選択的シミュレーションに基づいた評価関数の学習

大 崎 泰 寛[†]　小 谷 善 行[††]

シミュレーションによって獲得された勝率に基づいて着手を選択する研究が近年注目されている．ところがゲームにおいて限られた思考時間内に精度の高い勝率を獲得するためには多くのシミュレーション回数を要する問題点がある．そこで本論文では対局中にシミュレーションを行う代わりに，機械学習によって調整された評価関数からシミュレーション結果を推定する手法を提案した．先行研究はランダムシミュレーションによる勝率から評価関数を学習させたのに対して，本手法はゲームの知識を用いて選択的にシミュレーションした結果を学習に適用させた．その結果，ブロックスデュオにおいて先行研究の手法と比較して本手法が優れた対局結果を示すことを確認した．

A Learning Method of Evaluation Function Based on Selective Simulations

Yasuhiro OSAKI[†]　Yoshiyuki KOTANI[††]

The researches of selecting moves using the winning probability obtained through simulations are treated as hot topics in these years. However, there is the problem that obtaining the accurate winning probability through simulations needs too much times in limited think time in games. In this paper, we suggest a way of applying evaluation function based on machine learning to estimation of simulation outcome instead of actual running simulations during games. In contrast previous study evaluation function was approximated by the probability of winning, the study tries to apply the result of selective simulations, using domain-specific knowledge, to learning. Using BlokusDuo as a testing environment, we evaluate the learning result of approximating winning probability function and confirm that our learned player, in comparison to the other player based on the previous study, has been seen to yield better results of games.

1. はじめに

シミュレーションによって得られたサンプルに基づいて着手を選択する研究が，近年のゲーム情報学において大変盛んである．その中でもモンテカルロ法を木探索に拡張したアルゴリズムである UCT(Upper Confidence bounds applied to Trees)[1)]の登場がコンピュータ囲碁の棋力を向上に大きく貢献したことに注目が集まっている．コンピュータ囲碁の他にも，コンピュータ将棋やコンピュータブロックスデュオのゲームシステムに UCT を適用させた研究が行われている[7)8)]．また，シミュレーションにゲームの知識や様々なヒューリスティックを加えることで，より良いサンプルを得られることが知られ，コンピュータ囲碁の実力上位のプログラムがそれらを適用している．

ところが，シミュレーションにゲームの知識を導入することで，得られるサンプルの質は改善された反面，一回の playout にかかる時間が増加したため限られた思考時間内で得られるサンプル数が減少してしまうトレードオフ問題があった．また精度の高い勝率を得るためには多くのサンプルが必要であり，対局中に多くのシミュレーション回数が不可欠であった．

そこで本論文では，思考時間内にシミュレーションして勝率を得る代わりに，機械学習によって調整された評価関数を用いることでシミュレーション結果の推定を図った．本論文で提案する学習手法の目標は，学習データにおけるゲームの各局面の静的評価値を評価関数にもとづいた選択的シミュレーションによって得られる勝率に近似させることである．先行研究[9)]は単純にランダムシミュレーションによって得られた各局面の勝率を教師とした学習であったのに対して，提案手法は学習途中の評価関数を確率的に用いて選択的にシミュレーションをすることで，教師の質をさらに向上させた．学習によって得られた評価関数をもとに集められた“より起こりそうな”サンプルから評価関数を強化し，シミュレーション結果に対する推定精度を向上させることが本論文の論旨である．

† 東京農工大学大学院 工学府 情報工学専攻
Department of Computer and Information Sciences, Graduate school of Technology, Tokyo University of Agriculture and Technology
†† 東京農工大学大学院 工学府
Department of Computer and Information Sciences, Tokyo University of Agriculture and Technology

