# 戦略の動的推定による 2人対戦ゲーム接待AIの提案

杉本 直樹[1,a)]　鶴岡 慶雅[2]

**概要**：実用上の人間の対戦相手としてのゲーム AI には、強すぎず弱すぎない人間のプレイヤーと互角の実力が求められる。本研究では、プレイヤーの戦略を動的に推定しそれに応じて対戦 AI を変化させ、2人対戦ゲームにおいてゲーム固有の知識を利用することなく AI プレイヤーの実力を人間のプレイヤーと互角となるよう調整し、なおかつその AI プレイヤーが不自然な振る舞いを見せないようにする事を目的とする。本稿では、研究のため作成したパズルシミュレーションゲームにおいて提案手法による AI がランダム AI よりも長く試合を続けられる事を示した。

# Entertainment AI in Two-Player Games by Dynamic Strategy Estimation

Naoki Sugimoto[1,a)]　Yoshimasa Tsuruoka[2]

***Abstract:*** Game AI for entertainment is required to have equal abilities as a human player. The purpose of this research is to create a game AI which makes a close game against a given human player without using the domain knowledge and let the AI behave naturally by approximating the strategy of player dynamically. In this paper, we show that the AI adjusted by our method plays games longer than random AI in the simulation puzzle game we designed.

## 1. はじめに

プレイヤー同士が対戦するゲームにおいて、人間の代わりに、AI が相手を務めるという場面は多くある。AI にはゲームについての情報が与えられ、設計者が考案したアルゴリズムに従い行動を決定する。そのアルゴリズムにはゲームについて最適化されたヒューリスティックなものもあれば、MCTS [1] などの探索、強化学習などの機械学習による一般化されたものも存在する。それらを突き詰めたり組み合わせたりすることで、単純な 2 人対戦ゲームに限っては様々なジャンルにおいて AI のゲームの強さは人間のトッププレイヤーと同等、あるいはそれを上回る地点まで到達している。例えば将棋やチェスでは、AlphaZero が探索と機械学習を組み合わせることでゲームに特化していない手法において多くのトッププレイヤーを超える実力を実現した [2]。

一方で、ゲーム AI を人間であるプレイヤーの対戦相手として用いる場合には必ずしも AI の実力が高いことが求められるとは限らない。人間はゲームプレイにおいて簡単すぎる、あるいは難しすぎる難易度に楽しさを感じにくいことが知られており [3]、楽しさのためには AI との対戦ゲームで AI の実力が自分と大きく離れていることは望ましくない。近年では、プレイヤーを楽しませる囲碁をプレイする AI の研究 [4] や、体力に差が付かないようにアクションを選択する格闘ゲームの AI の研究 [5] など、人間と同等、あるいはそれ以下の実力の AI をどのように実現するか、という方向性の研究も行われている。

そのような AI を実現するためには、プレイヤーの実力

[1] 東京大学工学部電気電子工学科
Department of Electrical and Electronic Engineering, The University of Tokyo
[2] 東京大学大学院情報理工学系研究科電子情報学専攻
Department of Information and Communication Engineering, Graduate School of Information Science and Technology, The University of Tokyo
[a)] sugimoto@logos.t.u-tokyo.ac.jp

を推定しそれに応じて AI の行動を決定する必要がある。既存研究ではプレイヤーの実力の推定に関して、ゲームの状態に評価値を与えその大小によって現在の AI とプレイヤーの優劣を判断する手法 [6] や、ゲームの様々な要素にスコアを定義しプレイヤーの今までの累計スコアをプレイヤーの実力として推定する手法 [7] などがとられている。また AI の行動決定については、AI の次の行動により前述の評価値やお互いのスコアがどう変化するかを予測し、目標の値に近づくよう次の行動を決定するという手法や、AI が取りうる行動それぞれについてどれほど優れた行動であるかをあらかじめ定義しておき推定したプレイヤーの実力によってどの行動を選択するかを決定するといった手法 [8] が存在する。

