# ポリシーネットワークとバリューネットワークを併用した強化学習アルゴリズムのターン制戦略ゲームへの適用

木村 富宏[1,a)]

**概要**：ディープマインド社による AlphaGo/AlphaGo Zero/AlphaZero の発表により，ディープニューラルネットワークを使用した学習アルゴリズムが大きく注目を集めゲームアルゴリズムは急速な進歩をとげようとしている．しかしながら，AlphaZero の手法をターン制戦略ゲームに適用するにはゲームに必要なデータをニューラルネットワークで表現するには大きくなりすぎたり複雑になりすぎるといった問題があり，そのまま単純には適用できなかった．また，学習するための棋譜データの蓄積が少ない問題もあった．本研究では行動ユニットの情報をニューラルネットワークの入力側に配置することで出力側のニューラルネットワークの負担を軽減する探索手法を提案し，ポリシーネットワークとバリューネットワークを一つのネットワークに統合することで設計負担を減らし性能を向上させ、自己対戦で学習する AlphaZero の学習アルゴリズムの適用について報告する．作成した AI は学習していない大規模マップでも既存アルゴリズムに対して勝ち越すことができた．

# Application of reinforcement learning algorithm using policy network and value network to the turn-based strategy game

Kimura Tomihiro[1,a)]

***Abstract:*** Since the announcement of AlphaGo/AlphaGo Zero/AlphaZero by Deep Mind, the learning algorithm using deep neural networks have attracted much attention and the game algorithm is going to advance rapidly. However, the AlphaZero approach could not be implemented in a turn-based strategy game simply because the data required for the game was too large or too complex to be represented by neural networks. There was also a problem that there was little storage of game record data for learning. In this research, we propose a search method that reduces the burden on the neural network on the output side by placing the information of the action unit on the input side of the neural network, and the design burden by integrating the policy network and the value network into one network. This paper reports on the application of AlphaZero's learning algorithm, which eliminates the need to generate game record data by reducing self-learning and improving performance.

## 1. はじめに

オセロ、チェス、将棋のようなボードゲームにおいてゲームプログラムが人間のプロに勝利する出来事がつづいたが，囲碁に関してはプログラムはまだしばらくは人間のプロには勝てないだろうと思われていたところに，2016年に AlphaGo [1] が突然発表され囲碁のプロプレイヤーとの対戦で勝利するようになった．AlphaGo　は最初のバージョンではポリシーネットワークとバリューネットワークは分離され，ポリシーネットワークによる方策評価とバリューネットワークによる局面評価に加えて、モンテカルロ探索でよく使用される終局までのシミュレーションを行うロールアウトがあったが，引き続いて発表された AlphaGoZero [2] のバージョンではポリシーネットワークとバリューネットワークは一つに統合されロールアウトは

[1] 北陸先端科学技術大学院大学
Japan Advanced Institute of Science and Technology, JAIST
[a)] s1520752@jaist.ac.jp

廃止された．さらには AlphaGoZero の一般化アルゴリズムを目指したバージョンとして AlphaZero [3]　が登場し，AlphaGoZero に適用されたアルゴリズムがチェスと将棋にも適用可能であり高い性能が出ることが示された．同様の手法を囲碁や将棋以外の他のゲームに適用した場合の研究の進展についても期待されるところである．

さらに複雑なゲームの研究を考えた場合，ターン制戦略ゲームは選択肢の一つといえる．ターン制戦略ゲームは歴史的にみるとボードゲームのみしかなかった時代にはウォーシミュレーションゲームとして長く人々に親しまれ，ビデオゲームの時代となってからも，数多くの有名で人気のあるゲームタイトルが発売され，現在もひとつのカテゴリーとして人気のあるゲーム分野である．しかし，ターン制戦略ゲームに内臓される駒を行動させる思考ルーチンのレベルは AI アルゴリズムの設計の困難さから人間プレーヤーに比してもレベルが高いものはなかなか出てこなかった．

本研究では AlphaZero によって示された深層学習を使用したポリシーネットワークとバリューネットワークを組み合わせた探索手法と学習手法をターン制戦略ゲームへと適用し，過去の棋譜データを必要としない自己対戦で自ら強くなっていく思考アルゴリズムがより正確で高度な判断ができるようになることで，旧来のアルゴリズムでは正しく動作して正解にたどり着くことが困難だったマップで適切な動作ができるようになったり，強いプレーが行えるようになること，結果として，選択できるアルゴリズムの範囲が広がることを示す．