提案手法によって得られた評価関数の学習成果を測るために，ブロックスデュオを実験環境として，従来手法によって学習された評価関数と対局して比較させた．その結果 500 回の対局中で最大65.3%の勝率が得られ，従来手法に対して提案手法はより優れた学習法であることを示した．続いて，学習時のシミュレーション回数と同等の回数を対局中に実際シミュレーションするモンテカルロプレイヤに対して，探索なしで最大 36.0%，探索ありで最大 63.0%の勝率を得た．また，無作為に用意したテスト局面のシミュレーション結果と評価関数の静的評価値との絶対誤差について調べた．

以下 2 章では先行研究と本研究の位置づけを明記し，3 章では提案する学習アルゴリズムを示す．そして 4 章では実験結果を示し，5 章では結論と今後の課題について述べる．

2. 関 連 研 究

機械学習によるゲーム知識をシミュレーションに導入する研究が近年盛んにおこなわれている．文献 2)では，プロの指し手に含まれる各特徴の Elo-rating を算出し，シミュレーションに組み合わせることによって棋力の向上を図った．文献 3)は，best-of-n アルゴリズムがゲームの知識の利用法として特に効果的であると主張した．文献 4)では，盤面の局所的なパターンから構成された評価関数を TD(λ)によって学習して UCT アルゴリズムに導入したところ，思考時間内におけるサンプル数が大幅に減少したため十分なシミュレーションができなかったと述べた．

ところが，いずれの手法もゲームの知識を導入させることで乱数シミュレーションよりもサンプルの質が向上した反面，思考時間内のシミュレーション回数が減少する問題点があった．

そこで，シミュレーションによって得られた勝率に注目することで，ゲームに適用させた評価関数の学習アルゴリズムを大崎らは研究してきた[5)9)10)]．文献 5)10)は，ゲームの終端局面で環境から与えられる勝敗を報酬とするのみならず，各非終端局面からランダムシミュレーションによって近似される勝率を代理的な報酬と見立てて，環境から与えられた代理報酬の総和の最大化を目的とした強化学習[6)]の一種である TDMC(λ)学習アルゴリズムを提案した．

一方で文献 9)は，対局中にランダムシミュレーションをする代わりに，線形評価関数によってシミュレーション結果を推定するように学習させる手法を提案した．さらに，対局中に実際シミュレーションを行うモンテカルロプレイヤに対して，シミュレーション結果を学習させた評価関数が優位であることを実験から示した．

本論文は，学習中の評価関数を利用して選択的にシミュレーションを行った結果から評価関数を学習させる手法を提案する．そして，より良いサンプルを選択的に獲得することで教師の質を向上させた点が，先行研究とは異なった新規性である．

3. 選択的なシミュレーションに基づいた評価関数の学習アルゴリズム

評価関数を用いて学習局面から選択的にシミュレーションをした結果と，その局面の静的評価値との平均二乗誤差を目的関数として，最急降下法を用いて目的関数を最小化する方向に各特徴を調整する方法について説明する．

3.1 評価関数の構造

評価関数$f(w,x_t)$の構造について述べる．まず，エピソードの終端局面の時刻をTとしたとき，時刻ステップを$t\epsilon[1,T]$と定義する．そして，時刻tの局面P_tにおける静的な評価値v_tを単純な線形結合$v_t=f(w,x_t)=w^{tr}x_t$としたときに，その局面における重みベクトルは$w=(w_1,w_2,\cdots,w_N)$，特徴ベクトルは$x_t=(x_{t1},x_{t2},\cdots,x_{tN})$とそれぞれ長さが$N$のベクトルで表すことができる．局面$P_t$の形勢を表す特徴ベクトル$x_t$については，14×14 の正方からなるブロックスデュオの盤面に置かれたピースのマス目（以下スコア）や，置いたピースの角のマス（以下有効マス），置いたピースの辺のマス（以下死角マス），そして置いたピースの及ぼす勢力範囲の分布をもとに整数要素から構成されている．

