General 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

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、任意のゲームをうまくプレイできるプログラムを作成する試みである。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の学習に関しては、コンペティションにおける時間的制約により学習手法が制限されている。本論文では GGP においてそのような時間的制約から自由にシミュレーション方策の学習を行うことで、複数のシミュレーション方策を比較し、それらの性質を議論する。Tic-Tac-Toe を用いた実験では、平均二乗誤差は学習手法ごとで異なったが、勝率では差が見られなかった。これは用いたゲームの規模が小さかったためだと考えられる。

Learning Simulation Policies for Monte-Carlo Tree Search in General Game Playing

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

Abstract: General Game Playing (GGP) is an approach building a program which plays any game well by interpreting the game rule written in formal logic. Monte carlo tree search has recently succeeded in GGP, but learning methods of simulation policies are limited because of limitation of learning time in the competitions. In this paper, we try learning methods of simulation policies for GGP without considering the time limitation and discuss the behavior of various simulation policies. The experimental results on Tic-Tac-Toe show that mean square errors of those learning methods differ from each other. However, the rates of winning are not different. We think this is because tic tac toe is a very small game.

1. 背景と目的

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．

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から注目を集めている General Game Playing（GGP）[11] 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任意のゲームをうまくプレイできるプログラムを実現するという試みである．GGP プログラムは特定のゲームのみをプレイするプログラムと異なり，様々なゲームをうまくプレイできる必要があり，知識表現，探索，推論，学習といった人工知能

[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

技術を組み合わせながらプログラム自らが考える必要がある．GGP は人工知能技術のよりよい試金石となるとともに，ゲーム以外の問題もゲームとしてモデル化することにより扱うことが可能なため，現実世界の問題解決への応用という実用的価値も有している．

GGP における探索手法として現在広く使われているのが，ランダム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モンテカルロ木探索であり [7]，その性能はシミュレーションにおける手の選び方（これをシミュレーション方策，あるいは単に方策と呼ぶ）に大きく依存する [12]．しかし GGP においてどのような方策が効果的か，それを学習するにはどうするのがよいか，という点に関しては，過去のコンペティションにおいて厳しい時間的制約のため学習手法が限定されてきたこともあり，未解明の部分が大きい．そのため効果的な方策とその学習手法をより一層明らかにすることが GGP プレイヤの性能向上のために求められている．

本研究では，GGP におけるモンテカルロ木探索のシミュレーション方策に関し，これまでコンペティションにおける時間的制約により用いることが難しかった方策の学習手法を GGP プレイヤに実装し対戦実験を行い，その性能を評価する．その目的は GGP における効果的なシミュレーション方策とその学習手法を明らかにし，GGP プレイヤの性能向上に貢献することである．またシミュレーション方策とその学習手法に関し，特定のゲームにおいて見られた性質が GGP の様々なゲームにおいてどのように現れるかということを明らかにすることで，その一般性の評価にも貢献する．

2. Genera 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任意のゲームをうまくプレイできるプログラムを実現するという試みである．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，人工知能研究における大きな挑戦として広い関心を集めている [19]．

2.1 扱うことができるゲーム

一般的な GGP システムが扱うことができるゲームは次の条件を満たすものに限る [14]．

- 有限な状態をとる
- 決定的である（確率的要素を含まない）
- 完全情報である（プレイヤにとって隠された情報がない）
- プレイヤの数は 1 人でも 3 人以上でもよいが，開始から終了まで変化しない

2.2 Game Description Language

GGP においてゲームルールは Game Description Language（GDL）[14] によって記述される．GDL は Datalog という Prolog に似たクエリ言語を元にしており，表 1 のような専用の関係述語が用意されている．

2.2.1 GDL の記述例

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以下に示す．Tic-Tac-Toe は 3×3 の網目からなるゲームであり，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最初はセル $(1, 1)$ は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))
```

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

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

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)))
```