## 2. 既存研究

ターン制戦略ゲームは、市場で人気もあり歴史のある分野でありながら強い AI アルゴリズムの設計は困難な場合が多い．この理由としては，将棋や囲碁に比較しても探索空間が広大であること，扱う駒の種類や地形が豊富でデータ量が大きくなること，一つのターン内で複数の駒が同時に行動することが可能であるなどといった理由から探索木の分岐数が容易に計算爆発を起こすためである．

このようななかで研究を推進するためにターン制戦略ゲームに向けた学術研究用プラットフォーム TUBSTAP [4] が提唱された．TUBSTAP は過去のターン制戦略ゲームの有名タイトルのゲームルールや特徴を分析して必要最小限のルールを抽出して再構成しゲームとしてプレーできるようにした研究用プラットフォームである．

TUBSTAP を使用して既存のゲームアルゴリズムを適用する試みとして UCT 探索を使用するもの [5] やミニマックス法で局面を分割して探索する手法 [6] などが提案されたがなかなか性能が上がらなかった．

深層学習のターン制戦略ゲームへと適用する研究として経路探索マップでの動作決定に深層学習を使用するもの [9]，MCTS アルゴリズムの対戦棋譜から学習して対戦に使用するもの [10] などがあるが駒の数が一個と限られたり，リカレントネットワークを使用して出力ニューロン数を削減しているために，バリューネットワークのような他の出力を付加しにくい，マップデータや大量の対戦データを用意しなければならないという制約があった．

囲碁や将棋であれば過去の研究から大量の棋譜データの蓄積があるがターン制戦略ゲームではそのように活用できるデータがないのも研究上の課題であるが、自己対戦で学習するシステムであれば棋譜は自動生成されるのでこの点は解決できる．

## 3. 提案手法

今回提案する学習システムのシステム構成について説明する．システムは大きく分けてニューラルネットワーク部，TUBSTAP のゲームを実現する部分，自己対戦や探索などのアルゴリズムを実現する部分，対戦プログラム等に分けられる．AlphaZero のシステムと比較して同一の部分は，ロールアウトを使用しない MCTS 探索部分，着手決定にニューラルネットワークの確率出力を利用して自己対戦のみで学習を進める部分，ポリシーネットワークとバリューネットワークを統合した部分が主要な部分である．異なると言える部分は TUBSTAP に特有の部分として，駒の行動出力を表現するポリシーネットワークの出力表現にかかわる部分，探索の高速化のための PUCT 式にバイアスを加えた部分等である．なお，AlphaGo のアルゴリズムを彼らは APV-MCTS と呼んでいたが，ここでは A（非同期）の要素がないのでポリシーとバリューの要素がある AlphaZero タイプのアルゴリズムの呼称として PV-MCTS と呼ぶことにする．

### 3.1 アルゴリズム部

ここでは TUBSTAP の一つの駒 $u_i$ ごとの指し手を行動 $a_{i,j}$ とし各局面を表現するノードは $a_{i,j}$ を表現するエッジから次の局面のノードに遷移するものとする．本研究のノード展開においては図 1 のように選択されたユニット $u_i$ ごとに子ノードが展開されユニットの選択情報はノードの縦の接続で表現される．本研究ではユニットの指し手行動を表現するのに必要な（行動する駒のマス）×（移動先マス）×（攻撃先マス）の三つの組の情報のうち，行動する駒のマスの情報をノードの展開に集約することで、ニューラルネットワーク部の出力ニューロン数を削減する．この適用によりノードの展開はどの駒を移動させるかの情報がノードの横方向の展開となる．ニューラルネットワークでの推論は各ユニット $u_i$ ごとに自軍のユニット数の回数行い，選択確率 $P(s, u, a_{i,j})$ は各エッジに対応する．つまり自軍ユニットが n 個ある場合は，ニューラルネットワークによる推論は n 回分行われることになる．