表 1　評価関数の特徴

特徴	詳細
スコア	置いたピースのマス 最終的にスコアの多いプレイヤの勝ち
有効マス	置いたピースの角のマス いずれのスコアや自分の死角マスでない
死角マス	置いたピースの辺のマス 自分のピースは置けない
確定マス	自分の有効マスでかつ相手の死角マス
勢力範囲	有効マスからの距離が 3 以内のマス 勢力範囲同士の重複は認めるものとする

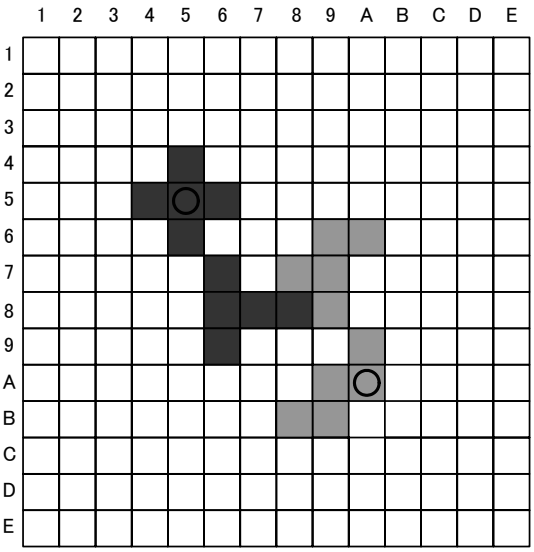

図 1　ブロックスデュオの盤面

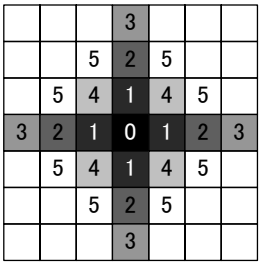

図 2　勢力範囲と有効マスからの距離

例えば，図 1 の座標(5,5)を覆った側（先手番）にとって濃く塗られたマスはスコアを示す．また，先手番にとって座標(7,4)は有効マスであり，座標(7,5)は死角マス，座標(9,9)は確定マスに該当する．そして本論文では，有効マスからのマンハッタン距離が 3 以内のマスを（図 2 参照），d=0～d=5 まで計 6 種類の勢力範囲として別個の特徴とした．ただし d=0 は有効マスのことを指し，勢力範囲に該当するためには対象のマスにいずれのプレイヤのピースも置かれていないかつ，死角マスでないことが条件となる．さらに有効マスからの最短経路のすべてに邪魔がないことを考慮するために，経路上に相手のピースがないことも勢力範囲の条件として加える．したがって，先手番にとって図 1 の座標(A,7)は勢力範囲 d=5 となるが，座標(A,B)は勢力範囲 d=5 には当てはまらない．また勢力範囲は各有効マスからそれぞれ計算したとき，複数の勢力範囲の距離を持ったマスが現れた場合は重複を認め，単純に重みを加算することにした．

評価関数は，上記の計 9 種類の特徴が異なる座標上に存在した場合に，それぞれ別個の特徴としてみなす固有特徴 14×14×9 種類と，いずれの座標上に存在しようがすべて同一のものと見なす共有特徴 1×9 種類から構成されている．この共有特徴を評価関数に導入することで，学習データに十分登場しないようなスパースな固有特徴が局面評価に大きな影響を与えることを回避させた．

3.2　重みベクトルの更新

局面P_tを root に選択的シミュレーションを行って得られた勝率r_tと，シグモイド関数$\sigma(x)$で正規化した評価値$V_t = \sigma(v_t)$との平均二乗誤差を最小にするために，最急降下法を用いて重みベクトルwを更新する．

まず，学習データ内の任意の 1 エピソードにおける目的関数OFを，

$$OF = \sum_{t=1}^{T} [r_t - V_t]^2 \qquad (1)$$

とする．目的関数OFは学習データの初手から対局終了を表すT手までに観測された二乗誤差の総和を表している．続いて，式(1)で示された目的関数OFを最小化するために重みベクトルwの勾配ベクトル$\nabla_w OF$を求める．

$$\nabla_w OF = -2k \sum_{t=1}^{T} [r_t - V_t]\sigma(v_t)(1 - \sigma(v_t))x_t \qquad (2)$$

