E-106

「2048」のアルゴリズム
Algorithm of game 2048

中江 康公†　　米田 貴‡
Nakae Yasuhiro　Yoneda Takashi

1. はじめに

ゲーム「2048」はイタリア人のガブリエレ・チルリによって開発された数字パズルゲームである[1]。2014 年 3 月にリリースされた以来、オンラインモバイルアプリケーションストアで多くダウンロードされ、話題となっているパズルゲームである。遊び方が非常に簡単だが、ゲームのパターンがランダムのため、人間ではこのゲームをマスターするのが極めて困難である。そこで、人間よりはるかに高い処理能力を持つコンピュータなら、どのように動き、どのようにゲームをクリアしていくのかを疑問に思った。なお、この高度情報社会において、コンピュータは様々な領域においてその優れた能力を貢献していることを踏まえ、本研究ではコンピュータが問題を解決するための方法、アルゴリズムを用いたゲーム AI について実験的調査を行いたい。これによって、今後のゲーム領域においての AI の応用に貢献することができると考えられる。また、アルゴリズムは人間が書いたプログラムで動作しているとしても、その動きを予想することは困難である。そのため、アルゴリズムが考える動きを観察すること自体が大変興味深いと考えられる。

2. 先行研究

2.1 「2048」とは

「2048」のゲームボードは図 1 のように、4×4 のマスで成り立っている。それぞれのマスは空（何も入っていない）の状態と数字が書かれているタイルを持つ状態の二つの状態がある。タイルの上に書かれている数字は必ず 2 の累乗数となっている。ゲームが始まるとき、2 つの 2 か 4（2 と 4 が出る割合は 9:1）のタイルがランダムに 16 マス内に置かれる。上下左右にタイルをスライドすると、タイルは指示された方向の壁まで移動する。タイルが複数存在する場合、すべてのタイルがスライドした方向の壁まで移動する。その後、タイルが存在しないマスで新たなタイル（2 か 4）がランダムに一つ置かれる。タイルが移動するとき、同じ数字のタイルがぶつかると 2+2=4、4+4=8 のように数字が足し合わされていき、その代わりに新しい足されたタイルが作られる。最終的に 2048 のタイルができればゲームクリアだが、それ以後も続けることはでき、さらに大きなタイルを作り出すことができる。また、マス内にタイルがすべて埋められ、完全にタイルが動かせない状態になるとゲームオーバーとなる。本研究では、ゲームクリアの条件はタイル 2048 を作ることとする。

このルールから、実際、2048 はシングルプレイヤーゲームではなく、パソコンとプレイヤーの間に行われる二人ゲームであることがわかる。パソコンがランダムにタイルを置き、その後にプレイヤーが一回だけ移動するという流れを永遠に繰り返す仕組みとなっている。これはチェスや将の二人零和有限確定完全情報ゲーム[2]（以後、完全情報ゲームと呼ぶ）によく似ている。そのため、本研究はこのゲームの枠組みに有効と証明されたアルゴリズムを使用する

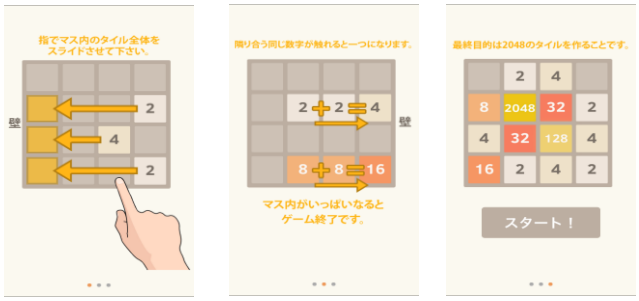


図 1
出典：ゲームのチュートリアル画面[1]

また、ゲーム内では、ゲームの進捗状況を示すためにスコア機能を用いている。スコアは、同じタイルどうしが結合すると、新しい足されたタイルの上に示されている 2 の累乗数だけ元のスコアに足されるように設定されている。タイル 2 と 4 だけがランダムに置かれるため、タイル 2048 を作った時のスコアをほぼ逆算することができる。最低（2 だけが置かれる場合）では 20000 点前後となることがわかる。言い換えれば、スコアが 20000 点前後になればゲームクリアと言える。

