モンテカルロ木探索を統合したプレイアウト方策の最適化

渡辺順哉[1,a)]　美添一樹[2,b)]　金子知適[1,c)]

概要：モンテカルロ木探索は囲碁を始めとするさまざまなゲームで有力とされている探索手法である．本研究ではシミュレーションバランシングと呼ばれる学習法を元にして，プレイアウト方策の学習をモンテカルロ木探索の枠組みの中で行うことを提案する．実験では，モンテカルロ木探索におけるノードを展開する閾値を調整することで，重みの収束の様子に差が観察された．また，9 路盤での Fuego を相手とした対戦実験ではシミュレーションバランシングの勝率をはっきりと上回ることは出来なかったものの，ノードを展開する閾値を大きくして方策の学習を行うことで勝率が上昇することがわかった．

Optimization of Playout Policy Integrated with Monte-Carlo Tree Search

Junya Watanabe[1,a)]　Kazuki Yoshizoe[2,b)]　Tomoyuki Kaneko[1,c)]

Abstract: Monte-Carlo tree search is a search method widely regarded that is effective in various games including the game of Go. In this paper, we present a learning method for the playout policy in Monte-Carlo tree search based on the Simulation Balancing method. In our experiments, we observed that the convergence rate of feature weights get affected by the expansion threshold in Monte-Carlo tree search. We also analyzed winning rate of our method on 9x9 board against Fuego. Though our method did not outperform Simulation Balancing clearly, we confirmed that the winning rate improves by increasing the expansion threshold.

1. はじめに

現在，さまざまなゲームにおいてコンピュータプレイヤの棋力は人間のトップを上回っている．しかし，囲碁というゲームにおいてはそのレベルはアマチュア 6，7 段レベルに留まっている [3]．この理由としては主に，囲碁においては局面の良し悪しを静的に判断する評価関数の作成が困難であり，その他のゲームで成功してきた評価関数を用いた mini-max 探索があまり有効でないことが挙げられる．

そこで現在，強い囲碁プログラムを作る上で広く用いられている探索手法がモンテカルロ木探索である．モンテカルロ木探索は「プレイアウト」と呼ばれるランダムなシミュレーションによってノードに報酬を与え，その結果を用いて有望なノードを中心にゲーム木を展開していく．また，プレイアウトでは，あるノードからランダムに手を選択し終局まで局面を進行させ，終局時に得られるスコアをノードに与える．モンテカルロ木探索は探索に局面の評価関数を必要とせず，この点では囲碁に適した探索手法であると言える．また，プレイアウトにおける手の選択では，一様ランダムに手を選択するのではなく，ある着手確率 (方策) に従って手を選択することが有力とされている [3]．従って，方策の質を高めることはモンテカルロ木探索において重要な問題である．

また，方策で用いる重みの学習は様々な観点から試みられている．例えば Minorization-Maximization 法 [5] と呼ばれる学習法では強いプレイヤーの棋譜になるべく手が一致するように方策を学習する．また，シミュレーションバランシング [1] と呼ばれる学習法ではプレイアウト方策のバ

[1] 東京大学大学院総合文化研究科
Department of General Systems Studies, Graduate School of Arts and Sciences, The University of Tokyo
[2] 東京大学大学院新領域創成科学研究科
Graduate School of Frontier Sciences, The University of Tokyo
[a)] watanabe@graco.c.u-tokyo.ac.jp
[b)] zoe.f16xl@gmail.com
[c)] kaneko@graco.c.u-tokyo.ac.jp

ランスを最適化するように学習を行う．これれらの学習法はいずれもプレイヤーの棋力を上昇させる点において成功を収めている．しかしながら，これらの学習法及びその他の学習法においても，モンテカルロ木探索の枠組みを目的関数に含むような学習手法は確立されていない．そこで，本研究では目的関数にモンテカルロ木探索の枠組みを導入し最適化を行なうことで，モンテカルロ木探索と相性の良い方策を学習しプレイヤーの棋力を向上させることを目的とする．

2. 関連研究

2.1 シミュレーションバランシング

シミュレーションバランシング [1] は方策のバランスを最適化する学習法であり，以下 2.2 節で定義する目的関数を 2.3 節で解説するアルゴリズムで最適化することで方策の学習を行なう．なお，バランスの定義やより詳細な計算内容については参考文献 [1] を参照のこと．

