Vol.2013-GI-29 No.3
2013/3/4

GGPにおける強さとバランスを両立したモンテカルロ木探索の方策の学習

藤田 康博[1]　浦 晃[2]　三輪 誠[3]　鶴岡 慶雅[2]　近山 隆[2]

概要：General Game Playing (GGP) は形式的に表現されたゲームルールを解釈することで、幅広い未知のゲームをうまくプレイできるプログラムを実現する試みである。GGP においてはモンテカルロ木探索が近年成功を収めているが、そのシミュレーション方策をどのように学習すべきかには未知の部分が多い．本研究では GGP の枠組みにおいて，方策の「強さ」とバランスという性質に関し着目し，既存の学習手法を組み合わせることで「強さ」とバランスを両立を目指す新たな学習手法を提案する．新たな学習手法は一部のゲームにおいて既存手法よりよい性能を示した．

Learning Strong And Balanced Policies for Monte-Carlo Tree Search in GGP

Fujita Yasuhiro[1]　Ura Akira[2]　Miwa Makoto[3]　Tsuruoka Yoshimasa[2]　Chikayama Takashi[2]

Abstract: General Game Playing (GGP) is an approach to building a program which can play various games well by interpreting the formal descriptions of their rules. Monte-Carlo tree search has recently been successfully applied in GGP, but it is still largely unknown how the simulation policies should be learned. In this paper, we focus on the strength and balance of policies and propose a new learning method for acquiring policies that are "strong" and balanced. The experimental results show that the new method performs better than existing methods in some games.

1. はじめに

過去のゲーム AI の研究では 1 つの特定のゲームを人間の世界チャンピオンのようにうまくプレイすることができるプログラムを開発することに主な関心が置かれてきた．しかしそのようなプログラムは事前に得られているそのゲーム固有の知識に大きく依存し，そこで使用される技術もそのゲームに高度に特化したものであるため，他のゲームやゲーム以外の問題を扱うことができない．さらに，そのようなプログラムでは，ゲームの分析やアルゴリズムの設計のほとんどの部分は事前に専門家や開発者により行われるため，プログラムが解く問題はほんの一部にすぎない．

現実世界の様々な問題に対し好ましい解を与え，人間の意思決定を補助できるような AI プログラムを実現するためには，特定のゲームをうまくプレイするための技術の研究だけでなく，未知のゲームを分析し適切に対処できるプログラムや，幅広いゲームに有効な技術に関する研究が必要であると考えられる．そのため注目すべき近年の研究の潮流として，General Game Playing とモンテカルロ木探索を挙げることができる．

2005 年の Association for the Advancement of Artificial Intelligence（AAAI）カンファレンスにおけるコンペティションの開催を皮切りに注目を集めている General Game Playing（GGP）[10] は，形式的に記述されたゲームルールを解釈することにより，人間が介入することなく幅広い

[1] 東京大学工学部電子情報工学科
Department of Infomatics and Communication Engineering, The University of Tokyo
[2] 東京大学大学院工学系研究科
Graduate School of Engineering, The University of Tokyo
[3] マンチェスター大学コンピュータ科学科
School of Computer Science, The University of Manchester

ゲームをうまくプレイできるプログラムを実現する試みである．GGP プレイヤは特定のゲームのみをプレイするプログラムと異なり，未知のゲームに対し，知識表現，探索，推論，学習といった AI 技術を組み合わせることにより，プログラム自らゲームを分析しプレイすることを要求される．GGP は幅広いゲームに対するゲームプレイング技術の有効性の評価を与えるテストベッドとしての役割を果たすとともに，ゲームとしてモデル化することができる多様な問題に対処可能なシステムとして，現実世界の問題解決への応用可能性という実用的価値も有している．

囲碁における成功をきっかけに盛んに研究されるようになったモンテカルロ木探索は，確率的シミュレーションを繰り返して局面を評価しながらゲーム木を探索する探索手法である．囲碁以外にも様々なゲームで有効性が示されており，特にゲーム固有の知識を必要とせず，シミュレーション回数を増やすことで幅広いゲームに適応できるという性質から，GGP への適用が成功を収めている [6]．

本研究の目的は幅広いゲームに有効な学習手法を提案することである．中でもモンテカルロ木探索の性能に影響を与えるシミュレーション方策の性質に着目し，よい性質を備えた方策の学習を目指す．

GGP の枠組みを用いることにより，幅広いゲームにおけるプレイヤの性能を評価することが可能になる．過去の GGP 研究においてはプレイヤの思考時間の制限を厳しく設定することが多かったが，本研究では学習の時間効率よりも学習される方策の性質に注目した評価を行うため，必要だと思われる学習・プレイ時間を確保する．

2. 関連研究

2.1 General Game Playing

General Game Playing（GGP）は形式的に表現されたゲームルールを解釈することにより，人間が介入することなく幅広いゲームをうまくプレイできるプログラムを実現する試みである．このようなアイディアは 1968 年の研究 [15] にまで遡ることができるが，2005 年の AAAI における国際コンペティションの開催を皮切りに，AI 研究における大きな挑戦として広い関心を集めている [21]．

