幾何情報のあるモデルアーキテクチャとペアワイズ学習による汎用ボードゲーム AI 手法

A General Method for Board Game AI Using Geometric Model Architecture and Pairwise Learning

千葉　直音
Nao Chiba
法政大学情報科学部コンピュータ科学科
E-mail: nao.chiba.7i@stu.hosei.ac.jp

Abstract

Board games have complex rules and many different states, which usually requires a lot of effort and cost to build an AI that can play them well, also makes it harder to test and evaluate the game, and slows down the development process. Therefore, reducing the time and effort needed to build an AI is important for efficient game development. In this paper, focusing on a two-player, finite, deterministic, perfect-information game, we propose a method to automatically generate an AI using only the board representation and the game rules, without adding game-specific algorithms. By contrast to AlphaZero, which produces a target by performing Monte Carlo tree search with many simulations and neural network inferences for each move, our method generates many subtrees of the game tree and computes an evaluation value for each position using retrograde analysis. We then introduce (1) Geometric Model Architecture and (2) the ranking order of positions by pairwise learning. By collecting more targets per move, our method aims to improve samples efficiently. We develop our evaluation code for the agent against a rule-based AI and adapt it to the AlphaZeroGeneral open-source framework. In our experiments, we measure and compare the win rates of our agent and AlphaZeroGeneral.

1. はじめに

ボードゲームのような戦略系ゲームは，複雑なルール体系や多様な局面が存在するため，それらを攻略できる AI を作成するには多大な労力とコストが必要となる．特に AI の設計や実装に長い時間を要すると，そのゲーム自体のテストプレイや検証が円滑に進まなくなり，結果として開発全体の進行速度に大きな影響を及ぼす．したがって，AI の構築にかける時間や労力を削減することは，効率的なゲーム開発で非常に重要な課題である．

人為的にゲーム専用のアルゴリズムを設計した場合，そのアルゴリズムは特定のルールや盤面構造に強く依存する．このような性質をもつ AI は，一度ルールを変更すると，それに応じて多くの修正作業を繰り返し行わなければならず，開発効率を著しく低下させる要因となる．よって，できる限りゲーム固有のアルゴリズムに依存せず，盤面とルールといった汎用的な情報のみを入力として AI を自動生成できる仕組みが求められる．このようなアプローチが実現できれば，人為的な修正作業を大幅に減らし，効率的かつ柔軟なゲーム開発が可能となる．

本研究ではこの課題に対処するため，離散的な盤面状態をもち，各マスがT種類の値をとる二人有限確定完全情報ゲームを対象に，盤面状態とルールのみを利用しゲームを攻略する AI を構築する方法を提案する．ボードゲームの幾何情報を入れたモデルアーキテクチャを用いてペアワイズ学習を行うことで，ゲームルールに特化したアルゴリズムに依存せず，汎用的に利用可能な AI の自動生成手法を実現する．本研究では，AlphaZero の一般的な実装として公開されている AlphaZeroGeneral のコードにルールベース AI との対戦評価コードなどを追加，改変する．実験ではオセロ，Connect4，五目並べで同スペックのコンピュータを用いて同時間の学習時間における AlphaZeroGeneral と自作エージェントそれぞれのルールベース AI に対する勝率を計測し比較する．

2. 関連研究

Silver ら [1]はルールのみを与え，モンテカルロ木探索(MCTS)と CNN などのニューラルネット(NN)を組み合わせ，自己対戦を繰り返す手法を用いて，チェス，将棋，囲碁の異なるゲームを同一のアルゴリズムで扱えるようにした．田中 [2]は後退解析により「どうぶつしょうぎ」を完全解明し，後手必勝であることを示した．全盤面がわかればこのように同一手法でボードゲーム AI をつくることも可能である。

3. AlphaZero の学習アルゴリズム

Silver ら [3]による AlphaZero は，状態sを入力として合法手の確率分布pと状態価値vを同時に出力するモデルを用いる．次の手順によりターゲットを作成する．

① 初期盤面から終局まで自己対戦を行う．
② 手番tごとにモデルを使ってモンテカルロ木検索(MCTS)を数百～数千回シミュレーションする．

③　得られた$(s_t, \pi_t(a))$に対し，自己対戦の終局の結果z_tをバックアップし，$(s_t, \pi_t(a), z_t)$を作成する．

しかし，この手法は初期にランダムプレイよりで安定するまで大量の自己学習が必要であり，1 手ごとに大量の MCTS シミュレーションと NN 推論を行って 1 つターゲットを作るなど，サンプル効率が良くない問題がある．実際，AlphaZero では第 1 世代 TPU が 5000 基，第 2 世代 TPU が 64 基用いられており [1]，膨大なマシンパワーや資材が必要であることが分かる．