2.2 目的関数

シミュレーションバランシングでは以下の関数 L を最小化することを目標とする．

$$L = E_\rho\Big[(E_{\pi_\theta}[z|s] - V^*(s))^2\Big] \tag{1}$$

ここで，z は報酬，ρ は局面集合，π_θ は方策，$V^*(s)$ は局面 s の minimax 値である．また，$E_{\pi_\theta}[z|s]$ は局面 s において方策 π_θ に従ってプレイアウトした際に得られる報酬 z の期待値である．なお，具体的な方策 π_θ としてはソフトマックス方策を用いる [4]．ソフトマックス方策では局面 s において手 a を選ぶ確率 $\pi_\theta(s,a)$ は

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

と定義される．ただし，$\phi(s,a)$ は特徴ベクトル，θ は重みベクトルである．

つまり，この学習法では局面の minimax 値とプレイアウトの平均報酬の平均自乗誤差を最小化するように方策の学習を行なう (本研究では minimax 値は-1, 1 の 2 値のいずれかという立場を取り，報酬 z の期待値は [-1, 1] の範囲の実数値となる)．

2.3 最適化

式 (1) を目的関数としたとき，重み θ の具体的な学習法としては方策勾配法を用いる．

まず，式 (1) の勾配を求めると以下のようになる:

$$\nabla_\theta L = -2E_\rho[(V^*(s) - E_{\pi_\theta}[z|s])\nabla_\theta E_{\pi_\theta}[z|s]].$$

ここで，$\chi(s)$ を $s_1 = s$ から始まる局面の進行 $\xi = (s_1, a_1, \cdots, s_T, a_T)$ の集合とすると，

Algorithm 1 式 (1) を目的関数としたオンライン学習

```
for all s_1 ∈ training set do
  V ← 0, g ← 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
  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} ∇_θ log π_θ(s_t, a_t)
  end for
  θ ← θ + α(V̂*(s_1) − V)g
end for
```

$$\begin{aligned}\nabla_\theta E_{\pi_\theta}[z|s] &= \sum_{\xi\in\chi(s)} \nabla_\theta(\pi_\theta(s_1,a_1)\cdots\pi_\theta(s_T,a_T))z(\xi) \\ &= E_{\pi_\theta}\left[z\sum_{t=1}^{T}\nabla_\theta log\ \pi_\theta(s_t,a_t)\right] \end{aligned}\tag{2}$$

そして，$\nabla_\theta \log \pi_\theta(s_t,a_t)$ の計算は次の様に行なわれる．

$$\nabla_\theta \log \pi_\theta(s_t,a_t) = \phi(s,a) - \sum_b \pi_\theta(s,b)\phi(s,b)$$

ただし，$\phi(s,a)$ は局面 s における手 a の持つ特徴ベクトル，b は全ての合法手を表す．以上の計算から，シミュレーションバランシングにおける方策勾配法による方策の学習はアルゴリズム 1 によって行なう [1]．

なお，V はプレイアウトの報酬の平均値，α はステップサイズと呼ばれる定数．M, N はプレイアウト数である．なお，minimax 値 $V^*(s)$ の値は実際には求めることが出来ないので，その代わりに十分時間をかけたモンテカルロ木探索による局面の評価値 $\hat{V}^*(s)$ を用いる．

3. UCT と Progressive Widening

本説では代表的なモンテカルロ木探索の 1 つである UCT アルゴリズム，及び本実験で用いた囲碁プログラムに実装した Progressive widening [5] と呼ばれる枝刈り手法について解説する．

3.1 UCT

UCT [6] はモンテカルロ木探索のアルゴリズムの 1 つである．まず，基本的なモンテカルロ木探索のアルゴリズムの概要を以下に示す [3]．なお，プレイアウトとはある局面から終局面までランダムに合法手を選択し，終局面のスコアを獲得するという一連の手続きである．

(1) ルートノードから，評価値の高いノードを順に選択しリーフノードまで到達．
(2) リーフノードからこれまでに行なわれているプレイアウトの回数が閾値に達しているならば，全ての合法手を展開し 1 回ずつプレイアウトを行なう．そうでなければリーフノードから 1 回プレイアウトを行い，得られたスコアを用いて，1 で巡ったノードの評価値を