ただし，kは以下式(3)のシグモイド関数のゲイン係数である．

$$\sigma(v_t) = \frac{1}{1 + exp(-kv_t)} \qquad (3)$$

式(2)から求められた任意の 1 エピソードの勾配ベクトル$\nabla_w OF$を利用することで，重みベクトルwを更新することができる．本論文における学習モデルでは，それぞれの着手ごとに逐次的な更新をせず，1 エピソードの終端までの差分の総和を求め，一括したバッチ更新を行った．

例えば，要素jの重みw_jを学習データにおける任意の 1 エピソードから更新する場合を考える．学習率をα，時刻tにおける要素jの特徴ベクトルをx_{tj}としたときw_jは以下の式(4)から更新される．

$$w_j := w_j - \alpha \frac{\partial OF}{\partial w_j}$$

$$= w_j + \alpha k \sum_{t=1}^{T} [r_t - V_t]\sigma(v_t)(1 - \sigma(v_t))x_{tj} \qquad (4)$$

1) 学習中の評価関数の重みベクトルwに基づいて自己対局から学習データを獲得する
   i) 各学習局面P_tにおける評価値V_t，特徴ベクトルx_tを保存する
   ii) 各学習局面P_tにおけるシミュレーション上の勝ち数win_num_tを 0 に初期化する
2) 各学習局面P_tに対して以下の処理を行う
   i) シミュレーション局面を$P'_t \leftarrow P_t$とする
      a) 局面P'_tの可能手をすべて生成する
      b) その中から無作為にn個可能手を選択する
         （ただし，可能手がn個に満たない場合は可能手のすべてを選択する）
      c) 選択した中から評価値が最も高かった局面を次局面としてP'_tを更新する
      d) 局面P'_tがゲーム終端に達するまで a)からの作業を繰り返す
   ii) もし終端局面P'_tが勝ち局面ならば，$win_num_t \coloneqq win_num_t + 1$とする
      （ただし，引き分け局面ならば，$win_num_t \coloneqq win_num_t + 0.5$とする）
   iii) シミュレーション回数がmに達するまで i) からの作業を繰り返す
   iv) 学習局面P_tにおける勝率r_tを求める； $r_t = \frac{1}{m} win\_num_t$
3) 更新式(4)から，重みベクトルwの各要素w_jを更新する
4) 更新回数が l に達するまで 1) からの作業を繰り返す

図 3　選択的シミュレーションに基づいた学習フロー

3.3 選択的シミュレーション

文献 3)において playout の質を向上させるために有効な手法であった best-of-n アルゴリズムを，本学習手法の選択的シミュレーションに図 3 の手順で適用した. best-of-n アルゴリズムはシミュレーション上の各学習局面において，ランダムに選んだn個の可能手の中から最も評価値の高いものを末端まで選択して勝率を求めるため，末端までランダムシミュレーションによって勝率を求めていた従来手法とは異なる．ただし，シミュレーション中に次局面の総数がnを下回った場合は，次局面すべての中から最も評価値の高かった局面を次局面とした．nを小さくするにつれてシミュレーションがランダムに近づき，nを大きくするにつれてシミュレーションに多くの時間がかかることに加えて，サンプルが評価関数に基づいて大きく偏ってしまう．したがって，ブロックスデュオのような膨大な状態数を持つゲームを学習環境とした場合，機械学習などから適切なnを求める必要がある．

この best-of-n アルゴリズムは単純だが，シミュレーション上の次局面すべてに対して逐一評価関数を呼び出す必要がないことから高速であり，また評価関数を確率的に利用できることからより良いサンプルを獲得できると考えられる．以下，本論文ではnを選択数と呼ぶ.

4. 実　　験

以下の学習パラメータにおいて評価関数の重みを調整した．そして提案手法を評価するために，ランダムシミュレーションを教師とした従来手法[9] によって得られた評価関数ならびに対局中にシミュレーションを行って着手を選択するモンテカルロプレイヤとの対局実験を行った.