2.2 研究対象のアルゴリズムについて

先行研究の一つとして、最初に、インターネットで今まで開発された「2048」のためのアルゴリズムはどのようなものかを調べた。その結果、扱う手法が全く違うアルゴリズムが多数存在していた。「2048」は将棋やチェスと同じジャンルのゲームであるため、それらのゲームに用いられるアルゴリズムを本研究の実験対象として扱いたい。それが Matt Overlan の Minimax with Alpha Beta Pruning[4]である。Minimax は二人零和有限確定完全情報ゲームに効率的なものであるとある程度証明されたている。また、AI

実際、この Overlan のアルゴリズムは二つの手法、Minimax と Alpha Beta Pruning を合成したものである。ほかの完全情報ゲームと同じく、「2048」も現在の局面から出現しうるすべての局面を予測する性質を持つ。例えば、とあるプレイヤーが移動する局面があるとすると、そこから上下左右いずれの手を選んでも、移動後の局面を予測することができる。もちろん、パソコンはタイルをランダムに置くが、4×4 マス上の空マスにしか置かれないため、予測することができる。したがって、これらの局面を樹形図（ゲーム木）で表すことができる。

Minimax は樹形図にある局面を探索しそれぞれの局面を評価関数で評価した上、以下の思考に基づいてプレイヤーがとる最善の手――最終的にゲームクリアできるように現在打つべき手――を導き出す手法だ。Minimax

†神戸大学附属中等教育学校，Kobe University Secondary
‡神戸大学附属中等教育学校，Kobe University Secondary

は常に相手――ここではパソコンを指す――が完璧な手を打つことを仮定している。そのため、相手にとっていい状況から自分にとって一番いい手を選ぶべき。つまり、相手の動きが自分に最大の攻撃をするように評価する。また、Minimax は理論上の最適解を探さない。なぜなら、最適解が適するときは相手が愚かである場合が多いからだ。もし、相手に完璧に手を打てば、損失を最小にすることができる。もし相手がミスを犯しても、一番悲観的な局面よりいい局面に達することができる。ここでは抽象的に述べたが、以下は「2048」の例を用いて説明する[6]。

とあるプレイヤーが移動方向を選択（以後、ターンと呼ぶ）する局面 A（層 0）を根とするゲーム木を想像する。プレイヤーのターンの局面を MAX 節点と呼ぶ。パソコンがランダムにタイルを置く相手のターンの局面を MIN 節点と呼ぶ。また、Minimax の作動がよりわかりやすく説明するために、局面 A から枝分かれていく最初の幾つかの局面を層 1 と呼び、層 1 のそれぞれの局面から枝分かれていく幾つかの局面を層 2、以下同様に定義する。D（D は自然数）を層の変数と定義するとき、局面 A、又は層 0 および D が複数（以後複数層と呼ぶ）のときには MAX 節点だけ並んでいる。D が奇数（以後奇数層と呼ぶ）のときには MIN 節点だけが並んでいる。この樹形図から最適解を求める過程を詳しく見ると、(1)いくつかの評価基準に基づいた評価関数が一番深い層から順番にゲーム木に存在するすべての局面を評価する。(2)もし一番深い層が奇数層なら、その前の偶数その各節点から枝分かれる四つの局面の中の一番評価が低い局面を選ぶ。偶数層なら、一番評価が高い局面を選ぶ。(3)層 D-1 で層 D が奇数層か偶数層かによって上で述べたと同様に各局面を評価する。このように、層 1 まで同じ動作を繰り返す。(4)層 1 の一番評価が低い局面になるように移動する、という四つの段階に分かれる。ここで層 D について注意すべき点がある。ゲーム「2048」において、ゲーム木全体が極めて膨大のため、人間より優れている処理能力を持つコンピュータですら終局まで捜索することは不可能だ。そのため、理論上では、ゲームクリア局面のある層に近づけるほど、または層 D が深いほど、最適解を導き出す可能性が大きくなる。このことは後の実験的調査で証明したい。