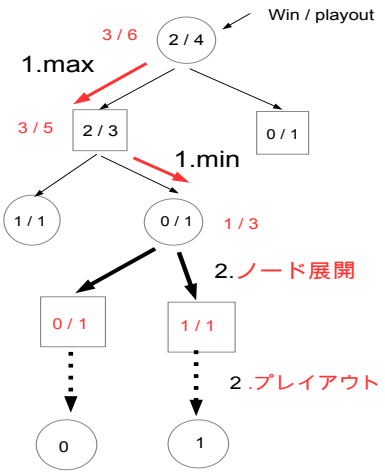

図 1 モンテカルロ木探索の例 (展開の閾値を 1 とした), ノード横の数字はプレイアウトの結果によって更新されたノードの勝率

更新.

(3) 手順 (1), (2) を制限時間まで繰り返す

(4) ルートノードからの訪問回数 (または勝率など) が最大のノードを選択

ここで，単純なモンテカルロ木探索としては，ノードの評価値はプレイアウトの結果から得られる勝率を用いるという方法も考えられる．しかし，そのようにノードの評価を単純に勝率とした場合，最初の数回のプレイアウトでノードの勝率が見誤られると，それ以降手順 1 で選択されなくなってしまう可能性がある．また，これはランダムなシミュレーションを用いるプレイアウトの性質上よく起こることである．この問題点に対して，ノードの評価を勝率に加えノードへの訪問回数も考慮するようにしたのが UCT アルゴリズムである．UCT アルゴリズムではノードへの訪問回数をバイアス項に含む UCB1 値を用いてノードを評価する．

$$UCB1 = V + C\sqrt{\frac{logN}{n}}$$

ここで，n はあるノードへの訪問回数，N は親ノードの訪問回数，C は定数である．この様に UCB1 値を用いてノードを評価することで訪問回数が少ないノードを高く評価し，手順 1 で再選択することが出来る様になるのである．

3.2 Progressive Widening

progressive widening は枝刈りの一手法である [5]．progressive widening では 3.1 節アルゴリズムの手順 1 において，全てのノードを同時に展開せず，ノードへの訪問回数に応じて，事前に計算した評価値の高いノードから徐々にノードを展開する．この様にノードを展開することで，評価値の高いノードを優先して探索することが出来る様になる．なお具体的には，あるノード S に評価値が n 番目のノードが追加されるのは，ノード S への訪問回数が以下で定義される t_{n-1} 回に達したときである．

$$t_{n+1} = t_n + 40 \times 1.4^n,$$
$$t_0 = 0.$$

4. 提案手法

本研究では学習後の方策がモンテカルロ木探索で用いられることを前提として，学習を効果的に行なう手法を提案する．なお，以下ではシミュレーションバランシング (SB) と本手法を区別するために，本手法を UCT バランシング (UCTB) と呼ぶことにする．

4.1 目的関数

本手法では具体的に次の目的関数 M を用いる手法を提案する．

$$M = E_\rho\left[\left(E_{UCT\pi_\theta}[z|s] - V^*(s)\right)^2\right] \quad (3)$$

ただし，$E_{UCT\pi_\theta}[z|s]$ は局面 s から方策 π_θ の元で UCT 探索を行い，UCT アルゴリズムが最も訪問したノードで得られる報酬の期待値と定義する．一方，シミュレーションバランシングでは $E_{\pi_\theta}[z|s]$ は局面 s からプレイアウトして得られる報酬の期待値である．この目的関数における期待値の定義の差がシミュレーションバランシングと本手法の違いである．そして本手法ではこの様に目的関数を定めることで，モンテカルロ木探索が有望と判断した局面から行なわれたプレイアウトを中心に学習することが出来る．また，モンテカルロ木探索では深さの異なるノードからプレイアウトが行なわれる．シミュレーションバランシングではルートノードから直接行なわれたプレイアウトを学習の対象にしており，学習後の方策が必ずしもモンテカルロ木探索内で最適であるかははっきりとしない．その点，本提案手法では，モンテカルロ木探索で行なったプレイアウトを学習対象とすることで，モンテカルロ木探索で用いる方策の学習がより効果的に行われることが期待される．