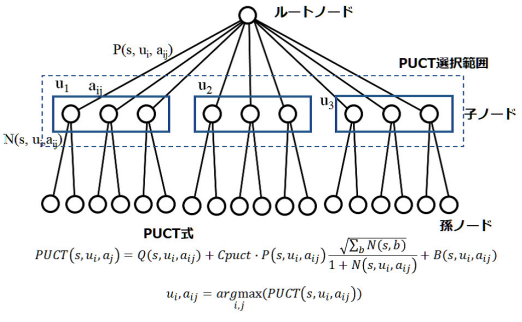



**図 1:** ノード展開の基本概念

基本アルゴリズムにおいては多くの点において AlphaZero と多くの点において同一とした．基本的な点にとしては、探索は深層学習によるポリシーとバリューの二つを使用した MCTS 型探索を行う．MCTS 探索ではニューラルネットワークのポリシーネットワークの出力である確率分布ベクトル $\mathbf{p}$ に基づいた確率探索を行い、最終的に最も探索回数の多いノードを着手とする．MCTS でよく使われるロールアウトは使用していない．自己対戦時にはあらかじめ展開されたノードの再利用を行っている．PUCT 値の選択にユニットの選択も含まれる形となっており，着手可能ユニット数が一個の場合は通常の PUCT 探索と同一のアルゴリズムとなる．探索中のノードの選択はすべての着手可能ユニットの着手可能手のなかから次式で計算される PUCT 値の最も高いノードを探索する．

$$\begin{aligned} PUCT(s,u_i,a_{ij}) =& Q(s,u_i,a_{ij}) \\ &+ Cpuct \cdot P(s,u_i,a_{ij}) \frac{\sqrt{\sum_b N(s,b)}}{1+N(s,u_i,a_{ij})} \\ &+ B(s,u_i,a_{ij}) \end{aligned} \tag{1}$$

$$a_{i,j}, u_{i,j} = \arg\max_{a,u}(PUCT(s,u_i,a_{ij})) \tag{2}$$

ここで $Q(s,u_i,a_{ij})$ はノードの勝利確率，$Cpuct$ は探索補正係数でここでは 0.8 の固定値、$P(s,u_i,a_{ij})$ は各ノードの選択確率、$N(s,u_i,a_{ij})$ はノードの訪問回数、$B(s,u_i,a_{ij})$ は探索の高速化のための本研究独自のバイアス項であり次のようになっている．

$$B(s,u_i,a_{ij}) = \frac{b_{attack}}{N(s,u_i,a_{ij})} \tag{3}$$

$b_{attack}$ はノードが攻撃ノードの場合，3.7 として攻撃ノードが優先的に選択されやすい設定としている．このようにアルゴリズムが PUCT 値による選択を行うと行動と同時にユニットも選択されるようになっている．探索木の根ノードにのみ広い範囲の探索のための Dirichlet ノイズを付加している．また、自己対戦時の着手選択時には、温度減衰付きボルツマン確率選択が行われる．ノード設計はニューラルネットの出力段の負担を軽減するために、各駒ごとに推論を行いすべての行動を統合する形にまとめた．各ノードに格納される情報は，訪問回数 $N$，選択確率 $P$，総 value 値 $W$，平均 value 値 $Q$, そのノードの子ノードへのリンク，等になる．

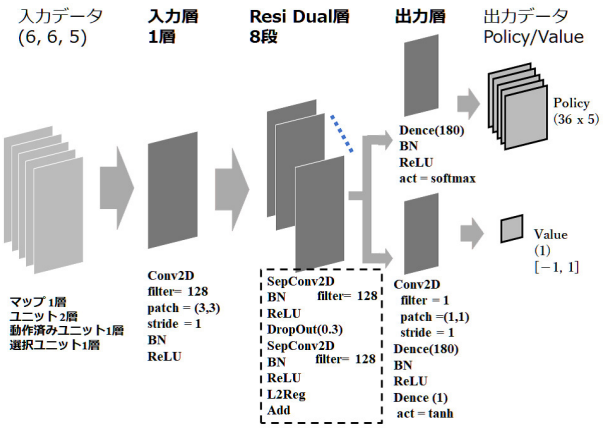



**図 2:** 設計したニューラルネットワークのブロック図

探索が葉ノードに到達してのち，訪問回数と value 値を $N(s_t,a_t) = N(s_t,a_t)+1$, $W(s_t,a_t) = W(s_t,a_t)+v$, $Q(s_t,a_t) = \frac{W(s_t,a_t)}{N(s_t,a_t)}$ によって更新していく．探索の最終段において最も訪問回数の多かったノードを最終的な着手とする．