4. 提案手法

本研究では，サイズNの離散的な盤面状態をもち，各マスがT種類の値をとる二人有限確定完全情報ゲームを対象として，盤面表現とルールのみを与え，ゲーム特有のアルゴリズムを組み込まずに AI を自動生成する手法を提案する．AlphaZero の手法は，各手番tの MCTS で大量の MCTS シミュレーションと NN 推論を行って 1 つターゲットを作成する．しかし，提案手法はここでゲーム木の部分木を多数生成し，簡易なミニマックス探索により各局面の評価値を算出する．そのうえで，これらの局面が勝率の高い順に並ぶよう順位関係と各盤面の評価値をターゲットとして得ることで，1 手ごとのターゲット収集量を多くしサンプル効率を改善することを目指す．

以下では，勝敗表現をすべて先手目線で行う．先手が勝利することを勝利，勝ち等と表現し，後手が勝利することを敗北，負け等と表現する．

4.1. 幾何情報を入れたモデルアーキテクチャ

本研究では AlphaZero と同様，2 つのヘッドを持つモデルを用いる．1 つ目のヘッドは方策ヘッドであり，エージェントはこの方策ヘッドの良い(先手番であれば高い，後手番であれば低い)状態を選んで行動する．2 つ目のヘッドは価値ヘッドであり，このヘッドはある盤面から方策ヘッド通りに行動した際の勝ちやすさを表現する．以下では入力sに対する方策を$\mathrm{model}(s).p$，価値を$\mathrm{model}(s).v$と表記する．また，価値ヘッドの重みは初期状態で非常に小さく初期化されている．モデルは幾何的情報を含ませるt種類のサブモデルと線形アテンションブロックから構成される(図 1).

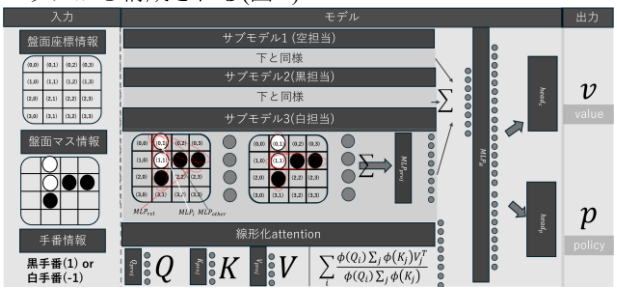

図 1　モデルアーキテクチャ

4.2. ゲームの部分木

本提案手法ではゲーム木の部分木を幅k，深さdで作成する．この部分木は各ノードに盤面状態sと仮評価値lv，を保有する．

4.2.1. ゲームの部分木の作成

部分木はモデルを用いた自己対戦を行いながら作成する．先手はモデルの方策が高い値の盤面を選択し，後手は方策の低い盤面を選択するように行動する．この時，collect_rate回に 1 回，深さd，幅，枝分かれkのゲーム探索木を作成する．この時作られる部分木は，現在の盤面を根として，k個の盤面ずつd回枝分かれするものである，この際，k個の盤面として，盤面iが選ばれる確率は

$$p(s_i) = \frac{\exp((-1)^t \mathrm{model}(s_i).p)}{\Sigma_j \exp\left((-1)^t \mathrm{model}(s_j).p\right)}, t = \begin{cases} 0\,(\text{先手番}) \\ 1\,(\text{後手番}) \end{cases}$$

である．ただし，次の手が終局である場合，その状態s_eに遷移する確率は一様で以下の通りである．

$$p(s_e) = \frac{1}{s_e\text{の数}}$$

4.2.2. 仮評価値lvと方策学習用wrの設定

部分木を作成後，各ノードで仮評価値lvを設定する．

①　葉の場合：

$$lv = \begin{cases} \mathrm{model}(s).v\,(\text{非終局}) \\ 1(\text{勝利}) \\ 0(\text{引き分け}) \\ -1(\text{敗北}) \end{cases}$$

$$wr = (\text{定数} - 1) \cdot \frac{\text{勝った回数} - \text{負けた回数}}{\text{ロールアウト回数}} + \text{定数} \cdot lv$$