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．

2.3 マッチの進行

GGP プレイヤは Game Manager（GM）と呼ばれるシステムと通信することによりゲームをプレイする．

まず GM は各プレイヤに次の情報を送信する．これを START 命令と呼ぶ．

- GDL により記述されたゲームルール
- 送信先プレイヤの役割（ゲームルールの role 関係述語で定められた役割のうちいずれか）
- startclock（第 1 ターンが始まるまでの思考時間）
- playclock（1 ターン毎の思考時間）

startclock が経過した後はターンの繰り返しによりゲームが進行する．各ターンでは，まず GM が直前に各プレイヤが選択した手（第 1 ターンでは NIL）を全プレイヤに送信する．これを PLAY 命令と呼ぶ．PLAY 命令を受信したプレイヤは playclock 以内に自分が選択する手を GM に送信しなければならない．GM は各プレイヤから手を受信し，それらをまた次の PLAY 命令で全プレイヤに伝える．もし GM がプレイヤから合法手でない手を受信するか，あるいは playclock 以内に手を受信しなかった場合は，GM がそのプレイヤの代わりに合法手からランダムに手を選ぶ．

ゲームの終了条件が満たされた場合，GM は PLAY 命令の代わりに各プレイヤに STOP 命令を送信し，各プレイヤの報酬を記録する．START 命令から STOP 命令までのプロセスをマッチと呼ぶ．

2.4 過去のコンペティション

2005 年から 2012 年現在まで毎年，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．

このうち 2007 年，2008 年及び 2012 年に優勝した CadiaPlayer と 2009 年及び 2010 年に優勝した Ary はモンテカルロ木探索を使用したプレイヤであり [7]，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 [3]．このような過去のコンペティションにおける実績から，シミュレーションに基づいた探索手法，とりわけモンテカルロ木探索の GGP に

表 2: 過去の GGP コンペティションの優勝プレイヤ [2], [18]

2005	ClunePlayer
2006	FluxPlayer
2007	CadiaPlayer
2008	CadiaPlayer
2009	Ary
2010	Ary
2011	TurboTurtle
2012	CadiaPlayer

おける有効性が示されているといえる．

3. 関連研究

3.1 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る．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．

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ると言える．

3.1.1 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）[13] 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 [13]．

3.1.2 シミュレーション方策

ランダム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，各状態において各プレイヤが手をどのような確率分布に従って選択するかに依存する．一様分布に従い手を選ぶのが最も単純な方策であるが，それによってはよい評価は得られない．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の方がよい評価をもたらす．したがって方策は実際のプレイヤの性能に大きな影響をもたらし [12]，方策を改善することはモンテカルロ木探索を用いたプレイヤの性能を向上をもたらす．

3.2 GGP における特徴の抽出と方策の学習

GGP において，状態や手に対する評価のようなゲーム固有の知識は，GM から与えられるゲームルールのみから獲得する必要がある．そのために一般に用いられるのが，自己対戦結果等を元にした，状態や手に対する評価の学習である．そのような学習を行うためには状態や手から特徴を抽出する必要がある．

これまでに GGP プレイヤにおいて用いられてきた特徴としては，手そのものや，真である命題 1 つ，真である命題 1 つと手の組というような単純なものもあれば [9], [16]，駒の種類や盤面上の場所といった高度な概念を用いたものもある [10]．そのような高度な概念は GDL による表現おける述語の形のようなパターンについてテンプレートとの比較を行うことにより得ることが可能であるが，ゲームによってはそのような概念が存在しない場合があり，また同じゲームでも人間が GDL の文法の範囲内でゲームルールをどのように表現するかにより結果が変わる恐れがある．一方，手や真である命題といった特徴は GDL により表現されたゲームである限り常に用いることができる．

局面や手の特徴を抽出することにより，その評価を学習し，それをシミュレーションにおいて手を選ぶための方策として利用することが可能になる．例えば CadiaPlayer の Move-Average Sampling Technique（MAST）[9] では，モンテカルロ木探索の逆伝搬ステップにおいて，各ノードに統計的情報を記録するだけでなく共通のテーブルに手のみに対する平均報酬を記録し，それをシミュレーションで手を選ぶために用いている．一方 Feature-to-Action Sampling（FAST）[10] は，駒の種類を抽出できたならば駒の種類毎の評価を，駒の場所を抽出できたならば駒の場所毎の評価を学習した上で，相手の駒を一対一で獲得する場合には駒の種類の価値を，新たな場所に駒を置くだけの場合にはその場所の価値を考慮し，シミュレーションにおける手を選ぶという手法である．

GGP におけるモンテカルロ木探索の方策の学習には以上のように複数の先行研究があるが，コンペティションにおける startclock 及び playclock の短さから，抽出する特徴，方策の学習手法，利用場面等に制限があり，比較的計算コストの大きい方策の学習手法の利用の妨げとなっている．例えば 2012 年のコンペティションにおける startclock は 30〜180 秒，playclock は 15〜40 秒であった [1]．

