）の平均得点の推移．

プロットしたものを図 5.2 に示す．

図 5.2 から，ステップ 6 終了時点である程度得点が収束していることが分かる．

5.4 $\alpha\beta$ 探索の追加

N タプルネットワークプレイヤについても，ニューラルネットワークプレイヤと同様に，$\alpha\beta$ 探索を組み合わせ，性能の向上が見られるかを確認する．

本研究における $\alpha\beta$ 探索プレイヤは，前節で作成した攻撃側評価関数または防御側評価関数を用いて $\alpha\beta$ 枝刈りありの Minimax 探索を行う．計算時間の制約から，探索の深さ d は最大 $d = 11$ とした．ただし，時間の都合で d は奇数のみとした．具体的には，攻撃側プレイヤでは攻撃側の評価関数を用い，防御側プレイヤでは防御側の評価関数を用いる．以降では，探索深さ d の攻撃側プレイヤを atk_d と表記し，探索深さ d の防御側プレイヤを def_d と表記する．1 手あたりにかかる時間については，最も時間のかかった $d = 11$ の場合，atk_11 で平均約 11.2 秒，def_11 で平均約 39.1 秒だった．

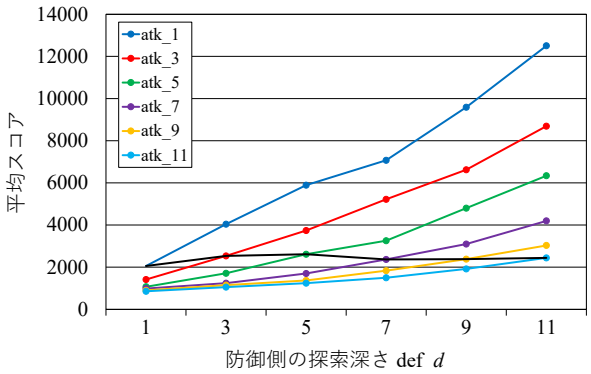


図 5.3　各攻撃側プレイヤの平均得点

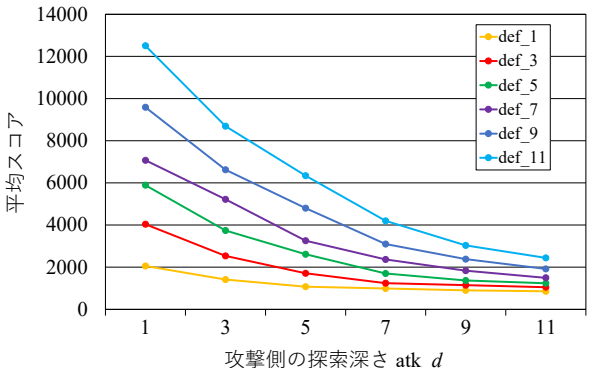


図 5.4　各防御側プレイヤの平均得点

5.5　対戦実験

攻撃側プレイヤと防御側プレイヤの両者について，探索深さ $d = 1, 3, 5, 7, 9, 11$ の総当たりの実験を行う．各攻撃側プレイヤと防御側プレイヤの組合せについて，攻撃側が最初の 50 手をランダムにプレイした局面からスタートして 100 回ずつ対戦を行った（合計 3600 試合）．対戦実験の全体は 2 週間で終了した．

対戦の結果，平均得点について，防御側プレイヤを横軸にプロットしたグラフを図 5.3 に，攻撃側プレイヤを横軸にプロットしたグラフを図 5.4 に示す．図 5.3 には追加で黒色の線が引かれている．これは，攻撃側プレイヤと防御側プレイヤで同じ探索深さの場合の結果を線で結んだものである．

5.5.1　対戦結果の考察

図 5.3，図 5.4 より，攻撃側プレイヤ，防御側プレイヤの両方で探索を深くすると強くなるという結果が得られた．また，攻撃側プレイヤでは，探索を深くしていくと徐々に得点の減少幅が小さくなっていることが分かる．これらの結果から，ニューラルネットワークプレイヤと同様の傾向にあることが分かった．

また，図 5.3 に示した同じ深さの探索を行うプレイヤ同士の対戦において，ニューラルネットワークプレイヤとは異なり，平均得点 2500 点程度になっていることが分かる．