②　葉以外の場合：そのノードの子のs, lv, wvを$\mathrm{child}.s, \mathrm{child}.lv, \mathrm{child}.wr$と表記すると，親の$lv, wr$は以下の通りである

$$lv = \mathrm{sum}(p(s_i) \cdot \mathrm{child_i}.lv)$$
$$wr = \mathrm{sum}(p'(s_i) \cdot \mathrm{child_i}.wr)$$

wrの伝播では，正規化されている遷移確率$p'(s)$を用いるがlvの伝播では正規化されていない遷移確率$p(s)$を用いている．これによって，伝播が続くとlvは小さくなっていくが，これは誤差を伝播させない意図と$\mathrm{model}(s).v$の精度が高い部分と低い部分の比較でターゲットが不安定になることを防ぐ意図がある．

4.2.3. 教師データの取得

方策ヘッドは 2 つの盤面とその差をどれだけにするかのターゲットを取得し，価値ヘッドはある盤面に対しその評価値をどれくらいにするかのターゲットを取得する．

①　方策ヘッド学習用の教師データ：あるノードの子で，wrがi番目に大きいノードをchild_iとする．子が 2 つ以上あるすべてのノードで

$$\mathrm{model}(\mathrm{child}_i.s).p - \mathrm{model}(\mathrm{child}_{i+1}.s).p$$

のターゲットを以下とする．

$$0.5 + 0.5 \cdot \frac{(\mathrm{child}_i.wr - \mathrm{child}_{i+1}.wr)}{\mathrm{margin}}$$

②　価値ヘッド学習用の教師データ：葉以外のすべてのノードで$\mathrm{model}(s).v$のターゲットをlvにする．

4.3. 学習

モデルの出力によるスコアzとそれに対応するターゲットをyで，方策ヘッドに関する損失関数は交差エントロピーを用いて以下の通り定める．

$$\mathrm{loss}_p = -\big(y\log\sigma(z) + (1-y)\log\big(1-\sigma(z)\big)\big)$$

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

また，価値ヘッドに関する損失関数は二乗誤差を用いて

$$\mathrm{loss}_v = \frac{1}{2}(z-y)^2$$

とし，全体での損失を$\mathrm{loss}_p + \lambda\mathrm{loss}_v$として学習する．

5. 実装

5.1. モデルアーキテクチャ

モデルはマスの取りうる状態t種類のサブモデル(5.1.1 節で示す)と，線形アテンションブロック(5.1.2 節で示す), MLP の

$$\mathrm{MLP_H}\colon \mathbb{R}^{2h+2+1} \to \mathbb{R}^{g}$$

$$\mathrm{head_p}\colon \mathbb{R}^{g} \to \mathbb{R}^{1}\ (\text{方策ヘッド})$$

$$\mathrm{head_c}\colon \mathbb{R}^{g} \to \mathbb{R}^{1}\ (\text{価値ヘッド})$$

をもち，出力は$[B]$の方策と価値である．$[B,S]$のマス情報を$[B,S,\mathrm{emb}]$にし，$[B,S,\mathrm{emb}]$, $[B,S,2]$, $[B]$をT種類のサブモデル，線形アテンションブロックに入力，その出力を上記の MLP にかける．

5.1.1. サブモデル

サブモデルは入力に$[B,S,\mathrm{emb}]$のマスの種類情報と$[B,S,2]$の座標情報，$[B]$の手番情報，$[B,S]$のマスク情報をとり，以下の MLP をもつ．

$$\mathrm{MLP_i}\colon \mathbb{R}^{emb+1} \to \mathbb{R}^{f}\ (\text{あるマス}_{\mathrm{i}}\text{の意味ベクトル})$$

$$\mathrm{MLP_{other}}\colon \mathbb{R}^{emb+1} \to \mathbb{R}^{f}\ \big(\text{他のマスの意味ベクトル}\big)$$

$$\mathrm{MLP_{rel}}\colon \mathbb{R}^{9+1} \to \mathbb{R}^{f}\ (\text{幾何的関連性を表す})$$