しかし、これらの手法にはいくつか問題点がある。まず 1 つは、プレイヤーの実力を推定するために、ゲームの設計者がプレイヤーの「強さ」について理解している必要がある点である。ゲームの要素にスコアを定義する、現在状態を正しく評価する、といったタスクのためには、ゲームのルールだけではなく、ゲームにおけるどの要素がどれほど重要であるかを把握している必要がある。次に、 AI が評価値やスコアを目標値に近づけるために最適な行動をした場合、場合によっては異常に速い、あるいは遅い行動や、あからさまに自分が不利になるための行動を取ることも望ましくない。これによりプレイヤーは、ゲームの勝敗を操作されているように感じてしまうことがあり、ゲームをプレイする楽しさが損なわれてしまう。最後に、既存の手法ではプレイヤーの実力は一次元の数値で表現されるため、プレイヤーの癖や狙いといった固有の戦略は推定されていないという点も問題である。最善の戦略が存在せず、戦略同士に相性があるような複雑なゲームにおいては、プレイヤーの戦略を含めた推定が AI の調整に必要だと考えられる。

本研究の目的は、AI を対戦相手とした 2 人対戦ゲームにおいてゲーム固有の知識を用いることなく一般にプレイヤーの強さを推定し、それに応じた強さでありながら不自然さを感じさせないような AI を用意する手法を作成し、プレイヤーが AI を相手とした対戦ゲームにおいてより楽しさを感じるようにすることである。また、本研究では、上で述べたような既存の手法での問題点が表れるゲームとして、研究のために開発した 2 人対戦パズルシミュレーションゲームを使用する。

## 2. 関連研究

### 2.1 ゲームの楽しさ

ゲームにおける楽しさとは何かについての研究は古くからなされており [9]、ゲームの難易度もその要素であることが示されている。Malone [10] や Lazzaro [3] は、程よい困難を乗り越えられる事がプレイヤーの満足度に繋がると述べている。また、Lomas は簡単すぎるゲームはプレイヤーを飽きさせ、関心を失わせてしまう事を示した [11]。ゲームの楽しさの指標とする要素は他にも多数提唱されているが、ゲームの難易度はその中で重要なものの 1 つである。

### 2.2 DDA (dynamic difficulty adjustment)

プレイヤーの実力を推定し、それに応じてゲームの難易度を様々な側面から調整するという研究は、DDA (dynamic difficulty adjustment) として進められている [12]。DDA の枠組みの中では、プレイヤーの実力や熟練度が何らかの方法で推定され、それを元にプレイヤーからの入力に補正がかけられる、ゲーム内の環境が変化する、AI の行動が変動するといった環境操作が行われる。その後一定期間の観測の後に、再びプレイヤーについての推定がなされる。このサイクルを続けることによって、ゲームの難易度を目標とする地点から遠ざからないようにしている。

### 2.3 CNN (convolutional neural network)

人間の脳神経系のニューロンを数理モデル化したものをいくつも組み合わせたネットワークであるニューラルネットワークの 1 つとして、CNN（convolutional neural network）という手法がある。CNN は、入力を畳み込み層 (convolution layer)、プーリング層 (pooling layer) に通すことで、入力が高次元で近くに存在する要素が比較的深い関係を持つ画像認識のようなデータセットについてより効率良く学習を進めることができる [13]。

畳み込み層では、入力データをその形を保ったまま特徴マップとして扱い、フィルターと呼ばれるパラメータを用いて畳み込み演算を行うことで入力データのサイズや次元数を調整しつつ入力の部分的な特徴を抽出することが出来る。

プーリング層では、あるサイズの領域の情報を集約するための層であり、畳み込み層の出力特徴マップのサイズを調整し、畳込み層の出力の小さなズレに対してネットワークを頑健にする目的がある。集約するための手段として、決めたサイズの領域の要素の最大値をとる max プーリング、平均値をとる average プーリングなどが存在し、画像認識などの分野においては、主に max プーリングが採用される。プーリング層は、畳込み層の出力した特徴マップに対してある決まった処理を行うだけの層であるため、学習すべきパラメータが存在しない。