### 3.2 ニューラルネットワーク部

ニューラルネットワーク部の設計は AlphaZero で使用された ResiDual [7] タイプの設計および Wide Residual Network [8] を参考にしているが，今回のゲームに適用するために細かい部分で多くの変更を行った（図 2）．独自の変更として Dropout 部を追加して収束効率を高めている．ターン制戦略ゲームに深層学習を適用しやすくするためポリシーネットワークの出力段のニューロン削減策として，駒の移動先（36 か所）と攻撃先（四方向）をデコードした形の出力（$36 \times 5$）とした（図 3）．この手法によりユニットの行動を表現する（行動する駒のマス）×（移動先マス）×（攻撃先マス）のデータを本来なら $36 \times 36 \times 5 = 6480$ の出力ニューロンが必要になる所を $36 \times 5 = 180$ に削減した．この手法は歩兵等に適用できるが自走砲に適用する際は攻撃先の追加が必要になる．なお，リカレントネットワークを利用して出力ニューロンの削減を図る手法もある [9], [10]．しかしながら，この手法では時分割で出力情報を出力するためバリューネットワークの情報を統合するのが難しくなる．

ニューラルネットワークへの入力データは 0 から 1 までにそれぞれ正規化された形で，マップ状の進入禁止と草原かをマス目状にした情報が一層，ユニットの位置情報と HP の値をマス目状にデコードしたもの二層，動作済みユニットの位置をマス目にした情報が一層，動作ユニットの位置

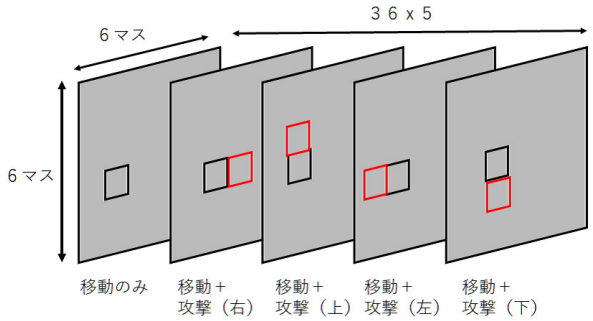



**図 3:** ポリシーネットワーク出力のデータ表現

をマス上においた情報が一層の計 5 層を 6 マス × 6 マス × 5 層としてまとめられて入力される．ニューラルネットワークの入力層は Conv2D が一層あり，続いて ResiDual Network が 8 段積層され，ここからポリシーヘッドとバリューヘッドに分岐して二つの出力へと接続される．ポリシー出力は行動予測の確率分布の形であり，バリュー出力は [-1, 1] の範囲の試合の勝敗予測を表すスカラー出力である．

### 3.3 ニューラルネットワークの学習

自己対戦によって生成されたゲームデータをリプレイメモリ [11] に蓄積して決められたタイミングでニューラルネットワークの学習を行う．リプレイメモリに格納する際にはデータの鏡映と回転を行いデータ量を 8 倍にする Data Augmentation を行っている．学習の際には学習率のスケジューリングを行い，オプティマイザーは SGD+momentum(0.9) を使用し，バッチサイズは 128 とした．損失関数は次式のように設定した．

$$loss = (z - v)^2 + D_{KL}(\pi \mid \mathbf{p}) + L2Reg \qquad (4)$$

ここで，$z$ は葉ノードにおける報酬値，$v$ はバリューの出力，$D_{KL}(\pi \mid p)$ はカルバック・ライブラー情報量，$L2Reg$ は L2 正則化項である．第一項と第二項の重みは同一とし L2 正則化定数は $1e^{-4}$ とした．自己対戦で学習するニューラルネットワークのトーナメントによる交代は行っていない．

## 4. 数値実験

ニューラルネットワークの学習の進展について損失データから確認する．さらにベンチマーク問題を使って提案するシステムの基本性能を確認し，最終的に対戦実験により性能を確認する．