2.1.1 Game Description Language

GGP においてゲームルールは Game Description Language（GDL）[13] という言語によって記述される．GDL は Datalog というクエリ言語を元にした言語であり，ゲームの局面を真である命題の集合として定義する．表 1 のような専用の関係述語が用意されている．

以下では，Tic-Tac-Toe と呼ばれるゲームを例に，実際のゲームルールがどのように GDL により記述されるかを示す．Tic-Tac-Toe の盤面は 3×3 の網目からなり，各セルは空であるかもしくは x または o がマークされている．x, o それぞれに対応する 2 人のプレイヤ xplayer，oplayer が交互に空のマスをマークし，先に 3 つ直線上にマークしたプレイヤが勝利者となる．

ゲームに登場するプレイヤが xplayer と oplayer の 2 人であることは次のように記述される．

```
(role xplayer)
(role oplayer)
```

先手プレイヤが xplayer であるということ及び初期状態では全てのセルが空であるということは，プレイヤの手番を表す述語 control 及びセルの状態を表す述語 cell を導入することによりそれぞれ次のように記述される．

```
(init (control xplayer))
(init (cell 1 1 b)) (init (cell 1 2 b)) (init (cell 1 3 b))
(init (cell 2 1 b)) (init (cell 2 2 b)) (init (cell 2 3 b))
(init (cell 3 1 b)) (init (cell 3 2 b)) (init (cell 3 3 b))
```

プレイヤの手番が交互に入れ替わることは，現在 xplayer の手番なら次は oplayer の手番，現在 oplayer の手番なら次は xplayer の手番ということであり，これらはホーン節の形で次のように記述される．

```
(<= (next (control xplayer))
    (true (control oplayer)))
(<= (next (control oplayer))
    (true (control xplayer)))
```

プレイヤは自分の手番では空のマスにマークすることができ，自分の手番でなければ何もできない．これは次のように記述される．GDL では前に?をつけることで変数を表し，ここではセル (?x, ?y) にマークするという手を (mark ?x ?y)，何もしないという手を noop と表している．

```
(<= (legal ?player (mark ?x ?y))
    (true (cell ?x ?y b)
    (true (control ?player)))
(<= (legal xplayer noop)
    (true (control oplayer)))
(<= (legal oplayer noop)
    (true (control xplayer)))
```

表 1: GDL に用意されている関係述語

(role P)	ゲームにプレイヤ P が登場する
(init X)	初期状態において命題 X が真である
(next X)	次状態において命題 X が真である
(true X)	現在の状態において命題 X が真である
(legal A)	手 A が合法手である
(does P A)	プレイヤ P が手 A を選ぶ
(goal P R)	プレイヤ P が報酬 R を得る
terminal	終端状態である

プレイヤ xplayer がセル (?x, ?y) にマークするという手を選んだ場合，次状態ではセル (?x, ?y) は x でマークされていなければならない．このようなプレイヤが選ぶ手と次状態の関係は次のように記述される．プレイヤ oplayer についても同様に記述される．

```
(<= (next (cell ?x ?y x))
    (does xplayer (mark ?x ?y)))
```

Tic-Tac-Toe は一方のプレイヤが直線上に 3 つ自分のマークを並べるか，あるいは空のマスが無くなることにより終了する．このことは，プレイヤ?player が直線上に 3 つ自分のマークを並べていることを (line ?player)，空のマスが無いことを open と表すならば，次のように記述される．ここで導入した述語 line 及び open がどのような条件で真となるかは別に記述しなければならないが，ここでは省略する．

```
(<= terminal
    (role ?player)
    (line ?player))
(<= terminal
    (not open))
```

GGP ではゲームの勝敗はプレイヤに与えられる報酬として定義される．ゲームが終了した場合，各プレイヤはゲームの状態に応じて報酬を得る．プレイヤの報酬は 0 以上 100 未満の整数でなければならない．直線上に 3 つ自分のマークを並べたプレイヤが報酬 100 を得るとする場合，それは次のように記述される．

```
(<= (goal ?player 100)
    (line ?player))
```

GDL による実際のゲームのルールの記述は以上のようなものになる．チェスのような複雑なゲームのルールも，記述量は増えるものの，同様に記述することができる．

2.1.2 扱うことができるゲーム

GGP システムが扱うことができるゲームの範囲は GDL の記述力によって規定され，以下の条件を満たす必要がある [13].