4.2 最適化

式 (3) の最適化はシミュレーションバランシングと同様に強化学習の 1 手法である方策勾配法で行なう．式 (3) の勾配を求めると以下のようになる：

$$\nabla_\theta M = -2E_\rho[(V^*(s) - E_{UCT\pi_\theta}[z|s])\underline{\nabla_\theta E_{UCT\pi_\theta}[z|s]}_{①}.$$

ここで，$E_{UCT\pi_\theta}[z|s]$ は方策 π_θ で UCT 探索を行い，訪問回数が最大のノードの勝率であるが，この勾配である上式 ① を正確に計算することは困難である．そこで，本手法では UCT におけるノード選択は学習の枠組みから外し，リーフノードから行なわれるプレイアウトを学習の対象とする．そして，① の計算は UCT で行なわれるプレイアウトに対して式 (2) と同様に計算することとする．なお，学

Algorithm 2 式 (3) を目的関数としたオンライン学習

for all $s_1 \in$ training set **do**
  $V \leftarrow 0, g \leftarrow 0$
  for $i = 1$ to M **do**
    simulate $(s_1, a_1, \ldots, s_T, a_T; z)$ using UCT+π_θ
  end for
  select most visited node N.
  $V \leftarrow N's\ win\ rate$
  for $j = 1$ to N **do**
    simulate $(s_1, a_1, \ldots, s_T, a_T; z)$ using UCT+π_θ
    $g \leftarrow g + \frac{z}{N(T-T'+1)} \sum_{t=T'}^{T} \nabla_\theta \log \pi_\theta(s_t, a_t)$
  end for
  $\theta \leftarrow \theta + \alpha(\hat{V}^*(s_1) - V)g$
end for

習に用いるプレイアウトは UCT が選択したノード以下で行われたものに限定せず，UCT 探索全体で行ったプレイアウトを対象とする．

実際の学習は Algorithm 2 を方策の重みが収束するまで繰り返す．なお，Algorithm 2 では UCT 探索を行いながら 1 局面毎に重みベクトルを更新している．

ここで，本手法での V は UCT において最も選択されたノードから行われたプレイアウトの報酬の平均値，M, N は UCT におけるプレイアウト数である．そして，a_t は $t = 1$ から $T^{'}$ の範囲では UCT がノードを降りる際に選択した行動とし，$t = T^{'}$ から T ではプレイアウト中に方策が選択した行動とする．

5. 実験結果と考察

本実験では提案手法とシミュレーションバランシングの性能を比較することを目的としており，実験条件はなるべく参考文献 [2] に揃えるようにした．

5.1 対局プログラム

本実験において対戦比較での使用，及び学習局面の生成に用いる囲碁プログラムは参考文献 [3] の実践編に記載されている囲碁プログラムを改変して用いた．また，本対局プログラムでは 3.2 節で解説した Progressive widening を実装し，特徴としては手番を考慮した 3x3 パターン，及び盤端からの距離を用いた．なお, progressive widening で用いる重みの学習は，コンピュータ囲碁の対局サイトである cgos 上で行われたレーティング 2700 台の強いコンピュータ囲碁プログラム同士の対局記録 [7] のうち 1000 局を用いて MM 法 [5] によって行なった．また，参考文献 [3] の実践編に記載されているプログラムを参考にして UCB1 値の定数項の値は 0.31 とした．

5.2 特徴

プレイアウト方策に用いる特徴は手番を考慮した 3x3 パターンに加え，参考文献 [2] を参考にして以下の 6 つの特徴を用いた．

(1) 直前に相手の打った手の周囲 8 カ所．さらに，以下の特徴 2 から 6 も含む．
(2) 直前にアタリをかけられた自分の連に接する敵連を取る．ただし，特徴 3 に含まれる場合は除く．
(3) 直前にアタリをかけられた自分の連に接する敵連を取る．ただし，助けた連がアタリになる．
(4) 直前にアタリをかけられた自分の連をノビる．
(5) 特徴 4 の条件が成り立ち，かつノビた連がアタリ．
(6) 直前にダメを 3 から 2 に減らされた自分の連に接する敵連の内，ダメ 2 でかつ打てば必ずその連が取れる．