### 4.1 自己対戦学習に使用する初期盤面設定

自己対戦で使用した初期盤面設定は 6 マス ×6 マスのマップで以下のようになっている．

- □ 歩兵が RED/BLUE ともに 1 個で HP と位置はすべてランダム．
- □ 歩兵が RED/BLUE ともに 2 個で HP と位置はすべてランダム．
- □ 歩兵が 1 個と 2 個の組み合わせでどちらが 2 個になるかはランダム．HP と位置はすべてランダム．
- □ 以前の研究 [10] で使用した 5 マス × 5 マスのマップで 2 個同士の対戦で使用した手筋マップを 50 種類程度．

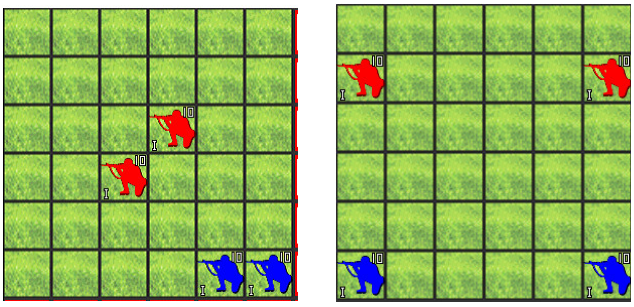
ここでいう手筋マップとは図 4 のような特定の重要な局面で優先して学習すべき手筋のある場合を特に取り出したマップである．図 4a は一見ではわかりにくいけれども RED に必勝手順が存在する局面であるがこのたびの学習で PV-MCTS がこの必勝局面を徐々に学習していったものである．この必勝局面で勝てるかどうかによって学習の進み具合を確認できる．学習局面に 1 個対 1 個などの設定があるのは、PV-MCTS の探索手法は特定の局面について少しづつ方策と価値を覚えていって有利な局面を見出していくような方式であるため，探索木の下の局面から覚えていくのがより早く学習を進めるからである．この場合は最終的な目的の局面が 2 個対 2 個の局面であったとしても，終局までには必ず 1 個対 2 個や 1 個対 1 個の局面を通過するためこのような局面の学習も必ず必要になるので，あらかじめ同時に学習を進めることで学習を加速させることが狙いである．なお，PV-MCTS での一回あたりのシミュレーション回数は 500 回でありこの設定は以下すべての実験で共通である．

### 4.2 自己対戦に要した時間

ニューラルネットワークの初期状態から学習を始めて 3 週間程度学習を継続した．しかし，毎日使用する手筋マップを入れ替えてデータを保存しながらなので、完全に連続してはいない．

### 4.3 ニューラルネットワークの学習の様子

ニューラルネットワークの学習の推移を損失データで確認する．通常の学習に使用した設定 ($b_{attack} = 3.7$, dropout あり, rate $= 0.3$) での損失の推移は図 5 のようになった．