Alpha Beta Pruning は Minimax の改良版と補充と考えてもいい。Alpha Beta Pruning を採用することで、最大深度層 D までのすべての節点の構築と捜索をする必要がなくなる。樹形図を構造していく過程において、今見つかった解より良い解が今後の層において見つからないと判断した場合、その枝の探索をやめる。このことを pruning（枝刈り）と呼ぶ。

Alpha Beta と Minimax の大きな違いは、前者は深度を優先に探索（深さ優先探索）して樹形図を構成している。また、Minimax の計算が膨大かつ複雑という問題を解決することができる。Minimax は探索する深度が深くなることにつれ、評価しなければならない局面が指数的に増加する。しかし、これは指数に増加する計算を有限の時間内で行うため、深度が深くなるほど、このアルゴリズムの探索効率が悪くなる。そのため、この 2 つのアルゴリズムを掛け合わせて用いた。このことによって、Minimax の効果を十分に発揮できる上、時間の節約にもつながる。

また、探索する層の深さが直接ゲームの結果に影響を与えるが、Overlan のアルゴリズムでは、AI が探査できる最大深度は制限されていない。その代わりに、思考する時間（以後思考時間と呼ぶ）に制限をかけた。フォーマルでは、探索できる時間は 100 ms と設定されている。設定された時間以内なら、層 1 から何層までも自由に探査できる。

2.3 評価関数

本研究は Minimax について主に議論をするため、アルゴリズムに用いられる評価関数についてはそのヒューリスティックについてだけ紹介する。

Overlan のアルゴリズムの最終的評価関数は四つの異なるヒューリスティック関数（また、部分評価関数）と呼ぶを足し合わせたものとなる。それを構成する異なる評価基準（ヒューリスティック）に基づく四つの評価関数を用いて局面を評価している。なお、評価関数及び四つの部分評価関数のソースコードは Overlan の GitHub[4]に掲載しているため、本論文では掲載しない。

1. Monotonicity(単調性)

これは左から右、上から下まで逓減、又は逓増する具合を指す。一般的に、局面が単調であるほど良い判断することができる。図 2 はこのヒューリスティックにおいての良い局面の例である[2]。具体的に、この関数は上下と左右を分けて以下の手順を踏まえて局面を評価する；(1)上下、なたが左右で隣り合う二つのタイルの差の対数（2 を底とする）をとる。この時、上と下を分けて考えたいため、上から下を引く、下から上を引く、の 2 パターンを用意する。その値は 2 から 2048 までの対数なので、その対数の範囲は 1 から 11 までの自然数である；(2)縦一列又は横にある三つの差をすべてパターン別で足し合わせ、一番数値の大きいものを残す。この時上下左右それぞれ差の最大値が残される；(3)上下の二つの最大値からさらに最大値を選び、左右も同様に最大をとる。この時、上下と左右の二つの最大値が残される；(4)二つの最大値を足し合わせ、この局面の単調さを表す数値となり、一つの評価となる。この関数を単独に用いる場合、大きなタイルをコーナーに置く傾向がみられる[2]。また、小さいタイルが孤立される局面を避ける効果も持つ。

2. Smoothness(平滑性)

単調性ヒューリスティックは、隣り合うタイルの対数（底を 2 とする）の差を 1 にする傾向を持つ。しかし、順調にゲームを進むには、毎回の移動でできるだけ多くの数のタイルを結合しなければならない。それを実現できるものはこのヒューリスティックである。平滑性とは一つのタイルとその隣り合うタイルの差を指す。一般的に、平滑の局面ほど、一回の移動で結合できるタイルの数が多くなり、良い曲と評価できる。図 3 は極めて平滑な局面の例である[2]。また、空のマスが多いほど、局面の評価が高くなるが、分散するほど評価が低くなる。

3. Max-value（最大値）

このヒューリスティックは局面にある最大値の対数（底を 2 とする）をそのまま評価にする。本研究はゲーム終了時にタイル 2048 を一つ作り出すことをゲームクリアだとみなしているため、アルゴリズムはタイル 1024 を１５個作っても本研究において成功とはいえない。このヒューリスティックを使用することによって、2048 を合成する傾向が増え、ゲームクリアへとつながる。