5.3 実験条件

まず，学習局面は 5.1 節に記載したプログラムのプレイアウト数を 1000，方策を一様ランダムとして，9 路盤での自己対戦の記録 5000 局から 1 局当り 1 局面をランダムに選択し，20000 プレイアウトの Fuego の評価値を教師例とした．また，具体的な方策 π_θ としては，θ を 5.2 節に記載した特徴の重みとしたソフトマックス方策 [4] を用いた．なお，ステップサイズは 10，プレイアウトの報酬は-1, 1 とし，プレイアウト数 M, N は共に 500 とした．

また，本実験では UCT におけるノード展開の閾値を 1，3，5，10，30 の 5 通りに設定し実験を行なった．展開の閾値を増やすことによって探索木が浅く広くなり，学習結果に影響を与えることが予測される．なお，本実験ではその他のパラメータである UCB1 値の定数項や progressive widening の式などは固定した．

5.4 学習結果

シミュレーションバランシングと提案手法でそれぞれ方策の学習を行い，学習の反復過程における目的関数の値の変化を図 2 に示す．ただし，「UCTB-i」は展開の閾値を i とした場合の提案手法を示すこととする．なお，目的関数の値は学習を 1 反復行うごとに保存した重みを用いて，学習後に学習局面とは別のテスト局面 500 局に対して計算した．図 2 からシミュレーションバランシング及び提案手法の両手法において，目的関数を減少させるように学習していることが分かる．また次の図 3 においてはシミュレーションバランシングと UCTB-30 に対して学習局面とテスト局面での目的関数の値の推移を示す．なお，学習局面に対する目的関数の値を計算する際には，計算時間を省略する目的で全学習面数 5000 のうち 500 局面を用いた．図 3 の結果から両手法ともに，反復回数が増えるにつれて学習局面に対する目的関数の値は減少していることが分かるが，UCTB-30 においてはテスト局面に対して横ばいとなっている様子が観察される．また，両手法ともに学習局面に対する目的関数の値の方がテスト局面に対する値より下がっている．次に，シミュレーションバランシングと UCTB-30 において，学習の反復回数と L1 ノルムの関係を

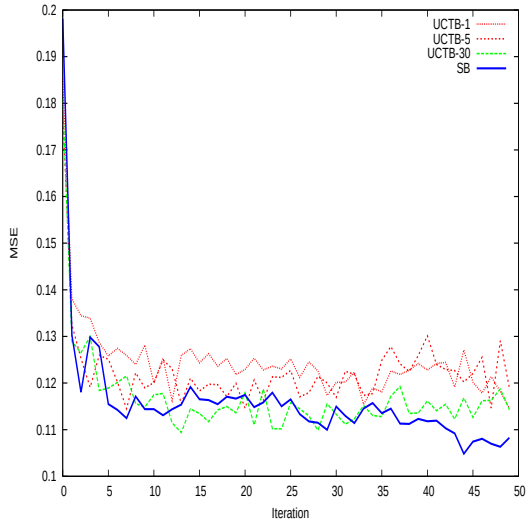

図 2　学習過程における目的関数の変化

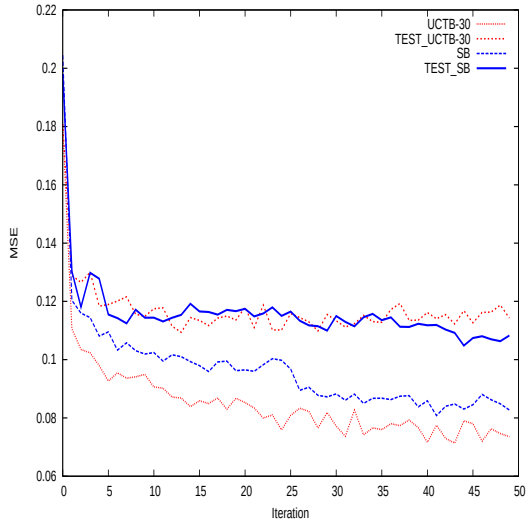

図 3　学習過程における目的関数の変化 (UCB-30 と SB に対して，学習局面とテスト局面の目的関数値を比較)