## 3. 提案手法

本研究では、既存の手法での問題点を改善する以下の手法によるゲーム対戦 AI を提案する。その概要を図 1 に示す。

提案手法による対戦 AI は対戦中、数ステップ毎に用意された AI の中から 1 つの AI を選び、自身の戦略を選ばれ

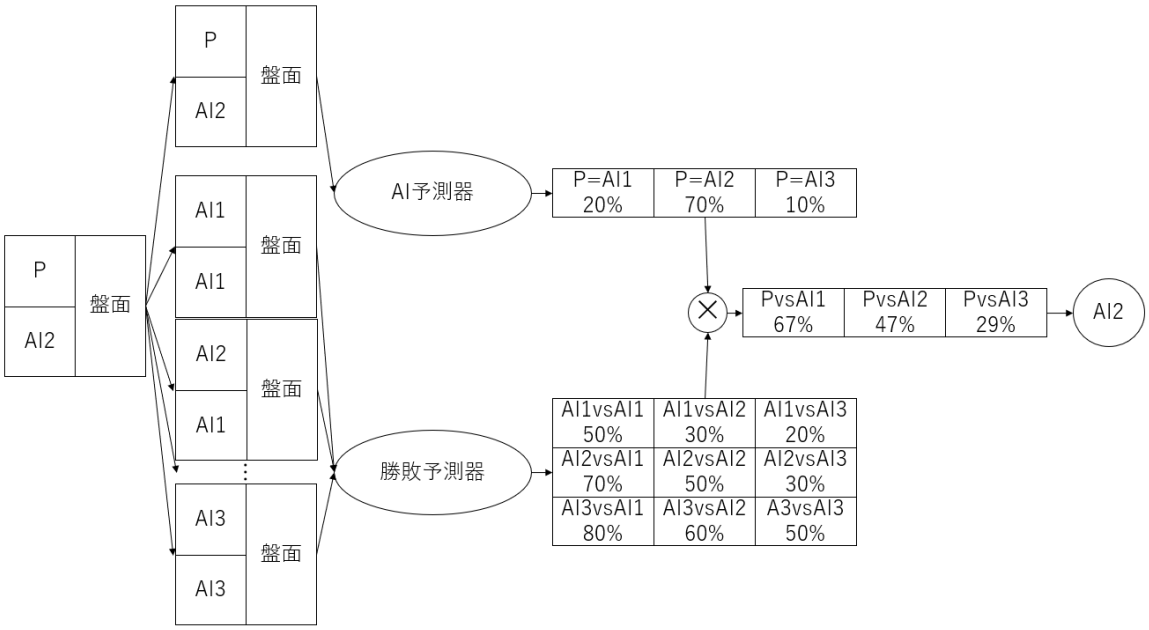


**図 1** 提案手法の概要

た AI と同じものに切り替える。AI 決定アルゴリズムの詳細を Algorithm1 に示す。

**Algorithm 1** Decide AI

**Require:** $P \in \mathbb{R}^{id}$ $W \in \mathbb{R}^{id*id}$ $r \in \mathbb{R}^{id}$
**Ensure:** $AI = \max\{r_i\}$
  **while** GamePlay **do**
    $wait(\tau)$
    $data \leftarrow GetData()$
    $P = EstimatePlayer(data, AI)$
    **for** $i = 0$ to $id - 1$ **do**
      **for** $j = 0$ to $id - 1$ **do**
        $W_{i,j} \leftarrow EstimateWinRate(data, i, j)$
      **end for**
    **end for**
    $r \leftarrow P^T \times W$
    $AI \leftarrow \arg\min_i\{|0.5 - r_i|\}$
  **end while**