- 状態数が有限である
- 決定的である（確率的要素を含まない）
- 完全情報である
- プレイヤの数がゲームの開始から終了まで変化しない

2.2 モンテカルロ木探索

モンテカルロ木探索は確率的なシミュレーションを利用したゲーム木の最良優先探索であり，シミュレーションを繰り返しながらメモリ上に探索木を作ることにより最善手を探索する．

2.2.1 Upper Confidence bounds applied to Trees

モンテカルロ木探索のバリエーションうち最もよく使用されるアルゴリズムに Upper Confidence bounds applied to Trees（UCT）[12] がある．UCT は選択ステップにおいて次の値が最大となるような子ノード j を選択する．

$$UCT = \overline{X_j} + 2C_p\sqrt{\frac{2\ln n}{n_j}} \qquad (1)$$

ただし n は親ノードの訪問回数，n_j は子ノード j の訪問回数，$\overline{X_j}$ は子ノード j を経由した場合の平均報酬，$C_p > 0$ は定数である．$n_j = 0$ の場合 $UCT = \infty$ となるため，全ての子ノードは少なくとも 1 度は訪問されることが保証される．右辺の第 1 項は平均報酬が高い子ノードほど選ばれやすいという傾向（exploitation）を，第 2 項は訪問回数が少ない子ノードほど選ばれやすいという傾向（exploration）を生み出している．C_p はこの 2 つの傾向のバランスを取るように設定される．

UCT に関しては，シミュレーション回数を増やすと平均報酬がミニマックス値（各プレイヤが常に最善手を選んだ場合の報酬）に収束することが示されている [12].

2.3 モンテカルロ木探索のシミュレーション方策

シミュレーションにおけるプレイヤの手の選び方，すなわち局面ごとに与えられる手の選択の確率分布をシミュレーション方策，あるいは単に方策と呼ぶ．シミュレーションを繰り返すことにより求められる平均報酬はシミュレーション方策に依存する．一様分布に従い手を選ぶのが最も単純な方策であるが，その場合の評価の精度は高くない．多くの場合，よい手を高い確率で，悪い手を低い確率で選ぶような方策を用いた方が正確な評価をもたらす．したがって方策の差はプレイヤの性能に大きな影響をもたらし [11]，方策を改善することによりプレイヤの性能を向上させることができる．

どのような方策が強いプレイングを可能にするかという点に関してこれまでに明らかになっている知見の 1 つに，「強い」方策とバランスのとれた方策の関係がある [11], [19].

方策を，次のようなソフトマックス関数の形で，状態 s において手 a を選択する確率として定義する．

$$\pi_\theta(s,a) = \frac{e^{\phi(s,a)^T\theta}}{\sum_b e^{\phi(s,b)^T\theta}} \qquad (2)$$

ただし $\phi(s,a)$ は状態 s と手 a に対する特徴ベクトル，θ は個々の特徴に対する重みベクトルである．これにより各状態に対し手の確率分布が与えられる．

状態 s のミニマックス値を $V^*(s)$ とし，プレイヤが手を選ぶことにより状態が s_t から s_{t+1} に遷移した場合に生じる状態のミニマックス値の変化を

$$\delta_t = V^*(s_{t+1}) - V^*(s_t) \tag{3}$$

と表し，重みベクトル θ に対して「強さ」（strength）$J(\theta)$ とインバランス（full-imbalance）$B_\infty(\theta)$ を定義する．

$$J(\theta) = \mathbb{E}_\rho[\mathbb{E}_{\pi_\theta}[\delta_t^2|s_t = s]] \tag{4}$$
$$B_\infty(\theta) = \mathbb{E}_\rho[(\mathbb{E}_{\pi_\theta}[z|s_t = s] - V^*(s))^2] \tag{5}$$

ただし $\rho(s)$ は探索において評価する状態 s の確率分布であり，$\mathbb{E}_\rho$ は $\rho(s)$ 上の期待値を，$\mathbb{E}_{\pi_\theta}$ は方策 π_θ を用いたシミュレーションを繰り返した上での期待値を表す．z は s_t から π_θ に従い終局までシミュレーションを行った場合の報酬を表す．$J(\theta)$ を最小化する π_θ を「強い」（strong）方策，$B_\infty(\theta)$ を最小化する π_θ をバランスのとれた（balanced）方策と呼ぶ．

直感的には，「強い」方策はよさそうな手をできるだけ高い確率で選ぶ一方，バランスのとれた方策は，シミュレーションを繰り返すことでその平均報酬への影響が互いに打ち消される限りにおいて，悪そうな手も選ぶ．

バランスのとれた方策をシミュレーションに用いることに関しては，5×5，6×6，9×9 の囲碁において有効性が示されている [11], [19]．一方，将棋において「強い」方策とバランスのとれた方策を UCT プレイヤに用いた結果，前者の勝率が高かったことも報告されている [23]．どのような方策が有効かはゲームによって異なるといえる．

2.3.1 Apprenticeship Learning

「強い」方策を学習する手法として，状態 s_l とそこでの近似的な最善手 a_l の組 (s_l, a_l^*) の集合 L を訓練データに用いる Apprenticeship Learning（AL）が提案されている [19]．これは方策が訓練データと同じ手を選ぶ対数尤度 $\log L(\theta)$ を勾配上昇法により最大化する手法である．

$$L(\theta) = \prod_{l=1}^{L} \pi(s_l, a_l)$$
$$\log L(\theta) = \sum_{l=1}^{L} \log \pi(s_l, a_l)$$
$$\nabla_\theta \log L(\theta) = \sum_{l=1}^{L} \nabla_\theta \log \pi(s_l, a_l) \tag{6}$$