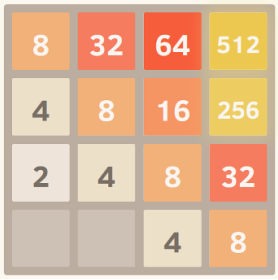


図 2

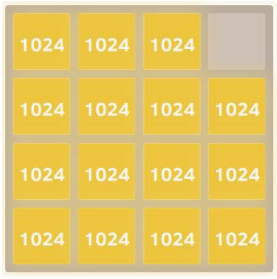


図 3

出典:[2]

4. Empty-cells(空マスの数)

１６マスが全部埋められるとゲームオーバーになるため、このヒューリスティックは空マスの数が少ない局面に低い評価を与える。これは空マスの数を一定以上に保つことができ、タイルの結合に十分の活用できる空間を与えるとともに、終盤にゲームオーバーになることを避ける効果を持つ。

この四つのヒューリスティック関数から出された評価を全部足し合わせた値が、最終的にある特定の局面の評価となり、Minimax で使用される。

実際に、この 4 つのヒューリスティック関数はあくまで本アルゴリズムの作者が一般理論に沿って考えた関数であるため、これらのヒューリスティックよりも優れているヒューリスティックが存在する可能性がある。しかし、本研究は研究の焦点をアルゴリズムとするため、評価関数の有効性についての議論を今後の課題にしたい。

2.4 本研究の目的

Overlan のアルゴリズムの有効性およびその改善点

前節で述べたように、このアルゴリズムを構成する関数はすべて理論上、ゲームクリアに繋がるものとなっている。実際にゲームで実行するときの効果は不明のため、プログラムを重複実行ことによって、その有効性を測る。

2.5 本論文の構成

上記の目的を達成するため、アルゴリズムの性能を設定できる変数が変化するとき、ゲームの結果にどのような影響を与えるのかを検証する必要がある。したがって、異なる思考時間、探索深度や評価関数において、スコアはどう変化するのかを調べる。第 3 章では、アルゴリズムについて実験的調査を行う。第 4 章は総括とする。

3. Matt Overlan のアルゴリズムの分析調査

3.1 目的

とあるアルゴリズムの現実における有効性を調べたいとき、そのアルゴリズムを適用した対象のプログラムを重複実行し、そこから得られたデータを分析するのが一般的である。本研究の場合、AI を搭載したゲームを重複実行し得られたデータからそれぞれの変数の間の関係性がわかる。今回の主な調査対象は思考時間、探索深度、と評価関数の三つの変数だが、このアルゴリズムは、深層 D を制限するではなく、一回の動きを決定するのにかかる時間を制限しているため、理論的に、思考時間が長くなるほど、探索できる深度が深くなり、タイル 2048 に達する回数が多くなる。つまり、思考時間が長くなるほど、スコアも大きくなるはずだ。このことから、以下の仮説を立てたい。

仮説１：思考時間とスコアは正の相関関係にある。
また、探索深度とスコアは正の相関関係にある。

また、予備調査でこのアルゴリズムの動きを何回か観察したところ、毎回全く違う動きが観察されたため、上下左右の回数やばらつきが結果とどのような相関をもつのか疑問に思った。具体例を挙げると、タイル 2048 の作成に成功したゲームの上下左右別の移動回数を見ると、あるゲームでは上に 200 回、下に 200 回、左に 200 回、右に 200 回移動したが、あるゲームでは上に 80 回、下に 330 回、左に 150 回、240 回移動した。前章で述べたように、Overlan は局面の単調性を一つの評価基準として用いている。理論上、ある局面において、大きいなタイルをコーナーに置き動かさないようにすれば単調性を保つことができる。しかし、四方向に均等に動かしてしまうと、人間の感覚的に、大きなタイルをコーナーに保ちにくい。逆に、固定したいタイルの位置に合わせてばらつきのある動きをすれば、そのタイルの位置を「固定」しやすくなると考えられる。これはあくまで理論に基づいた予測で、実際の所実験的調査を欠けている。上下左右の移動回数のばらつきを表す数値として、四方向へのそれぞれの移動回数の標準偏差を使用することができると考えた。標準偏差の大小と移動回数のばらつきは比例するため、以下の仮説を立てることができる。

仮説２：スコアと移動の標準偏差が正の相関にある

3.2 方法

準備物：

1.アルゴリズムのソースコード[4]