3.3 「強い」方策とバランスのとれた方策

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 [12], [17]．

3.3.1 方策の「強さ」とバランス

ここでは方策を，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．

$$J(\theta) = \mathbb{E}_\rho[\mathbb{E}_{\pi_\theta}[\delta_t^2 | s_t = s]]$$

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

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．

3.3.2 Apprenticeship Learning

「強い」方策を学習する手法として，状態 s_l とそこでの最善手 a_l の組を訓練データとし，それらと可能な限り同じ手を選ぶように学習する Apprenticeship Learning（AL）が提案されている [17]．

この学習手法は 1 つの訓練データ (s_l, a_l) に対して次のように重みベクトル θ を更新する．

$$\Delta\theta = \alpha\psi(s_l, a_l) \tag{4}$$

ただし，

$$\psi(s,a) = \nabla_\theta \log \pi(s,a) \tag{5}$$

$$= \phi(s,a) - \sum_b \pi_\theta(s,b)\phi(s,b) \tag{6}$$

であり，α は学習率である．

3.3.3 Policy Gradient Simulation Balancing

バランスのとれた方策を学習する手法として Policy Gradient Simulation Balancing（SB）が提案されている [17]．これはミニマックス値の近似値が知られている多数の状態を訓練データとして，シミュレーションの平均報酬がミニマックス値に近づくように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．

擬似コード中の V は平均報酬をどの方向に変更する必要があるかを意味するもので，偏り（bias）と呼ばれる．g は平均報酬を増やすために重みベクトルをどの方向に変更する必要があるかを意味するもので，方策勾配（policy gradient）と呼ばれる．これらはそれぞれ別々に行われる M 回及び N 回のシミュレーションの結果より推定される．$\hat{V^*}(s)$ は状態 s のミニマックス値の近似値を表す．α は学習率である．

Algorithm 1 Policy Gradient Simulation Balancing[17]

```
θ ← 0
for all s_1 ∈ trainingdata 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
```

3.3.4 囲碁における評価

以上の 2 つの学習手法のうち，5×5 と 6×6 の囲碁において SB が，評価精度，プレイヤの強さともに優れていることが示されている [17]．また 9×9 の囲碁においても SB が有効であることが示されている [12]．

4. 提案手法

GGP における学習には 3.2 節で述べたような時間的制約という困難があるが，コンペティションの設定は GGP という試みに対して必然的なものではなく，そのような制約をひとまず考慮から外し多様な手法を試し評価することも，GGP における効果的な方策やその学習手法を明らかにするためには有益だと考えられる．そうして得られた知見は厳しい時間的制約の下での学習にも役立つことが期待される．

本研究では，GGP において，コンペティションのような厳しい時間的制約を考慮から外した上で，効果的な方策の性質に迫るような複数の方策学習手法の評価を行う．取り上げる手法は Apprenticeship Learning（AL），及び Policy Gradient Simulation Balancing（SB）の 2 つである．それらを用いて学習と対戦を行うことにより，GGP における「強い」方策とバランスのとれた方策の関係及びそれらを学習する手法の有効性を評価する．

AL と SB を利用するためには訓練データがなければならない．[17] ではプレイヤとは別の高性能なプログラムによりシミュレーションを繰り返すことで訓練データを作成しているが，GGP ではプレイヤは事前にどんなゲームをプレイするか知らないという前提があるため，ゲーム毎に評価のための外部プログラムを使用して訓練データを作成する方法を採ると GGP の枠組みからは外れてしまう上，

評価に使用できるゲームの種類も制限されてしまう．そこで本研究では学習の前段階として，ゲームの初期状態から終局までランダムなシミュレーションを行い，その中で現れた状態からこれまでに選ばれていない状態を 1 つ選び出す操作を繰り返すことにより多数の状態を得た上で，各状態に対し，UCT を用いて十分な回数のシミュレーションを行うことにより，AL に対してはその状態における最善手，SB に対してはその状態のミニマックス値の近似値を求め，それらを訓練データとして用いる．

5. 評価

4 章で述べた 2 つの学習手法を GGP プログラムに実装し，それぞれ学習を行いその経過を観察した．

5.1 評価設定

学習手法を実装するための GGP プログラムとしてポツダム大学により提供されている Basic Player[4] を用い，方策の学習以外の点に関しては同条件となるようにした．Game Master プログラムとして，GGP Server Project により提供されている GameController[5] を使用した．