- ゲイン係数 $k = 0.02$
- 学習率 $\alpha = 0.5$
- 選択的シミュレーションの回数 $m = 100$
- 更新回数 $l = 10{,}000$
- 選択数 $n = 1, 2, 4, 8, 16, 32$

学習データは, ε-greedy 方策に基づいて自己対局から獲得した．このとき，評価関数の更新回数 l に応じて ε を徐々に低下させることで, 学習の収束を図った.

また，無作為に用意したテスト局面のシミュレーション結果と静的評価値との絶対誤差を調べた.

4.1 提案手法の学習結果

上記の学習パラメータにおいて，評価関数の各特徴を学習させた結果，以下の図 4 のような学習曲線が得られた．ただし，選択数は 4 として，前述の共有特徴の重みのみを図示したものとする.

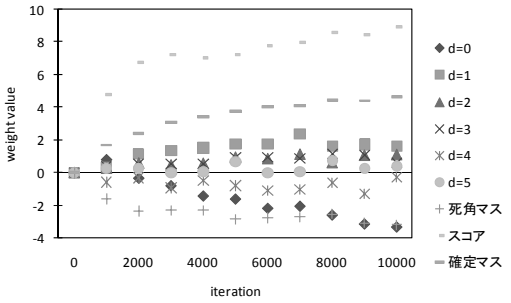

図 4　各共有特徴の重みの変化

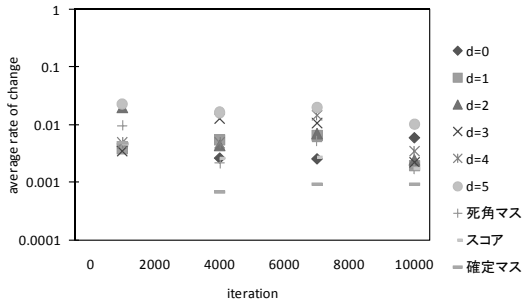

図 5　各固有特徴の平均変化率の変化

盤上のいずれの座標に置かれていてもすべて同じ特徴と見なす共有特徴の重みは図 4 のように得られた．図 4 から，スコアの価値が最も高いことがわかった．また，各固有特徴の収束の程度を調べるために一棋譜あたりにおける各固有特徴の要素jの変化率Δw_j^lを以下の式(5)に示した．

$$\Delta w_j^l = \left| \frac{w_j^{l+1} - w_j^l}{w_j^l} \right| \tag{5}$$

ただし，w_j^lは l 回目の更新後のw_jの値として，要素jの 14×14 全座標における変化率の平均$\overline{\Delta w_j^l}$を図 5 に示した．更新回数が増加するにつれて平均変化率は減少し，最終的にはすべての固有特徴の平均変化率が 1%を下回ることがわかった．ただし，固有特徴ごとに収束の程度は異なり，学習終了時に勢力範囲 d=5 と確定マスとの間に 10 倍程度の開きがあった．この理由には，確定マスは有利な手番側に多く見られる特徴であるため比較的学習されやすかったのに対して，勢力範囲 d=5 は有利不利問わずあらゆる局面に表れるため学習されにくかったことが考えられる．

4.2　先行研究との対局結果

従来手法はランダムシミュレーションから勝率を獲得しているため，best-of-1 アルゴリズム，つまり選択数 1 による選択的シミュレーションと同一の学習手法と見なすことができる．評価関数の構造や学習パラメータはすべて同じものを用い，それぞれの重みベクトル$\boldsymbol{w}$の各要素はすべて初期化した．お互いのプレイヤの思考時間は一手 1 秒とし，$\alpha\beta$法による反復深化を用いて探索し，思考時間終了時の段階で得られている最善手を返すものとした．そこで先後合わせて 500 回対局を行い，従来手法（best-of-1）に対する提案手法（best-of-2 から best-of-64 まで）の勝率を求めた結果が図 6 である．ただし，引き分けは 0.5 勝と見なして勝率を計算したものとして，best-of-1 に対する best-of-1 の勝率を便宜上 0.5 と定めた．また，似た対局結果になることを避けるためお互いのプレイヤは初手のみランダムにペントミノを置くように決めた．その結果，2 手目終了までに 315^2 通りの局面が生成されるために十分ばらつくことが予想される．