２．JavaScript の開発環境が整ったパソコン一台のみ（ソースコードの編成は Visual Studio 2017 を使用する）

パソコンのスペックは、

CPU：intel® Core i7-6700 @ 3.40GHz

メモリ：16.0 GB

３．ブラウザ（本研究では Microsoft Edge を使用する）

調査手順：

１．GitHub からソースコードをダウンロードし、minSearchTime(思考時間)を 10 ms から 10 ms ずつ変えて 150 ms までゲームを重複実行する。ゲームがタイル 2048 の合成に成功する、又はゲームオーバーになるとき、ゲームを止める。

２．毎回ゲームが終了するときの以下の 3 つのデータを記録する

①ゲームが終了したときのスコア(score)。

②上下左右それぞれ移動した回数(move)。

③毎ターン探索した深度(depth)。

3.思考時間ごとの平均スコア、平均移動回数、平均深度を算出する。また、上下左右のそれぞれの回数から標準偏差を算出する。

４．提示した仮説に応じて、適切なフラフを作成する。

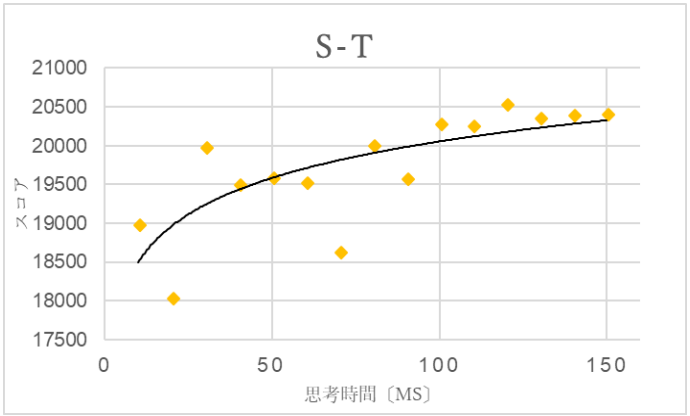
3.3 結果

データをすべてに本論文に載せることは不可能のため、各思考時間のデータを代表するもののみ図 4 に記載する。図 4 からグラフ 1 （スコア―思考時間、以後 S－T と呼ぶ）、グラフ 2 (平均深度―思考時間、以後 D-T と呼ぶ)とグラフ 3（スコア―探索深度、以後 S－D と呼ぶ）を作成した。グラフ 1 は 10 ms から 150 ms の間の平均スコアを対

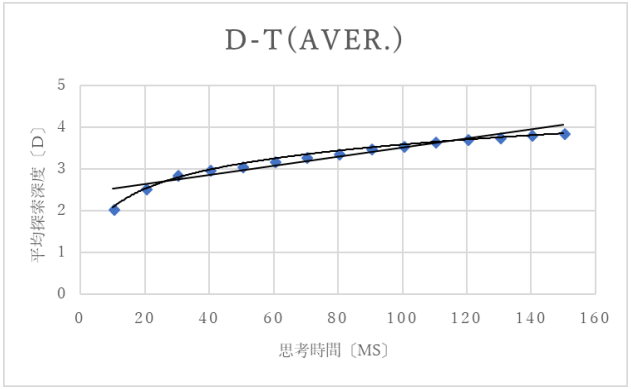
数関数でフィットした図である。グラフ 2 は 10 ms から 150 ms の間の平均深度を一次関数および対数関数でフィットした図である。グラフ 3 は各思考時間の平均スコアと探索深度の関係を一次関数でフィットした図である。グラフ 1 で対数関数を当てはめた理由は、2048 に達するときのスコアが 20000 点前後になるため、いくら思考時間を長く設定しても、この点数に大幅超える得点は得られないと考えた。それに対して、グラフ 2 の最大深度と思考時間に関係は、理論上、思考時間の長さと最大深度は比例の関係にあるが、グラフに分布しているデータを線でつなぐと対数グラフに類似しているからだ。