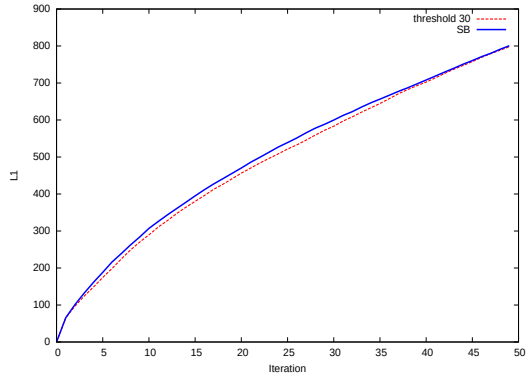

図 4　学習の反復回数毎の L1 ノルムの値

以下の図 4 に示す．図 4 から，反復回数の前半部においてはグラフの傾きは減少しながらも後半部ではほぼ直線になり，またグラフ全体においては L1 ノルムの値は単調増加している様子が観察される．この結果と図 3 の結果を合わせると過学習が行われていると考えられる．

表 1　Fuego に対する勝率．勝率の 95 %信頼区間は全て約 ±1.6%.

反復回数	10	20	30	40	50	60
SB	52.2%	54.6%	51.7%	56.8%	57.2%	58.3%
UCTB-1	38.4%	41.9%	38.9%	40.6%	43.6%	
UCTB-3	43.3%	45.7%	42.5%	44.0%	44.9%	
UCTB-5	42.9%	44.0%	46.6%	46.8%	48.5%	
UCTB-10	43.5%	47.9%	50.5%	50.7%	46.1%	
UCTB-30	48.5%	50.2%	48.8%	52.8%	53.1%	51.0%

次に学習後の方策を用いて 9 路盤での対戦比較を行ない，結果を表 1 に示した．対戦相手には囲碁プログラム Fuego[*1] を用い，プレイアウト数は 50 とした．また，対戦中に 3 コウによる同一局面の繰り返しが発生した場合は無勝負とし，対戦結果に含めないこととした．ここで，方策を一様ランダムとした場合の勝率を 1000 局の対戦によって計測した結果，勝率は 20.5% であった．表 1 の結果から，シミュレーションバランシングで勝率の最大値は 58.3%，提案手法では展開の閾値を 30 とした場合に最大値 53.1% パーセントを記録しており，勝率の最大値に関して提案手法がシミュレーションバランシングを下回っていることが分かる．

また，展開の閾値を大きくすることで勝率の最大値が上昇している．本実験条件においては展開の閾値の最大値は 30 としたが，本実験結果からこの展開の閾値をより大きくすることでさらに勝率の最大値が上昇する可能性があると考えられる．

なお，展開の閾値によっては一定の反復回数を超えた後は勝率が減少している．これは上で考察したように，過学習したことによるものであると考えられる．この点に対しては，既存研究 [2] においても反復回数が一定の値を超えるとシミュレーションバランシングの勝率が減少するということが報告されており，今後の研究においては学習局面の数を増やすことや目的関数に正則化項を導入する必要があると考えられる．

5.5　シミュレーションバランシングと提案手法における重みの比較

本節ではシミュレーションバランシングと提案手法において学習後の重みの値を比較する．まず，シミュレーションバランシングと提案手法に対して，それぞれ表 1 の勝率が最大となる反復回数における特徴 1 から 6 の重みの値を以下の表 2 に示す．

また，図 5 から図 10 ではシミュレーションバランシングと提案手法において，学習の反復回数が増えるにつれて特徴 1 から 6 の重みがどのように変化しているかを示した．なお，横軸は学習の反復回数，縦軸は特徴の重み θ_i とした．図 5 の結果から，シミュレーションバランシングでは

[*1] Fuego 開発版 revision 1981

表 2 各学習法における勝率最大時の特徴 1〜6 の重み θ_i

	SB	UCTB-1	UCTB-3	UCTB-5	UCTB-10	UCTB30
特徴 1	3.20	0.30	1.14	1.88	1.70	2.21
特徴 2	5.45	4.92	3.93	4.77	4.85	4.80
特徴 3	2.15	1.88	0.87	1.02	1.37	2.01
特徴 4	2.27	4.48	4.02	3.43	2.48	2.35
特徴 5	-3.82	-3.26	-3.15	-3.79	-4.21	-4.49
特徴 6	3.03	1.13	0.79	2.78	2.36	2.89