Algorithm1 において、$GetData$ によって得られる $data$ とは盤面の状況である。

提案手法による対戦 AI は一定のタイムステップ $\tau$ 毎に、$EstimatePlayer$ により現在のゲームの状況から対戦プレイヤーを用意された AI に対する確率分布として推定し、これを $P$ とする。

$$P_i = p(L_e = i) \tag{1}$$

ここで $L_e$ は対戦相手の AI ラベルであり、$p(L_e = i)$ とは対戦相手がラベル $i$ を割り当てられた AI となる確率である。

続いて、現在のゲームの状況、対戦する 2 つの AI のラベルからその片側の勝率を算出する $EstimateWinRate$ を用いて各 AI 同士の勝率を行列 $W$ として得る。

$EstimatePlayer$、$EstimateWinRate$ 関数として、教師あり学習によって学習した CNN を利用する。

それぞれの学習には、事前に行った AI 同士の対戦データから得られた次の入出力学習データを用いる。

(1) EstimatePlayer
  (a) 入力教師データ：一時点でのゲームの盤面、対戦する片方の AI ラベル
  (b) 出力教師データ：もう片方の AI ラベル

(2) EstimateWinRate
  (a) 入力教師データ：一時点でのゲームの盤面、対戦する AI ラベル 1、AI ラベル 2
  (b) 出力教師データ：AI ラベル 1 の勝敗

EstimiatePlayer 関数については softmax 法を利用することで確率分布を出力する。EstimateWinRate 関数については勝利を 1、敗北を 0 としたロジスティック回帰を出力する。

この手法はゲームの戦略同士に相性が存在し絶対的な強さの序列が存在しない場合であっても、相手プレイヤーと互角な AI を用意することが可能で、ゲームの強さについての深い理解や AI 同士の強さの序列の決定を必要とせず、ゲームに特有の構造や情報を用いないため一般的に様々なゲームに適用することが出来る。またゲームの各段階では対戦相手となる AI は用意された AI のいずれかであるため、設計された 1 つ 1 つの AI が自然な挙動をすると仮定すれば不自然な手、露骨な手は現れにくいと考えられる。

一方で様々な個性を持つ独立した AI を多数用意しなくてはならないため、それが困難であるゲームについてはこの手法を適用することは難しい。また可能であっても、AI

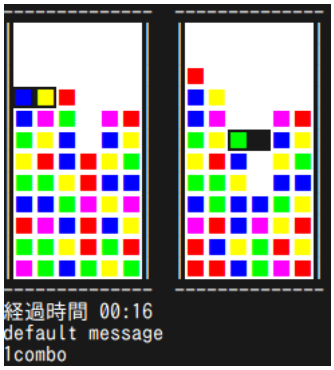


図 2　研究用ゲームのプレイ画像

が 1 つも存在しない状況からこの手法を導入するのは手間がかかってしまうのが問題である。

## 4. 実験

本稿では、研究のために作成したパズルシミュレーションゲーム上において、複数の AI を用いて学習した提案手法による AI が、その AI 決定アルゴリズムをランダムにした AI と比較して学習に用いなかった AI を相手とした時に同等、あるいはそれ以上に 50%に近い勝率を持ちつつより長く試合を続けられた事を示す。図 5 に示すのは、本研究のために開発した 2 人対戦パズルゲームのプレイ画面である。以下が、このゲームのルールである。

- プレイヤーは縦 12* 横 6 マスからなる場を所有する
- 場のマスそれぞれは空白、または様々な色のパネルが存在する
- 一定時間毎に、下に空白の存在しないパネルは 1 マス上に移動し、一番下のマスには新たにパネルが生成される
- 一番上のマスのパネルがせり上がろうとしたプレイヤーは敗北する
- 同じ色のパネルが縦、または横に 3 つ並ぶと消滅する
- パネルの下が空白のマスの場合、そのパネルは下のマスに移動する
- プレイヤーは、任意の隣り合うパネル、または空白の位置を入れ替えることができる
- 一度にたくさんのパネルを消したり、連続してパネルを消すと、相手の場の一番上のマスに妨害パネルが出現する
- 妨害パネルは、3 つ並んでも消滅せず、隣接するマスにある妨害パネル以外のパネルが消滅する際に消滅する

実験環境とする 2 人対戦ゲームとしてこのようなアクションパズルゲームを採用するのは、以下の理由により、このゲームにおいて適切な AI の調整をすることが既存手法の問題点を解決することに繋がるからである。