各訓練データ (s_l, a_l) に対して確率的勾配上昇法により次のように重みベクトル θ を更新する．

$$\Delta\theta = \alpha\psi(s_l, a_l) \tag{7}$$

ただし，

$$\psi(s,a) = \nabla_\theta \log \pi(s,a) = \phi(s,a) - \sum_b \pi_\theta(s,b)\phi(s,b) \tag{8}$$

であり，α は学習率である．

2.3.2 Policy Gradient Simulation Balancing

バランスのとれた方策を学習する手法として，ミニマックス値の近似値 $\hat{V}^*(s)$ が知られている状態 s の集合を訓練データに用いる Policy Gradient Simulation Balancing（SB）が提案されている [19]．これはインバランス $B_\infty(\theta)$ を勾配降下法により最小化する手法である．

$$b(s) = V^*(s) - \mathbb{E}_{\pi_\theta}[z|s]$$
$$g(s) = \nabla_\theta \mathbb{E}_{\pi_\theta}[z|s]$$
$$B_\infty(\theta) = \mathbb{E}_\rho[b(s)^2]$$
$$\nabla_\theta B_\infty(\theta) = -2\mathbb{E}_\rho[b(s)g(s)] \tag{9}$$

偏り $b(s)$ は平均報酬をミニマックス値に一致させるためにどれだけ増減させるべきかを意味し，方策勾配 $g(s)$ は平均報酬を増減するために θ をどのように更新すべきかを意味する．各訓練データ s に対し，$b(s)$，$g(s)$ をそれぞれ M,N 回のシミュレーションにより近似し，確率的勾配降下法により重みベクトル θ を更新する．その擬似コードを Algorithm 1 に示す．α は学習率である．

Algorithm 1 Policy Gradient Simulation Balancing[19]

```
θ ← 0
for all s_1 ∈ training data do
    V ← 0
    for i = 1 to M do
        simulate (s_1, a_1, ··· , s_T, a_T; z) using π_θ
        V ← V + z/M
    end for
    g ← 0
    for j = 1 to N do
        simulate (s_1, a_1, ··· , s_T, a_T; z) using π_θ
        g ← g + z/(NT) Σ_{t=1}^{T} φ(s_t, a_t)
    end for
    θ ← θ + α(V̂*(s_1) − V)g
end for
```

2.4 GGP プレイヤ

2.4.1 過去のコンペティション

2005 年から 2012 年まで毎年，AAAI のカンファレンスまたは International Joint Conferences on Artificial Intelligence（IJCAI）の中で GGP の国際コンペティションが開催されている．過去のコンペティションで優勝したプレイヤの名前を表 2 に示す．

2007 年，2008 年及び 2012 年に優勝した CadiaPlayer[8]

は初めて GGP においてモンテカルロ木探索を用いたプレイヤである [6]．2009 年及び 2010 年に優勝した Ary[14] もモンテカルロ木探索を用いており，また 2011 年に優勝した TurboTurtle も詳細は不明であるがシミュレーションを用いたプレイヤである [2].

モンテカルロ木探索は，ゲームルールさえ明らかであればゲーム固有の知識を持たずに開始することができ，シミュレーションの数を増やすことで幅広いゲームに適応できるという柔軟性を持つため，GGP に適した探索手法であり，その有効性がコンペティションにおける実績に現れていると言える．

2.4.2 GGP におけるシミュレーション方策の獲得

GGP においてシミュレーション方策を獲得するための様々な手法が提案されている．Move-Average Sampling Technique（MAST）[9] は，ある局面において良い手はおそらく似たような別の局面においても良い，というヒューリスティックに基づき，モンテカルロ木探索の最中に局面から独立した手ごとの平均報酬を保持しつつ，それによってシミュレーションにおいて良い手を高い確率で選ぶようにバイアスをかける手法であり，幅広いゲームにおいて有効性が示されている．これを発展させたものとして，手と状態命題の組ごとにこれを行う Predicate-Average Sampling Technique（PAST）や，直近に選ばれた手の連続ごとにこれを行う手法 [20] も提案されている．

さらに TD 学習によって，各状態命題の価値 [17], [18] やゲームルールのパターンマッチングにより認識された盤面における駒の種類または位置の価値 [9] を学習し，シミュレーション方策として用いることが提案されている．

3. 提案手法

本研究では，方策の「強さ」を最適化する Apprenticeship Learning（AL）とバランスを最適化する Policy Gradient Simulation Balancing（SB）の 2 つの方策学習手法を踏まえ，「強さ」とバランスを同時に学習するための AL と SB の結合手法（AL+SB）を提案する．これは近似的な最善手 $\hat{A}^*(s)$ 及びミニマックス値の近似値 $\hat{V}^*(s)$ が知られている状態 s の集合 L を訓練データに用いて，訓練データに対する対数尤度 $\log L(\theta)$ とインバランス $B_\infty(\theta)$ 同時に最適化する．そのための目的関数を

$$O(\theta) = \log L(\theta) - B_\infty(\theta) \tag{10}$$

と定義し，これを勾配上昇法により最大化する．

$$\begin{aligned}\nabla_\theta O(\theta) &= \nabla_\theta \log L(\theta) - \nabla_\theta B_\infty(\theta) \\ &= \sum_{s\in L} \psi(s, \hat{A}^*(s)) + 2\mathbb{E}_\rho[b(s)g(s)]\end{aligned} \tag{11}$$