**(a)** RED に必勝手順があるマップ　**(b)** 各攻撃が必要なマップ

**図 4:** 手筋マップの例

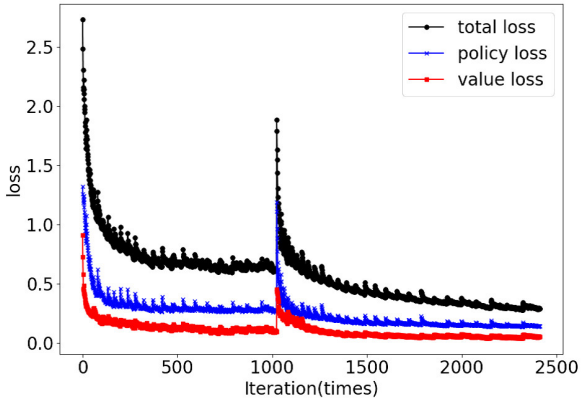



**図 5:** 損失の推移 ($b_{attack} = 3.7$，ドロップアウトあり)

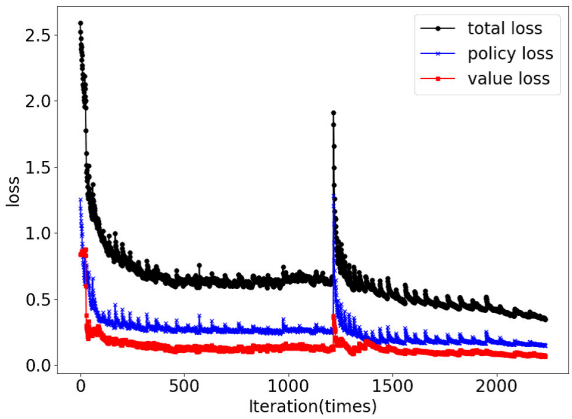



**図 6:** 損失の推移 ($b_{attack} = 0$)

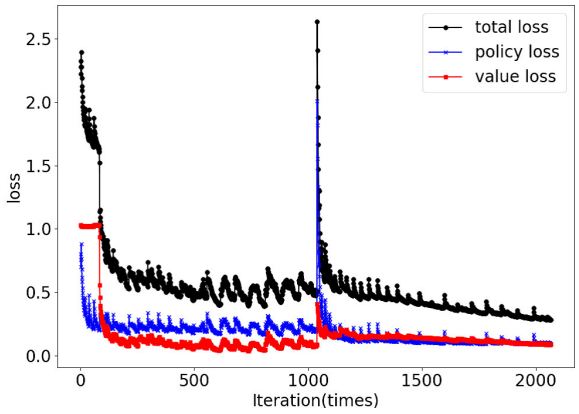



**図 7:** 損失の推移 (Dropout なしの場合)

比較のためにバイアス係数の $b_{attack}$ を使用しなかった場合 (図 6 ) とドロップアウトを使用しなかった場合の損失 (図 7 ) のグラフを示す．

図 6 の $b_{attack} = 0$ の設定では一見，収束が早く行われているようにみえるが，実際の駒の動きは攻撃が適切に行うことができない場合が多く，学習が進まなくなってしまうことから，早いうちから過学習が発生していると推定される．図 7 のドロップアウトなしの設定でも比較的損失が低下するのが早いものの，学習が停滞することが多くドロップアウトを設定したほうが安定して収束する可能性が高い．

**表 1:** ベンチマーク問題　pinch01 での検証結果
**Table 1** Results at map pinch01.

| 検証アルゴリズム | 成功 | 失敗 |
|---|---|---|
| pMC | 2 | 8 |
| MCTS | 10 | 0 |
| PV-MCTS | 10 | 0 |

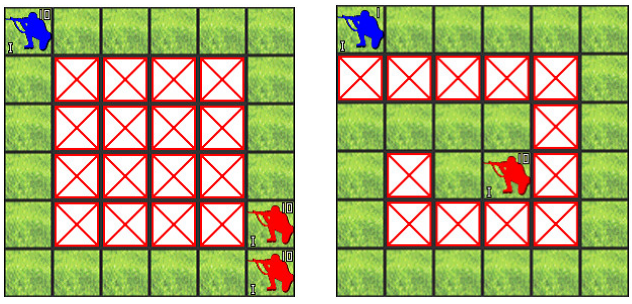
### 4.4　ベンチマーク問題による基本性能の確認

作成した AI の基本性能を評価するため以前作成されたベンチマーク問題集 [12] をベースにする図 8 のような挟み撃ち問題 (pinch01) と経路探索問題 (pathfind01) を用意し，成功するかどうか検証した．成功の基準は挟み撃ちマップであれば規定のターン数内で BLUE を挟み撃ちして相手を全滅させられたら成功，経路探索マップは BLUE までにたどり着いて相手を全滅させられたら成功である．

ここでは対戦相手として用意した原始モンテカルロ法（pMC）と MCTS タイプのものも同時に評価した．pMC と MCTS はできる限り強くするため一手あたりのシミュレーション回数を多めに設定し，結果として動作時間は最大 30 秒程度と長めなっている．シミュレーション回数は pMC が枝あたりに 100 回、MCTS がトータルで 2000 回とし、双方ともロールアウトを試合終了まで行って勝敗を判定する．これは以下の評価でも共通の設定である．

PV-MCTS の場合は二つのマップを合わせて 50 回程度の試合で学習させた．二つのマップとも BLUE 側のアルゴリズムは MCTS を使用した．これらのマップに限って学習した場合は学習時間は一時間弱であった．以上の結果をまとめたものが表 1 と表 2 である．