まず、このゲームには格闘ゲームにおける体力やレースゲームにおけるタイムのような、ゲームの準絶対的な評価指標が存在しない。このゲームでは画面の上までパネルが埋まってしまうと負けとなるが、自分の色のコマの多い方が勝利するオセロにおいて自分の色が少ないことが不利とは限らないように、このゲームにおいて自分のパネルが多いことは一度に多くのパネルを消して相手を攻撃するチャンスとも捉えられ一概に評価することはできない。よってプレイヤーの戦略をゲーム固有の評価値などから推定することは難しく、より一般的な方法を考えなくてはならない。

次に、リアルタイムで進行するゲームであるため、比較的 AI を強く、弱くすること自体は簡単であり、またそれが人間が見た時に露骨である点も重要である。このゲームで相手に勝つために何が重要かが分からずとも、とにかく AI を速く、もしくは遅く動かせばより強い、弱い AI は簡単に作成できてしまう。そのため、勝敗のために最適な行動を AI が取ることでゲームをコントロールした場合、およそ自然とは思えない AI の挙動が観測される。

またこのゲームでは実力差が明確である場合試合が早期に決着するため、1 試合のゲームプレイ時間の長さが、異なる手法間でプレイヤーとの実力差を定量的に比較する際の指標とすることが可能である。

そして、このゲームではアクションこそ少ないものの、画面の上まで溜まってしまったパネルを下に降ろすタスクや、ただパネルを消すタスク、相手を攻撃するタスクなどを並行して、あるいはどれかに集中してこなす必要があり、プレイヤーそれぞれに強さとは違った固有の戦略が存在する。そのため戦略の相性が存在する可能性があり、適切な AI の調整にはプレイヤーの強さだけでない固有の戦略の推定が必要だと考えられる。それはより複雑なゲームでは一般的で、このゲームのような単純なゲーム上でその点を考慮してプレイヤーの戦略を推定することはそれらのゲームにおける研究の足掛かりとしても有用である。

### 4.1 実験方法

本実験では、上記のパズルシミュレーションゲームにおいて 10 種類の AI を用意し EstimatePlayer 関数、EstimateWinRate 関数の学習に利用し、これとは別の 5 種類の AI と 1000 試合ずつ対戦を行い、それぞれに対する提案手法による AI とランダム AI の勝率、平均試合時間を求めた。このゲームではここにおけるランダム AI とは、提案手法と同様の一定のタイムステップ毎に提案手法の AI の学習で利用した AI の中の 1 つにランダムで切り替わる AI である。

提案手法の AI について、EstimatePlayer 関数、EstimateWinRate 関数の実装の詳細を以下に示す。

EstimateWinRate 関数のモデルは、畳込み層、ReLU 活性化層、プーリング層、全結合層、ReLu 活性化層、全結合層、シグモイド活性化層から構成されており、ハイパーパラメータは以下のようになっている。

- 畳み込み層：カーネルサイズ 3、出力チャンネル数 10、パディング 0、ストライド 1
- プーリング層：プールサイズ 2
- 全結合層：入出力ユニット数がそれぞれ（300,150），（150,1）

損失関数には平均二乗誤差を採用している。

このネットワークへの入力は、横 14、縦 12、チャンネル 18 の 3 次元データとなっている。このゲームの各プレイヤーの盤面は横 6、縦 12 グリッドで構成されており、入力のうち横 1-6 マスが片方のプレイヤーの盤面、9-14 マスがもう片方のプレイヤーの盤面を表現している。中央の 2 マスは、畳み込み層でお互いの盤面が混ざってしまわないよう全てのチャンネルに 0 を埋めている。各チャンネルの意味は次のようになっている。