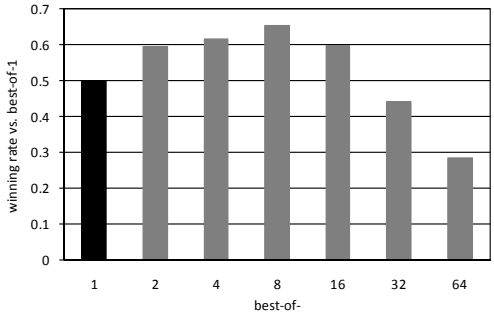

図 6　先行研究に対する勝率の変化

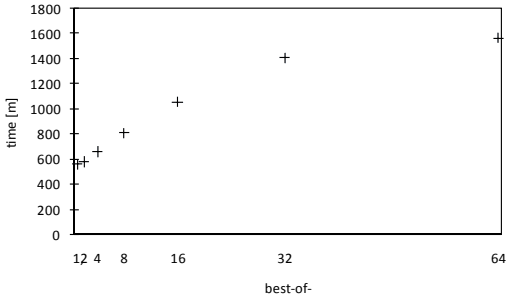

図 7　選択数と学習時間の関係

図 6 から，選択数が 2, 4, 8, 16 のとき先行研究よりも高い勝率をあげることがわかった．特に選択数が 8 のときに最も高い勝率 65.3%が得られた．また，選択数が 32 と 64 のときは従来手法に負け越していることがわかった．この原因には，ブロックスデュオは終盤に近づくにつれ極端に可能手が減るためシミュレーションのランダム性が失われ，同一の偏ったサンプルから学習されたことが考えられる．

表 2　先行研究との勝率と学習時間の関係

選択数	選択数 1 に対する勝率	学習時間 [／best-of-1]
1	0.500	1.000
2	0.593	1.037
4	0.616	1.178
8	0.653	1.447
16	0.596	1.886
32	0.440	2.509
64	0.283	2.792

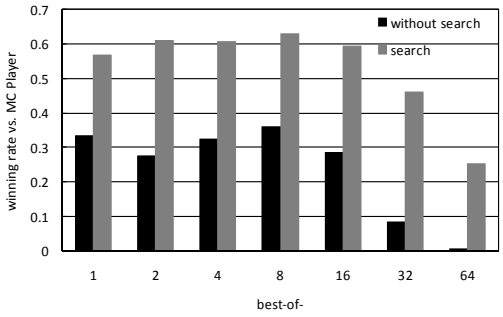

図 8　モンテカルロプレイヤに対する勝率の変化

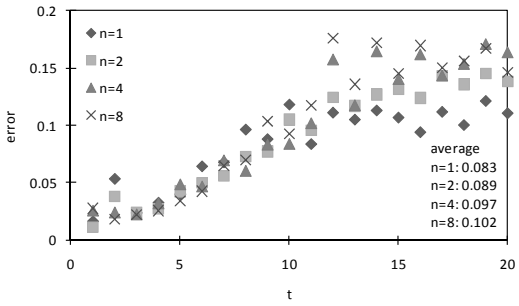

図 9　局面の手数と絶対誤差の関係

また，各選択数の学習時間を計測したものが図 7 である．学習時間とは，図 3 のとおり更新時間とシミュレーション時間との和から求められる．いずれの選択数においても更新時間は等しいことが考えられるが，選択数に比例してシミュレーション時間は増加すると考えられる．ところが，図 7 より明らかに学習時間と選択数の線形性は保たれていないことがわかる．この理由には，ブロックスデュオはゲームの終盤に近づくと可能手が序盤と比べて極端には少ない局面が多くなるため，シミュレーション上の可能手が選択数を下回るためであると考えられる．