第 6 章

異なるプレイヤ同士の対戦実験

第 4 章，第 5 章で得られたニューラルネットワークプレイヤと N タプルネットワークプレイヤについて，直接対戦することで性能を評価する．各プレイヤの探索深さ d は，着手にかかる時間が近くなるように，以下の 3 つの条件を考えた．短い探索時間と中程度の探索時間は 100 回ずつ，長い探索時間は 50 回ずつ対戦を行った．学習環境には表 6.1 を用いた．

それぞれの対戦実験の結果を表 6.2，6.3，6.4 に示す．また，これらの結果を棒グラフで表したものを図 6.1 に示す．

- 短い探索時間：ニューラルネットワークプレイヤの探索深さ $d = 3$，N タプルネットワークプレイヤの探索深さ $d = 7$．この設定では，1 手あたりの計算時間はおよそ 50 ms である．
- 中程度の探索時間：ニューラルネットワークプレイヤの探索深さ $d = 5$，N タプルネットワークプレイヤの探索深さ $d = 9$．この設定では，1 手あたりの計算時間は 500 ms から 1000 ms である．
- 長い探索時間：ニューラルネットワークプレイヤの探索深さ $d = 7$，N タプルネットワークプレイヤの探索深さ $d = 11$．この設定では，1 手あたりの計算時間は完全にはバランスしておらず，N タプルネットワークプレイヤはニューラルネットワークプレイヤより 4, 5 倍程度の時間を使う．

表 6.1　実験に用いた計算機環境

CPU	Intel Core i3-8100 BOX
	(4 コア, 4 スレッド, 3.60GHz)
メモリ	16 GB
GPU	ZOTAC GeForce GTX 1080 Ti Mini
	ZT-P10810G-10P (GPU メモリ 11 GB)
OS	Linux version 4.15.0-139-generic
Python	3.8.8 (conda 4.10.3)
TensorFlow	tensorflow-gpu 2.4.1 **
g++	7.5.0