$$\mathrm{MLP_{proj}}\colon \mathbb{R}^{f} \to \mathbb{R}^{h}\ (\text{次元の拡張})$$

$[B,S,emb+1]$の埋め込みベクトルの盤面情報を$\mathrm{MLP_i}$にかけ，$[B,S,f]$でマス$_{\mathrm{i}}$の意味ベクトル①を生成する．次に，$[B,S,k,emb+1]$の近傍kを，$\mathrm{MLP_{other}}$にかけ，$[B,S,k,f]$でマス$_{\mathrm{i}}$の近傍kの意味ベクトル②を生成する．マス$_{\mathrm{i}}$の近傍kの相対座標を用いて，9つの幾何情報(x, yの差や角度など)を得る．$[B,S,k,9+1]$の相対情報を$\mathrm{MLP_{rel}}$にかけ，$[B,S,k,f]$でマス$_{\mathrm{i}}$と近傍kの幾何的関係ベクトル③を得る．②と③の積を 2 次元方向に和を取り$[B,S,f]$のマス$_{\mathrm{i}}$の近傍kの関係ベクトル④を得る．①と④の積$[B,S,f]$を$\mathrm{MLP_{proj}}$にかけ，$[B,S,h]$のテンソルを得る．

5.1.2. 線形アテンション

アテンションでは各マスiで

$$\mathrm{Attention}(i) = \sum_j \frac{\exp\big(q_i^T k_j\big)}{\sum_{j'}\exp\big(q_i^T k_{j'}\big)} v_i$$

となり，全体のアテンション計算に$O(S^2)$の計算量が必要であるが，

$$\exp(q_i^T k_j) \approx \phi(q_i)^T \phi(k_j)$$

$$\phi(x) = 1 + \begin{cases} x\ (x \geq 0) \\ \exp(x) - 1\ (x < 0) \end{cases}$$

とすれば

$$\mathrm{Attention}(i) \approx \sum_j \frac{\phi(q_i)^T\phi\big(k_j\big)}{\sum_{j'}\phi(q_i)^T\phi\big(k_{j'}\big)} v_i = \frac{\phi(q_i)^T\Sigma_j\phi\big(k_j\big)v_j^T}{\phi(q_i)^T\Sigma_j\phi\big(k_j\big)}$$

となり，$\Sigma_j\phi\big(k_j\big)$, $\Sigma_j\phi\big(k_j\big)v_j^T$を先に計算すれば$\mathrm{Attention}(i)$が$O(1)$で求まり，全体として$O(S)$の計算量で済む [4]．本実装では以下の MLP により，Q, K, V を出力する．

$$\mathrm{Q_{proj}}\colon \mathbb{R}^{\mathrm{emb}+2+1} \to \mathbb{R}^{\mathrm{da}}$$

$$\mathrm{K_{proj}}\colon \mathbb{R}^{\mathrm{emb}+2+1} \to \mathbb{R}^{\mathrm{da}}$$

$$\mathrm{V_{proj}}\colon \mathbb{R}^{\mathrm{emb}} \to \mathbb{R}^{\mathrm{h}}$$

5.2. データ収集

データ収集を C++で実装する．Python 側で作成したモデルを TorchScript に保存し，C++側で利用する．Python 側から$[d, \mathrm{collect_rate}, k]$を渡し，collect_rateステップごとに幅$k$，深さ$d$の部分木を作成する．部分木は構造体 Node，Connect，Graph から生成する．Node はlvを保有し，Connect は辺を表す．Graph は出発ノードと Connect の集合を保有する．Python から呼び出されたこのコードはペアワイズ学習用の(s_h, s_l, t_p)と MSE 学習用の(s,t)を返す．

5.3. 学習

学習は Python で行う．5.2 節で得られる(s_h, s_l, t_p), (s,t)に対して，損失関数を以下の通り定める．

$$\mathrm{loss_p} = \mathrm{F.binary\_cross\_entropy\_with\_logits}(\mathrm{model}(s_h).p - model(s_l).p, t_p)$$

$$\mathrm{loss_v} = \mathrm{F.mse\_loss}(\mathrm{model}(s).v, t)$$

6. 実験

6.1. 手順

本実験では五目並べ，オセロ，Connect4 で同スペックのコンピュータを用いて同時間の学習時間における AlphaZeroGeneral と自作エージェントそれぞれのルールベース AI に対する勝率を計測し比較する実験を行う．以下，AlphaZeroGeneral のエージェントを AA，提案手法エージェントを MA，ルールベースエージェントを RA と表記する．AA，MA の行動は方策ヘッドで貪欲に選択を行う．これはテスト時に探索を行わずモデルの推論だけで行動することである．