以上の実験結果を表 2 にまとめた．本項から，ゲームの環境に対して適切な選択数であるならばランダムシミュレーションよりも選択的シミュレーションの方が優れた学習となるが，選択数が大きすぎるとシミュレーションの改良に悪影響を与えることがいえる．

4.3　モンテカルロプレイヤとの対局結果

前述の評価関数と対局中にシミュレーションを行うモンテカルロプレイヤを対局させた．モンテカルロプレイヤは各可能手に対してランダムシミュレーションを 100 回行い，最も勝率の高かったものを着手とした．一方，評価関数は探索なしの一手読みと一手 1 秒の$\alpha\beta$法による反復深化探索を試した．対局条件は 4.2 と同じで先後合わせて 500 回対局した結果を図 8 に示す．探索なしの一手読みでは選択数が 8 のときの勝率 36.0%が最大であり，選択数が 64 のとき勝率はわずか 0.6%しかなかった．また選択数が 1 のときの勝率は 33.4%に留まった．両プレイヤは局面が内包する理論上一意に定められた勝率の推定から最善の手を選択するという同じ方策であったにも関わらず，勝率は 50.0%前後でなかったことから，学習回数が不十分であること，または重みベクトル$\boldsymbol{w}$の特徴が十分でなかったことが考えられる．続いて，一手 1 秒の$\alpha\beta$法による反復深化探索で対局した結果，選択数が 1 から 16 においてモンテカルロプレイヤに勝ち越すことがわかった．その中でも，選択数が 8 のときは最大で 63.0%の勝率をあげることがわかった．また 4.2 同様，ある程度選択数が大きくなると勝率が下がる傾向があった．

4.4　ランダム局面に対する評価関数の予測性能

未知のテスト局面に対して，学習した評価関数の勝率の予測性能について調べた．学習した評価関数をもとに無作為に生成した初手から 20 手目までの各 100 局面をテスト局面として利用した．ただし，テスト局面の勝率r_tと局面の静的評価値$\sigma(v_t)$との絶対誤差を$|r_t - \sigma(v_t)|$とし，各テスト局面に付与された勝率r_tは 10000 回の選択的シミュレーションから獲得した．選択数 1 から 8 までの予測結果を図 9 に示す．図 9 より，選択数が大きいほど絶対誤差の平均値が大きくなることがわかった．この理由には選択数が大きいと勝率がより勝敗を反映する値に近づくため，勝率がより極端な値をとるからであると考えられる．例えば選択数を全可能手と定めると，best-of-n アルゴリズムの性質から統計的な要素がシミュレーションから排

除されるため各局面に対して一意のr_tが求まる．また，選択数が 1 の場合は前述のとおり，ランダムシミュレーションと同義であるため，r_tは勝率と一致する．そのため選択数が大きいほど，局面の静的評価値では近似しにくかったと考察している．

また，手数が深くなるにつれて絶対誤差が大きくなる理由には，手数が深くなると局面の静的評価が難しくなることが考えられる．ブロックスデュオにおいて，相手のピースの侵入をいかに防ぐかがゲームの勝敗に大きく関与するというゲームの性質がある．ゲームの序盤では，お互いのプレイヤが十分にピースを置いていないため，侵入について考慮する必要がないため局面の静的評価はしやすい．一方でゲームの中盤以降では，ピースが多く置かれているため相手の陣地への侵入について深く考えなくてはならない．そのため，序盤に比べて中盤以降の方が絶対誤差が大きくなったことが考えられる．このことから，ゲーム木探索で得られた動的な評価値を選択的シミュレーションの結果に近似させることが今後の研究の発展課題になるだろう．