思考時間〔ms〕	スコアの平均値	平均探索深度〔D〕
150	20422	3.881
140	20410	3.850
130	20370	3.780
120	20549	3.741
110	20275	3.672
100	20304	3.583
90	19594	3.514
80	20026	3.395
70	18649	3.306
60	19541	3.217
50	19602	3.099
40	19512	3.004
30	19994	2.888
20	18058	2.551
10	19003	2.066

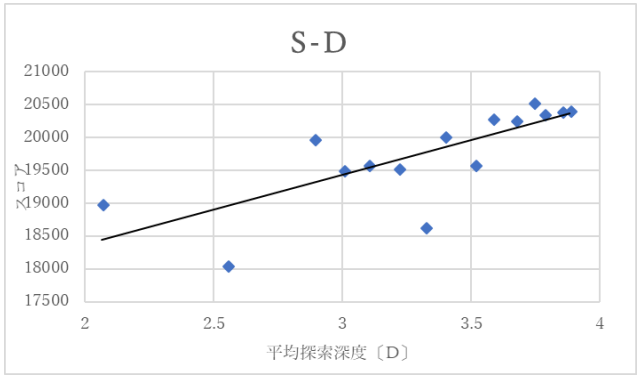
図 4



グラフ 1

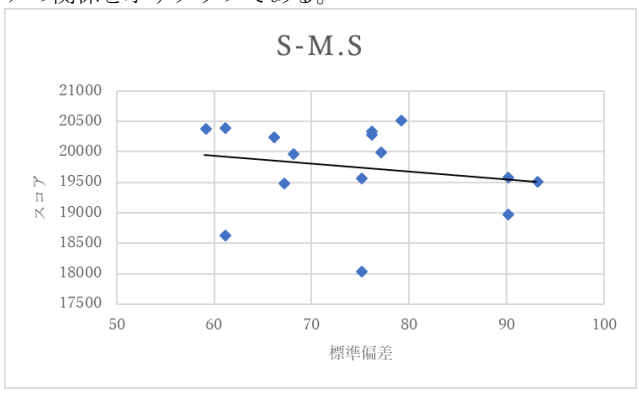


グラフ 2

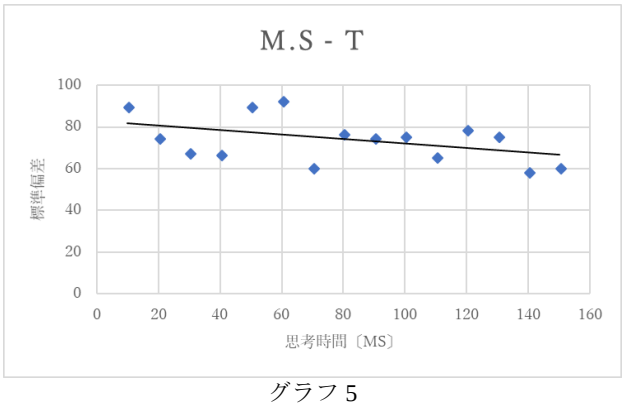


グラフ 3

本研究では掲載していない全体のデータから、グラフ 4（スコア—標準偏差、以後 S-M.S と呼ぶ）とグラフ 5（標準偏差—思考時間、以後 M.S-T と呼ぶ）を作成した。グラフ 5、 6 は移動の標準偏差について、思考時間時間とスコアの関係を示すグラフである。



グラフ 4



グラフ 5

3.4　考察

グラフ 1 と 2 から、スコアと思考時間、平均探索深度と思考時間が指数に比例し、正の相関関係にあることが分かった。グラフ 3 から、探索深度とスコアが比例し、正の相関関係にあることが分かった。したがって、仮説 1 は正しい。このことは、Overlan のアルゴリズムの最大深度 D 又は思考時間のどちらかを変えても又は制限を加えても、その値に応じるスコアをある程度予測することができる。また、思考時間が 150ms の時、成功率が 95 %であった。これは実験を行うパソコンのスペックによって差が生じるが、その差は論理上スペックの差に比例するため、実験結果に普遍性を持つと考えられる。しかし、グラフ 1 において、指数グラフの傾きは 150 ms においてまだ 0 に近づいてい