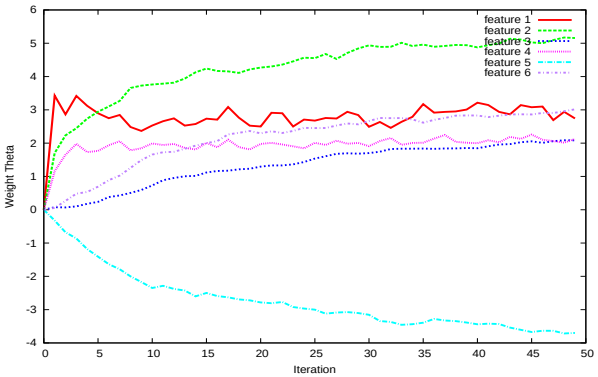

図 5 シミュレーションバランシングにおける学習集過程での重みの変化

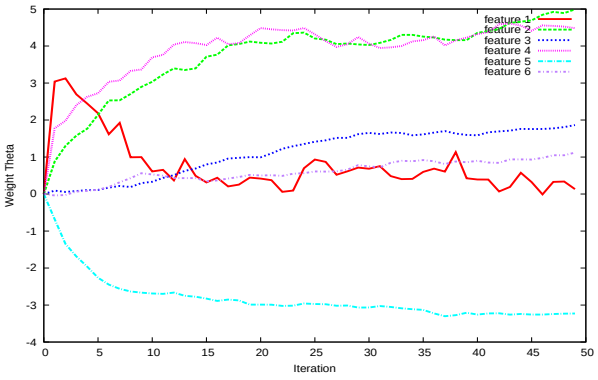

図 6 提案手法における学習過程での重みの変化 (UCTB-1)

特徴 1 の重みが比較的同じ値で安定して推移しているのに対して，提案手法では最初の数反復で特徴 1 の重みが上昇し，その後徐々に値が下がり収束に向かっていいる様子が観察される．また，UCTB-1 においてはその他 3 つのグラフと比較して特徴 4 の重みが大きく学習されており，加えて特徴 1 の値が振動しながら 0 に近づいている様子が観察される．

また，UCTB-10 と UCTB-30 の重みの推移は UCTB-1 と比較すると，シミュレーションバランシングの重みの推移に近い．これは，展開の閾値を大きくすることで探索木が浅く広くなり，結果としてシミュレーションバランシングに近い学習をしている可能性があると考えられる．

6. 結論

本研究では，代表的なモンテカルロ木探索である UCT

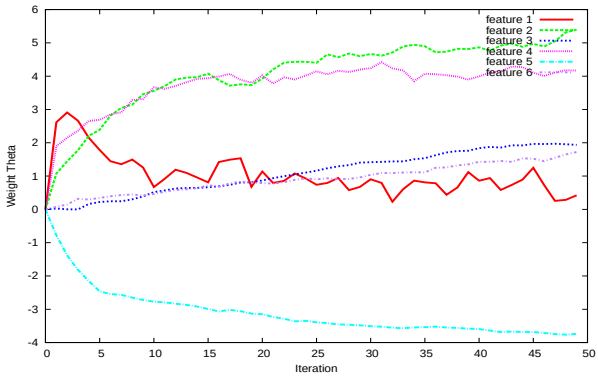

図 7 提案手法における学習過程での重みの変化 (UCTB-3)

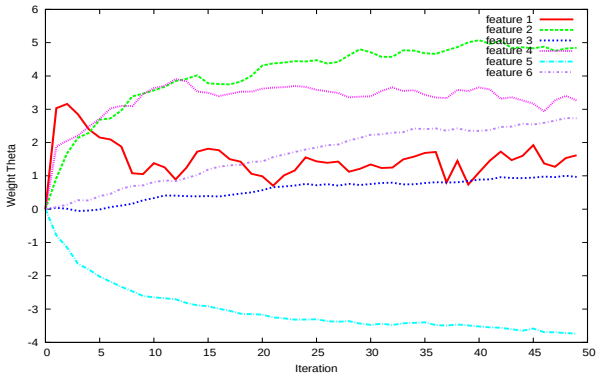

図 8 提案手法における学習過程での重みの変化 (UCTB-5)