学習を行うゲームとして Tic-Tac-Toe を用いた．GDL 記述としては Dresden GGP Server[6] で提供されているものを用いた．

2 つの学習手法に共通の条件として，勝利時の報酬を 1，敗北時の報酬を −1，引き分け時の報酬を 0 として扱った．学習率は共に 0.01 とした．学習率を 1 あるいは 0.1 とした場合は学習開始直後に手の確率分布が極端に偏ってしまい，学習の成果が得られなかったためである．

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手と真である 2 命題の組み合わせを用いた．例えば「(cell 1 1 x)(cell 1 2 x)(mark 1 3)」という特徴はセル (1,1) とセル (1,2) が x でマークされている状態でセル (1,3) にマークするという状況を意味する．特徴ベクトル $\phi(s,a)$ の特徴「(cell 1 1 x)(cell 1 2 x)(mark 1 3)」に対応する要素は，状態 s において (cell 1 1 x) と (cell 1 2 x) の 2 命題が真であり，かつ手 a が (mark 1 3) である場合に限り 1 であり，そうでなければ 0 である．手と真である 2 命題の組み合わせという特徴は過去の GGP の方策学習の例と比較して豪華なものといえる．特徴の数が増えると十分な学習のためにより長い時間が必要となってしまうが，特徴の表現力が小さいほど方策の学習の幅が狭くなり学習手法による差が観察しづらくなると考えられるため，このような特徴を採用した．

バランスのとれた方策の学習の際には，M，N はともに 100 とし，SB の訓練データとして，上述の方法により抽出された重複のない 500 の状態を用い，各状態に対し UCT による 1 万回のシミュレーションを行うことで得られた平均報酬をミニマックス値の近似値とした．AL に対しても同様に 500 の状態に対しシミュレーションを行い，平均報酬が最も高い子ノードに対応する手を最善手とみなした．

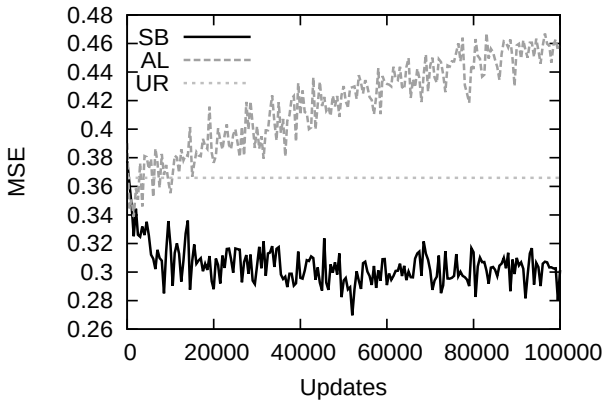

図 1: 100 回のシミュレーションの平均報酬の，テストデータに対する MSE（Tic-Tac-Toe）

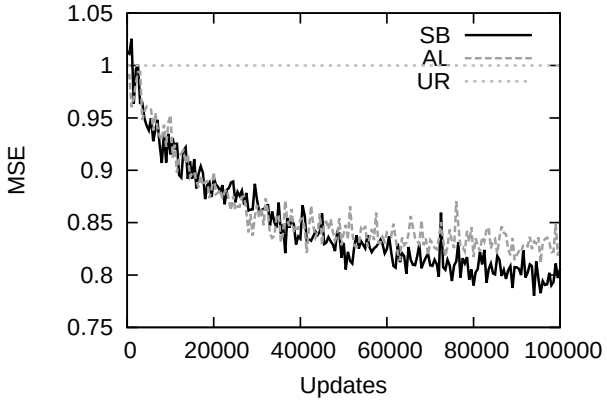

図 2: 1 回のシミュレーションの報酬の，テストデータに対する MSE（Tic-Tac-Toe）

5.2 学習経過

SB の訓練データを作成したのと同じ方法により 100 の状態からなるテストデータを用意し，500 の訓練データ全てついて 1 回ずつ重みベクトルを更新する度に，その方策による評価の正確さに関する 2 種類の誤差を調べた．

1 つは，その時点までに学習した方策を用いた，テストデータの各状態に対するシミュレーション 100 回の平均報酬の，テストデータの値に対する誤差の 2 乗の全テストデータに対する平均（Mean Squared Error，MSE）であり，これによりシミュレーションを繰り返して得られる平均報酬による評価がどれだけ正確かを評価する．