ない。つまり、今回実験が行われるパソコンでは、150 ms より長い思考時間を設定すれば、成功率がより高くなると予測できる。なお、グラフ２のデータが対数関数にフィットした理由について詳しく考えたい。本実験は、2048 の合成に成功すればゲームクリアという制限をかけているため、毎ターンの探索においてとある一定の層まで探索すれば、タイル 2048 の合成にほぼ成功することが予測される。言い換えれば、とある局面から枝分かれる層において、とある層 D まで探索を進み、minimax 法を用いて決めた移動方向は、層 D よりも深い層まで探索を進んだときに決めた移動方向と同じになる場合がある。このことは本章の実験で検証することができないが、グラフから、本実験に用いるパソコンにおいて、平均層 4 に近づけるほど、対数グラフの傾きが 0 に近づき、成功率が 100%に限りなく近づけると考える。2048 の制限を外した場合の思考時間と平均深度の関係は今後の課題としたい。

グラフ 3 は、思考時間ごとの平均スコアを用いるデータを使用しているため、グラフ 1 と 2 を結合したグラフとしてみなすことができると考えられる。したがって、グラス３はそれ自体意味を持たず、スコアと平均深度の従来の関係を測り場合、より多い数のデータを一つのグラフに示せば妥当な結論に導くことができると考えられる。

グラフ４と５は仮説 2 を検証するためのグラフだが、それを果たすことができないと思われる。確かに、グラフから、移動方向標準偏差はそれぞれ時間とスコアは反比例の関係にあると読み取れるが、その相関関係はわずかである。加えて、前章で述べたように、大きいタイルをコーナーに「固定」するためにばらつきのある移動方向をとるという理論は、少ない移動回数において成り立つは、必ずしもゲームの最初から最後まで成り立つとは限らない。したがって、移動回数の標準偏差をグローバルな変数として扱うのは妥当ではない。ゆえに、標準偏差に基づく議論はこの研究に不適切であると考えられる。上下左右別移動回数とスコアの関係性を調べるには、標準偏差以外の変数が必要だと考えられ、今後の課題としたい。

4. 結論

仮説１は正しいであることは分析結果から証明された。しかし、仮説２はまだ不明である。全体の考察として、思考時間を長くしたり探索深度深くしたりして、ゲームがプレイヤーに有利に進むように設定をすれば、ゲームクリアになりやすくなり、このアルゴリズムの実用性を証明することができた。それに対して、議論を立てる段階において誤りの主張や考えをとったことや、データの使い方に偏りがあったことが大きな反省すべき点である。また、課題として残された無制限時の探索深度とスコアの関係と移動回数のばらつきによる影響を正確に導き出すために、データの再収集を行う必要があると思われる。本研究の問いに対しては、実用性のみを検証することができた。しかし、ここからの延伸として、評価関数の可能性や多様性に関係する。例えば、スコアと４方向への移動回数のばらつきの関係を明確にすることができるなら、それに基づく新たな評価関数を考案することができる。実際に、第 3 章の実験的調査で使用された評価関数はただアルゴリズムの作者が考えた一通りだけである。より高度な実験と議論ができるようになるため、今後は課題解決に向けて努めたい。

参考文献

[1] Gabriele Cirulli ほか「gabrielecirulli/2048 」 https://github.com/gabrielecirulli/2048 (2018.7.27 閲覧)

[2] stackoverflow.com 「What is the optimal algorithm for the game 2048?」 https://stackoverflow.com/questions/22342854/what-is-the-optimal-algorithm-for-the-game-2048 (2018.7.27 閲覧)

[3] 川合慧『情報』第 13 版，p.121. 東京大学出版会 (2006).

[4] Matt Overlan「ovolve/2048-AI 」 https://github.com/ovolve/2048-AI (2018.7.27 閲覧)

[5] Bruce Rosen 「CS 161 Recitation Notes - Minimax with Alpha Beta Pruning」 http://web.cs.ucla.edu/~rosen/161/notes/alphabeta.html (2018.6.15 閲覧)

狩野祐東 『確かな力が身につく JavaScript「超」入門』初版. SB Creative (2015).

紀平拓男 『プログラミングの宝箱 アルゴリズムとデータ構造 第２版』第２版. SB Creative (2011).

w3schools.com 「Javascript/JS Tutorial」 https://www.w3schools.com/js/default.asp (2018.7.27 閲覧)