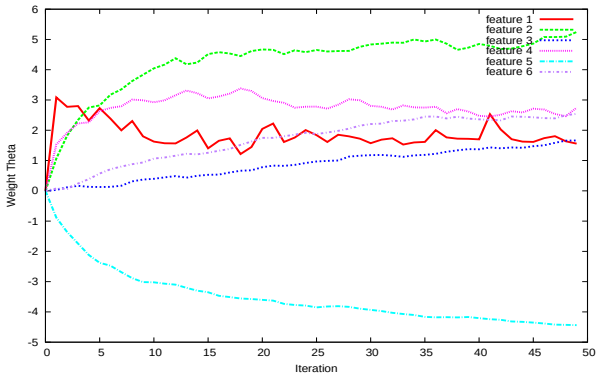

図 9 提案手法における学習過程での重みの変化 (UCTB-10)

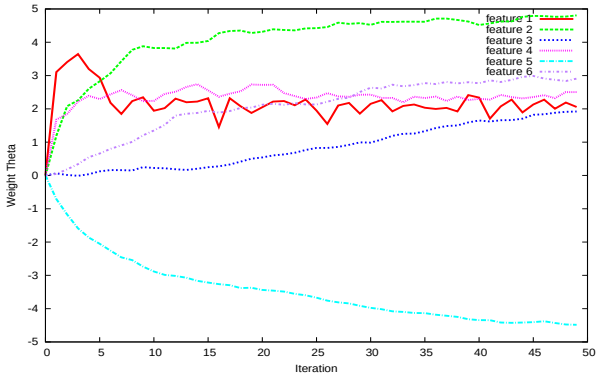

図 10 提案手法における学習過程での重みの変化 (UCTB-30)

アルゴリズムとプレイアウト方策の学習法であるシミュレーションバランシングを組み合わせた方策の学習法を提案した．実験結果から Fuego を対戦相手とした勝率の最大値はシミュレーションバランシングをはっきりと上回る結果は得られなかった．しかしながら，展開の閾値を増やすことで勝率が上昇する傾向が見られたことから，この展開の閾値をさらに大きくすることで勝率がさらに上昇する可能性がある．また，展開の閾値を 30 とした場合では展開の閾値を 1 とした場合よりも特徴 1 から 6 の重みの収束がシミュレーションバランシングにおける重みの収束の様子に近づいていることが観察された．仮に展開の閾値を 500 とすれば学習においてノードが全く展開されないため，シミュレーションバランシングと同じ学習結果が得られることは明らかであるが，展開の閾値を 30 から 500 に上げるにつれて単純に学習結果がシミュレーションバランシングに近づくかは明らかではなく，この点についても調査する必要がある。また，展開の閾値を変更することで勝率や学習の各反復回数での目的関数の値に差が出たことから，学習中の木の成長の仕方やその形が学習に影響を与えたと考えられる．また，UCT を始めとするモンテカルロ木探索では展開の閾値以外にも，UCB1 値の定数項や progressive widening の式などの要素があり，今後はこれらの要素が学習に与える影響も調査する必要がある．また，本実験で使用したプログラムには時間の都合上で RAVE の実装を行わなかったが，RAVE はプレイアウトの結果を広く探索木のノードに伝える効果があり，本提案手法に組み込むことで学習結果に影響を与えることが予測される．

謝辞　本研究は過去の渡辺研究室における卒業研究のアイディアを元にして，実験条件などを拡張したものです．研究の基本的な考え方から細部の取り組み方にいたるまで，1 年間丁寧にご指導していただいた当時の指導教員である渡辺治先生に心から感謝いたします．

参考文献

[1] Silver, D., Tesauro, G.: Monte-Carlo Simulation Balancing. In: ICML (2009).
[2] Huang, S., Coulom, R. and Lin, S.: Monte-carlo simulation balancing in practice. In: ICCG (2010).
[3] 松原仁 編, 美添一樹 · 山下宏 著 コンピュータ囲碁 モンテカルロ法の理論と実践. 共立出版 (2012)
[4] Sutton, R. J, Barto, A. G.: Reinforcement Learning An Introduction. The MIT Press, Massachusetts (1998).
[5] Coulom, R.: Computing Elo ratings of move patterns in the game of Go. International Computer Games Association Journal, 30(4):198-208, 2007.
[6] Kocsis, L. Szepesvari, C.: Bandit based Monte-Carlo planning. 15th European Conference on Machine Learning, pp.282-293, 2006.
[7] `http://www.yss-aya.com/cgos9_201101_201204_2500over.cab`