もう 1 つは，各状態に対する 1 回のシミュレーションの報酬のテストデータの値に対する誤差の 2 乗を，各状態につき 100 回のシミュレーションに対して求めたものを平均したものであり，これによりシミュレーション 1 回の報酬のみによる評価がどれだけ正確かを評価する．

Tic-Tac-Toe に対する，100 回のシミュレーションの平均報酬に関する MSE，1 回のシミュレーションに関する

MSE をそれぞれ図??に示す．UR（Uniform Random）は一様な確率分布に従い手を選ぶ方策を指す．AL，SB ともに，1 回のシミュレーションに関する MSE は学習に従い UR より小さくなっていることが確認できる．しかしながら，100 回のシミュレーションの平均報酬に関する MSE に関しては，SB ではこれについても小さくなっているものの，AL では学習開始直後は UR を下回るものの上昇に転じてしまっている．この理由として考えられるのは，AL による学習が進行するにつれて方策がほとんど決まった手しか選ばなくなってしまい，シミュレーションを繰り返しても評価精度が向上していないということである．

そこで AL と SB の両方について，図 1 の右端まで学習が進んだ場合の，初期状態（全てのセルが空）に対する手の確率分布を確認してみると，SB による方策は高確率のものからおおよそ 0.57，0.29，0.10 のような分布を与えている一方，AL による方策は高確率のものから 0.94，0.03，0.01 のような分布を与えている．この傾向は初期状態だけでなく多くの状態において見られる．これはほとんど決まった手しか選んでいないということであり，先程の説明と合致している．

もし方策がこのような高い確率で常に最善手を選ぶことができているならば，平均報酬はほぼミニマックス値に一致し，MSE はほぼ 0 になると考えられるが，学習した結果はそれとは異なるものである．これは，ほとんど常に最善手を選ぶような方策を学習するには本実験で用いた特徴だけでは表現力が不足しており，その結果として AL による方策が少なくない局面で最善手ではない手を高い確率で選んでしまっているという説明が可能である．この場合，いくらシミュレーションを繰り返してもその局面では最善手はほとんど選ばれず，平均報酬もミニマックス値と乖離してしまう．

このように AL による「強い」方策より，SB によるバランスのとれた方策の方が，シミュレーションを繰り返した場合の平均報酬のテストデータの値に対する MSE を小さくすることができるという点は，囲碁における研究 [17] と一致している．

学習した方策を UCT に用いて Tic-Tac-Toe において対戦実験を行った結果を表 3 に示す．この表から，どの方策を用いた場合でも引き分けが圧倒的多数であり，方策間に差が見られなかった．Tic-Tac-Toe は小さいゲームであるため，UCT による木の展開により十分な探索が行えるからだと考えられる．

表 3: UCT を用いた対戦結果

方策	勝ち	引き分け	負け
AL (vs SB)	11	974	15
AL (vs UR)	39	949	12
SB (vs UR)	13	986	1
AL (vs AL)	79	421	0
SB (vs SB)	1	499	0
UR (vs UR)	11	489	0

6. おわりに

本論文では GGP において時間的制約から自由にシミュレーション方策の学習を行うことで、複数のシミュレーション方策を比較し、それらの性質を議論した。Tic-Tac-Toe を用いた実験では、平均二乗誤差は学習手法ごとで異なったが、勝率では差が見られなかった。これは用いたゲームの規模が小さかったためだと考えられる。そのため，今後はより大きな規模のゲームでも同様の学習実験をしていきたい．

参考文献

[1] Gamemaster. `http://gamemaster.stanford.edu/`. Accessed: 18/09/2012.

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

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

[4] General Game Playing Potsdam. `http://www.ggp-potsdam.de`. Accessed: 18/09/2012.

[5] GGP Server. `http://sourceforge.net/apps/trac/ggpserver/`. Accessed: 18/09/2012.

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

[7] 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.

[8] G. Chaslot, S. Bakkes, I. Szita, and P. Spronck. Monte-carlo tree search: A new framework for game ai. In *Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference*, pp. 216–217, 2008.

[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] H. Finnsson and Y. Björnsson. Cadiaplayer: Search-Control Techniques. *KI-Künstliche Intelligenz*, Vol. 25, No. 1, pp. 9–16, 2011.

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

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

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

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

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

[16] 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.

[17] 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.

[18] 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.

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

[20] R.J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, Vol. 8, No. 3, pp. 229–256, 1992.