擬似コードを Algorithm 2 に示す．α 及び β は学習率であり，「強さ」とバランスの間の調和を保つように定める．

Algorithm 2 結合手法（AL+SB）

```
θ ← 0
for all s_1 ∈ training data do
    V ← 0
    for i = 1 to M do
        simulate (s_1, a_1, ··· , s_T, a_T; z) using π_θ
        V ← V + z/M
    end for
    g ← 0
    for j = 1 to N do
        simulate (s_1, a_1, ··· , s_T, a_T; z) using π_θ
        g ← g + z/(NT) Σ_{t=1}^{T} φ(s_t, a_t)
    end for
    θ ← θ + α(V̂*(s_1) − V)g + βψ(s_1, Â*(s_1))
end for
```

この結合手法と AL 及び SB をそれぞれ GGP プレイヤに適用し，学習と対戦実験を行うことにより，複数のゲームにおける「強い」方策とバランスのとれた方策の関係を解析しつつ結合手法の有効性について検証する．

表 2: 過去の GGP コンペティションの優勝プレイヤ [1], [20]

2005	ClunePlayer[7]
2006	FluxPlayer[16]
2007	CadiaPlayer
2008	CadiaPlayer
2009	Ary
2010	Ary
2011	TurboTurtle
2012	CadiaPlayer

4. 評価

AL，SB，結合手法の 3 つを GGP プレイヤに実装し，それらを用いてシミュレーション方策を学習させ，その経過を観察するとともに対戦に用いて有効性を評価した．

4.1 評価設定

GDL を処理しゲームを進行するための推論機構として，GDL を元に動的にコードを生成する gdlcc[22] を用いた．評価のためのゲームには，Dresden GGP Server[4] で提供されている Crisscross，Breakthrough（5×5 に縮小したもの）を利用するとともに，どうぶつしょうぎ [5] を GDL により記述したもの [3] を利用した．どうぶつしょうぎは公式ルールのものに加えて，つかまえた駒が自分ではなく相手の持ち駒になるバリエーション（以下，どうぶつしょうぎ O）を用意した．

ゲームの報酬は GDL により 0 から 100 の範囲の整数として定義されるが，学習アルゴリズム内で用いる際は -1

から 1 の範囲の実数に直した．AL の α，SB の α，結合手法の α 及び β はいずれも 0.05 に設定した．SB と結合手法の M, N はどちらも 100 とした．

各ゲームについて，重複のない 1200 の状態をランダムに生成し，その各状態をルートとして 1 万回の UCT 探索を行うことにより近似的な最善手とミニマックス値を得た上で，そのうち 1000 状態を訓練データ，200 状態をテストデータとして用いた．訓練データ全てについて 1 度だけ重みの更新を行うのを学習の 1 イテレーションとし，100 イテレーションの学習を行った．各イテレーションの直後に，学習中の方策について，テストデータの最善手と同じ手を選ぶ確率の平均（Correct Rate，CR）と，方策を用いて 100 回のシミュレーションを行った場合の平均報酬とテストデータのミニマックス値の平均二乗誤差（Mean Squared Error，MSE）をそれぞれ計算した．

特徴ベクトル $\phi(s,a)$ の各要素は，対応する特徴があれば 1，なければ 0 の 2 値とし，抽出する特徴としては真である命題と手の組み合わせを用いた．例えば「(cell 1 1 x)(mark 1 2)」という特徴はセル (1,1) が x でマークされている場合にセル (1,2) にマークするという状況を意味する．この特徴は単純でゲームの特性を捉えるためには不十分であるように見えるが，4.2 節及び 4.3 節で示すようにこれだけでも方策の「強さ」やバランスを学習し GGP プレイヤを強くすることが可能である．組み合わせる命題の数を増やすと特徴の表現力は高まるが，特徴の数が劇的に増えてしまい，学習した方策を用いたシミュレーションが遅くなってしまう．

各ゲームについて，学習された 3 種類の方策を用いるプレイヤと一様な確率でランダムに手を選ぶ方策（Uniform Random，UR）を用いるプレイヤを互いに対戦させ，それらの性能を評価した．方策が与える確率に従って直接手を選ぶ方法（Direct），方策を用いて各合法手についてそれぞれ 100 回のシミュレーションを行った上で最も平均報酬の高い手を選ぶ単純モンテカルロ（Simple-MC），方策を用いて合法手の数 ×100 の回数の UCT 探索を行った上で最も平均報酬の高い手を選ぶ方法の 3 つの方策の利用法についてそれぞれ先手 500 回，後手 500 回，合計 1000 回の対戦を行った．