- 1-6 チャンネル : 各色パネルの存在を表現するワンホットな入力であり、対応する盤面の座標にその色のパネルが存在すれば 1、なければ 0 となる
- 7 チャンネル：その座標のパネルが消える最中であった場合に、消え始めてから完全に消えるまでに対してどれほど時間が経ったかを 0 1 の実数で入力する
- 8 チャンネル：各プレイヤーのフィールドが今どれほどの速さでせり上がっているかを入力として敷きつめる
- 9-18 チャンネル：それぞれがラベル付された AI に対応し、盤面を持つ AI のチャンネルには 1 を敷きつめ他のチャンネルには 0 を敷きつめる。

EstimatePlayer 関数のモデルは、EstimateWinRate 関数のモデルと比較して最後の全結合層の出力が 10 ユニットとなっており、その後の活性化関数がシグモイド関数から softmax 関数に変更されている点で異なる。またその入力についても、片方の盤面に対応する入力の 9-11 チャンネルが全て 0 となっている点で異なる。また損失関数には多クラス交差エントロピー誤差関数を採用している。

### 4.2 実験結果と考察

実験結果を表 1 に示す。

**表 1** 提案手法とランダム AI の比較

| AI | 提案手法 | | ランダム AI | |
|---|---|---|---|---|
| 指標 | 勝率 | フレーム数 | 勝率 | フレーム数 |
| AI1 | 0.530 | 600 | 0.535 | 592 |
| AI2 | 0.450 | 576 | 0.515 | 572 |
| AI3 | 0.650 | 556 | 0.655 | 550 |
| AI4 | 0.885 | 465 | 0.890 | 457 |
| AI5 | 0.400 | 626 | 0.340 | 602 |

勝率は特定の AI に対して提案手法、ランダム AI が勝利した割合を表し、フレーム数は行った全試合のフレーム数の平均を取ったもので試合時間の指標となっている。勝率については提案手法による AI とランダム AI の間でどの AI に対しても大きな違いは見られなかった一方で、試合時間に関してはランダム AI よりも提案手法による AI の方が比較的長くなっていることが見て取れる。

まず提案手法による AI の試合時間が長くなったことによって、AI 決定アルゴリズムが動作はしているということが分かる。AI 決定アルゴリズムは CNN を用いて 2 つの関数が学習された上で動作すれば、自分が不利な状況ではより強い AI を選択し敗北を避けようとし、自分が有利な状況ではより弱い AI を選択し相手の敗北を避けようとする。その結果ゲームの決着が先延ばしになり、試合時間が長くなると考えられる。

一方で、提案手法による AI とランダム AI の間でどの AI に対しても勝率に関して差が見られなかったのには、様々な原因が考えられる。

- CNN の学習不足
  本実験で用いた学習データは学習用に用意した AI10 種類の、同じ AI 同士を含む対戦の各時点データ約 100,000 件ほどであるが、このゲームの状態数は膨大なもので、さらに本実験の環境においてはプレイヤーのラベルや試合の経過時間なども CNN の入力であるためもっと多くのデータを学習に利用するべきであった可能性がある。また、集めた学習用データは全て特定の 2 つの AI が始めから対戦したデータから得ているため、強い AI が弱い AI に追い詰められている、などといった特定の状況のデータが不足しており、運用の際にそれらを仮定して現在の状況での各 AI 同士の勝率を求める事は難解である。これらの問題の対策には、単純にデータ数を増やす他、学習時にはゲームの開始時点で各プレイヤーの盤面をそれぞれランダムに初期化する、学習データを収集するゲームの最中にもプレイヤーの AI を切り替えるなどの方法が考えられる。
- *EstimatePlayer* 関数、*EstimateWinRate* 関数のモデルが不適切である
  今回用いた 2 つの関数のモデルは盤面やプレイヤーの AI ラベルを全てまとめた 3 次元構造を入力とした単純な CNN モデルであったが、このモデルが不適切であるために学習が進まない、あるいは学習が正しく進んだとしても 2 つの関数が理想的な出力を返さない可能性が考えられる。モデルの改善の案としては、各プレイヤーの盤面をそれぞれ独立した畳み込み層、プーリング層に通し特徴を抽出した後でその他の情報と繋ぎあわせる事でプレイヤーに割り当てられる盤面単位での特徴抽出をより容易にする、数フレーム前の盤面の情報も含む時系列データを入力することで盤面の推移を特徴として捉えることを可能にするなどが挙げられる。