結果としては pMC は原始モンテカルロ法であり木探索が入っていないせいか失敗があったが、経路探索問題では 8 回も成功していて性能は高い．また MCTS はすべての試行で成功した．これらの問題に対しては性能が高いと言える．これはどちらのアルゴリズムに対してもそれなりに長い計算時間を与えて動作させているためと思われる．



**(a)** pinch01　　**(b)** pathfind01

**図 8:** アルゴリズム検証用マップ

表 2: ベンチマーク問題　pathfind01 での検証結果
**Table 2** Results at map pathfind01.

| 検証アルゴリズム | 成功 | 失敗 |
|---|---|---|
| pMC | 8 | 2 |
| MCTS | 10 | 0 |
| PV-MCTS | 10 | 0 |

表 3: PV-MCTS の対戦マップ 01 での対戦結果
**Table 3** Match results at map01.

| 対戦相手 | 勝ち | 引分け | 負け | 勝率 (%) |
|---|---|---|---|---|
| pMC | 88 | 4 | 8 | 88 |
| MCTS | 85 | 3 | 12 | 85 |

PV-MCTS は学習する手間があるものの，探索自体は高速で行うことができすべて成功することができているので基本的な動きは問題ないと言える．

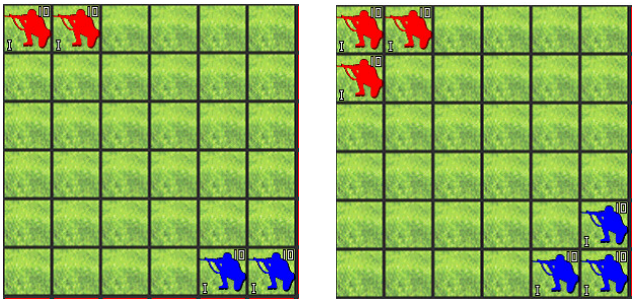
### 4.5 対戦による性能評価

自己対戦によって学習した PV-MCTS の性能評価のため，対戦実験を行った．

対戦に使用したマップは図 9a である．このマップでは歩兵ユニットが二個づつの配置となっており，最大ターン数は 16 ターンまでと設定し勝利条件は相手を全滅した時のみとし全滅がおきずに最大ターンに到達した場合はすべて引分けとした．先手の設定は 50 試合を PV-MCTS とし，後半 50 試合を後手として先手後手の有利不利条件がなくなるようにした．対戦結果が表 3 である．

対戦マップが単純で障害物がない設計になっているのは逃げ場や隠れ場所がないことでアルゴリズムの純粋な読みの能力を測定するためである．また，できる限り移動可能マスを増やすことで 1 ターン目から計算爆発が起きやすくなりアルゴリズムの限界性能を見やすくする設定にもなっている．

また，学習したニューラルネットワークの汎化性能を検証し，未知局面に対する対応能力を測定するため，学習していない設定での対戦を行ったのが対戦マップ 02 における対戦である．マップ 02 の条件は歩兵ユニットが 3 個づつの条件であるが，自己対戦による学習ではこの設定は一切入っていない．先手後手は試合の半分で入れ替えている．対戦結果が表 4 である．学習を行っていない設定の対戦マップにもかかわらず勝率が 70% と 65% となっており PV-MCTS は良い戦いをしたといえる．このような動作ができているのは初期条件が 3 駒同士でも戦闘が進むにつれて 2 駒同士や 1 駒同士の状態になる瞬間があり，このような時に 2 駒マップで学習した経験が探索を助けているものと思われる．



(a) 対戦マップ 01　　(b) 対戦マップ 02

図 9: 対戦用マップ

表 4: PV-MCTS の対戦マップ 02 での対戦結果
**Table 4** Match results at map02.

| 対戦相手 | 勝ち | 引分け | 負け | 勝率 (%) |
|---|---|---|---|---|
| pMC | 14 | 0 | 6 | 70 |
| MCTS | 13 | 3 | 4 | 65 |

表 5: ポリシーのみでのマップ 01 での対戦結果
**Table 5** Only policy match results at map01.

| 対戦相手 | 勝ち | 引分け | 負け | 勝率 (%) |
|---|---|---|---|---|
| pMC | 86 | 4 | 10 | 86 |
| MCTS | 85 | 7 | 8 | 85 |