5. おわりに

本論文は，学習中の評価関数を確率的に利用して選択的にシミュレーションを行った結果から評価関数を調整する新しい学習手法を提案した．本手法は，ランダムシミュレーションによって得られる勝率よりもより実際の勝敗に近い値を学習に適用することで評価関数の強化を図った．そして実験環境をブロックスデュオとして，対局実験から提案手法による評価関数が従来手法よりも優れた対局成績を収めることがわかった．特に提案手法のシミュレーション時の選択数を 8 としたとき，従来手法に対して 1.447 倍の学習時間をかけて 65.3%と最も高い勝率をあげることがわかった．続いて，対局中に実際シミュレーションをして着手を選択するモンテカルロプレイヤに対して，選択数を 8 として学習した場合，探索なしで最大 36.0%，探索ありで最大 63.0%の勝率をあげることがわかった．またこのとき，未知のテスト局面の選択的なシミュレーション結果と局面の静的評価値との絶対誤差が平均して 10.2%であることが実験からわかった．

今後の課題として，静的な評価値ではなくゲーム木探索によって得られた動的な評価値をシミュレーション結果に近似させることから，学習成果の向上について調べることがあげられる．

続いて，文献 4)において学習によって得られた評価関数の利用方法として実験された，ガウス雑音やソフトマックス分布などの他の選択的なシミュレーション手法と best-of-n アルゴリズムを比較することがあげられる．いずれの選択方法においても，シミュレーションの速度と精度の関係が重要となるだろう．

また UCT の改良策として，本手法で得られた評価関数を UCT の木探索部分およびシミュレーション部分に導入することが考えられる．特に，木探索部分における末端で展開した新規局面の初期値推定 [4)11)]に適用することで，優先度の高い局面からシミュレーションが可能になると期待できる．

参考文献

1) Levente Kocsis, Csaba Szepesvári, "Bandit Based Monte-Carlo Planning", Proceedings of the 15th European Conference on Machine Learning, pp.282-293, Sept. 2006.

2) Rémi Coulom, "Computing Elo Ratings of Move Patterns in the Game of Go", Proceedings of the 9th International Conference on Computer Games, pp.113-124, Apr. 2007.

3) Peter Drake, Steve Uurtamo, "Move Ordering vs Heavy Playouts: where sould heuristics be applied in Monte Carlo Go?", Proceedings of the 3rd annual North American Game-On Conference, pp.35-39, Sep. 2007.

4) Sylvain Gelly, David Silver, "Combining Online and Offline Knowledge in UCT", Proceedings of the 24th International Conference on Machine Learning, pp.273-280, Jun. 2007.

5) Yasuhiro Osaki, Kazutomo Shibahara, Yasuhiro Tajima, Yoshiyuki Kotani, "An Othello Evaluation Function Based on Temporal Difference Learning using Probability of Winning", IEEE Symposium on Computational Intelligence and Games, pp.205-211, Dec. 2008.

6) Richard Sutton, Andrew Barto, "Introduction to Reinforcement Learning", MIT Press, 1998.

7) 佐藤佳州，高橋大介，"モンテカルロ木探索によるコンピュータ将棋"，第 13 回ゲームプログラミングワークショップ, pp.1-8, Nov. 2008.

8) 中村秋吾，三輪誠，近山隆，"静的評価関数を用いた UCT の改善"，第 12 回ゲームプログラミングワークショップ, pp.44-51, Nov. 2007

9) 大崎泰寛，築地毅，但馬康宏，小谷善行，"勝率に近似させた評価関数の性能について"，情報処理学会 ゲーム情報学研究会報告, vol.2009-GI-22 No.5, Jun. 2009.

10) 大崎泰寛，柴原一友，但馬康宏，小谷善行，"TDMC(λ)に基づく評価関数の調整"，第 13 回ゲームプログラミングワークショップ, pp.73-79, Nov. 2008.

11) 但馬康宏, 小谷善行, "モンテカルロ法における勝率近似関数の組み込み方法"，第 13 回ゲームプログラミングワークショップ, pp.100−103, Nov. 2008.