4.2 学習経過

Crisscross，Breakthrough，どうぶつしょうぎ，どうぶつしょうぎ O の学習経過をそれぞれ図 1，2，3，4 に示す．

AL について見ると，全てのゲームにおいて CR がほぼ単調に増加しており，最終的には 5 割前後に達している．良い手を高い確率で選ぶように方策が学習されていることがわかる．一方 MSE の変化は単調ではなく，ある程度下がった後に上昇に転じるという傾向が見られ，強い手を高い確率で選ぶことが必ずしも MSE の現象に結びつかないことが読み取れる．これは囲碁や Tic-Tac-Toe においても確認されている現象である [19], [24]．AL は特定の手の選択確率をひたすら上昇させるように学習を行うため，方策が与える手の確率分布が偏りがちになり，シミュレーションを繰り返しても決まった手しか選ばなくなる傾向にあり，これが平均報酬をミニマックス値から遠ざける原因になっている可能性がある．

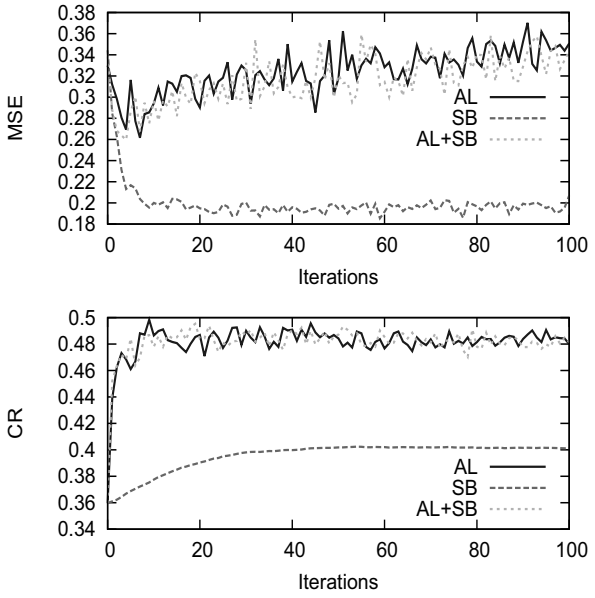

図 1: Crisscross の学習経過

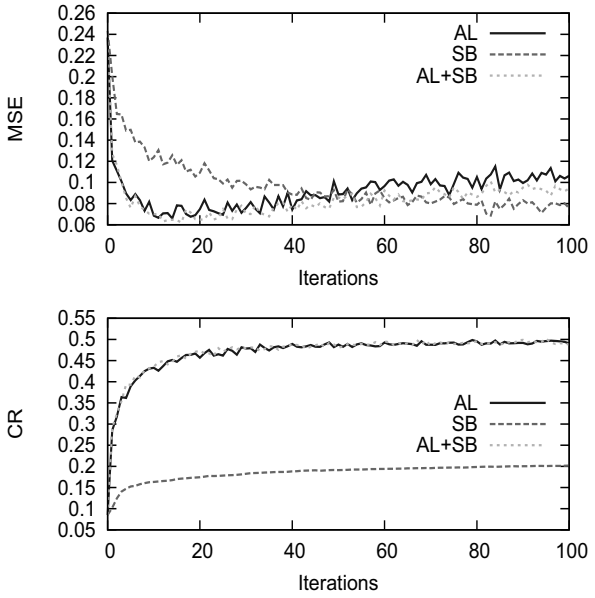

図 2: Breakthrough の学習経過

SB について見ると，AL と同様に全てのゲームにおいて CR が単調に増加しており，SB により学習された方策も良い手を高い確率で選ぶ傾向があることがわかる．SB のアルゴリズムは手の価値に根ざしたものではないのにもかかわらずこのような傾向を持つことは興味深い．MSE は全てのゲームにおいて単調に減少しており，平均報酬をミニ