AlphaGo の当初の論文では，ポリシーネットワークやバリューネットワーク、ロールアウトの組合せや単体での性能評価が行われていたが，ここではロールアウトは使用していないのでバリューネットワークを使用しないでポリシーネットワーク単体での評価を試みる．バリューネットワークの効果を失くして，ポリシーネットワークのみでの動作として，葉ノードにおける $v$ をターミナル値 $z$（勝利:1, 引分け:0，敗北: -1）におきかえた場合のポリシーネットワークのみのバージョンの対戦マップ 01 での対戦成績が表 5 である．勝率的には差があまりないが終盤には MCTS 特有のゆるみの挙動がみうけられた．PV-MCTS の探索に要する時間は一手あたり１０秒から３０秒程度であったが，3 個のユニットのマップでは 1 分程度にのびるようになった．pMC と MCTS は 20 秒から３０秒程度である．

## 5. 考察

挟み撃ち問題や経路探索問題においては，広い範囲を探索しなければならないため，旧来のアルゴリズムでは深い探索が必要であるが PV-MCTS では学習をしている状態であれば短時間の探索で正しい答えを出すことができる．また，ここで用意した対戦用のマップのように広い探索が必要な場合，旧来のアルゴリズムでは短いターンですらなかなか正しい探索を行えなくなることがあるが，PV-MCTS では学習した効果を発揮するなら正しい行動選択を行うことができ結果として対戦成績は良好となる．そして，２駒同士の対戦結果が 3 駒同士の対戦に役立ったように学習した結果はある程度推論がおよぶ範囲であれば応用が可能

であるということができる．また通常のニューラルネットワークによる行動決定ではポリシーネットワークのみでも行うことができるがさらにバリューネットワークを組み合わせることで他の利点も得ることができるようになる．

## 6. まとめ

AlphaZero によって広く紹介されたディープニューラルネットワークを使用した自己対戦による学習アルゴリズムの手法をターン制戦略ゲームに適用した場合について検証し報告した．ターン制戦略ゲームに PV-MCTS の手法を適用することはデータの表現方法やニューラルネットワークの設計において変更を要する部分があり直接そのまま適用できないが，適切な変更を行って活用できる形を示した．測定された結果は従来のアルゴリズムだけでは動作するのが難しかったり性能が出にくいマップ上でも高い性能を示すことが判明した．

### 参考文献


[1] Silver, D., et al.: Mastering the game of Go with deep neural networks and tree search. Nature 529(7587), pp. 484–489 (2016)

[2] Silver, D., et al.: Mastering the game of Go without human knowledge. Nature 550(7674), pp. 354–359 (2017)

[3] Silver, D., et al.: A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. Science 362(6419), pp. 1140–1144 (2018)

[4] 村山公志朗ら，学術研究用プラットフォームとしての大戦略系ゲームのルール提案，ゲームプログラミングワークショップ 2013 論文集 GPW2013，pp. 146-153 (2013)

[5] 佐藤直之，藤木翼，池田心，ターン制戦略ゲームにおける局面評価値構成のための局面分割および単純化ゲームのオフライン木探索，ゲームプログラミングワークショップ 2015 論文集，pp. 61-68 (2015)

[6] 加藤千裕，三輪誠，鶴岡慶雅，近山隆：ターン制ストラテジーゲームにおける戦術決定のための UCT 探索とその効率化，ゲームプログラミングワークショップ 2013 論文集，pp. 138-145 (2013)

[7] He, K., et al.: Deep Residual Learning for Image Recognition, The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770-778 (2016)

[8] Zagoruyko, S., Komodakis, N.: Wide Residual Networks, CoPR, abs/1605.07146, (2016)

[9] 木村富宏：ターン制戦略ゲームへの深層学習の適用，情報処理学会 第 41 回ゲーム情報学 (GI) 研究発表会，(2019)

[10] Kimura, T., Ikeda, K.: Designing policy network with deep learning in turn-based strategy games, 16th Advances in Computer Games Conference (ACG), (2019)

[11] Mnih, V., et al.: Human-level control through deep reinforcement learning, Nature 518(7540), pp. 529–533 (2015)

[12] 木村富宏，池田心：ターン制戦略ゲームにおけるベンチマークマップの提案，GPW-16, pp. 36-43, (2016)