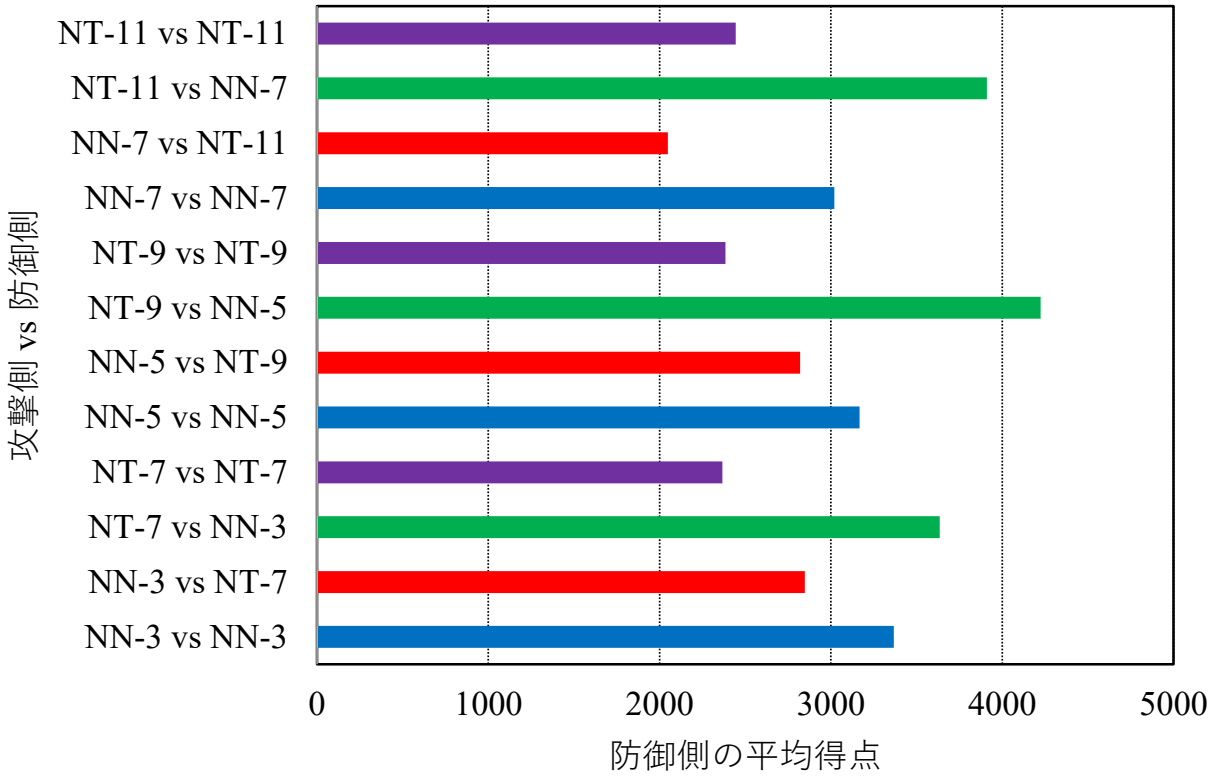


図 6.1　異なるプレイヤ間の対戦の結果

表 6.2　異なるプレイヤ間の対戦結果：短い探索時間

		防御側	
		NN-3	NT-7
攻撃側	NN-3	3367	2848
	NT-7	3635	2366

NN-3: ニューラルネットワーク 深さ $d = 3$

NT-7: N タプルネットワーク 深さ $d = 7$

表 6.3　異なるプレイヤ間の対戦結果：中程度の探索時間

		防御側	
		NN-5	NT-9
攻撃側	NN-5	3167	2820
	NT-9	4224	2384

NN-5: ニューラルネットワーク 深さ $d = 5$

NT-9: N タプルネットワーク 深さ $d = 9$

表 6.4　異なるプレイヤ間の対戦結果：長い探索時間

		防御側	
		NN-7	NT-11
攻撃側	NN-7	3020	2048
	NT-11	3911	2444

NN-7: ニューラルネットワーク 深さ $d = 7$

NT-11: N タプルネットワーク 深さ $d = 11$

第 7 章

考察

ニューラルネットワークプレイヤ同士の対戦結果と N タプルネットワークプレイヤ同士の対戦結果を比較すると，探索を深くすると強くなる点と，攻撃側プレイヤにおいて，探索を深くしていくと徐々に得点の減少幅が小さくなっている点については共通していることが分かる．しかし，同じ深さの探索を行うプレイヤ同士の対戦において，ニューラルネットワークプレイヤでは徐々に平均得点が下がっていたのに対し，N タプルネットワークプレイヤではどの深さでも平均得点 2500 点程度になっている．

次に，異なるプレイヤ間の対戦結果を考察する．表 6.2，表 6.3 から，防御側プレイヤに着目すると，どちらの対戦相手に対してもニューラルネットワークプレイヤの方が平均得点が高い．つまり，防御側プレイヤについては，ニューラルネットワークプレイヤの方が優れていると考えられる．一方，攻撃側プレイヤに着目すると，防御側がニューラルネットワークプレイヤのときは，ニューラルネットワークプレイヤが N タプルネットワークプレイヤよりも平均得点が低く，防御側がが N タプルネットワークプレイヤのときは，N タプルネットワークプレイヤがニューラルネットワークプレイヤよりも平均得点が低い．つまり，攻撃側プレイヤについては，防御側がニューラルネットワークプレイヤのときはニューラルネットワークプレイヤが優れており，防御側がが N タプルネットワークプレイヤのときは N タプルネットワークプレイヤが優れていると考えられる．

一方，表 6.4 では，防御側プレイヤに着目すると，どちらの対戦相手に対してもニューラルネットワークプレイヤの方が平均得点が高い．つまり，防御側プレイヤについては，ニューラルネットワークプレイヤの方が優れていると考えられる．また，攻撃側プレイヤに着目すると，どちらの対戦相手に対してもニューラルネットワークプレイヤの方が平均得点

が低い．つまり，攻撃側プレイヤについても，ニューラルネットワークプレイヤの方が優れていると考えられる．

したがって，一部の条件を除き，ニューラルネットワークプレイヤの方が優れているという結果になった．この結果は 1 人用 2048 とは逆の結果であるため興味深い．これは，本研究で用いた N タプルの組合せが通常の 2048 に特化して提案されたものであり，対戦型 2048 では，本研究で用いた N タプルの組合せよりも優れた N タプルの組合せが存在するのではないかと予想している．