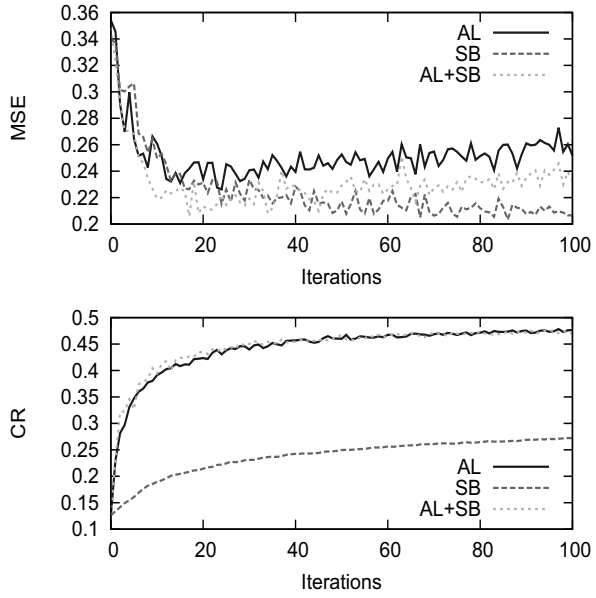

図 3: どうぶつしょうぎの学習経過

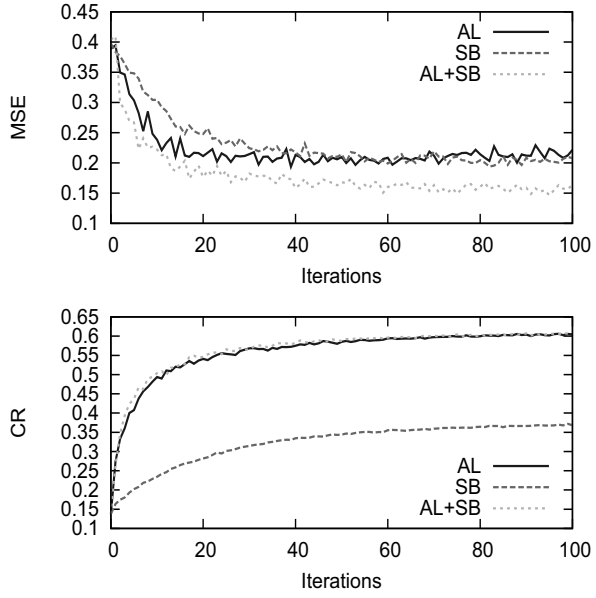

図 4: どうぶつしょうぎ O の学習経過

マックス値に近づけるように方策が学習されていることがわかる．

AL と SB を比較すると，全てのゲームにおいて AL は SB より高い CR を維持する．最終的な MSE は全てのゲームにおいて SB が AL より低い値を実現しているが，Breakthrough とどうぶつしょうぎ O では学習の前半では AL の方が低い MSE を維持している．ゲームによっては良い手を高い確率で選ぶように学習することが大きく・早く MSE を下げることにつながることがわかる．

結合手法である AL+SB について見ると，全てのゲームにおいて CR はほぼ AL と同じ値で推移し，AL と同程度に「強い」方策が学習できていることが確認できる．MSE については，Crisscross と Breakthrough では AL とほぼ同じか少し小さい値を推移しており，α と β を等しく設定した場合には AL が方策の性質へ与える寄与が SB のそれに比べて大きく支配的になりがちだと考えられる．しかしどうぶつしょうぎでは前半では AL・SB のいずれと比べても低い MSE を維持し，最終的な MSE も AL と SB のほぼ中間に落ち着いている．どうぶつしょうぎ O では常に最も低い MSE を維持している．ゲームによっては SB の寄与も無視できないということがわかる．最終的にはどうぶつしょうぎとどうぶつしょうぎ O において，AL と同程度に「強く」，かつ AL より（どうぶつしょうぎ O では，加えて SB より）バランスのとれた方策を学習できている．

4.3 対戦成績

AL，SB，AL+SB の 3 種類の手法で学習した方策を用いたプレイヤと，一様な確率でランダムに手を選ぶ方策（UR）を用いたプレイヤの 4 つのプレイヤの間で対戦を行った際の，他のプレイヤとの対戦において獲得した報酬の合計の一覧を表 3 に示す．

方策に従って直接手を選ぶプレイ方法（Direct）では，全てのゲームにおいて AL 及び AL+SB が最も多くの報酬を獲得しており，それらによって学習された方策の「強さ」を裏付けている．

SB は 3 つのプレイ方法の中では単純モンテカルロ（Simple-MC）で比較的多くの報酬を獲得しており，これは手の評価に用いる平均報酬の精度を向上させる SB の性質が現れていると言える．Breakthrough の Simple-MC では SB が最も多くの報酬を獲得しているが UCT では AL 及び AL+SB に大きく負けており，単純モンテカルロでよい結果をもたらす方策が UCT においてもよいとは限らないといえる．

本研究の提案する AL+SB は，どうぶつしょうぎとどうぶつしょうぎ O において単純モンテカルロ，UCT どちらに利用した場合でも最も多くの報酬を獲得した．これは学習の経過において CR と MSE の両方について比較的よく学習が行われていたことと合致する結果であり，ゲームによっては方策の「強さ」とバランスの両立を目指す学習手法が強いプレイングを実現するために有効であることを示している．

5. おわりに

本研究では GGP において，モンテカルロ木探索のシミュレーション方策の「強さ」とバランスという性質に着目し学習を行い，学習手法の特性がどのように方策に現れ，それがプレイヤの性能にどのように影響を与えるかについての議論を行った上で「強さ」とバランスの両立を目指すための結合手法を提案しその性能を評価した．

方策の「強さ」とバランスのプレイヤの性能への影響は，ゲームによって，またその方策の利用方法によって大きく

表 3: 他のプレイヤとの 3000 回の対戦で獲得した報酬（Crisscross を除いて，1 回の対戦で勝ち:100，引き分け:50，負け:0 の報酬を獲得する）