6.1.1. 五目並べ

11×11 の五目並べを対象に学習し実験を行う．RA の方策を以下の通りとする．

① 次の手で勝てる盤面であれば勝つ．
② 相手が次勝ってしまう場合は防ぐ．

これ以外の場合，以下の③から⑤のスコア計算により手を確率的に決める．eは両端が空いているかいないかで値が変わる数値である．

③ 後1, 2, 3個で自分が 5 連揃う時，それぞれ$+1000e$, $+200e$, $+25e$とする．
④ 後1, 2, 3個で相手が 5 連揃う時，それぞれ$-900e$, $-150e$, $-18e$とする
⑤ 列cに対して，$-0.15|c - \mathrm{center}|$とする．

6.1.2. オセロ

8×8 オセロを対象に学習し実験を行う．RA の方策は図 2 のスコア [5]によって最も高いスコアである場所に石を置く．

68	-12	53	-8	-8	53	-12	68
-12	-62	-33	-7	-7	-33	-62	-12
53	-33	26	8	8	26	-33	53
-8	-7	8	-18	-18	8	-7	-8
-8	-7	8	-18	-18	8	-7	-8
53	-33	26	8	8	26	-33	53
-12	-62	-33	-7	-7	-33	-62	-12
68	-12	53	-8	-8	53	-12	68

図 2　オセロ RA のスコア図 [5]

6.1.3. Connect4

6 行×7 列の Connect4 を対象に学習し実験を行う．RA の方策を以下の通りとする．

① 次の手で勝てる盤面であれば勝つ．
② 相手が次勝ってしまう場合は防ぐ．

これ以外の場合，以下の③から⑥のスコア計算により手を確率的に決める．

③ 後1, 2, 3個で自分が 4 連揃う時，それぞれ$+120$, $+12$, $+1$とする．
④ 後1, 2, 3個で相手が 4 連揃う時，それぞれ-140, -14, -1とする．
⑤ k個$(k \geq 2)$のフォークに対して，$+220 + 60(k-2)$とする．
⑥ 列cに対して，$-2|c - \text{center}|$とする．

6.2. 結果

以下では，それぞれのゲームで MA，AA の先手，後手の勝率の推移を記述している．横軸が学習時間(s)であり，縦軸が RA に対する勝率である．

6.2.1. 五目並べ

五目並べの勝率の推移を図 3 に示す．

① 先手：MA の勝率は 0.1〜0.6 を上下しつつ，後半にかけて徐々に上がっていく．一方，AA はほぼ常に 0 で，ルールベース AI にほとんど勝てていない．
② 後手：AA，MA ともにほぼ 0 で推移している．

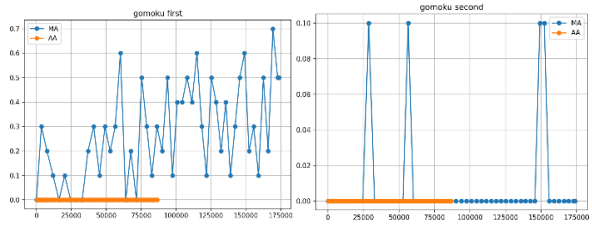

図 3　五目並べでの実験結果

6.2.2. オセロ

オセロの勝率の推移を図 4 に示す．初期は AA と MA が近いか，MA の方がやや良い区間もあるが，学習が進むにつれて AA の勝率が上がり，0.5 近くまで到達している．MA はむしろ終盤でやや下がり気味になる．

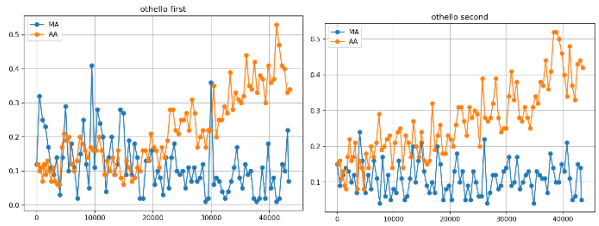

図 4　オセロでの実験結果

6.2.3. Connect4

Connect4 の勝率を図 5 に示す．

① 先手：グラフの前半では AA が 0.8〜1.0 近い高い勝率を出している区間があるが，学習が進むと急激に勝率が落ち，後半では 0 付近まで下がっている．MA は全体としては低めだが，所々で 1.0 近いスパイクが立っている．
② 後手：MA が 0.8〜1.0 近いスパイクを先手よりも頻繁に出している．