- 学習用の AI のバリエーション不足
  本実験のために用意した 10 種類の AI は、どれも基本的な動作を行うためその強さや戦略に明確な差異が無く、途中の AI の変化が試合の勝敗を覆すほどの影響を持たなかった可能性が考えられる。これについては、多種多様な AI を用いて学習を行うことで改善が見込まれる。

## 5. 終わりに

本稿では研究のために作成されたパズルシミュレーションゲーム上において、提案手法による AI がランダムに AI を切り替える AI と比べて複数の AI に対して長く試合を続けられる事を示した。しかし、勝率についてランダムな AI と差が生まれない原因については不明であり、その調査が課題である。

また、本実験では試合時間の長さを対戦するプレイヤーが互角であることの指標として用いたが、実際にこのゲームを人間がプレイした際に試合時間が長くなることによって自分と対戦相手の実力がより拮抗していると感じられるかどうかについても、実験を行い確かめる必要がある。

その他現段階で考えられるこの手法の課題としては、現在の提案手法においてプレイヤーは用意した AI の要素の組み合わせのような形で表現されており、その時点ではプレイヤーの戦略は強さという一次元の数値とは違って推定されているものの、その利用段階ではそれぞれの AI についての要素を別々に利用し計算しており、推定したプレイヤー固有の戦略を AI の調整に活かせているとは考えにくい点が挙げられる。今後の研究では推定したプレイヤーの戦略を AI の調整にどう活用するかを中心に模索していきたい。

### 参考文献


[1] Coulom, R.: Efficient selectivity and backup operators in Monte-Carlo tree search, *International conference on computers and games*, Springer, pp. 72–83 (2006).

[2] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T. et al.: Mastering chess and shogi by self-play with a general reinforcement learning algorithm, *arXiv preprint arXiv:1712.01815* (2017).

[3] Lazzaro, N.: Why we play games: Four keys to more emotion without story (2004).

[4] 池田　心: モンテカルロ碁における多様な戦略の演出と形勢の制御: 接待碁 AI に向けて (2012).

[5] Demediuk, S., Tamassia, M., Raffe, W. L., Zambetta, F., Mueller, F. F. and Li, X.: Measuring Player Skill Using Dynamic Difficulty Adjustment, *Proceedings of the Australasian Computer Science Week Multiconference*, ACSW '18, New York, NY, USA, ACM, pp. 41:1–41:7 (online), DOI: 10.1145/3167918.3167939 (2018).

[6] Buro, M.: The evolution of strong Othello programs, *Entertainment Computing*, Springer, pp. 81–88 (2003).

[7] Jennings-Teats, M., Smith, G. and Wardrip-Fruin, N.: Polymorph: dynamic difficulty adjustment through level generation, *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, ACM, p. 11 (2010).

[8] 中川明紀，逢坂翔太，柴崎智哉ほか: ニューラルネットワークによる格闘ゲーム AI の難易度調整及び行動多様性向上手法，全国大会講演論文集，No. コンピュータと人間社会, pp. 801–802 (2008).

[9] Yannakakis, G. N. and Hallam, J.: Towards optimizing entertainment in computer games, *Applied Artificial Intelligence*, Vol. 21, No. 10, pp. 933–971 (2007).

[10] Malone, T.: *What makes computer games fun?*, Vol. 13, No. 2-3, ACM (1981).

[11] Lomas, D., Patel, K., Forlizzi, J. L. and Koedinger, K. R.: Optimizing challenge in an educational game using large-scale design experiments, *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, pp. 89–98 (2013).

[12] Hunicke, R.: The case for dynamic difficulty adjustment in games, *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*, ACM, pp. 429–433 (2005).

[13] Krizhevsky, A., Sutskever, I. and Hinton, G. E.: Imagenet classification with deep convolutional neural networks, *Advances in neural information processing systems*, pp. 1097–1105 (2012).