ゲーム	プレイ方法	AL	SB	AL+SB	UR
Crisscross	Direct	224100	199050	**224775**	102075
	Simple-MC	193800	**278400**	175950	101850
	UCT	187350	**259575**	215850	87225
Breakthrough	Direct	242700	103900	**244200**	9200
	Simple-MC	176500	**194700**	168900	59900
	UCT	**204100**	153200	203200	39500
どうぶつしょうぎ	Direct	**228750**	123800	223850	23600
	Simple-MC	161300	137850	**183400**	117500
	UCT	157100	149600	**166150**	132650
どうぶつしょうぎ O	Direct	**224700**	134300	219100	21950
	Simple-MC	163100	164550	**183300**	89100
	UCT	169400	139200	**178850**	139700

変化することを確認するとともに，さらに一部のゲームにおいてはそれらの両立を目指す結合手法がいずれか一方を目指す学習手法よりも有効であることを示した．

今後は方策の性質についてゲームごとの特性を考慮しさらに詳しい解析を進めるとともに，方策の「強さ」とバランスの望ましい両立の方法や限界について検証を進めたい．

参考文献

[1] General Game Playing. `http://games.stanford.edu/`. Accessed: 18/09/2012.

[2] General Game Playing. `http://stanfordacm.com/files/General_Game_Playing.pdf`. Accessed: 18/09/2012.

[3] muupan/dobutsushogi github. `https://github.com/muupan/dobutsushogi`. Accessed: 04/02/2013.

[4] Welcome to the Dresden GGP Server! - Dresden GGP Server. `http://130.208.241.192/ggpserver/`. Accessed: 18/09/2012.

[5] どうぶつしょうぎ official website —lpsa—. `http://www.joshi-shogi.com/doubutsushogi/`. Accessed: 04/02/2013.

[6] C. Browne, E. Powley, D. Whitehouse, S. Lucas, P. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A survey of monte carlo tree search methods. *Computational Intelligence and AI in Games, IEEE Transactions on*, No. 99, pp. 1–1.

[7] J. Clune. Heuristic evaluation functions for general game playing. In *Proceedings of the National Conference on Artificial Intelligence*, Vol. 22, p. 1134. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.

[8] H. Finnsson. *Cadia-player: A general game playing agent*. PhD thesis, Master's thesis, Reykjavik University, 2007., 2007.

[9] H. Finnsson and Y. Björnsson. Simulation control in general game playing agents. In *Proceedings of the International Joint Conference on Artificial Intelligence Workshop on General Game Playing, Pasadena, California*, pp. 21–26, 2009.

[10] M. Genesereth, N. Love, and B. Pell. General game playing: Overview of the AAAI competition. *AI magazine*, Vol. 26, No. 2, p. 62, 2005.

[11] S.C. Huang, R. Coulom, and S.S. Lin. Monte-Carlo simulation balancing in practice. *Computers and Games*, pp. 81–92, 2011.

[12] L. Kocsis and C. Szepesvári. Bandit based monte-carlo planning. *In Proceedings of the European Conference on Machine Learning 2006*, pp. 282–293, 2006.

[13] N. Love, T. Hinrichs, D. Haley, E. Schkufza, and M. Genesereth. General game playing: Game description language specification, 2008.

[14] J. Méhat and T. Cazenave. Ary, a general game playing program. In *Board Games Studies Colloquium*, 2010.

[15] J. Pitrat. Realization of a general game-playing program. In *4th IFIP Congress*, Vol. 2, pp. 1570–1574, 1968.

[16] S. Schiffel and M. Thielscher. Fluxplayer: A successful general game player. In *Proceedings of the National Conference on Artificial Intelligence*, Vol. 22, p. 1191. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.

[17] S. Sharma, Z. Kobti, and S. Goodwin. Knowledge generation for improving simulations in uct for general game playing. *AI 2008: Advances in Artificial Intelligence*, pp. 49–55, 2008.

[18] S. Sharma, Z. Kobti, and S. Goodwin. Learning and knowledge generation in general games. In *Computational Intelligence and Games, 2008. CIG'08. IEEE Symposium On*, pp. 329–335. IEEE, 2008.

[19] D. Silver and G. Tesauro. Monte-Carlo simulation balancing. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 945–952. ACM, 2009.

[20] M.J.W. Tak, M.H.M. Winands, and Y. Bjornsson. N-Grams and the Last-Good-Reply Policy Applied in General Game Playing. *Computational Intelligence and AI in Games, IEEE Transactions on*, Vol. 4, No. 2, pp. 73–83, 2012.

[21] M. Thielscher. General game playing in AI research and education. *KI 2011: Advances in Artificial Intelligence*, pp. 26–37, 2011.

[22] K. Waugh. Faster state manipulation in general games using generated code. *Proceedings of the 1st general intelligence in game-playing agents (GIGA)*, 2009.

[23] 関栄二, 三輪誠, 鶴岡慶雅, 近山隆. 将棋におけるモンテカルロ木探索の特性の解明. 第 17 回ゲームプログラミングワークショップ, pp. 68–75, 2012.

[24] 藤田康博, 浦晃, 三輪誠, 鶴岡慶雅, 近山隆. Genral game playing におけるモンテカルロ木探索のシミュレーション方策の学習. 第 17 回ゲームプログラミングワークショップ, pp. 38–45, 2012.