第 8 章

まとめ

本研究では，対戦型 2048 においてニューラルネットワークプレイヤと N タプルネットワークプレイヤの 2 種類のプレイヤを，同じルールの元で学習し，$\alpha\beta$ 探索を組み合わせることで性能を強化した．さらに，作成した 2 つのプレイヤを直接対戦させて性能の評価を行った．その結果，2 つのプレイヤは探索を深くすると強くなる点と，攻撃側プレイヤにおいて，探索を深くしていくと徐々に得点の減少幅が小さくなっている点については共通していることが分かった．一方，同じ深さの探索を行うプレイヤ同士の対戦において，ニューラルネットワークプレイヤと N タプルネットワークプレイヤでは異なる傾向が見られることが分かった．また，異なるプレイヤ間の対戦結果から，実行時間の条件をおおよそ揃えたニューラルネットワークプレイヤと N タプルネットワークプレイヤ間の対戦では，多くの対戦条件において，ニューラルネットワークプレイヤの方が優れていることが分かった．

謝辞

本研究を行うにあたり，指導教員である松崎公紀教授には研究活動や論文執筆などに多くのご指導をいただきました．また，本論文の副査を引き受けていただいた高田喜朗教授，竹内聖悟講師に心からお礼申し上げます.

参考文献

[1] 寺田　実：対戦型 2048，情報処理学会夏のプログラミング・シンポジウム [2015] 報告集，pp. 19–22 (2016).

[2] 岡　和人，松崎公紀：システム的選択による N-tuple networks の “対戦型 2048” への適用，情報処理学会第 58 回プログラミング・シンポジウム，pp. 193–202 (2017).

[3] K. Matsuzaki: Developing Value Networks for Game 2048 with Reinforcement Learning, *Journal of Information Processing*, Vol. 29, pp. 336–346, 2021.

[4] 横山智洋，松崎公紀：ニューラルネットワークと強化学習による対戦型 2048 プレイヤの作成，情報処理学会第 62 回プログラミング・シンポジウム (2021).

[5] Matsuzaki, K.: Developing Value Networks for Game 2048 with Reinforcement Learning, *Journal of Information Processing*, Vol. 29, pp.‘ 336-346 (2021).

[6] 小田駿斗，松崎公紀：攻撃側が置くタイルの数を選択できる対戦型 2048 に対するニューラルネットワークプレイヤの学習，情報処理学会第 63 回プログラミング・シンポジウム (2022).

[7] 小田駿斗，松崎公紀：対戦型 2048 におけるニューラルネットワークプレイヤの α β 探索による強化，第 27 回ゲームプログラミングワークショップ (2022).

[8] 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.

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

[10] Wu, I.-C., Yeh, K.-H., Liang, C.-C., Chang, C.-C. and Chiang, H.: Multi-Stage Temporal Difference Learning for 2048, *Technologies and Applications of Artificial*

Intelligence, Lecture Notes in Computer Science, Vol. 8916, pp. 366–378 (2014).

[11] Yeh, K.-H., Wu, I.-C., Hsueh, C.-H., Chang, C.-C., Liang, C.-C. and Chiang, H.: Multi-stage temporal difference learning for 2048-like games, *IEEE Transactions on Computational Intelligence and AI in Games*, Vol. 9, No. 4, pp. 369–380 (2016).

[12] Jaśkowski, W.: Mastering 2048 with Delayed Temporal Coherence Learning, Multi-Stage Weight Promotion, Redundant Encoding and Carousel Shaping, *IEEE Transactions on Computational Intelligence and AI in Games*, Vol. 10, No. 1, pp. 3–14 (2018).

[13] Matsuzaki, K.: Developing 2048 Player with Backward Temporal Coherence Learning and Restart, *Proceedings of Fifteenth International Conference on Advances in Computer Games (ACG2017)*, pp. 176–187 (2017).

[14] H. Guei et al.: Optimistic Temporal Difference Learning for 2048, *IEEE Transactions on Games*, Vol. 14, No. 3, pp. 478–487 (2021).

[15] Matsuzaki, K.: Systematic selection of N-tuple networks with consideration of inter-influence for game 2048. In: Technologies and Applications of Artificial Intelligence (TAAI 2016) (2016).