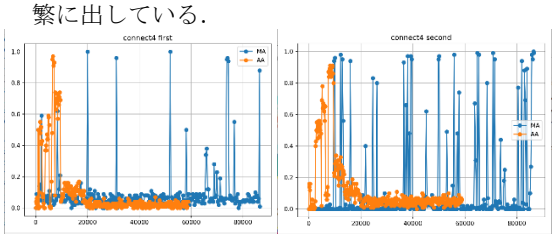

図 5　Connect4 での実験結果

7. 議論

勝率が低くなる原因として，モデルアーキテクチャが過度に視覚情報に頼りすぎる点がある．オセロなどは，五目並べ、Connect4 に比べ視覚的に明らかに勝率が上がったりする手が少ない．本提案手法のモデルアーキテクチャは視覚的に結びつける用のパラメータが多く，視覚的にわかりにくいものは，学習が不安定になると考えられる．実際に，角に置くなどの視覚的明らかな有効手は学習により最善手に浮上していた．五目並べでも三三を作る手などが最善手としてあげられている．そのため，視覚的に明らかな有効手が多い五目並べや Connect4 は，高い勝率を記録していると考える．

ランダムロールアウトの勝率と，実際の勝利が合わない場合も，勝率が上がりにくいと考えられる．例えば，五目並べの後手の場合，初期はランダムロールアウトの勝率を頼りに進んでいくが，先に先手の有効な方策が定まっているため，勝率の高い手が方策の低い手となり，ランダム勝率の低い手を選ぶようになっていくと考えられる．この時，Connect4 のように合法手が少なければ有効手にたどり着けて，価値を方策決定で支配的にしていけるが，五目並べのように合法手が多いと有効手が見つからずにランダム勝率と敗北の伝播に翻弄されることになる．これは後手の五目並べの勝率が上がらなかった原因であると考えられる．

さらに，図 6 は Connect4 における損失の変化であり，Connect4 のように 1 手で勝敗が大きく変化するようなボードゲームはそれに対応できている時とできていない時で，大きく勝率が変化すると考えられる．AA もオセロの時より勝率の変化が認められ，ランダム勝率に方策の一部を支配されている MA はより安定していない．五目並べ先手も 1 手のミスが大きく影響するが，Connect4 程ではないため，変化が抑えられているのだと考える．

上記から五目並べ先手が比較的優秀な結果であったのは，オセロなどよりも勝利への道が常に視覚的に分かりやすく，1 手のミスの影響も Connect4 に比べて小さいからであると考えられる．この場合、部分木を用いたペアワイズ学習により，従来手法より高速な学習が可能であった．

図 6　Connect4 における損失の遷移

8. おわりに

本研究では，特に，視覚的に分かりやすく合法手の多いボードゲームで既存の手法よりも高速に学習できる可能性のある汎用的なボードゲーム学習アルゴリズムを提案した．本研究では先手後手対称的なゲームを対象に実験を行ったが，AA は先手後手ルールが対称なものとして常に手番視点の盤面に変換した状態が用いられている．しかし，本手法は盤面の変換は用いず，先手後手非対称のゲームにも対応できるようにしているため，そのようなゲームでの実験も対象になりうる．

今後の課題として，ボードゲーム以外へのシミュレーションへの活用なども可能か検討する．また，ランダム勝率の支配率を変更し，どのように学習時間がかわるかなども調べる必要がある．

文　献

[1] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan and D. Hassabis, "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm," *Science,* vol. 362, no. 6419, pp. 1140-1144, 2018.

[2] 田中哲郎, "「どうぶつしょうぎ」の完全解析," 情報処理学会研究報告：ゲーム情報学(GI), vol. 22, no. 3, pp. 1-8, 2009.

[3] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel and D. Hassabis, "Mastering the game of Go without human knowledge," *Nature,* vol. 550, no. 7676, pp. 354-359, 2017.

[4] A. Katharopoulos, A. Vyas, N. Pappas and F. Fleuret, "Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention," *Proceedings of Machine Learning Research,* vol. 119, pp. 5156-5165, 2020.

[5] "もりとにーのブログ," [Online]. Available: https://tony-mooori.blogspot.com/2015/10/aic.html.